Федеральное государственное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований Российской академии наук»
(ФГУ ФНЦ НИИСИ РАН)

# ТРУДЫ НИИСИ РАН

ТОМ 5 N 2

## МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

## ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2015

**Тематика номера:**
Моделирование динамических природных и рукотворных процессов, моделирование в микроэлектронике, автоматизация управленческой деятельностью, математические исследования и вопросы численного анализа, построение программ реального времени

Журнал публикует оригинальные статьи по следующим областям исследований: математическое и компьютерное моделирование, обработка изображений, визуализация, системный анализ, методы обработки сигналов, информационная безопасность, информационные технологии, высокопроизводительные вычисления, оптико-нейронные технологии, микро- и наноэлектроника, вопросы численного анализа

**The topic of the issue:**
Modeling of dynamic natural and man-made processes, modeling in microelrctronics, automation of administrative activities, mathematical investigations and problems of numerical analysis, construction of real-time programs

The Journal publishes novel articles on the following research arias: mathematical and computer modeling, image processing, visualization, system analysis, signal processing, information security, information technologies, high-performance computing, optical-neural technologies, micro- and nanoelectronics, problems of numerical analysis

Заведующий редакцией: Ю.Н. Штейников

Издатель: ФГУ ФНЦ НИИСИ РАН,
117218, Москва, Нахимовский проспект 36, к. 1

---

**СОДЕРЖАНИЕ**

## V. ПОСТРОЕНИЕ ПРОГРАММ РЕАЛЬНОГО ВРЕМЕНИ

# Обобщенный подход к описанию распределения энергий спиновой системы

**Б.В. Крыжановский[1], Л.Б. Литинский[2]**

*1-член-корреспондент РАН, 2- кандидат физико-математических наук*

**Аннотация:** Исследован спектр энергий системы $N$ бинарных спинов. Показано, что конфигурационное пространство состояний можно разбить на $N$ классов, распределение энергий внутри которых в асимптотическом пределе $N \to \infty$ хорошо аппроксимируется гауссовой плотностью. Для каждого из классов в самом общем виде получены точные выражения для первых трех моментов распределения энергий, в том числе – при наличии неоднородного магнитного поля. Получены выражения для дисперсии квазиэнергий в локальном минимуме. Излагаются результаты компьютерного моделирования распределения энергий для стандартной модели Изинга и моделей спинового стекла Эдвардса-Андерсона и Шеррингтона-Киркпатрика. На этой основе проведено систематическое обоснование нового метода расчета статистической суммы.

**Ключевые слова:** статистическая сумма, гауссово распределение энергий.

## 1. Введение

В [1,2] было сделано краткое сообщение о новом методе вычисления статистической суммы системы $N$ бинарных спинов $s_i = \{\pm 1\}$, $i = 1,...,N$, состояние которой задается конфигурационным вектором $\mathbf{S} = (s_1, s_2, ..., s_N)$ и энергией:

$$E \equiv E(\mathbf{S}, \mathbf{H}) = -\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} s_i s_j - \sum_{i=1}^{N} H_i s_i \,. \quad (1)$$

Здесь $\mathbf{H} = (H_1, H_2, ..., H_N)$ - вектор магнитного поля, $\mathbf{T} = \left(T_{ij}\right)$ - симметричная матрица с нулевой диагональю ($T_{ii} = 0$).

Основная идея метода очень проста и может быть пояснена следующим образом. Рассмотрим статистическую сумму $Z = \sum_{\mathbf{S}} \exp\left[-\beta E(\mathbf{S})\right]$, где суммирование ведется по всем состояниям $\mathbf{S}$, а $\beta$ - обратная температура. Для вычисления $Z$ необходимо знать распределение энергий в пространстве состояний $\mathbf{S}$, однако в общем случае оно неизвестно. Наш подход состоит в том, что все множество состояний разбивается на $N$ классов, распределение энергий в каждом из которых аппроксимируется гауссовой плотностью с известными средним значением и дисперсией.

Разбиение пространства на классы производится следующим образом. Выберем в качестве начальной конфигурации произвольную точку пространства $\mathbf{S}_0 = (s_1^{(0)}, s_2^{(0)}, ..., s_N^{(0)})$. Назовем классом $\Omega_n$ множество конфигураций, отличающихся от $\mathbf{S}_0$ противоположными значениями $n$ спинов. Фактически, $\Omega_n$ является $n$-окрестностью начальной конфигурации $\mathbf{S}_0$. Совокупность $n$–окрестностей $\Omega_n$, $n = 0,1,2,...,N$, исчерпывает все множество конфигураций, а величину $Z$ можно представить в виде:

$$Z = \sum_{n=0}^{N} \sum_{\mathbf{S}\in\Omega_n} \exp\left[-\beta \cdot E(\mathbf{S})\right]. \quad (2)$$

В общем случае неизвестно, как распределены энергии состояний из $\Omega_n$. Эмпирический факт состоит в том, что это распределение всегда является одногорбым. Более того, оно хорошо аппроксимируется гауссовой плотностью вероятности. Поясним, что имеется в виду. Пусть $\bar{E}_n$ - среднее значение энергии для состояний из $\Omega_n$, а $\sigma_n^2$ - дисперсия распределения (значения этих характеристик можно, например, оценить по случайной выборке). Тогда при больших $N$, по меньшей мере в энергетическом интервале $\bar{E}_n - 3\sigma_n \le E \le \bar{E}_n + 3\sigma_n$ эмпирическое распределение энергий хорошо ложится на кривую $\exp[-\frac{1}{2}(E-\bar{E}_n)^2/\sigma_n^2]/\sqrt{2\pi}\sigma_n$. Следовательно, если получить аналитические выражения для $\bar{E}_n$ и $\sigma_n$, суммирование в (2) по состояниям из $\Omega_n$ можно заменить интегрированием по гауссовой плотности. Это позволит свести выражение (2) к интегралу от экспоненты, в показателе которой стоит большая константа $N$. Подобные интегралы вычисляются стандартным методом Лапласа (метод перевала).

Первые полученные на этом пути результаты изложены в [1,2], однако обоснование метода в эти публикации не уместилось. Настоящая работа посвящена описанию спектра энергий спиновой системы и систематическому обоснованию изложенного в [1,2] метода. Вычисление статистической суммы – одна из ключевых задач статистической физики. Первоначально она разрабатывалась исключительно в контексте физических приложений. Однако со временем вычисление статистической суммы оказалось востребованным и для других задач – при анализе

нейронных сетей [3], в задачах комбинаторной оптимизации [4], машинного обучения [5] и др.

Первые попытки описания формы энергетической гиперповерхности (1) были предприняты в [6,7], где предложено разбиение пространства конфигурационных состояний на микроканонические ансамбли и получены асимптотические выражения для первых двух моментов распределения энергий в этих ансамблях. Более точные выражения для среднего и дисперсии приведены в работах [8]-[10]. Эти выражения успешно использованы при разработке алгоритмов минимизации [8]-[15] и для описания сложных нейронных систем [16]-[19].

Статья имеет следующую структуру. Раздел 2 содержит вывод строгих выражений для $\bar{E}_n$ и $\sigma_n$. В разделе 3 получены оценки для дисперсии квазиэнергий в локальном минимуме, которые требуются для применения нашего подхода в моделях со случайной матрицей $\mathbf{T}$. В разделе 4 излагаются результаты компьютерного моделирования, демонстрирующие, насколько хорошо распределение энергий состояний из $\Omega_n$ можно аппроксимировать гауссовой плотностью. Здесь приводятся графики, на которых результаты машинного эксперимента сопоставляются с гауссовой кривой. В разделе 5 проводится анализ экспериментальных данных и обсуждение полученных результатов. В Приложение отнесены сложные технические детали.

## 2. Вычисление моментов распределения

Обозначим через $\mathbf{S}_0 = (s_1^{(0)}, s_2^{(0)}, ..., s_N^{(0)})$ начальную конфигурацию. В качестве $\mathbf{S}_0$ может выступать как случайная, так и специально выбранная конфигурация. Ее $n$–окрестность $\Omega_n$ состоит из $L = C_N^n$ конфигураций, отличающихся от $\mathbf{S}_0$ значениями $n$ координат. Все они имеют одинаковую относительную намагниченность $m = \mathbf{S}\mathbf{S}_0^+ / N = 1 - 2n/N$. Покажем, как вычисляются среднее значение и дисперсия распределения энергий состояний из $\Omega_n$:

$$\bar{E}_n = \frac{1}{L}\sum_{\mathbf{S}\in\Omega_n} E(\mathbf{S}), \quad \sigma_n^2 = \frac{1}{L}\sum_{\mathbf{S}\in\Omega_n}\left[E(\mathbf{S}) - \bar{E}_n\right]^2, L = C_N^n. \quad (3)$$

Сначала все вычисления проделаем для нулевого магнитного поля ($\mathbf{H} = 0$), затем обобщим их на случай $\mathbf{H} \neq 0$. Значение энергии для начальной конфигурации $\mathbf{S}_0$ в отсутствие магнитного поля обозначим через $E_0 = E(\mathbf{S}_0, \mathbf{H} = 0)$:

$$E_0 = -\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} s_i^{(0)} s_j^{(0)}. \quad (4)$$

Пусть $\mathbf{S}^{(k)} = (s_1^{(k)}, s_2^{(k)}, ..., s_N^{(k)})$ - одна из конфигураций класса $\Omega_n$, $k = 1, 2, ..., L$. Ее компоненты можно выразить через компоненты начальной конфигурации $\mathbf{S}_0$: $s_i^{(k)} = s_i^{(0)} a_i^{(k)}$. Здесь $\mathbf{A}_k = (a_1^{(k)}, a_2^{(k)}, ..., a_N^{(k)})$ есть вектор, у которого $n$ компонент равны -1, и $N - n$ компонент равны +1. Соответственно, энергию в точке $\mathbf{S}^{(k)}$ можно представить в виде:

$$E(\mathbf{S}^{(k)}) = -\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} s_i^{(0)} s_j^{(0)} a_i^{(k)} a_j^{(k)} \chi_{ij}, \quad (5)$$

где $\chi_{ij}$ есть единичный симметричный тензор 2-го порядка: $\chi_{ij} = 1$ если $i \neq j$, $\chi_{ij} = 0$ если $i = j$. Множеством векторов $\{\mathbf{A}_k\}$ полностью определяется класс $\Omega_n$, а усреднение по нему сводится к суммированию по всем векторам $\mathbf{A}_k = (a_1^{(k)}, a_2^{(k)}, ..., a_N^{(k)})$, $k = 1, 2, ..., L$. Подставляя выражение (5) в (3) можно вычислить произвольный момент распределения энергий в классе $\Omega_n$.

Нетрудно заметить (см. Приложение), что задача нахождения моментов распределения энергий сводится к усреднению по классу $\Omega_n$ величины $a_{i_1}^{(k)} a_{i_2}^{(k)} ... a_{i_r}^{(k)} \chi_{i_1 i_2 ... i_r}$, где $\chi_{i_1 i_2 ... i_r}$ есть единичный симметричный тензор $r$-го порядка: $\chi_{i_1 i_2 ... i_r} = 1$ если все индексы различны, $\chi_{i_1 i_2 ... i_r} = 0$ если совпадают хотя бы два индекса. Иными словами, задача сводится к отысканию среднего от произведения $r$ компонент вектора $\mathbf{A}_k$:

$$K_r = \frac{1}{L}\sum_{\Omega_n} a_{i_1}^{(k)} a_{i_2}^{(k)} ... a_{i_r}^{(k)} \chi_{i_1 i_2 ... i_r} \equiv \frac{1}{L}\sum_{k=1}^{L} a_{i_1}^{(k)} a_{i_2}^{(k)} ... a_{i_r}^{(k)} \chi_{i_1 i_2 ... i_r} \quad (6)$$

Действительно, подставляя (6) в (3) и меняя порядок суммирования для среднего значения энергии получим:

$$\bar{E}_n = -\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} s_i^{(0)} s_j^{(0)} \cdot \frac{1}{L}\sum_{k=1}^{L} a_i^{(k)} a_j^{(k)} \chi_{ij}. \quad (7)$$

Из общих соображений понятно, что множитель $K_2 = L^{-1}\sum a_i^{(k)} a_j^{(k)} \chi_{ij}$ в (7) не зависит от конкретного значения пары индексов $i$ и $j$, и его можно вынести за знак суммирования. Тогда (7) принимает вид:

$$\bar{E}_n = K_2 E_0.$$

Вычисление величин $K_r$ ($r = 1, ..., 6$) и точных выражений для первых трех моментов распределения энергий вынесем в Приложение. Здесь приведем только окончательные выражения для среднего и дисперсии:

$$\begin{aligned} \bar{E}_n &= E_0 \frac{(N-2n)^2 - N}{N(N-1)}, \\ \sigma_n^2 &= \frac{8n(N-n)}{N(N-1)(N-2)(N-3)} \times \\ &\times \left[ A\left( \mathrm{Sp}\,\mathbf{T}^2 - \frac{E_0^2}{N(N-1)} \right) + NB\sigma_\lambda^2 \right]. \end{aligned} \quad (8)$$

Здесь $\mathrm{Sp}\,\mathbf{T}^2$ есть след матрицы $\mathbf{T}^2$, $\sigma_\lambda^2$ - дисперсия квазиэнергий $\lambda_i = h_i s_i^{(0)}$, где $h_i$ - локальное поле, действующее на *i*-й спин (в начальной конфигурации $\mathbf{S}_0$) и использованы следующие обозначения:

$$A = 4(n-1)(N-n-1), \quad B = 2[(N-2n)^2 - (N-2)],$$
$$\sigma_\lambda^2 = \frac{1}{N}\sum_{i=1}^{N}\lambda_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N}\lambda_i\right)^2, \quad \mathrm{Sp}\,\mathbf{T}^2 = \sum_i\sum_j T_{ij}^2, \qquad (9)$$
$$\lambda_i = h_i s_i^{(0)}, \quad h_i = \sum_{j\neq i} T_{ij} s_j^{(0)}, \qquad E_0 = -\sum\lambda_i.$$

В асимптотическом пределе ($N\to\infty$) выражения (8) принимают очень простой вид:

$$\bar{E}_n = E_0\cdot m^2,$$
$$\sigma_n^2 = 2\mathbf{Sp}\,\mathbf{T}^2\cdot(1-m^2)\left[(1-m^2)(1-\varepsilon_0^2) + 2m^2\bar{\sigma}_\lambda^2\right], \qquad (10)$$

где $m = 1-2n/N$ есть относительная намагниченность и введены безразмерные величины:

$$\varepsilon_0^2 = \frac{E_0^2}{N^2\,\mathrm{Sp}\,\mathbf{T}^2}\ , \ \bar{\sigma}_\lambda^2 = \frac{N\sigma_\lambda^2}{\mathrm{Sp}\,\mathbf{T}^2}. \qquad (11)$$

При наличии магнитного поля выражения для среднего и дисперсии несколько усложняются (см. Приложение). В асимптотическом пределе $N\to\infty$ эти выражения принимают вид:

$$\bar{E}_n(\mathbf{H}) = m^2 E_0 - m\,\mathbf{H}\mathbf{S}_0^+ \qquad (12)$$

$$\sigma_n^2(\mathbf{H}) = \sigma_n^2 + (1-m^2)\times$$
$$\times\left[\|\mathbf{H}\|^2 - \frac{1}{N}(\mathbf{H}\mathbf{S}_0^+)^2 + 4m\left(\mathbf{H}\mathbf{T}\mathbf{S}_0^+ - \frac{E_0}{N}\mathbf{H}\mathbf{S}_0^+\right)\right], \qquad (13)$$

где $\mathbf{S}_0^+$ есть транспонированный вектор $\mathbf{S}_0$, а $E_0$ и $\sigma_n^2$ даются выражениями (4) и (10) соответственно. Отметим специальный случай, когда все компоненты магнитного поля $\mathbf{H} = (H_1, H_2, ..., H_N)$ одинаковы по модулю: $|H_i| = H$. Если начальную конфигурацию $\mathbf{S}_0$ выбрать параллельной магнитному полю, то $\mathbf{H}\mathbf{S}_0^+ = HN$ и выражения (12), (13) существенно упростятся:

$$\bar{E}_n(\mathbf{H}) = E_0\cdot m^2 - mHN,$$

$$\sigma_n^2(\mathbf{H}) = \sigma_n^2.$$

Отметим, что все моменты распределения энергий целиком определяются моментами квазиэнергий $\lambda_i = h_i s_i^{(0)}$ для *начальной конфигурации* $\mathbf{S}_0$. Среднее значение $\bar{E}_n$ определяется величиной $E_0 = -\sum\lambda_i$, т.е. средним значением $\lambda_i$. Дисперсия $\sigma_n^2$ включает в себя зависимость от дисперсии квазиэнергий $\sigma_\lambda^2$. Третий момент распределения энергий включает в себя (см. Приложение) еще и зависимость от среднего значения величин $\lambda_i^3$ и так далее.

Для однородных спиновых систем, в которых все спины имеют одинаковое окружение, вычисление моментов распределения сильно упрощается, если за начальную точку $\mathbf{S}_0$ взять основное состояние системы (глобальный минимум энергии). Для модели Изинга на гиперкубе основное состояние хорошо известно (см. обсуждение этого вопроса в п. 4.1), т.е. известны величины $E_0$ и $\sigma_\lambda^2$. Для систем со случайной матрицей $T_{ij}$ (как в модели Шерингтона-Кирпатрика спинового стекла или в модели Эдвардса-Андерсона) возникает вопрос - как вычислить энергию основного состояния $E_0$ и дисперсию квазиэнергий $\sigma_\lambda^2$? В следующем разделе дается ответ на этот вопрос.

## 3. Распределение квазиэнергий в минимуме

Рассмотрим спиновую систему с матрицей связей $\mathbf{T}$, элементы которой можно рассматривать как независимые случайные величины, подчиняющиеся некоторому распределению с нулевым средним и стандартным отклонением $\sigma_T$. Нас интересует распределение квазиэнергий $\lambda_i = h_i s_i^{(0)}$, где $h_i$ есть локальное поле (9), действующее на *i*-й спин. Отметим, что величиной $\lambda_i$ определяется энергия, приходящаяся на *i*-й спин. Действительно, энергия состояния равна сумме квазиэнергий $E = -\sum\lambda_i$ и, следовательно, среднее значение квазиэнергии равно $\bar{\lambda}_i = -E/N$.

Нетрудно заметить, что если начальная конфигурация $\mathbf{S}_0$ выбрана случайным образом, то величины $T_{ij}s_i^{(0)}s_j^{(0)}$ также являются случайными. В этом случае квазиэнергии $\lambda_i$, как суммы большого числа независимых случайных величин $T_{ij}s_i^{(0)}s_j^{(0)}$, распределены по нормальному закону с дисперсией $\sigma_\lambda^2 = N\sigma_T^2$.

Иное дело, если конфигурация $\mathbf{S}_0$ отвечает минимуму энергии (ниже будем рассматривать только этот случай). Величины $T_{ij}$ и $s_i^{(0)}s_j^{(0)}$ теперь не являются независимыми, поскольку корреляцией между ними обусловлено появление минимума. Однако это обстоятельство можно преодолеть. Следуя [20] выделим из матрицы $\mathbf{T}$ член $\mathbf{T}_0$, ответственный за образование минимума. Для этого представим матрицу $\mathbf{T}$ в виде:

$$\mathbf{T} = \mathbf{T}_0 + \mathbf{T}_1, \qquad (14)$$

где

$$\mathbf{T}_0 = r_0\sigma_T\mathbf{S}_0^+\mathbf{S}_0\ , \quad \mathbf{T}_1 = \mathbf{T} - \mathbf{T}_0.$$

Вес $r_0$ находим из условия некоррелированности элементов матриц $\mathbf{T}_0$ и $\mathbf{T}_1$. Вычисляя ковариацию матричных элементов и полагая ее равной нулю получим

$$r_0 = -\frac{E_0}{N(N-1)\sigma_T}. \qquad (15)$$

При таком выборе веса $r_0$, из (14)-(15) получим соотношение

$$\mathbf{S}_0\mathbf{T}_1\mathbf{S}_0^+ = 0\,,$$

показывающее, что вклад от $\mathbf{T}_1$ в энергию $E_0$ строго равен нулю и $E_0 = -\mathbf{S}_0\mathbf{T}_0\mathbf{S}_0^+$. Таким образом, минимум в конфигурации $\mathbf{S}_0$ обусловлен только влиянием $\mathbf{T}_0$. Кроме того, стандарты элементов матриц $\mathbf{T}_0$ и $\mathbf{T}_1$ при условии (15) имеют вид:

$$\sigma_0 = r_0\sigma_T \quad , \qquad \sigma_1 = \sigma_T\sqrt{1-r_0^2}\,.$$

Отсюда получаем соотношение $\sigma_T^2 = \sigma_0^2 + \sigma_1^2$, подтверждающее, что матрица $\mathbf{T}$ представлена как сумма двух некоррелированных случайных матриц $\mathbf{T}_0$ и $\mathbf{T}_1$.

С учетом (14)-(15) квазиэнергию в конфигурации $\mathbf{S}_0$ можно представить в виде:

$$\lambda_i = \sqrt{N}\sigma_1(\gamma + x_i)\,, \tag{16}$$

где

$$\gamma = \frac{|E_0|}{N^{3/2}\sigma_1}\ , \quad x_i = \frac{1}{\sqrt{N}\sigma_1}\sum_{j\neq i} T_{1,ij}s_i^{(0)}s_j^{(0)}\,. \tag{17}$$

Из проведенного анализа следует, что $T_{1,ij}$ и $s_i^{(0)}s_j^{(0)}$ некоррелированы, а величины $T_{1,ij}s_i^{(0)}s_j^{(0)}$ подчиняются некоторому распределению с нулевым средним и стандартом $\sigma_1$. Это означает, что в пределе $N >> 1$ $x_i$ можно было бы рассматривать как нормально распределенную величину с нулевым средним и стандартом $\sigma_x = 1$. Необходимо учесть, однако, что в минимуме все спины направлены вдоль своих локальных полей, т.е. $\lambda_{i0} \geq 0$, причем $\sum\lambda_{i0} = -E_0$. Это означает, что безразмерные величины $x_i$ удовлетворяют условиям: $x_i \geq -\gamma$ и $\sum x_i = 0$. Наложение этих условий кардинально меняет моменты распределения величин $x_i$.

С учетом сказанного задача о распределении квазиэнергий $\lambda_i$ сводится к следующей. Имеется набор случайных величин $x_i$, каждую из которых в отдельности можно рассматривать как нормально распределенную с нулевым средним и единичной дисперсией. Каково будет их совместное распределение, если случайные величины $x_i$ не считать независимыми, поскольку на них наложены условия $x_i \geq -\gamma$ и $\sum x_i = 0$?

Совместное распределение величин $x_i$ запишется в виде:

$$P_N(x_1, x_2, ..., x_N) = \frac{1}{P_0}\prod_{i=1}^{N} P(x_i)\,\delta(x_1 + x_2 + ... + x_N)\,, \tag{18}$$

где $P_0$ - нормировочная константа и введены обозначения:

$$P(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2} \quad \text{при} \quad x \geq -\gamma\,, \quad P(x) = 0 \quad \text{при} \quad x < -\gamma\,.$$

Определим сначала значение $P_0$. Интегрируя (18) по всем $x_i \geq -\gamma$ и используя интегральное представление дельта-функции получим:

$$P_0 = \frac{1}{2\pi}\int_{-\infty}^{\infty} f_\omega^N d\omega\,,\ f_\omega = \frac{1}{\sqrt{2\pi}}\int_{-\gamma}^{\infty} e^{-\frac{1}{2}x^2 - i\omega x}dx\,. \tag{19}$$

Интеграл (19) оценим методом седловой точки. Из условия $df_\omega / d\omega = 0$ получим выражение для седловой точки $\omega = -i\delta$, где $\delta$ является решением уравнения

$$\delta = \frac{1}{\sqrt{2\pi}\,\Phi_0}e^{-\frac{1}{2}(\gamma-\delta)^2}\,,\ \Phi_0 = \frac{1}{\sqrt{2\pi}}\int_{-\gamma+\delta}^{\infty} e^{-\frac{1}{2}t^2}dt\,, \tag{20}$$

а величина $P_0$ принимает вид:

$$P_0 = \frac{1}{\sqrt{2\pi N(1-\gamma\delta)}}\Phi_0^N e^{\frac{1}{2}N\delta^2}\,.$$

Рассмотрим теперь маргинальное распределение величины $x_i$. Проводя в (18) интегрирование по всем остальным координатам $x_j$, $j \neq i$ и опуская индекс получим:

$$\overline{P}(x) = \frac{1}{2\pi P_0}P(x)\int_{-\infty}^{\infty} f_\omega^{N-1}e^{-i\omega x}d\omega\,.$$

Оценка этого интеграла методом седловой точки дает выражение:

$$\overline{P}(x) = \begin{cases} \dfrac{1}{\sqrt{2\pi}\,\Phi_0}e^{-\frac{1}{2}(x-\delta)^2}, & \text{когда } x \geq -\gamma, \\ 0, & \text{когда } x < -\gamma. \end{cases} \tag{21}$$

Полученное распределение удовлетворяет условиям нормировки, а среднее и дисперсия маргинального распределения (21) определяются выражениями

$$\overline{x} = 0\,, \quad \sigma_x^2 = 1 - \gamma\delta\,.$$

Подставляя в (21) вытекающее из (16) соотношение $x_i = \lambda_{i0}/\sqrt{N}\sigma_1 - \gamma$ получим искомое выражение для маргинального распределения квазиэнергий. Для среднего и дисперсии квазиэнергий в минимуме получается:

$$\overline{\lambda_i} = \frac{1}{N}|E_0|\,, \quad \sigma_\lambda^2 = N\sigma_1^2(1-\gamma\delta)\,. \tag{22}$$

Зависимость величины $1-\gamma\delta$ от приведенной глубины минимума $\gamma \approx |E_0|/N^{3/2}\sigma_T$ можно получить численным решением уравнения (20). Эта зависимость показана на рисунке 1. Как видим, с ростом $\gamma$ величина $1-\gamma\delta$ возрастает. На значимом участке значений $\gamma \sim 1.5$ получаем: $1-\gamma\delta \sim 0.55$. Компьютерные эксперименты на случайных матрицах показывают, что среднее значение энергии минимума можно оценить выражением

$$|E_0| \approx 1.5\cdot N^{3/2}\sigma_T\,, \tag{23}$$

а для энергии основного состояния справедлива оценка: $|E_0| \approx 1.6\cdot N^{3/2}\sigma_T$. Иными словами, в большинстве случаев для оценки параметра $\gamma$ можно

использовать значение $\gamma \sim 1.55$. Соответственно, для дисперсии квазиэнергий в минимуме можно использовать подтвержденное экспериментами оценочное выражение:

$$\sigma_\lambda^2 \approx 0.55 \cdot N\sigma_T^2 .$$

Оценки для $\sigma_\lambda^2$ проверялись в многочисленных экспериментах на случайных матрицах различного типа: матрицы с равномерным или гауссовым распределением матричных элементов, хеббовские матрицы, матрицы модели Эдвардса-Андерсона. В эксперименте генерировалось 200 матриц определенного типа, для каждой матрицы жадным алгоритмом Монте-Карло определялись $10^6$ минимумов, в каждом из которых определялась величина $\sigma_\lambda^2$ и усреднялась по всем минимумам. Размерность $N \times N$-матриц варьировалась в экспериментах от $N = 10^2$ до $N = 2 \cdot 10^4$. Результаты эксперимента хорошо согласуются с выражением (22). На рис.2 показаны результаты эксперимента с матрицами модели Шеррингтона-Киркпатрика размерностей $N = 500, 1000, 5000, 20000$. Сплошная кривая построена по формуле (22), причем приведена нормированная величина $\bar{\sigma}_\lambda^2 = \sigma_\lambda^2 / N\sigma_T^2$. Экспериментальные результаты даны в той же нормировке.

Резюмируем результаты настоящего раздела: *i*) если в качестве начальной конфигурации $\mathbf{S}_0$ выбран минимум энергии, то в выражения (8) и (10) для дисперсии $\sigma_n^2$ следует подставлять оценочные значения $\sigma_\lambda^2 \sim 0.6 \cdot N\sigma_T^2$ и $\bar{\sigma}_\lambda^2 \sim 0.6$ соответственно; *ii*) при отходе от минимума величина $\sigma_\lambda^2$ быстро возрастает: если в качестве начальной конфигурации $\mathbf{S}_0$ выбрана случайная точка вдали от минимума энергии, то для оценки дисперсии $\sigma_n^2$ в выражения (8) и (10) следует подставлять значения $\sigma_\lambda^2 = N\sigma_T^2$ и $\bar{\sigma}_\lambda^2 = 1$ соответственно.

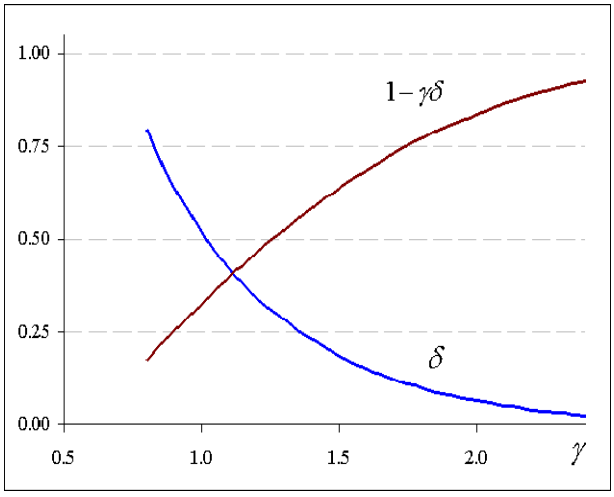



**Рис.1.** Зависимость величин $\delta$ и $1 - \gamma\delta$ от нормированной глубины минимума $\gamma$ (17)
.

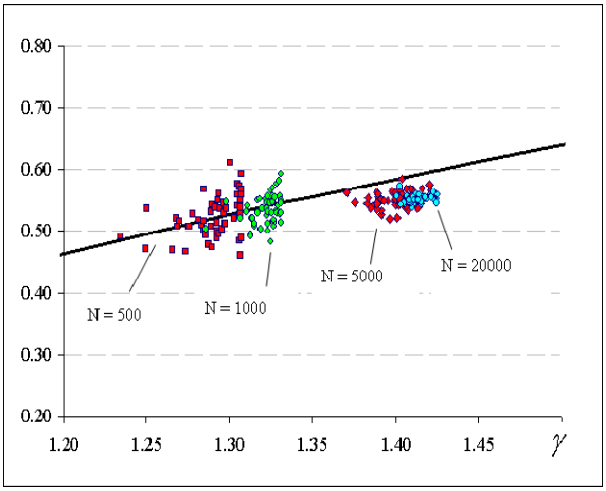



**Рис.2.** Зависимость дисперсии $\bar{\sigma}_\lambda^2$ от глубины минимума $\gamma$. Сплошная кривая – теория (23), маркеры – эксперимент.

# 4. Компьютерное моделирование

Прежде чем переходить к описанию компьютерных экспериментов, условимся по двум пунктам. Во-первых, для простоты будем говорить «распределение энергий в $n$-окрестности» вместо более длинного «распределение энергий состояний, принадлежащих $n$-окрестности»; во-вторых, до конца следующего раздела будем использовать символ $E_n$ вместо использовавшегося ранее $\bar{E}_n$ для обозначения среднего значения энергии в $n$-окрестности $\Omega_n$.

**4.1. Условия проведения экспериментов.**

При заданных $\mathbf{T}$ и $\mathbf{S}_0$ энергию $E(\mathbf{S}^{(k)})$ можно понимать как случайную величину, поскольку в сумме (5) множитель $a_i^{(k)} a_j^{(k)}$ случайным образом принимает значения $\pm 1$. Многое в распределении энергий $E(\mathbf{S}^{(k)})$ зависит от типа матрицы $\mathbf{T}$, от начальной конфигурации $\mathbf{S}_0$ и от номера $n$ окрестности $\Omega_n$.

В настоящем сообщении мы ограничимся известными физическими матрицами. Модель, на которой традиционно проверяют новые методы вычисления статистической суммы – модель Изинга на гиперкубической решетке. Размерность решетки обозначают через $D$, и различают одномерную ($D = 1$), планарную ($D = 2$) и кубическую модели Изинга ($D = 3$). Обычно учитывают только взаимодействие между ближайшими соседями, так что каждый спин связан с $2D$ другими спинами, и связи

между спинами одинаковы. В $N$-мерной строке матрицы $\mathbf{T}$ всего лишь $2D$ элементов отличны от нуля и равны друг другу. Когда $N \to \infty$, такая матрица характеризуется колоссальной разреженностью. Еще один тип исследованных у нас матриц отвечает $D$-мерной модели Эдвардса-Андерсона спинового стекла. Эта матрица подобна одноименной матрице Изинга, только связи между спинами не одинаковы, а являются случайными числами, полученными из стандартного нормального распределения. При $N \to \infty$ эта матрица тоже характеризуются огромной разреженностью. Напротив, в модели Шеррингтона-Киркпатрика спинового стекла каждый спин связан со всеми остальными, а величины связей - случайные числа, полученные из стандартного нормального распределения. Эти матрицы являются массивными (не разреженными). Размерность матриц в экспериментах варьировалась в пределах $N \sim 10^2 - 10^4$.

В качестве начальной конфигурации $\mathbf{S}_0$ в наших экспериментах выбиралась либо случайная конфигурация, либо - основное состояние (если основное состояние найти невозможно, использовался глубокий минимум по энергии). Выделенная роль основного состояния продиктована удобствами, возникающими для модели Изинга. Основным состоянием в этом случае является конфигурация $\mathbf{e} = (1,1,...,1)$. Если положить $\mathbf{S}_0 = \mathbf{e}$, то оказывается, что дисперсия $\sigma_\lambda^2$ (9) равна нулю: $\sigma_\lambda^2 = 0$. Это существенно упрощает выражения (8), (10) для дисперсии $\sigma_n^2$. Кроме того, при $\mathbf{S}_0 = \mathbf{e}$ минимальное значение энергии для любой окрестности $\Omega_n$ асимптотически стремится к $E_0$: $\min\limits_{\mathbf{S}\in\Omega_n} E(\mathbf{S}) \to E_0 \ \forall n$. Это очень полезное свойство, поскольку нижнюю энергетическую границу $\min\limits_{\mathbf{S}\in\Omega_n} E(\mathbf{S})$ необходимо знать для уравнения состояния (см. [1]-[2]). Сказанное относится к модели Изинга на гиперкубе. Мы предполагаем, что и для матриц других типов выбор основного состояния в качестве $\mathbf{S}_0$ позволяет отождествить нижнюю границу $\min\limits_{\mathbf{S}\in\Omega_n} E(\mathbf{S})$ с $E_0$.

Для характеристики расстояния между $\mathbf{S}_0$ и $\Omega$-окрестностью, удобнее вместо $n$ использовать относительную характеристику $x = n/N$. Все эксперименты проводились без магнитного поля ($\mathbf{H} = 0$), поэтому значение $x$ варьируется от 0 до ½.

Поясним, что показано на графиках. Зафиксировав $(N \times N)$-матрицу $\mathbf{T}$, начальную конфигурацию $\mathbf{S}_0$ и значение параметра $x$, мы генерировали большое число $K \sim 10^6 - 10^7$ случайных конфигураций из $n$-окрестности ($n = xN$), вычисляли их энергии $\{E_i\}_{i=1}^M$ и частоты появления $\{K_i\}_{i=1}^M$: $\sum_{i=1}^M K_i = K$. В модели Изинга энергии образуют дискретный набор значений, с фиксированным расстоянием между ближайшими энергиями. Поэтому в результате эксперимента получается не гистограмма, а дискретное распределение. Через $E_i$ мы обозначаем именно различные значения энергий. Их число $M$ значительно меньше общего числа случайных испытаний $K$.

Полученную дискретную кривую удобно сравнивать с графиком стандартной гауссовой плотности, поэтому эмпирические энергии $E_i$ трансформировали в $(E_i - E_n)/\sigma_n$, где $E_n$ и $\sigma_n^2$ даются выражениями (8). Обозначим через $p_i^{(th)}$ значения стандартной гауссовой плотности:

$$p_i^{(th)} = \frac{1}{\sqrt{2\pi}} \exp\left[ -\frac{1}{2}\left( \frac{E_i - E_n}{\sigma_n} \right)^2 \right], i = 1,...,M \ .$$

Сумма всех $p_i^{(th)}$ отлична от 1: $\sum\limits_{i=1}^M p_i^{(th)} = c_0 < 1$. Чтобы сравнивать $p_i^{(th)}$ с экспериментальными вероятностями появления энергий $E_i$, необходимо нормировать последние на константу $c_0$:

$$p_i^{(ex)} = c_0 \frac{K_i}{K}, \ i = 1,...,M \ . \qquad (24)$$

Теперь теоретические и экспериментальные вероятности нормированы одинаково: $\sum_{i=1}^M p_i^{(th)} = \sum_{i=1}^M p_i^{(ex)}$.

На всех графиках в этом разделе по оси абсцисс отложены трансформированные энергии $(E_i - E_n)/\sigma_n$. Целые числа на этой оси - 1, 2, 3, 4… - отвечают исходным энергиям $E_i$, отстоящим от среднего значения энергии $E_n$ на $\sigma_n$, $2\sigma_n$, $3\sigma_n$, $4\sigma_n$ и так далее. Это замечание окажется полезным при обсуждении результатов для модели Изинга на решетках различных размерностей $D$ (см. раздел 5).

Каждый отдельный рисунок состоит из двух частей – верхней и нижней. На графике в верхней части рисунка маркером показаны нормированные экспериментальные вероятности (24). Сплошная кривая в этой части рисунка представляет график стандартной гауссовой плотности. Если маркеры хорошо ложатся на сплошную кривую, можно считать, что экспериментальные вероятности хорошо согласуются с гауссовым распределением. Еще один график, характеризующий степень отличия экспериментальных вероятностей от гауссовой плотности, помещен в нижней части рисунка: для энергий $(E_i - E_n)/\sigma_n$ маркерами показана величина десятичного логарифма $\lg(p_i^{(ex)}/p_i^{(th)})$. Если $p_i^{(ex)} \approx p_i^{(th)}$ этот логарифм приблизительно равен 0. Если же $p_i^{(ex)}$ и $p_i^{(th)}$ заметно отличаются, величина логарифма показывает, на сколько порядков различаются значения двух вероятностей. Мы увидим, что в интервале энергий $|E_i - E_n|/\sigma_n \le 3$ этот график, как правило, близок к нулю, а заметно отличается он от нуля вне этого интервала.

**4.2. Результаты для модели Изинга.**

**4.2.1)** На рис.3 показаны результаты четырех экспериментов для 3D модели Изинга с матрицей размерности $N = 10 \times 10 \times 10 = 1000$. В качестве начальной конфигурации используется основное состояние ($\mathbf{S}_0 = \mathbf{e}$), значение параметра $x$ меняется от 0.5 до 0.005. Число случайных испытаний $K = 10^7$. Четыре панели рисунка показывают, как по мере убывания $x$ увеличивается отличие реального распределения энергий от гауссианы. Панель a) демонстрирует, что при $x = 0.5$ экспериментальные вероятности прекрасно описываются гауссовой плотностью. Видно (см. нижнюю часть рисунка), что в интервале $[-4, 4]$ эксперимент и гауссова плотность совпадают практически идеально, а расхождение между ними (не очень большое) происходит вне этого интервала. Панель b) показывает, что и для меньшего значения $x = 0.25$ наблюдается почти такое же хорошее согласие экспериментальных точек с гауссовой плотностью. Хотя нижняя часть рисунка свидетельствует о некотором перекосе распределения. Панель c) отвечает маленькому значению параметра $x$: $x = 0.05$. Здесь очевидно систематическое расхождение между экспериментальными вероятностями и гауссовой плотностью. У распределения имеется длинный левый хвост, на котором экспериментальные вероятности на 2-3 порядка превосходят теоретические значения. Отметим, что число $M$ различных энергий здесь много меньше, чем на предыдущих двух панелях. Наконец, панель d) демонстрирует графики для еще меньшего значения $x = 0.005$. Здесь имеется только 5 различных значений энергии. Для энергий в районе 0 расхождение между теорией и экспериментом не очень большое, но на левом хвосте экспериментальные вероятности на много порядков превосходят теоретические значения (см. нижнюю часть рисунка).

Вывод: распределение энергий не является гауссовым в окрестностях $\Omega_n$, отвечающих малым значениям $x = n / N$. Для $N = 1000$ это происходит при $x$, меньших критического значения $x_c \approx 0.15$: $x \le x_c$.

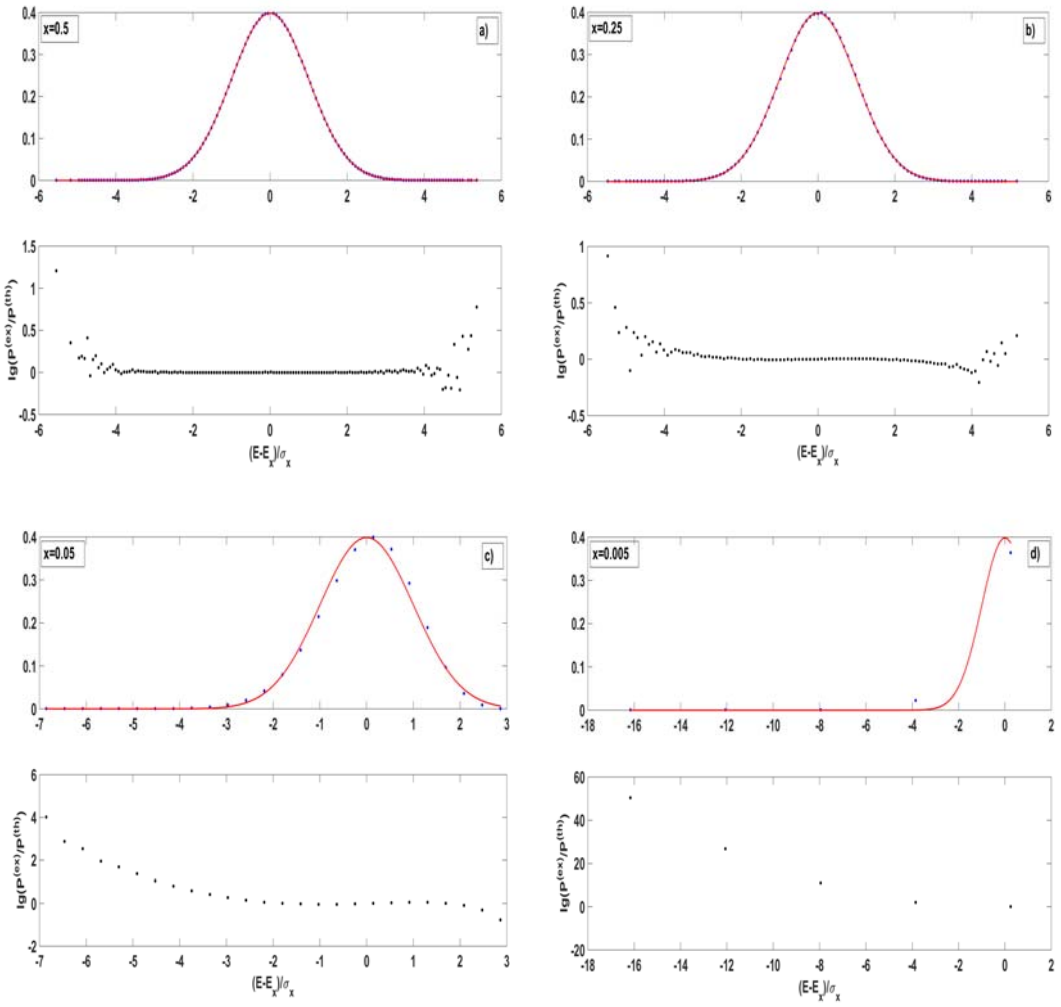



**Рис. 3.** 3D-Изинг, $N = 10 \times 10 \times 10 = 1000$, $\mathbf{S}_0 = \mathbf{e}$ - основное состояние, $K = 10^7$.

**4.2.2).** С увеличением размерности $N$, критическая область, в которой гауссова плотность плохо описывает экспериментальные данные, стремится к 0. Для модели Изинга несложно проделать соответствующие оценки. Действительно, аппроксимация распределения гауссовой плотностью возможна, когда пик распределения достаточно удален как от верхней границы распределения, так и от нижней - только в этом случае гауссова кривая не будет грубо обрезана ни слева, ни справа. Для изинговых матриц минимальное и максимальное значения энергий в $n$-окрестности имеют вид:

$E_{\min} \approx E_0 + 4q/\sqrt{n}$, $E_{\max} = E_0 + 4qn$, $E_n = E_0 \cdot (1-2x)^2$

где $E_0 = -Nq$, а $q = 2D$ - число ближайших соседей. Условие удаленности пика распределения от границ имеет вид:

$$E_n - E_{\min} >> \sigma_n, \quad E_{\max} - E_n >> \sigma_n. \qquad (25)$$

Из второго неравенства в (25) (а на рис. 3-5 видно, что при малых $x = n/N$ гауссиана зарезается именно справа) получим искомое условие:

$$x >> x_c = \frac{1}{\sqrt{qN}}.$$

Таким образом, при $x < x_c$ аппроксимация реального распределения гауссовой плотностью не корректна. Однако ширина $x_c$ этой области в асимптотическом пределе стягивается в нуль.

На рис.4 это наглядно проиллюстрировано графиками с результатами экспериментов для 1D модели Изинга с $K = 10^6$. Как и раньше $\mathbf{S}_0 = \mathbf{e}$, только теперь фиксированным является значение параметра $x$: $x = 0.05$, а размерность задачи варьируется от $N = 500$ до $N = 20000$. Мы не будем подробно анализировать графики, показанные на четырех панелях рисунка. Обратим внимание на то, что по мере увеличения размерности $N$ экспериментальные вероятности все лучше ложатся на гауссову кривую, и при $N = 20000$ согласие между экспериментальными точками и гауссовой плотностью почти хорошее. Можно быть уверенным, что для еще больших значений размерности ($N \sim 10^5$), экспериментальные точки идеально лягут на гауссову кривую.

Подведем итог: в пределе $N \to \infty$ можно считать, что критическая область, где гауссова аппроксимация распределения энергий не работает, стягивается в 0: $x_c \to 0$.

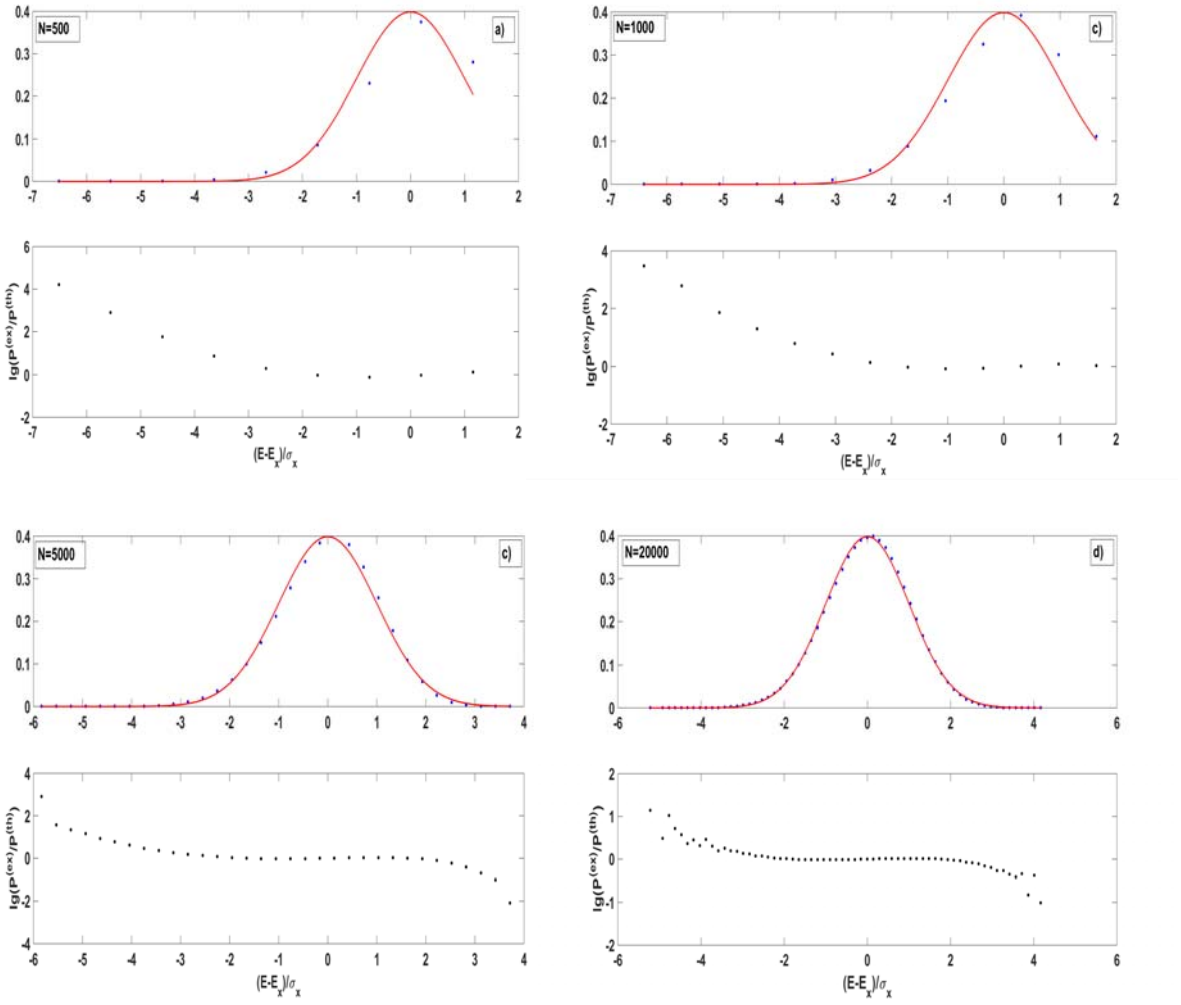



**Рис. 4**. 1D-Изинг, $\mathbf{S}_0 = \mathbf{e}$ - основное состояние, $x = 0.05$, $K = 10^6$.

**4.2.3)** На рис.5 показаны результаты экспериментов для трехмерной модели Изинга с матрицей очень большой размерности $N = 33 \times 33 \times 33 = 35937$. В качестве начальной конфигурации используется основное состояние ($\mathbf{S}_0 = \mathbf{e}$), а параметр $x$ меняется от 0.003 до 0.139. Число испытаний $K$ в разных экспериментах колеблется от $3 \cdot 10^6$ до $4 \cdot 10^7$. Мы видим, что для столь большой размерности экспериментальные вероятности неплохо ложатся на гауссову кривую уже начиная с $x = 0.028$. А для еще больших значений $x$ согласие эксперимента с теорией можно считать идеальным - см. нижние две панели рисунка. Это - еще одна иллюстрация того, как с ростом $N$ уменьшается критическая область, в которой гауссова аппроксимация распределения энергий не работает.

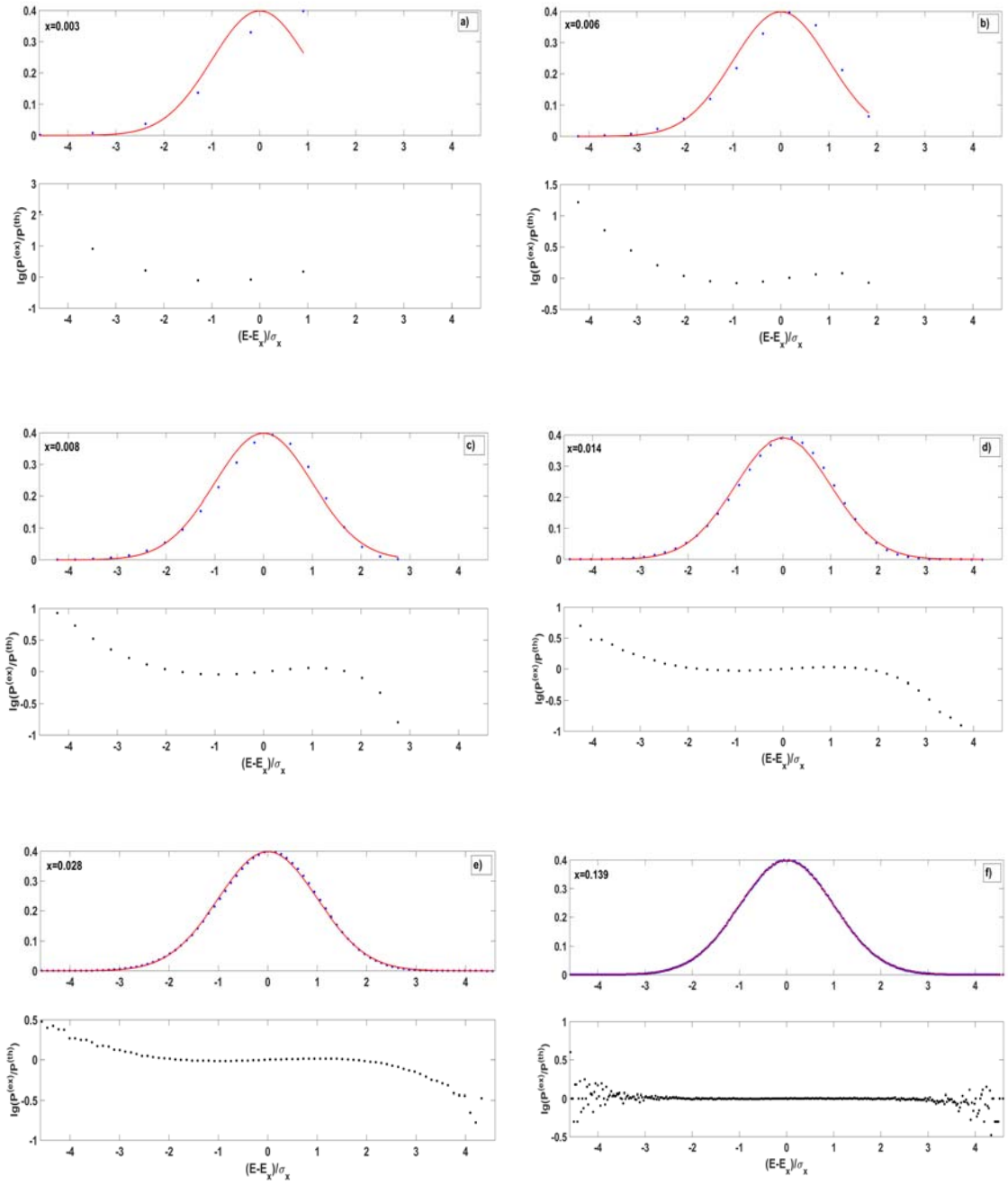



**Рис. 5**. 3D-Изинг, $N = 33 \times 33 \times 33 = 35937$, $\mathbf{S}_0 = \mathbf{e}$ - основное состояние, $K \sim 5 \cdot 10^6$.

**4.2.4)** На рис.6 показаны результаты экспериментов для трехмерной модели Изинга с матрицей большой размерности $N = 35937$, однако теперь в качестве начальной выступает случайная конфигурация: $\mathbf{S}_0 = \mathbf{S}_{\text{rand}}$. Значения $x$ меняются от 0.00003 до 0.0028, число испытаний $K$ в разных экспериментах колеблется от 3-х до 9-ти миллионов. Панель a) отвечает 1-окрестности $\Omega_1$ начальной конфигурации ($x = 0.00003$); панель b) отвечает 3-окрестности $\Omega_3$ ($x = 0.00008$) и так далее; последняя панель f) отвечает окрестности $\Omega_{100}$ ($x = 0.0028$). Графики на панели b) показывают, что уже для 3-окрестности начальной конфигурации экспериментальные вероятности $p_i^{(ex)}$ неплохо ложатся на гауссову кривую в достаточно широком интервале энергий. Начиная же с $x = 0.00028$ согласие между экспериментальными вероятностями и гауссовой плотностью можно считать практически идеальным в интервале энергий $|E_i - E_n| / \sigma_n \le 3$.

Мы видим, что для модели Изинга выбор в качестве $\mathbf{S}_0$ случайной конфигурации существенно увеличивает область, в которой выполняется основное предположение о гауссовом характере распределения энергий в $n$-окрестности. Легко понять, что для матриц, элементы которых являются случайными числами, это будет тем более справедливо. Однако при таком выборе начальной конфигурации трудно оценить нижнюю границу энергий для $n$-окрестности, что затрудняет получение уравнения состояния [1], [2].

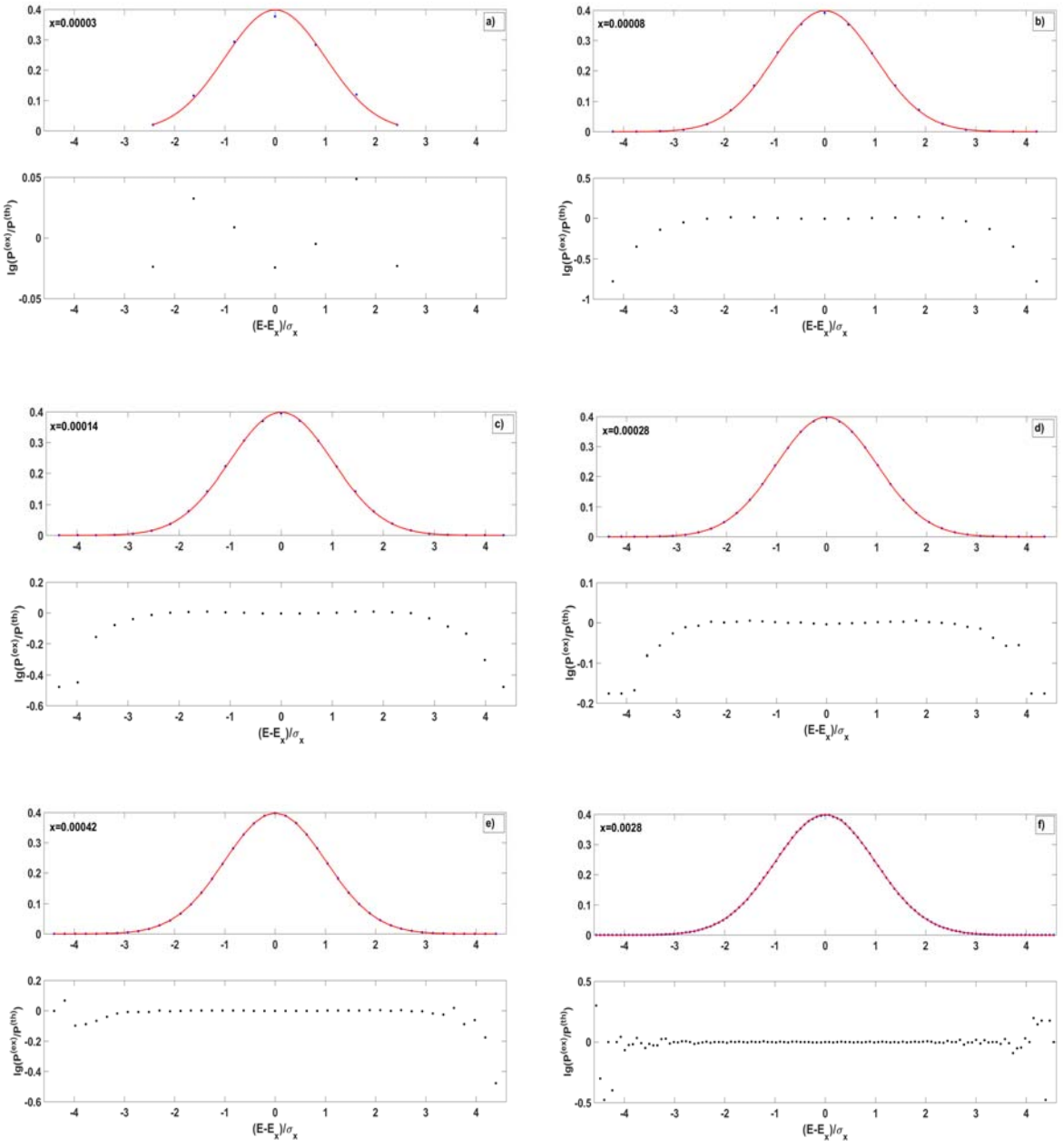



**Рис. 6**. 3D-Изинг, $N = 35937$, $\mathbf{S}_0$ - случайная конфигурация,

### 4.3. Результаты для моделей спинового стекла.

**4.3.1)** На рис.7 показаны результаты экспериментов для трехмерной модели Эдвардса-Андерсона спинового стекла. Матрица имеет размерность $N = 10 \times 10 \times 10 = 1000$, ее отличные от нуля матричные элементы суть случайные числа, полученные из стандартного гауссова распределения. Параметр $x$ принимает значения $x = 0.5, 0.03$ и $0.003$. Левый столбец графиков отвечает использованию в качестве $\mathbf{S}_0$ основного состояния, правый столбец отвечает использованию в качестве $\mathbf{S}_0$ случайной конфигурации. Во всех экспериментах число случайных конфигураций $K = 2 \cdot 10^7$.

Когда $x = 0.5$, независимо от того, какая конфигурация используется в качестве начальной $\mathbf{S}_0$, оба графика демонстрируют хорошее совпадение экспериментального распределения с гауссовой плотностью – см. панели a) и b) рисунка. Затем, когда значение параметра $x$ становится малым ($x = 0.03$) и очень малым ($x = 0.003$), графики ведут себя по-разному в зависимости от начальной конфигурации. Если в качестве $\mathbf{S}_0$ взята случайная конфигурация – см. графики на панелях d) и f) - экспериментальные вероятности по-прежнему хорошо укладываются на гауссову кривую. Если же в качестве $\mathbf{S}_0$ берется основное состояние - графики на панелях c) и e), - уменьшение $x$ сопровождается увеличением различий между экспериментом и гауссовой кривой. Распределение становится несимметричным, с обрубленным левым хвостом. Здесь глобальный минимум $E_0$ выступает в качестве нижней границы для энергий конфигураций, близких к $\mathbf{S}_0$.

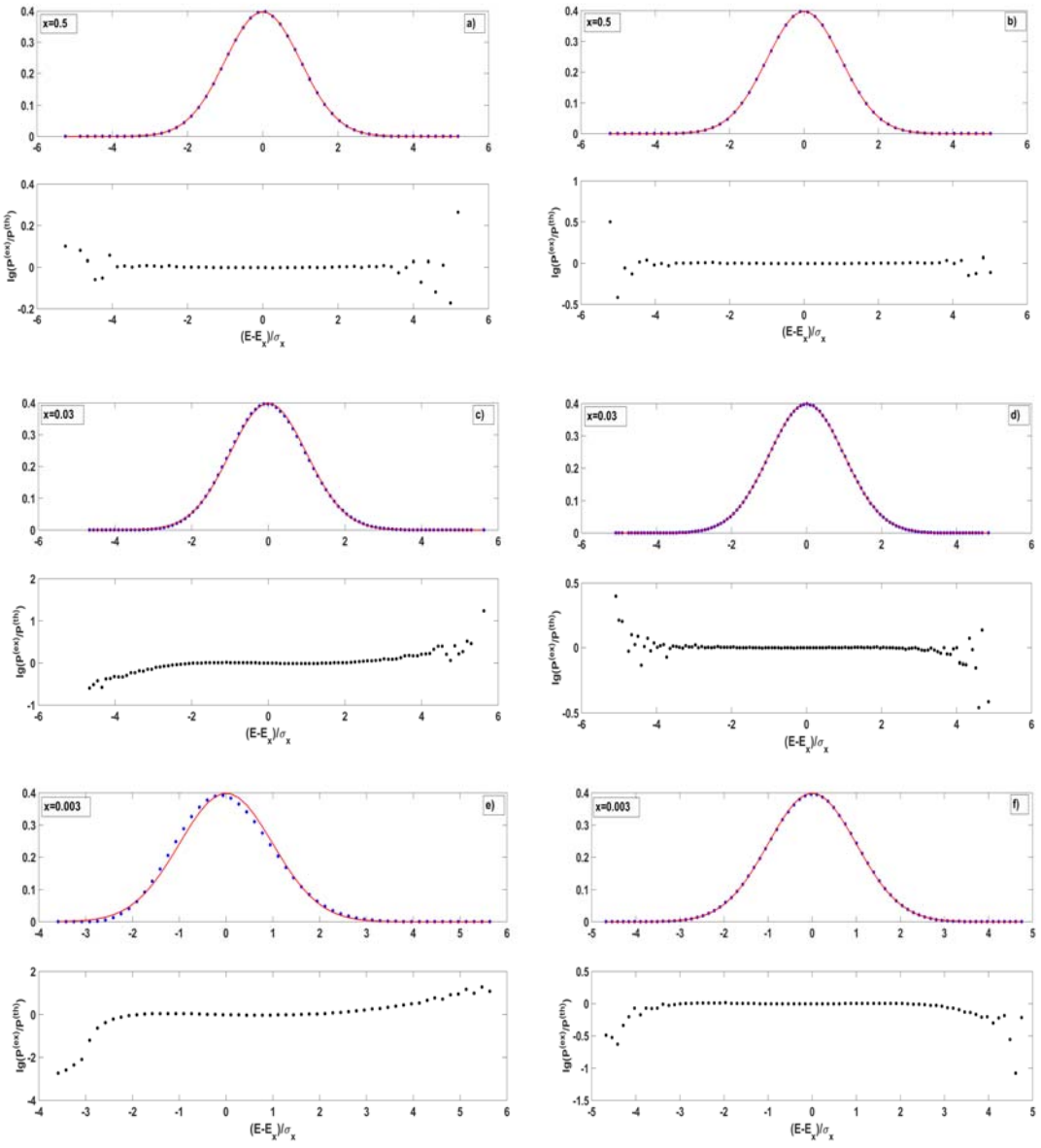



**Рис. 7.** Модель Эдвардса-Андерсона, $D = 3$, $N = 1000$, $K = 2 \cdot 10^7$. В качестве $\mathbf{S}_0$ : слева – основное состояние, справа - случайная

**4.3.2)** На рис.8 аналогично показаны результаты экспериментов для модели Шеррингтона-Киркпатрика спинового стекла. Размерность матрицы $N = 1000$, а $N(N-1)/2$ ее матричных элементов получены случайным образом с помощью стандартного гауссова распределения. Параметр $x$ принимает значения $x = 0.5, 0.01$ и $0.003$. Как и на предыдущем рисунке, графики левого столбца отвечают использованию в качестве $\mathbf{S}_0$ основного состояния, графики правого столбца отвечают случайной конфигурации в качестве $\mathbf{S}_0$. Во всех экспериментах число испытаний $K = 2 \cdot 10^7$.

Результаты экспериментов в целом подобны результатам, полученным для модели Эдвардса-Андерсона (см. рис. 7). Использование в качестве $\mathbf{S}_0$ случайной конфигурации даже при очень малых значениях параметра $x = 0.003$ демонстрирует хорошее согласие экспериментальных данных с гауссовой плотностью (см. панель f). Если же в качестве начальной конфигурации $\mathbf{S}_0$ берется основное состояние, то уже для $x = 0.01$ различия между гауссовой плотностью и экспериментом становятся заметными – см. нижнюю кривую на панели c). Для меньшего значения $x = 0.003$ эти различия еще возрастают – см. панель e) рисунка.

Подчеркнем, что если существенно увеличить размерность матрицы $N$, можно и для левого столбца графиков при малых значениях $x$ добиться хорошего согласия между экспериментом и гауссовой плотностью - по крайней мере, на центральном интервале энергий $|E_i - E_n| / \sigma_n \le 3$.

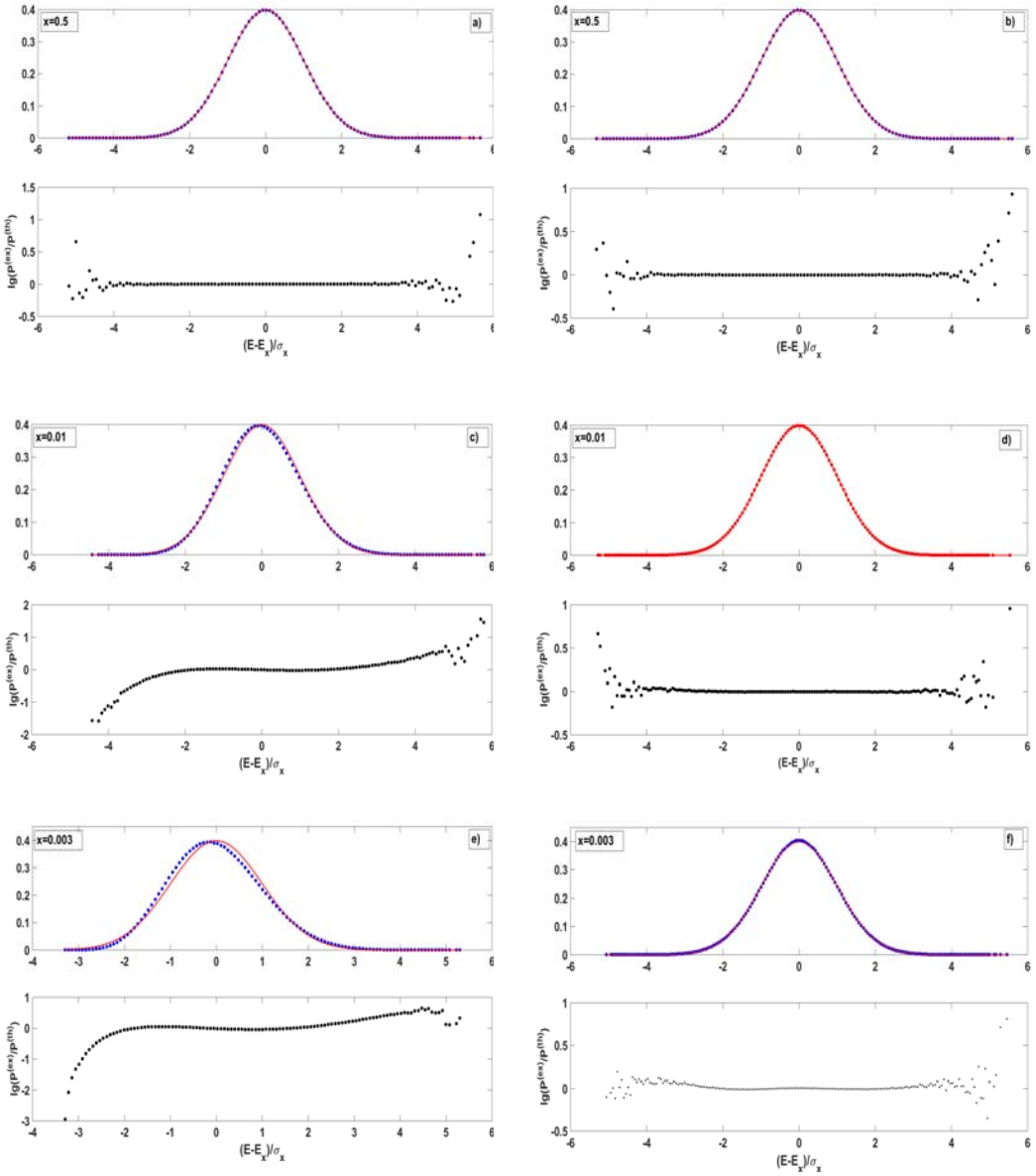



**Рис. 8.** Модель Шеррингтона-Киркпатрика, $N = 1000$, $K = 2 \cdot 10^7$. В качестве $\mathbf{S}_0$ : слева – основное состояние, справа – случайная конфигурация.

## 5. Обсуждение результатов и перспективы

Ключевым для нашего подхода является предположение от том, что распределение энергий в $n$-окрестности $\Omega_n$ можно с хорошей степенью точности аппроксимировать гауссовым распределением. На качественном уровне это можно обосновать следующими соображениями. Нормируем энергию таким образом, чтобы устранить ее зависимость от размеров спиновой системы $N$. (Для моделей Изинга достаточно перейти к энергии, приходящейся на один спин: $\tilde{E} = E / N$, а для систем со случайной матрицей - к $\tilde{E} = E / \sqrt{N \cdot \mathrm{Sp}\,\mathbf{T}^2}$.) Опираясь на (10)-(11) можно показать, что дисперсия величины $\tilde{E}$ с ростом $N$ стремится к нулю как

$$\tilde{\sigma}_n^2 = \frac{1}{N} c(m),$$

где $c(m)$ - не зависящая от $N$ функция намагниченности $m$: в случае моделей Изинга $c(m) = 2q(1-m^2)^2$, а для случайной матрицы $c(m) = 2 \cdot (1-m^2)\left[(1-m^2) + 2m^2 \bar{\sigma}_\lambda^2\right]$.

В указанной нормировке величина $\tilde{E}$ будет изменяться на конечном отрезке $[-q, +q]$ в случае модели Изинга, либо на отрезке $[-1.6, +1.6]$ для моделей со случайной матрицей. Ее дисперсия при этом стремится к нулю как $1/N$. Это означает, что при $N \to \infty$ распределение величины $\tilde{E}$ стремится к $\delta$-функции, которая хорошо аппроксимируется гауссианом. Это подтверждается и выражением (A5) для третьего момента распределения: в указанной нормировке член $R$ в выражении (A5) при $N \to \infty$ становится исчезающе мал, что отвечает гауссову распределению.

Подчеркнем, что аппроксимация распределения энергий гауссианом справедлива только для его центральной части, в некоторой окрестности среднего значения $E_n$. О хвостах распределения в общем случае мы ничего не можем сказать. Для конкретных моделей эксперимент показал следующее.

**i).** Если в качестве $\mathbf{S}_0$ выбрать основное состояние, то распределение энергий асимметрично относительно среднего значения $E_n$, что становится особенно заметно при малых значениях $x = n/N$:

- в случае моделей Изинга распределение имеет толстый левый хвост (в сторону глобального минимума) и быстро спадающий правый хвост (в сторону высоких энергий); перекос в распределении уменьшается с ростом числа ближайших соседей $q$;
- в случае моделей со случайной матрицей картина обратная: распределение имеет быстро спадающий левый хвост и толстый правый хвост.

**ii).** Если в качестве $\mathbf{S}_0$ выбрать случайную конфигурацию, то распределение симметрично относительно $E_n$:

- в случае моделей Изинга распределение имеет тонкие быстро спадающие хвосты (левый и правый), и отличие хвостов от нормального распределения уменьшается с ростом $x = n/N$;
- в случае моделей со случайной матрицей картина обратная: распределение имеет толстые хвосты, а отличие хвостов от нормального распределения возрастает с ростом $x = n/N$.

Под «толстым» или «тонким» хвостом имеется в виду, что экспериментальная кривая проходит выше или ниже теоретической гауссовой кривой, соответственно.

Обсудим, наконец, применение описанного подхода к задаче вычисления статистической суммы. Заменяя в выражении (2) суммирование по $n$-окрестности интегрированием экспоненты $\exp(-\beta E)$ с гауссовой плотностью, с помощью стандартных вычислительных приемов можно получить уравнение состояния, решение и исследование которого дает такие характеристики, как критическая температура, тип фазового перехода, значения критических показателей и т.п. Такая работа была проделана нами в [1], [2] для $D$-мерной модели Изинга, и результаты оказались неоднозначными: для $D \geq 3$ получающиеся значения критической температуры хорошо совпадают с общепринятыми значениями, а для $D < 3$ наши результаты кардинально расходятся с хорошо известными точными результатами - см. таблицу 1.

**Таблица 1.** Критические значения обратной температуры $\beta_c$.

| | $D$=2 | $D$=3 | $D$=4 |
|---|---|---|---|
| Точное значение | 0.4407 | 0.2216 | 0.1489 |
| Наша модель | 0.3912 | 0.2146 | 0.1464 |

(Для $D = 3$ и $D = 4$ в качестве точных значений взяты результаты компьютерного моделирования [21] и [22] соответственно.)

Видно, что для $D \geq 3$ наш подход дает значения, очень близкие к общепринятым. Более того, неограниченно увеличивая $D$, и наращивая число $q = 2D$ неравных нулю матричных элементов в строке, в предельном случае $q = N - 1$ мы фактически переходим к классической модели среднего поля. Эта модель хорошо изучена и описывается известным уравнением Брэгга-Вильямса [23]. То же самое уравнение получается и у нас при $q = N \to \infty$. Иными словами, чем больше число $q$ отличных от нуля матричных элементов, тем более точным оказывается наш метод вычисления статистической суммы.

Напротив, для двумерной модели Изинга наше значение критической температуры отличается от хорошо известного точного значения больше чем на

10% (см. таблицу). Еще хуже, что у нас фазовому переходу второго рода предшествует скачок намагниченности (фазовый переход первого рода) - это расходится с фазовой диаграммой для двумерного Изинга. Здесь наш метод дает структурно неверное решение. Результаты для одномерного Изинга не отражены в таблице 1, однако и в этом случае наши результаты «структурно» неверны: у нас получается фазовый переход второго рода, в то время как в одномерной системе – и это хорошо известно - фазовый переход отсутствует. Нам кажется, мы понимаем причину возникновения этих дефектов.

Начнем с того, что результаты компьютерных экспериментов позволяют уточнить формулировку основного предположения: для любого конечного значения $x$ можно указать такое $N$, что распределение энергий в $n$-окрестности $\Omega_n$, где $n = xN$, в энергетическом интервале $E_n \pm k\sigma_n$ будет хорошо описываться гауссовым распределением. Наблюдавшиеся в эксперименте значения коэффициента $k$ колебались в районе 3: $k \approx 2-4$.

Заметим, что для модели Изинга $\sigma_n \sim \sqrt{q}$, следовательно, чем больше размерность гиперкуба $D = q/2$, тем в более широком энергетическом интервале (при фиксированном значении $x$) распределение энергий можно считать гауссовым. Чем больше $q$, тем шире энергетический интервал, на котором наша аппроксимация работает, тем более точным является наш подход.

На рис.9 для 2D- и 3D-Изинга сравнивается поведение кривых $\lg\left(p_i^{(ex)} / p_i^{(th)}\right)$ для матриц одной и той же размерности $N = 36000$, при одинаковых значениях параметра $x$. На верхней панели даны графики для $x = 0.014$, на нижней панели – для $x = 0.028$. На обеих панелях кривая, отвечающая 3D-Изингу, проходит более полого, чем кривая, отвечающая 2D-Изингу. Различия в поведении кривых становятся заметными начиная с $\left|(E - E_x)/\sigma_x\right| = 2$ и достигают больших значений на хвостах. Это означает, что в случае 3D-Изинга распределение энергий в $n$-окрестности аппроксимируется нормальным распределением намного лучше, чем в случае 2D-Изинга.

Конечно, хотелось бы понять, почему критическим оказывается значение $D = 3$, а для решеток меньших размерностей наш подход перестает работать? В дальнейших исследованиях мы надеемся ответить на данный вопрос. Равно как и разобраться в том, как описать хвосты распределения энергий в $n$-окрестности? Если центральная часть распределения энергий в $\Omega_n$ при $n = xN$ и $x \neq 0$ хорошо описывается гауссовым распределением, то с хвостами распределения все непонятно. Скорее всего, по мере уменьшения $x$ от ½ в сторону нуля, все большую роль будет играть асимметрия распределения – об этом свидетельствует поведение кривых на графиках. Этот вопрос требует дальнейших исследований.

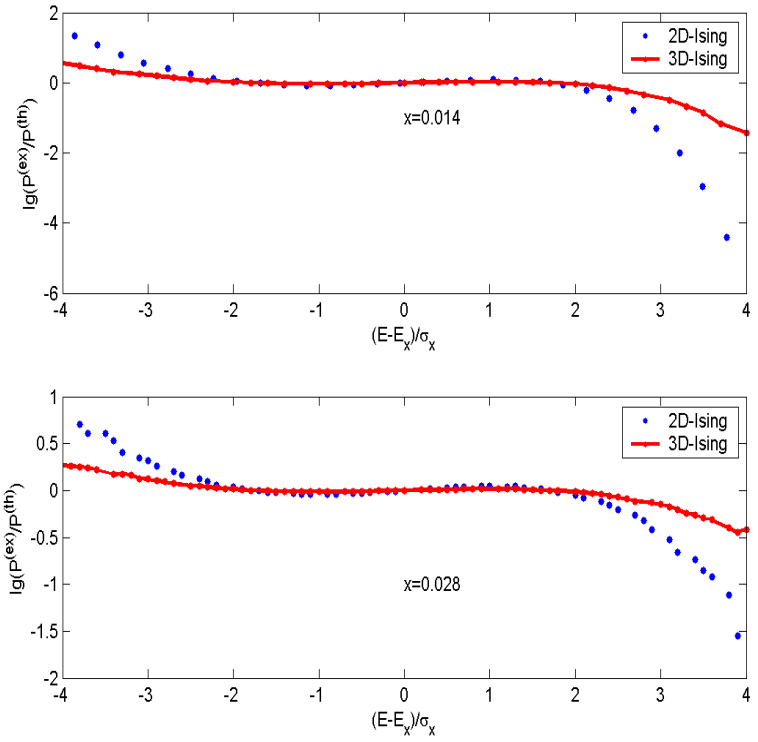



**Рис.9**. Сравнение значений $\lg\left(p_i^{(ex)} / p_i^{(th)}\right)$ для 2D- и 3D-изинговых матриц размерности $N$=36000.

# 6. Приложение

Пусть какие-то $n$ компонент вектора $\mathbf{A}^{(k)}=(a_1^{(k)},a_2^{(k)},...,a_N^{(k)})$ равны -1, а остальные компоненты равны +1. Окрестность $\Omega_n$ начальной конфигурации $\mathbf{S}_0$ определяется множеством всех таких векторов $\{\mathbf{A}^{(k)}\}$, где $k=1,2,...,L$, $L=C_N^n$. Компоненты любого вектора $\mathbf{S}^{(k)}\in\Omega_n$ получаются перемножением компонент начальной конфигурации $\mathbf{S}_0$ и подходящего вектора $\mathbf{A}^{(k)}$: $s_i^{(k)}=s_i^{(0)}a_i^{(k)}$. В разделе 2 отмечалось, что для вычисления моментов распределения энергий необходимо уметь усреднять величины $a_{i_1}^{(k)}a_{i_2}^{(k)}...a_{i_r}^{(k)}\chi_{i_1i_2...i_r}$, где $\chi_{i_1i_2...i_r}=1$ когда значения всех индексов различны, и $\chi_{i_1i_2...i_r}=0$ в противном случае. Иными словами, задача сводится к вычислению величин:

$$K_r=\overline{a_{i_1}^{(k)}a_{i_2}^{(k)}...a_{i_r}^{(k)}\chi_{i_1i_2...i_r}}\equiv\frac{1}{L}\sum_{k=1}^{L}a_{i_1}^{(k)}a_{i_2}^{(k)}...a_{i_r}^{(k)}\chi_{i_1i_2...i_r}\ .$$

Для первых трех моментов распределения энергий нам потребуются величины $K_r$ с $r=1,...,6$. Начнем с самого простого случая $r=1$. Значение $K_1=\overline{a_i^{(k)}}$ рассчитать очень просто: величина $a_i^{(k)}=-1$ в $n$ случаях из $N$, и $a_i^{(k)}=+1$ в остальных $N-n$. Следовательно, ее среднее значение равно: $\overline{a_i^{(k)}}=(N-2n)/N$.

Перейдем к вычислению $K_2=\overline{a_i^{(k)}a_j^{(k)}\chi_{ij}}$. Легко сообразить, что при $i\neq j$ знаки $a_i^{(k)}$ и $a_j^{(k)}$ совпадают в $n(n-1)+(N-n)(N-n-1)$ случаях, и противоположны в остальных $2n(N-n)$ случаях. Число возможных случаев равно $N(N-1)$ и, следовательно, искомое среднее равно:

$$K_2=\overline{a_i^{(k)}a_j^{(k)}\chi_{ij}}=\frac{[n(n-1)+(N-n)(N-n-1)]-2n(n-1)}{N(N-1)}\ .$$

Рассуждая подобным образом получаем выражения для всех необходимых нам величин $K_r$:

$$\begin{aligned}
&K_1=\frac{N-2n}{N},K_2=\frac{(N-2n)^2-N}{N(N-1)},\\
&K_3=\frac{(N-2n)^3-(N-2n)(3N-2)}{N(N-1)(N-2)},\\
&K_4=\frac{(N-2n)^4-2(N-2n)^2(3N-4)+3N(N-2)}{N(N-1)(N-2)(N-3)},\\
&K_6=1-\frac{4(N-n)\cdot n\cdot\kappa}{N(N-1)(N-2)(N-3)(N-4)(N-5)},\\
&\kappa=3(N-2n)^2+6(N-2n)^3(2n-5)+(N-2n)^2\times\\
&(28n^2-120n+125)+(N-2n)(32n^3-180n^2+\\
&+340n-210)+4(n-1)(n-2)(4n^2-18n+23).
\end{aligned}\qquad\text{(A1)}$$

Вычисление моментов распределения энергий начнем со случая, когда магнитное поле отсутствует: $\mathbf{H}=0$. Для первого момента в разделе 2 было получено соотношение $\bar{E}_n=K_2\cdot E_0$, из которого с учетом (A1) вытекает первое из двух выражений (8). Вычислим теперь среднее от квадрата энергии:

$$\overline{E_n^2}=\frac{1}{L}\sum_{k=1}^{L}E^2(\mathbf{S}^{(k)})\ .$$

Выразим вектор $\mathbf{S}^{(k)}$ в виде покомпонентного произведения векторов $\mathbf{A}^{(k)}$ и $\mathbf{S}^{(0)}$, и представим квадрат энергии в виде:

$$E^2(\mathbf{S}^{(k)})=-\sum_{i,j,l,r=1}^{N}T_{ij}T_{lr}s_i^{(0)}s_j^{(0)}s_l^{(0)}s_r^{(0)}\cdot a_i^{(k)}a_j^{(k)}a_l^{(k)}a_r^{(k)}\chi_{ij}\chi_{lr}\quad\text{(A2)}$$

Поскольку мы умеем усреднять произведение компонент вектора $\mathbf{A}^{(k)}$ с различными индексами, добавим в (A2) сомножитель, тождественно равный единице, и распишем получающиеся выражения:

$$\begin{aligned}
\chi_{ij}\chi_{kr}\ &=\ \chi_{ij}\chi_{kr}(\delta_{ik}+\chi_{ik})(\delta_{ir}+\chi_{ir})(\delta_{jk}+\chi_{jk})(\delta_{jr}+\chi_{jr})=\\
&=(\delta_{ik}\delta_{jr}+\delta_{ir}\delta_{jk})\chi_{ij}+(\delta_{ik}\chi_{ijr}+\delta_{ir}\chi_{ijk}+\delta_{jk}\chi_{ijr}+\delta_{jr}\chi_{ijk})+\chi_{ijkr}.
\end{aligned}\qquad\text{(A3)}$$

Первый член в правой части (A3) при подстановке в (A2) дает вклад, не содержащий компонент $s_i^{(0)}$, второй член дает произведение из двух таких компонент, а третий – из четырех. Подставляя (A3) в (A2) и проводя суммирование по всему классу $\Omega_n$, получим:

$$\begin{aligned}
\overline{E_n^2}&=2\mathrm{Sp}\,\mathbf{T^2}+4K_2\sum_{i=1}^{N}\sum_{j=1}^{N}(T^2)_{ij}s_i^{(0)}s_j^{(0)}\chi_{ij}+\\
&+K_4\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{r=1}^{N}T_{ij}T_{kr}s_i^{(0)}s_j^{(0)}s_k^{(0)}s_r^{(0)}\chi_{ijkr},
\end{aligned}\qquad\text{(A4)}$$

где $\mathrm{Sp}\,\mathbf{T^2}$ есть след матрицы $\mathbf{T^2}$: $\mathrm{Sp}\,\mathbf{T^2}=\sum_{i,j=1}^{N}T_{ij}^2$.

Проводя несложные преобразования и компонуя члены (A4) можно получить более компактное выражение для величины $\overline{E_n^2}$. Аналогичным образом можно получить и выражение для третьего момента $\overline{E_n^3}$. Приведем выражения для всех трех моментов:

$$\begin{aligned}
\bar{E}_n&=E_0\frac{(N-2n)^2-N}{N(N-1)}\\
\overline{E_n^2}&=\bar{E}_n^2+\sigma_n^2\\
\overline{E_n^3}&=\bar{E}_n^3+3\bar{E}_n\sigma_n^2-R\ ,
\end{aligned}\qquad\text{(A5)}$$

где дисперсия $\sigma_n^2$ дается выражением:

$$\begin{aligned}
\sigma_n^2&=2(1+K_4-2K_2)\sum_i\sum_j T_{ij}^2+\\
&+4(K_2-K_4)\sum_i\lambda_i^2+(K_4-K_2^2)E_0^2,
\end{aligned}\qquad\text{(A6)}$$

и

$$\lambda_i=\sum_j s_i^{(0)}T_{ij}s_j^{(0)}\ ,\ E_{30}=-\sum_{i,j}(T^3)_{ij}s_i^{(0)}s_j^{(0)}\ ,$$

$$R = 8\,(1-3K_2+3K_4-K_6)\cdot SpT^3 + 2\,(K_2-2K_4+K_6)\times$$
$$\times\left[8\sum_{i,j}T_{ij}^3 s_i^{(0)}s_j^{(0)} - 24\sum_{i,j}(T^2)_{ii}\lambda_i + 12\left|E_{30}\right|\right] -$$
$$-16(K_4-K_6)\sum_i \lambda_i^3 + (K_6-3K_2K_4+2K_2^3)\cdot\left|E_0\right|^3 + \qquad \text{(A7)}$$
$$+6\left|E_0\right|\cdot\left[\begin{array}{l}(K_6+2K_2^2-K_2K_4-2K_4)\cdot SpT^2 + \\ +2(K_4+K_2K_4-K_2^2-K_6)\sum_i \lambda_i^2\end{array}\right]$$

Как следует из выражений (A5)-(A7), первые три момента распределения энергий определяются первыми тремя моментами распределения квазиэнергий $\lambda_i$ (энергия $E_0$ связана со средним значением квазиэнергии соотношением $E_0 = -\sum\lambda_i$). Учет этого факта после ряда упрощений с использованием (A1) позволяет получить из (A6) относительно простое выражение для дисперсии $\sigma_n^2$, приведенное в (8).

Для практических целей наибольший интерес представляет вид полученных выражений в пределе $N\to\infty$. В этом случае выражения (A1) принимают вид:

$$K_1 = m, \qquad K_2 \approx m^2 - \frac{1-m^2}{N},$$
$$K_3 \approx m^3 - \frac{3m(1-m^2)}{N}, \qquad K_4 \approx m^4 - \frac{6m^2(1-m^2)}{N},$$
$$K_5 \approx m^5 - \frac{10m^3(1-m^2)}{N}, \quad K_6 \approx m^6 - \frac{15m^4(1-m^2)}{N},$$

где $m = 1-2n/N$ есть относительная намагниченность и сохранены члены $\sim O(\mathrm{N}^{-1})$. Подставляя эти выражения в (A5)-(A6) получим асимптотические выражения для среднего $\bar{E}_n$ и дисперсии $\sigma_n^2$, приведенные в (10).

Выражение для третьего момента в этом пределе определится асимптотическим значением величины $R$:

$$R = 8\left(1-m^2\right)^3\cdot \mathrm{Sp}\mathbf{T}^3 - 16m^4(1-m^2)\sum_i\lambda_i^3 +$$
$$+2m^2\left(1-m^2\right)^2\cdot\left[8\sum_{i,j}T_{ij}^3 s_i^{(0)}s_j^{(0)} - 24\sum_i(T^2)_{ii}\lambda_i - 12E_{30}\right].$$

В заключение приведем точные выражения для среднего и дисперсии при наличии магнитного поля:

$$\bar{E}_n(\mathbf{H}) = K_2E_0 - K_1\,\mathbf{HS}_0^+,$$
$$\sigma_n^2(\mathbf{H}) = \sigma_n^2 + 4(K_1-K_3)\cdot\mathbf{HTS}_0^+ - 2(K_3-K_1K_2)E_0\cdot\mathbf{HS}_0^+ +$$
$$+(1-K_2)\cdot\left\|\mathbf{H}\right\|^2 + (K_2-K_1^2)\cdot(\mathbf{HS}_0^+)^2,$$

где $\sigma_n^2$ дается выражением (A6). В асимптотическом пределе $N\to\infty$ эти выражения принимают вид (12) и (13).

Авторы благодарны Я.М.Карандашеву, В.М.Крыжановскому и М.Ю.Мальсагову за разработку алгоритмов расчета спектров энергии и помощь в проведении многочисленных экспериментов.

Работа поддержана грантами Российского Фонда Фундаментальных Исследований №15-07-04861 и №13-01-00540.

# Generalized approach to description of energy distribution of spin system

**B.V. Kryzhanovsky, L.B. Litinskii**

**Abstract:** We examined energy spectrums of some particular systems of $N$ binary spins. It is shown that the configuration space can be divided into $N$ classes, and when $N$ tends to infinity the energy distributions in these classes can be approximated by the normal distributions. For each class we obtained the expressions for the first three moments of the energy distribution, including the case of presence of a nonzero inhomogeneous magnetic field. We also derived the expression for the variance of the quasienergy distribution in the local minimum. We present the results of computer simulations for the standard Ising model and the Sherrington-Kirkpatrick and Edwards-Anderson models of spin glass. Basing on these results, we justified the new method of the partition function calculation.

**Keywords**: Partition function, normal distribution of energies.

# Литература


1. B.Kryzhanovsky, L.Litinskii. Generalized Bragg Williams Equation for System with an Arbitrary Long-Range Interaction // Doklady Mathematics. - No. 3( 90), 2014. P. 784–787.
2. Boris Kryzhanovsky and Leonid Litinskii. Approximate method of free energy calculation for spin system with arbitrary connection matrix // Journal of Physics: Conf. Ser. – No.574, 2015. P. 012017. http://arxiv.org/abs/1410.6696.
3. D. Amit, H. Gutfreund, H. Sompolinsky. Spin-glass models of neural networks // Phys. Rev A32. - No.2, 1982. P.1007.
4. O.C. Martin, R. Monasson, R. Zecchina. Statistical mechanics methods and phase transitions in optimization problems // Theoretical Computer Science. - No. 265(1-2), 2001. P. 3-67. http://arxiv.org/abs/cond-mat/0104428.
5. G.E. Hinton, S. Osindero, Y. The. A fast learning algorithm for deep belief nets // Neural Computation. - v. 18, 2006. P.1527-1554.
6. B.V.Kryzhanovsky. Shape of a Local Minimum and Probability of Its Detection in the Binary Optimization Problem // Differential Equations. - v.44, 2008. P. 1188-1190.
7. B.V.Kryzhanovsky, V.M.Kryzhanovsky. The shape of a local minimum and the probability of its detection in random search // Lecture Notes in Electrical Engineering. - v. 24, 2009. P.51-61.
8. Ya.M.Karandashev, B.V.Kryzhanovsky. Efficient Increasing of Global Minimum Basin of Attraction // Optical Memory & Neural Networks. - v. 19, No.2, 2010. P. 110-125.
9. I.M.Karandashev, B.V.Kryzhanovsky. Attraction Area of Minima in Quadratic Binary Optimization // Optical Memory and Neural Networks (Information Optics). - v.23, No.2, 2014. P.84-88.
10. I.M. Karandashev and B. V. Kryzhanovsky. Increasing the attraction area of the global minimum in the binary optimization problem // Journal of Global Minimization. - v. 56, No.3, 2013. P. 1167-1185.
11. Y.M.Karandashev, B.V.Kryzhanovsky. Transformation of Energy Landscape in the Problem of Binary Minimization // Doklady Mathematics. - v.80, No.3, 2009. P.927-931.
12. Iakov Karandashev and Boris Kryzhanovsky. Mix-Matrix Transformation Method for Max-Cut Problem // Lecture Notes in Computer Science. - v. 8681, 2014. P.323.
13. Y. Karandashev, B. Kryzhanovsky. Binary minimization: Increasing the attraction area of the global minimum in the binary optimization problem // Lecture Notes in Computer Science. - v. 6353, 2010. P. 525-530.
14. I.M.Karandashev and B.V.Kryzhanovsky. Transformation of Edge Weights in Graph Bipartition Problem // Lecture Notes in Computer Science. - v. 6792, 2011. P. 25–31.
15. I.Karandashev, B.Kryzhanovsky. The Mix-matrix Method in the Problem of Binary Quadratic Optimization // Lecture Notes in Computer Science. - v. 7552, 2012. P. 9-16.
16. Ya. M. Karandashev, B.V. Kryzhanovsky, L. B. Litinskii. Strong Instability of the Minima Spectrum of a Quadratic Binary Functional // Doklady Mathematics. - v. 83, No. 1, 2011. P. 116–120.
17. B.V. Kryzhanovsky, A.L. Mikaelian and A.B. Fonarev. Vector Neural Net Identifing Many Strongly Distorted and Correlated Patterns // Photonics Asia-2004. Proc. of SPIE, vol. 5642 (SPIE, Bellingham, WA 2005), P. 124-133.
18. Y. Karandashev, B. Kryzhanovsky, L. Litinskii. Local Minima of a Quadratic Binary Functional with a Quasi-Hebbian Connection Matrix // Lecture Notes in Computer Science. - v. 6354, 2010. P. 41-51.
19. Iakov Karandashev, Boris Kryzhanovsky and Leonid Litinskii. Weighted patterns as a tool to improve the Hopfield model // Physical Review E. – v.85, 2012. P. 041925.
20. B.V.Kryzhanovsky. Expansion of a matrix in terms of external products of configuration vectors // Optical Memory & Neural Networks. - v.17, No.1, 2008. P.62-68.
21. W.J.Blote, L.N.Shchur, A.L.Talapov. The cluster processor: new results // Int. J. Mod. Phys. C. - v.10, 1999. P. 1137. http://arxiv.org/abs/cond-mat/9912005.
22. H.Lundow, K.Markstrom. Critical behavior of the Ising model on the 4-dimensional lattice // Physical Review E. – v. 80, 2009. P.031104. http://arxiv.org/abs/1202.3031.
23. Р. Бэкстер. Точно решаемые модели в статистической механике. –М.: Мир, 1985.

# Термические преобразования скелета пород при добыче нефти с закачкой в пласт воздуха

**В.А. Юдин[1], А.В. Королёв[2], И.В. Афанаскин[2], С.Г. Вольпин[2]**

*1 – кандидат физико-математических наук, 2 – кандидат технических наук*

**Аннотация:** В обзоре рассмотрены термические преобразования минералов скелета пород баженовской свиты при добыче нефти с закачкой в пласт воздуха (ТГВ). По вероятности термического преобразования при ТГВ компоненты пород свиты разделены на три группы. Первая – кероген, термические превращения которого при ТГВ весьма вероятны. Вторая группа –гидрослюды, монмориллонит, гипс, пирит, которые при атмосферном давлении значительно меняют свойства даже при невысоких температурах и их изменения при ТГВ возможны. Третья группа – каолинит, хлориты, карбонатные минералы (кроме гипса), кремнезём, кварц, полевые шпаты, слюды, термические преобразования которых при ТГВ представляются маловероятными. Степень и характер термических преобразований должны быть экспериментально исследованы с учётом реальных условий залегания пород баженовской свиты.

**Ключевые слова:** термогазовое воздействие, закачка в пласт воздуха, пиролиз керогена, термические преобразования минералов, баженовская свита.

## Введение

По данным большинства аналитиков [1] производство и потребление энергии в мире в ближайшие 2–3 десятилетия будет расти.

Возрастёт доля возобновляемых источников энергии, но их вклад в общее производство энергии составит только около 30% [2].

Роль невозобновляемых углеводородных источников энергии - нефти, газа, угля - останется определяющей в снабжении человечества энергией; около 60% от общего производства первичной энергии придётся на нефть и газ [1-3].

На территории России, составляющей 12,8% территории Земли, сосредоточено 12-13% *прогнозных ресурсов* и около 12% *разведанных запасов* нефти [1-5]. Однако, *доказанные запасы* нефти в России невелики, и при сохранении нынешних темпов её добычи их хватит менее чем на 25 лет [1]. Поэтому многие специалисты прогнозируют возможное значительное падение нефтедобычи в России в недалёком будущем [4, 5]. Наблюдается и **ухудшение качества** *остаточных доказанных запасов*, поскольку на *трудноизвлекаемые запасы (ТИЗ)* приходится не менее 55-58% *разведанных запасов* России [3-5].

*Благоприятные для добычи запасы* нефти в России характеризуются и высокой степенью выработанности, которая на эксплуатируемых месторождениях превысила 50%, и высокой обводненностью пластов - в среднем 70% [4, 5].

**Для поддержания нефтедобычи на приемлемом уровне в текущем столетии эта ситуация требует принятия срочных мер.**

Значительным стратегическим резервом поддержания нефтедобычи в России являются нетрадиционные коллекторы: кремнисто-глинистые и карбонатно-кремнисто-глинистые высокобитумонасыщенные породы, в первую очередь, баженовской свиты Западной Сибири, доманиковых отложений Волго-Уральской провинции, хадумского горизонта Предкавказья [6-9].

Согласно некоторым исследователям, суммарные *ресурсы* нефти только в баженовской свите оцениваются в размере 0,8–2,1 трлн.т [10-13]. Консервативная оценка *прироста добычи* составляет 10-15 млн.т нефти в год в течение 20 лет, то есть 2.0-3.0 млрд.т накопленной добычи [14]. В литературе приводятся даже экстремальные оценки - 30 млрд.т, 35-50 млрд.т [10-15].

Весьма перспективным способом разработки таких месторождений является метод с закачкой в пласт воздуха и созданием в нефтенасыщенном пласте подвижного очага низкотемпературного окисления – термогазовое воздействие **(ТГВ)** [10–17, 41-43].

Особенностью ТГВ является то, что при инициировании в пласте подвижного очага окисления нефти путём закачки воздуха происходит и оттеснение нефти к добывающим скважинам газами горения, и повышается температура пласта, что делает возможным начало пиролиза керогена, входящего в состав скелета нефтесодержащих пород. Получаемая in-situ дополнительная нефть может быть добыта.

Физические и технологические особенности ТГВ предъявляют повышенные требования к подготовке соответствующих проектных документов разработки, даже на стадии опытно-промышленных работ [18, 19]. В обязательном порядке необходимо многовариантное численное моделирование всего процесса, и для оценки экономической рентабельности проекта до начала его реализации, и для выбора оптимального варианта ТГВ - как по расположению скважин, так и по режиму закачки агентов в пласт [18, 19].

На этапе же эксплуатации объекта методом ТГВ совершенно необходим контроль и регулирование внутрипластовых процессов, в первую очередь, перемещения фронта окисления для снижения риска прорыва воздуха к добывающим скважинам [18, 19].

Главной же составной частью таких процедур является численное гидродинамическое моделирование пластовых процессов, по результатам которого и должны приниматься технологические решения [20-23].

При этом необходимо численно моделировать все процессы, происходящие при обсуждаемом способе добычи, включая изменение свойств пород и пластовых флюидов под воздействием высоких температур.

Последнему вопросу уделяется явно недостаточное внимание. Как правило, ограничиваются учётом изменения вязкости пластовой нефти за счёт повышения температуры. Изменение параметров скелета породы при нагреве пород при моделировании ТГВ обычно не учитывается [24–28, 41-43].

В настоящем обзоре предпринята попытка проанализировать важность и необходимость учёта этого фактора для повышения достоверности численного моделирования ТГВ, а следовательно, и для повышения обоснованности экономических оценок и технологических решений, принимаемых на основе результатов расчёта.

## 1. Краткая характеристика состава пород баженовской свиты

Поскольку основное приложение ТГВ ожидается на отложениях типа баженовской свиты, целесообразно дать представление об особенностях состава и строения этих пород. Детальное изложение геологии, особенностей формирования отложений баженовской свиты выходит далеко за рамки данного обзора. Отметим только, что нефтематеринских пород такого типа в мире встречается достаточно много (битуминозные глины свиты Монтерей, штат Калифорния, США; битуминозные глины Огайо, США; верхний девон, штат Кентукки, США; сланцы Баккен и Вудфорд палеозойского возраста, центральная часть США; сланцы Иглфорд, штат Техас, США; глинистые сланцы мелового возраста – Клаггет и Гаммон, район Скалистых гор, США; сланцы силурийского возраста, Республика Алжир; глинистые сланцы Дувернэй девонского возраста, провинция Альберта, Канада; свита Ла Луна мелового возраста в Венесуэле; отложения ходжаипакской свиты (верхняя юра), Средняя Азия; и другие [29]).

По данным специалистов геологического факультета МГУ [30, 31] минеральный состав отложений баженовской свиты весьма сложен. Они содержат примерно 14% органической составляющей, в которой, из этих 14%, на жидкие углеводороды приходится примерно 3 % и на твердый компонент или кероген – 11%. Минеральная часть (85-86%) включает кремнезем – 40%, глинистый компонент – 23-25%, и карбонатные минералы – 12-13%, полевые шпаты – 2-3%, пирит – 5%.

Более наглядно указанный состав отражён на рис.1 [30-35]. Сходный состав пород баженовской свиты в районе действия «Юганснефтегаз» приведен в работе [36].

Детальное исследование пород баженовской свиты другими исследователями [37, 38] показало, что они характеризуются набором тех же основных породообразующих компонентов, что указаны выше: минералы группы кремнезёма, глинистые, карбонатные минералы, пирит, органическое вещество (ОВ).


**Рис. 1.** Состав пород баженовской свиты по данным [30-35]

Кремнистые минералы присутствуют в породах преимущественно в виде агрегатов кварца-халцедона и представляют собой в различной степени перекристаллизованные остатки скелетов радиолярий.

Обработка результатов рентгенофазового анализа более чем по 200 образцам [37] показала, что концентрация кремнёзема в исследованных образцах пород составляет 60-90%.

Результаты обработки данных рентгенофлуоресцентного и рентгенофазового анализов [37] позволяют утверждать, что содержание *глинистых минералов* в породах баженовской свиты не превышает 30%, чаще всего составляет 5-15%, в среднем 11%. Среди глинистых минералов преобладают гидрослюда и смешаннослойные минералы ряда гидрослюда-монтмориллонит, с числом разбухающих пакетов около 30%. На территории Красноленинского свода в составе глинистых минералов пород присутствует хлорит. В баженовской свите на долю каолинита приходится в среднем 10%, а на Средне-Назымской площади суммарная доля каолинита и хлорита составляет 10%.

Органическое вещество баженовской свиты относится к морскому гумусо-сапропелевому типу [37], сформированному из некромы фито-, зоопланктона, бактерий, с небольшой долей наземной растительности. Массовая концентрация органического углерода (Сорг) в породах достигает 30%, средние значения составляют 5%. Массовая

концентрация битумоида, выделенного из пород при экстракции их органическими растворителями, составляет 0,1-1,7% на породу.

*Карбонатные минералы* [37] представлены кальцитом и доломитом, в единичных случаях встречается сидерит. Встречаются породы, обогащённые биогенным карбонатом; в таких породах массовая доля карбонатных минералов не превышает 35%.

Основная часть карбонатных минералов, изученных в работе [37], относится к вторичным минералам, образованным в результате хемогенного замещения биогенного кремнезёма, которое происходило с различной интенсивностью в литогенезе. В зависимости от интенсивности процессов карбонатизации, доля карбонатного материала в отдельных прослоях может достигать 95%.

*Пирит* также является постоянным компонентом пород [37]. В баженовской свите пирит присутствует в виде маломощных прослоев (до нескольких сантиметров), конкреций и стяжений. Его концентрация, как правило, коррелируется с содержанием в породах ОВ.

Подробное литологическое исследование пород баженовской свиты проведено и в диссертации [38]. Согласно этой работе во многих скважинах в баженовской свите встречаются аргиллиты, слагая слои и пачки от 0,5 до 6 и более метров. Иногда встречается их частое переслаивание с породами глинисто-кремнистого состава.

В зависимости от содержания породообразующих компонентов аргиллиты баженовской свиты подразделены [38] на кремнистые, кремнисто-пиритовые, кремнисто-углеродистые, кремнисто-пиритово-углеродистые, карбонатные.

Комплекс глинистых минералов в отложениях баженовской свиты [38] представлен гидрослюдой, смешаннослойным минералом типа гидрослюда-монтмориллонит при подчиненном количестве монтмориллонитового компонента, каолинитом, хлоритом.

Весьма сложным является не только минеральный состав скелета, но и характер порового пространства пород баженовской свиты [39, 40]. Встречаются порово-трещинный, трещинный, трещинно-кавернозный типы коллекторов.

Фильтрационно-емкостные свойства коллекторов пласта невысоки и резко неоднородны по разрезу и латерали, что в значительной степени обусловлено интенсивностью и характером процессов вторичных преобразований [39].

В целом, по данным различных исследователей [30–40] минеральный скелет пород баженовской свиты состоит из большого числа компонентов: органическое вещество (кероген); кремнезёмные минералы (агрегаты кварца-халцедона, опал, тридимит, кристобалит, скрытокристаллический аутигенный кварц, микрокристаллический аутигенный кварц, зёрна кварца); глинистые минералы (гидрослюда, монтмориллонит, хлорит, каолинит, гидрослюда + монтмориллонит); карбонатные минералы (кальцит, доломит, сидерит, анкерит, смешанные известково-доломитовые разности, гипс); пирит; полевой шпат (натриевый и калиевый), слюда (мусковит); аргиллиты (кремнистый, кремнисто-углеродистый, кремнисто-пиритово-углеродистый, карбонатный)

# 2. Характер теплового воздействия на скелет пород при ТГВ

Рассмотрим, каким температурным воздействиям и какой длительности подвергаются породы при осуществлении ТГВ. При правильном режиме ТГВ температура пласта не должна превышать 400 °С [11, 18, 19, 41–43, 46]. Однако если, хотя бы на время, процесс перешёл в более высокотемпературную стадию – стадию высокотемпературного горения (ВГ), максимальная температура может подняться до 600 °С и даже выше [18, 19, 41–43, 46].

При численном моделировании должна быть предоставлена возможность охватить весь диапазон температур от начальной пластовой (80°С) до 700-800°С. Соответственно, в исходных данных должны содержаться сведения о параметрах пород во всём данном интервале температур.

Длительность теплового воздействия при ТГВ проиллюстрируем на примере моделирования процесса внутрипластового горения (ВГ), выполненного в диссертации [44] – смотри рис. 2 и рис. 3.

Условия расчёта: классическое сухое ВГ, начальная пластовая температура - 30°С, начальное пластовое давление - 10 МПа, эффективная толщина - 20 м, пористость - 25%, проницаемость - 700 мД, начальная вязкость нефти - 1000 сПз, теплоемкость породы пласта и окружающих пород - 2216 кДж/(м$^{3}$·$^{0}$С), теплопроводность породы пласта и окружающих пород - 10 кДж/(м·час·°С).

Как видно из рисунков, **температурное воздействие в несколько сот градусов охватывает в каждый момент времени зону в несколько десятков метров и длится, по меньшей мере, несколько месяцев.** При ТГВ максимальная температура прогрева будет меньше – не выше 400°С [11, 18, 19, 41-43, 46]. Тем не менее, при любом режиме окисления пластовой нефти должно наблюдаться длительное тепловое воздействие на породу при температуре в несколько сотен градусов.

Пространственный, временной и температурный масштаб такого теплового воздействия приводит к необходимости проанализировать возможность необратимых термических преобразований пород и их параметров после прохождения фронта окисления и вытеснения пластовой нефти.

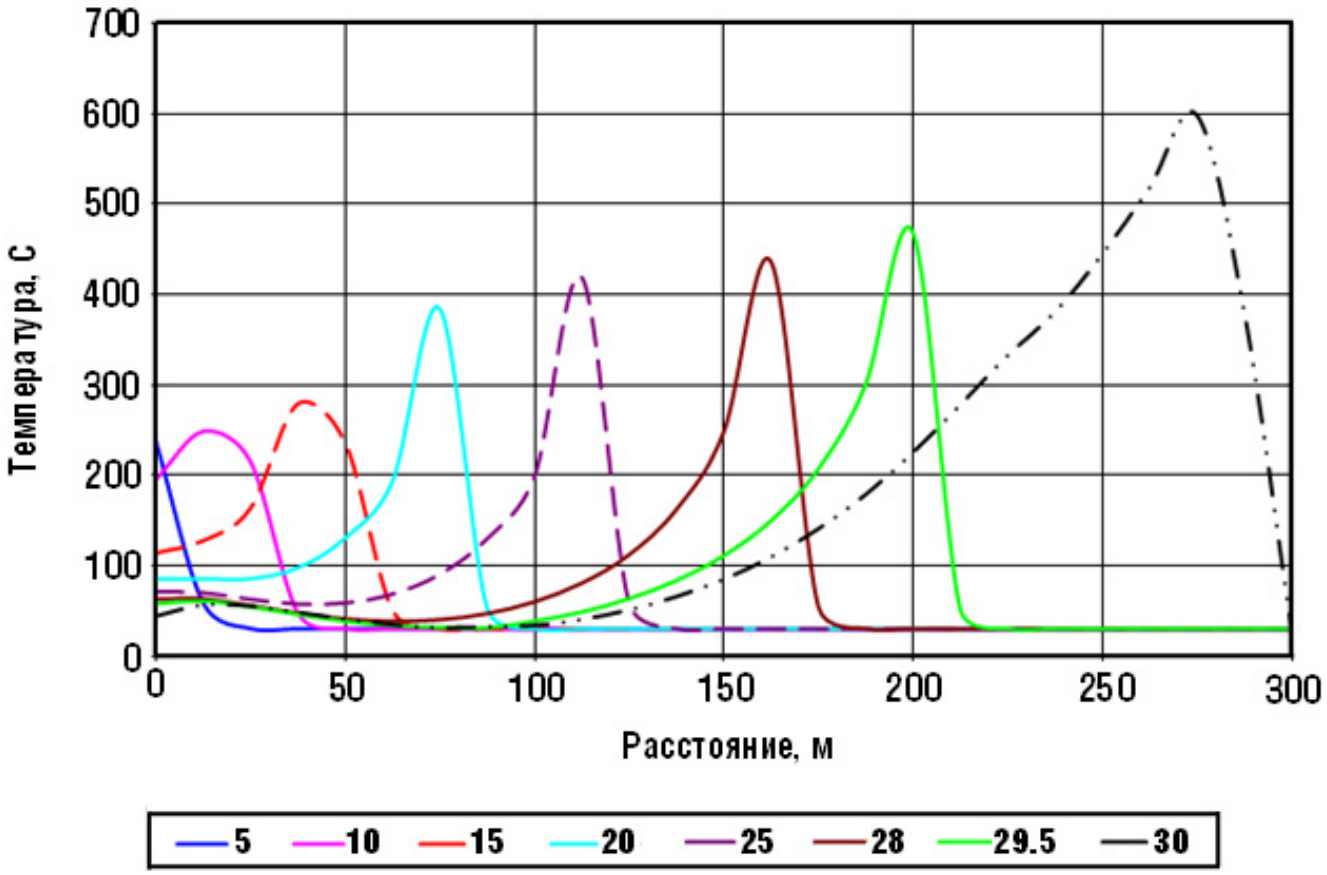



**Рис. 2.** Зависимость температуры пород при ВГ вдоль пласта в различные моменты времени
Шифр кривых – время, годы

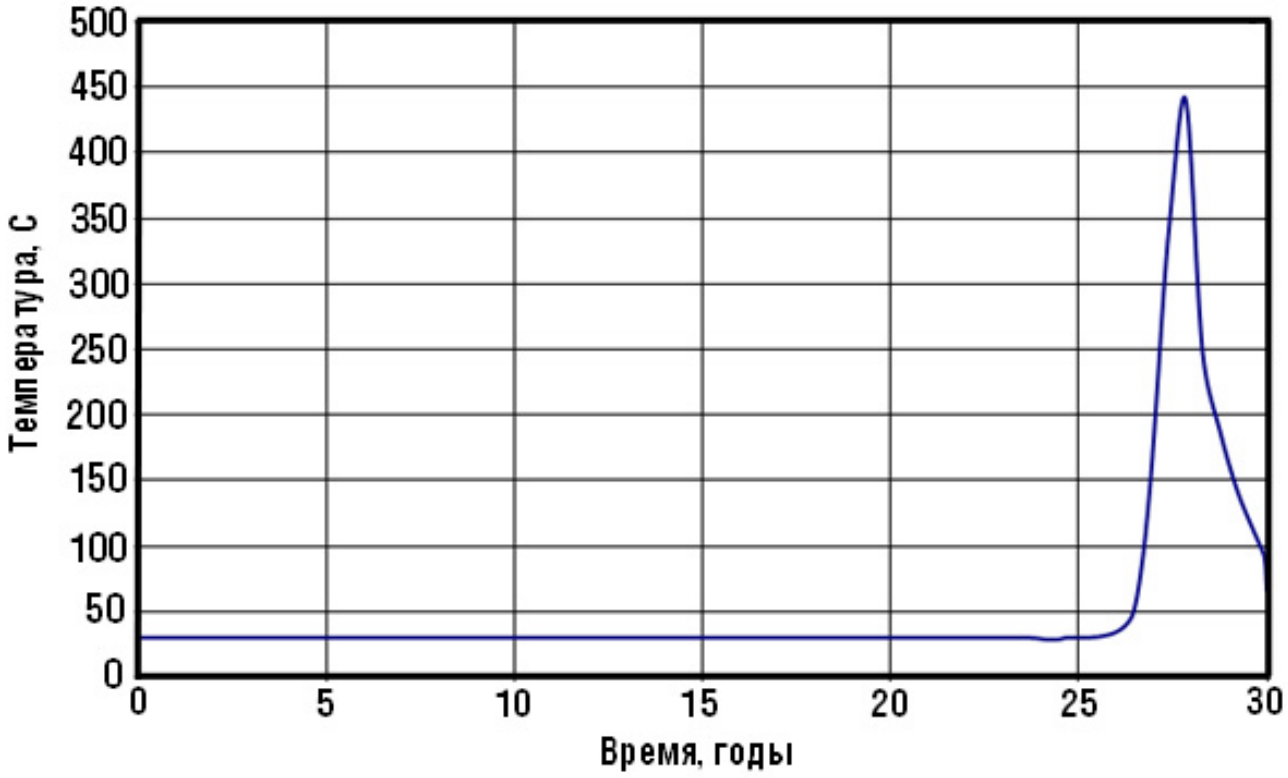



**Рис. 3.** Зависимость температуры пород от времени в средней точке пласта при ВГ

# 3. Термическое преобразование керогена при ТГВ

Напомним, что КЕРОГЕН — это часть рассеянного органического вещества осадочных пород (низких стадий преобразования), нерастворимая в органических растворителях.

Кероген представляет собой ассоциацию разнородных детритных и тонкодисперсных органических остатков, преобразованных большей частью в анаэробных условиях. Структуру керогена представляют в виде макромолекулы, составленной конденсированными карбоциклическими ядрами, соединёнными гетероатомными связями или алифатическими цепочками. Согласно теории появления органических нефтяных материалов, остатки растений и морских организмов под воздействием высоких температур и давления преобразуется, в первую очередь, в кероген, затем в битум и, наконец, в нефть и газ.

Типичный состав разных типов керогена и примеры его расчётных молекулярных формул приведены в табл. 1 [45].

**Таблица 1**

Состав нефте- и газообразующего керогена

| Сведения об объекте и элементный состав керогена | Кероген | | |
|---|---|---|---|
| | нефтеобразующий | | газообразующий |
| Возраст | пермский | Меловой | миоцен |
| Глубина (м) | 1980 | 1600 | 4400 |
| Географическое положение | Техас | Альберта (Канада) | Луизиана |
| C (весовые %) | 80 | 86 | 80 |
| H (весовые %) | 8 | 6,3 | 5 |
| O (весовые %) | 10 | 6,6 | 13 |
| N (весовые %) | 2 | 1,1 | 2 |
| Эмпирическая формула | $C_{67}H_{80}O_6N$ | $C_{72}H_{63}O_4N$ | $C_{67}H_{50}O_4N$ |

Кероген даже при сравнительно невысокой температуре нагрева претерпевает значительные превращения, поскольку, по сравнению со всеми углеводородами нефтяного происхождения, обладает наиболее сложными и неустойчивыми к термическому воздействию молекулами.

Так, согласно работам [16, 17, 41–43, 46], при температурах 320-380°C происходит разрыв связи углерод-углерод в алифатических цепях и узлах кольцевых структур с образованием метильных и метиленовых групп в молекулах с числом атомов углерода до 40, которые в нефтях составляют 40-60%.

Согласно данным монографии [45], в лабораторных условиях атмосферного давления свободные углеводороды освобождаются из образцов керогена при температуре 200-300°C. При ступенчатом нагреве образцов наблюдаются два пика выхода углеводородов из керогенсодержащих глин: первый представляет свободные углеводороды, выделившиеся при термодистилляции в результате нагревания до 250°C; второй – характеризует углеводороды, образованные при пиролитическом расщеплении молекул керогена породы.

Имеется значительное число экспериментов по изучению термических превращения керогена, содержащегося в горючих сланцах. Хотя существуют различия в химическом составе и иных свойствах керогена, сформировавшегося в различных геологических условиях, эти данные дают качественное представление о возможном поведении керогена пород и баженовской свиты.

В справочнике [47] указано, что при нагревании горючего сланца при атмосферном давлении, в лабораторной реторте кероген разлагается на воду, газ, смолу, термобитум и кокс. При этом вода разложения появляется при 270-290°C, появление газа отмечается при 325-350°C, а смола появляется при той же температуре, что и газ, но с некоторым запозданием. Образование термобитума происходит одновременно с образованием газа и смолы.

Последующий нагрев до 450-500°C сопровождается образованием смолы (65-67%), газообразных продуктов (10-15%) и твердого сланцезольного остатка (полукокса) [45, 47].

Практически аналогичные данные приводятся в справочнике [48]: начало термического разложения сланца наблюдается при 170-180°C; при 70-290°C начинается активное выделение пирогенной влаги, при 325-350°C – газа и смолы. Процесс полукоксования заканчивается в основном при 450-500°C, но дальнейшее разложение твердого остатка продолжается и при более высоких температурах.

Согласно [48] первичной стадией термического разложения является деполимеризация макромолекул керогена. При этом он переходит в пластическое состояние, образуя своеобразный продукт – термобитум. Продукты расщепления макромолекулы имеют разные размеры. Мелкие осколки (преимущественно, содержащие гетероатомы)

покидают систему в виде газов и паров. Но в основной массе термобитума продукты первичного разложения керогена состоят из крупных обломков макромолекулы. Эти осколки вступают во взаимодействие, основным результатом которого является образование более крупных, относительно термостабильных молекул.

Таким образом, при термическом превращении керогена (обычно называемом пиролизом) происходит не только разложение самого керогена, но и крекинг новообразованных углеводородов, с образованием из них газов и низкомолекулярных ароматических углеводородов [45, 48].

Подобное разложение керогена – процесс эндотермический. Теплота разложения керогена в интервале температур 200-550°С оценена методом количественной термографии на уровне (-125± 2) ккал/кг [51], а калориметрическим методом – в интервале от (–100) до (–172) ккал/кг.

Общая картина выхода различных продуктов при изотермическом разложении керогена схематично представлена на рис. 4, заимствованном из справочника [47].

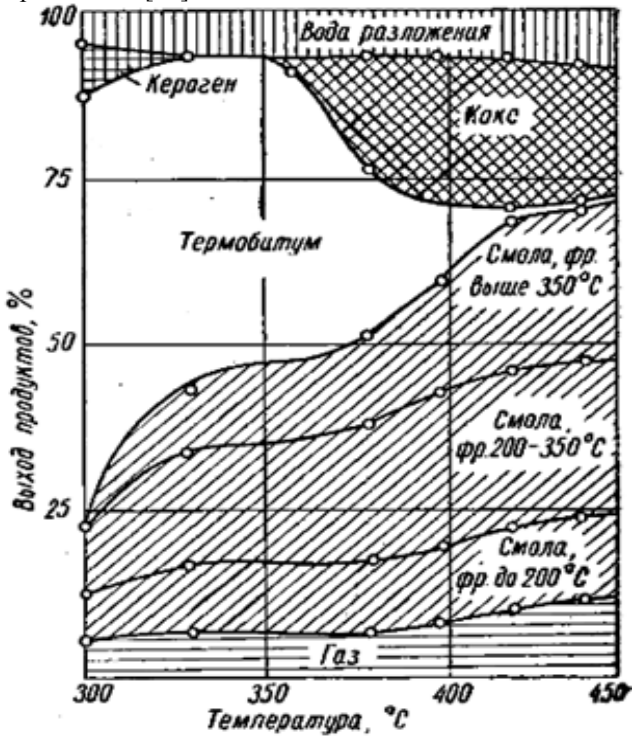



**Рис.4.** Схема выхода продуктов изотермического разложения керогена [47]

**Весьма примечательно, что при температуре около 325°С весь кероген полностью разложился на указанные продукты.**

Такой процесс наблюдался и экспериментально в работе [49], в которой зафиксирован выход различных типов нефти при нагревании пород баженовской свиты:

- ✓ природной, то есть исходно жидкой, нефти из трещинной пустотности пород, объёмом около 2,6%, которая занята природной нефтью;
- ✓ «синтетической нефти», то есть нефти, образованной при пиролизе керогена.

При этом объём нефти второго типа превосходил объём исходно подвижной нефти более чем в два раза.

В результате термического преобразования керогена должна существенно увеличиться пористость пласта. Результаты измерения пористости образца горючего сланца в процессе нагрева показаны в табл. 2 [47].

Об этом свидетельствуют и результаты экспериментов, выполненных на образцах пород баженовской свиты – смотри рис. 5 из работ [50, 51].

Естественно, приведенные данные носят достаточно иллюстративный характер, но они однозначно свидетельствуют о том, что термическое преобразование керогена пород баженовской свиты в какой-то степени будет иметь место при ТГВ. Этот процесс должен вызывать рост пористости и, соответственно, проницаемости пород за фронтом окисления. За счёт пиролиза керогена может быть добыто какое-то количество жидких углеводородов, дополнительно к уже имеющейся в пустотном пространстве жидкой нефти.

Соответственно, в зоне за фронтом окисления должны измениться и теплофизические параметры пород. Это может существенно изменить расчётный режим закачки воздуха для поддержания устойчивого перемещения фронта окисления к добывающей скважине.

Для корректного проектирования и регулирования ТГВ целенаправленные экспериментальные исследования в этом направлении представляются совершенно необходимыми.

**Таблица 2**

Изменение пористости и удельного веса образца горючего сланца при нагреве [47]

| Показатель | До нагрева | Температура нагрева, $^0$С | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 350 | 400 | 450 | 500 | 550 | 600 | 650 |
| Истинный удельный вес, г/см$^3$ | 1,72 | 2,02 | 2,42 | 2,58 | 2,61 | 2,63 | 2,65 | 2,66 |
| Кажущийся удельный вес, г/см$^3$ | 1,49 | 1,35 | 1,23 | 1,00 | 1,14 | 1,04 | 0,88 | 0,71 |
| **Пористость, %** | **13** | **33** | **49** | **61** | **56** | **60** | **67** | **73** |

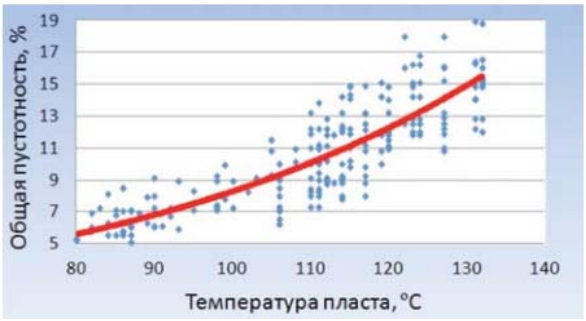



**Рис. 5.** Фильтрационно-емкостные характеристики пород баженовской свиты [50]

# 4. Термическое преобразование минералов скелета при ТГВ

***Глинистые минералы***. Нагревание глин приводит к необратимым изменениям их физико-химических свойств. Происходят изменения структуры, строения и фазового состояния глин [52], которые проявляются в эндо- и экзотермических эффектах на кривых нагревания. На всех кривых отмечается первый эндотермический пик, в интервале сравнительно низких температур 100-300°С, который связан с удалением связанной воды в процессе нагрева при атмосферном давлении. При таком процессе многие глинистые породы дают усадку, что сопровождается изменением объема от нескольких до 25-30%, а в некоторых случаях и больше. Развиваемое при этом давление набухания может достигать 1,0-1,5 МПа.

**Подобное уменьшение объёма глин кардинально увеличивает проницаемость глинистых пород.** Однако этот эффект наблюдался в процессе нагрева глин при атмосферном давлении. Неясно, будет ли он и до какой степени осуществляться в термобарических условиях залегания пород баженовской свиты: температурах 80-130°С, эффективном сжатии пород в 40-50 МПа.

Каолинит при дальнейшем нагревании до 500-600°С теряет кристаллизационную воду [53], поглощая тепло, а при 1000-1200°С разлагается с выделением тепла [54].

При нагреве гидрослюд в атмосферных условиях первый эндотермический эффект [55] обнаруживается при температурах 100-250°С и связан с удалением свободной воды.

Второй эндотермический эффект у гидрослюд при атмосферном давлении появляется при температуре 260-300°С, и обусловлен выделением воды, связанной с обменными ионами.

Третий эндотермический эффект [55], обусловленный выделением цеолитной воды, в атмосферных условиях наблюдается у гидратированных слюдистых минералов в интервале температур 400-700°С, но может продолжаться до 900°С и выше [54].

Другие, структурные, превращения гидрослюд начинаются при температурах выше 850°С[54].

Можно предположить, что вероятность заметной дегидратации гидрослюды при ТГВ зависит от возможности выделения содержащейся в гидрослюде воды и её удаления от места образования. Насколько этот эффект будет значимым в термобарических условиях залегания пород баженовской свиты при ТГВ – можно определить только при специальных экспериментальных исследованиях.

Все монтмориллониты теряют адсорбционную воду в интервале температур 100–200°С, из-за чего в монтмориллоните образуется «сжатая структура», и при нагревании выше этой температуры такое изменение структуры становится необратимым. Завершается подобное превращение при температурах 200-540°С [56]. При дальнейшем нагревании до температуры 700°С монтмориллонит начинает терять конституционную воду и переходит в безводную модификацию. При повышении температуры до 900-950°С начинается кристаллизация шпинели и появление других продуктов превращения монтмориллонита.

***Пирит.*** Термическая диссоциация пиритов различных отложений в вакууме начинается в интервале температур 200-575°С [57], с его диссоциацией на FeS и $S_2$ При нагреве в воздушной среде эти продукты диссоциации пирита начинают окисляться [57], при этом происходит горение серы в газовой фазе, выделяется кислород, двуокись серы и $Fe_2O_3$ [57].

Приведенные выше данные – весьма косвенные. Они не учитывают реальных условий нагрева пирита при ТГВ в горной породе: сжатие до 40-50 МПа; реальный состав пирита в баженовской свите, наличие примесей и дефектов; расположение кристаллов пирита в замкнутом объёме, среди остальных компонентов скелета; и т.д., и т.п.

Скорее всего, в реальной породе скорость и температура начала разложения пирита будут зависеть от возможности ухода газообразных продуктов реакции от места расположения разлагающихся зёрен пирита.

**Тем не менее, можно предположить, что возможность разложения или окисления пирита при ТГВ существует и этот вопрос должен быть изучен экспериментально.**

***Карбонатные минералы***. Большая часть механизмов термического разложения кальцита – высокотемпературные [58]. Максимум скорости разложения чистого кальцита (марки «ч.д.а.») соответствует температуре около 870°С [58]. Лишь при нагреве в вакууме начало разложения кальцита сдвигается примерно на 250°С в сторону более низких температур – до температуры около 400°С, а максимум разложения соответствует 620°С [58], что обусловлено более лёгким удалением продуктов реакции.

При атмосферных условиях установлено наличие ряда эффектов, приводящих к разрушению кальцита и при более низких температурах. К ним относятся: раскалывание минерала при нагревании, обусловленное разными знаками коэффициентов термического расширения минерала по кристаллографическим осям; а также декрепитация – растрескивание, пыление и фонтанирование минерала, причину появления которой связывают с наличием в минерале газово-жидких включений [59].

**При правильном режиме ТГВ термическое разложение известняка и доломита в пластовых условиях представляется маловероятным.** Но при переходе на режим внутрипластового горения (с температурами 600°С и выше) следует ожидать значительных превращений карбонатных компонентов пород баженовской свиты, с заметным изменением структуры и объёма порового пространства, проницаемости и, вероятно, теплофизических свойств.

***Гипс***. При атмосферных условиях гипс при нагревании сравнительно легко дегидратируется и в зависимости от степени нагревания даёт ряд продуктов, значительно отличающихся по своим свойствам [60]. Степень обезвоживания гипса зависит от температуры и длительности нагревания, а также от давления водяных паров. Уже при нагревании до 65°С двуводный гипс начинает медленно переходить в полуводный, а при 107-115°С двуводный гипс уже быстро теряет часть воды и превращается в полуводный гипс. Полуводный гипс полностью обезвоживается в интервале температур 170-210°С, а при дальнейшем нагревании переходит в растворимый ангидрит в интервале температур 220-360°С. При нагревании выше 450°С растворимый ангидрит переходит в нерастворимый при температурах от 450 до 750°С.

Можно предположить, что в породах баженовской свиты, при термобарических условиях их залегания, скорость и температура появления термических преобразований гипса, скорее всего, будет определяться возможностью удаления воды из зоны реакции. Но значимость таких преобразований гипса при ТГВ может быть определена только при специальных исследованиях.

***Слюды***. Чистые слюды содержат сравнительно небольшое количество кристаллизационной воды, в отличие от гидрослюд, содержащих заметное количество воды разных видов; поэтому слюды достаточно термоустойчивы [61]. Разложение светлой калиевой слюды – мусковита – происходит при 650°С. Биотит испытывает дегидратацию при 750-850°С, а роговая обманка, например, устойчива до 900-1000°С [62]. Эндотермические эффекты, связанные с удалением гидроксильных групп из слюдяных минералов, наблюдаются при температурах 760-780°С и 810-840°С.

**Таким образом, минералы слюды, имеющие в своём составе минимальное количество воды, вряд ли будут испытывать какие-либо изменения при ТГВ.**

***Полевые шпаты***. У достаточно чистых разностей полевых шпатов термические превращения начинаются при достаточно высоких температурах [63]. Температура плавления чистого $KAlSi_3O_8$ при атмосферном давлении равна 1150°С. Чистые альбит $NaAlSi_3O_8$ и анортит $CaAl_2Si_3O_8$ при давлении $10^5$ Па плавятся при 1118 и 1550°С соответственно.

**На основании этих литературных данных при реализации ТГВ термические превращения полевых шпатов в составе пород баженовской свиты представляются маловероятными.**

***Кремнезём***. Диоксид кремния существует при атмосферном давлении в семи кристаллических модификациях [64]. Не вдаваясь в детали весьма непростых цепочек превращений различных полиморфных модификаций кварца, подчеркнём, что все они отличаются друг от друга оптическими свойствами, плотностью, коэффициентом линейного расширения, строением кристаллической решетки и другими характеристиками.

Но подавляющее большинство этих превращений – высокотемпературные [64]. Кварц термодинамически устойчив, начиная от низких температур до 870° С, тридимит – от 870 до 1470°С, а кристобалит – от 1470°С до точки плавления, равной 1625° С. Все кварцевые модификации характеризуются и высокой температурой плавления – выше 1000°С.

**На основании этих данных можно полагать, что значительные изменения свойств кремнезёма и кварца при ТГВ весьма маловероятны.**

# Заключение

Как видно из приведенных данных, все компоненты пород баженовской свиты, по вероятности их термического разложения при ТГВ, могут быть разделены на три группы.

**Первая группа** состоит из керогена, термические превращения которого при ТГВ, в какой-то степени, будут иметь место. Они вызывают увеличение пористости пород, связности порового пространства, проницаемости, изменение теплофизических параметров пород за фронтом окисления.

**Вторая группа** – это компоненты пород, которые при атмосферном давлении испытывают довольно значительные изменения свойств уже при сравнительно невысоких температурах. Это, в основном, компоненты, в которых при атмосферном давлении происходят процессы высвобождения воды различного типа: гидрослюды, монмориллонит, гипс. Насколько эти процессы будут значимы при термобарических условиях залегания пород баженовской свиты – может быть выяснено только в результате специальных экспериментальных исследований.

К этой группе относится и пирит, при атмосферном давлении или в вакууме разлагающийся при невысоких температурах.

**Третью группу** составляют компоненты, термические преобразования которых при ТГВ представляются маловероятными: карбонатные минералы (кроме гипса), кремнезём, полевые шпаты, слюды.

Все эти вопросы должны быть экспериментально исследованы с учётом реальных условий залегания пород баженовской свиты, включая величину эффективного сжатия пород и пространственное распределение различных компонентов скелета.

Возможно, такие эксперименты целесообразно проводить на образцах пород, не рассматривая детально степень преобразования отдельных компонентов.

Эти исследования должны обязательно охватывать и область температур, характерных для режима высокотемпературного горения, поскольку и при

проектировании разработки, и при дальнейшем её регулировании, обязательно необходима оценка возможности перехода процесса в режим высокотемпературного горения и последствий этой ситуации.

# Rock matrix components thermal transformations due to application of HPAI method

**V.A. Yudin, A.V. Korolev, I.V. Afanaskin, S.G. Volpin**

**Abstract:** In presented review rock matrix components thermal transformations due to application of HPAI method of oil field development on bazen suite oil fields are discussed. All matrix components are divided in three groups according to probability of their transformations under rock heating during HPAI. The first is kerogen, its thermal dissociation looks the most probable. Second group includes hydromica, montmorillonite, gypsum, pyrite, that easily change their structure and parameters as a result of low-temperature heating under atmosphere pressure and, possibly, during HPAI. In third group are carbonate minerals, silica, feldspar, mica that have very high temperature threshold for structural transformations. The measure and the character of thermal transformations of bazen suite formations should be investigated in a series of special experiments under reservoir.

**Keywords**: HPAI, kerogen pyrolysis, rock matrix thermal transformations, bazen suite.

## Литература

1. BP Energy Outlook 2035. January 2014. http://www.slideshare.net/BP_plc/bp-energy-outlook-2035-2014-booklet
2. Paul Chefurka. World Energy and Population, 2007. http://www.wprr.ru/mirovaya-energiya-i-naselenie-perspektivy-s-2007-po-2100-gg
3. А.М.Мастепанов. Топливно-энергетический комплекс России на рубеже веков: состояние, проблемы и пути решения. Том 1. М.: ИАЦ «Энергия», 2009.
4. М.Д.Белонин, Ю.В.Подольский. Состояние сырьевой базы и прогноз возможных уровней добычи нефти в России до 2030 г. // Минеральные ресурсы России, 2006, № 5. http://www.vipstd.ru/gim/content/view/88/279
5. В.П.Якуцени, Ю.Э.Петрова, А.А.Суханов. Динамика доли относительного содержания трудноизвлекаемых запасов нефти в общем балансе // Нефтегазовая геология. Теория и практика, 2007(2). www.ngtp.ru
6. А.М.Брехунцов, И.И.Нестеров. Нефть битуминозно-глинистых и карбонатно-кремнисто-глинистых пород // Инновационные технологии оценки, моделирования и разработки залежей нефти баженовской свиты: Научно-практическая конференция им. Н.Н. Лисовского - 28 сентября 2010.
7. Ю.Е.атурин, В.П.Сонич, А.Г.Малышев, А.А.Кошелева. О возможном пути интенсификации нефтеизвлечения из отложений баженовской свиты // Освоение ресурсов трудноизвлекаемых и высоковязких нефтей: Международная конференция – Краснодар, 1999.
8. Ю.Е.Батурин, В.П.Сонич, А.Г.Малышев. и др. Проблемы и перспективы освоения баженовской свиты // Нефтяное хозяйство, 2001, № 9.
9. В.В.Ананьев, В.М.Смелков, Н.В.Пронин. Прогнозная оценка ресурсной базы мендым-доманиковых отложений как основного источника углеводородного сырья центральных районов Волго-Уральской нефтегазоносной провинции. «VIPStudio ИНФО», 2007. http://www.vipstd.ru/gim/content/view/459/
10. В.И.Грайфер, А.А.Боксерман, Н.М.Николаев. и др. Интеграция тепловых и газовых методов увеличения нефтеотдачи – основа технико-технологического комплекса разработки месторождений нетрадиционных ресурсов и трудноизвлекаемых запасов нефти // Rusnanotech: Международный форум по нанотехнологиям – Москва, 2010.
11. В.И.Кокорев. Технико-технологические основы инновационных методов разработки месторождений с трудноизвлекаемыми и нетрадиционными запасами нефти. Дис. на соиск. уч. степ. докт. техн. наук. М.: ИПНГ РАН, 2010.
12. А.А.Боксерман, В.А.Савельев, М.С.Джафаров. и др. Термогазовое воздействие – инновационная технология разработки месторождений Сибири // Энрекон-2010: Международная конференция – Москва, 2010.
13. М.Ф.Ямбаев. Основные особенности термогазового метода увеличения нефтеотдачи применительно к условиям сложнопостроенных коллекторов (на основе численного моделирования). Дис. на соиск. уч. степ. канд. техн. наук. М.: ОАО «ВНИИнефть им. ак. А.П. Крылова», 2006.
14. А.Шпильман. Ввод в разработку таких месторождений, как Имилорское, им. В.И. Шпильмана и Гавриковское, может стабилизировать добычу нефти в Югре // Агенство нефтегазовой информации, 29.11.2011. http://www.angi.ru/news.shtml?oid=2782534
15. А.А.Боксерман, В.И.Грайфер, В.И.Кокорев и др. Термогазовый метод увеличения нефтеотдачи // «Интервал», 2008, №7. – с. 6-33.

16. В.В.Плынин. Термогазовый метод и баженовская свита // Интернет-портал сообщества ТЭК EnergyLand.info http://www.energyland.info/analitic-show-50375
17. А.Соломатин. Термогазовое воздействие и месторождения Сибири // Интернет-портал сообщества ТЭК EnergyLand.info http://www.energyland.info/analitic-show-52541
18. S.Partha. In-situ combustion handbook - principles and practices. Final Report, November 1998. Performed Under Contract No. DE-AC22-94PC91008 (Original Report Number NIPER/BDM-0374). «BDM Petroleum Technologies», BDM-Oklahoma, Inc. Bartlesville, Oklahoma, National Petroleum Technology Office U.S. DEPARTMENT OF ENERGY, Tulsa, Oklahoma.
http://repository.icse.utah.edu/dspace/bitstream/123456789/5336/2/DOE-PC-91008-0374-OSTI_ID-3175-.pdf
19. Д.Г.Антониади, А.Р.Гарушев, Б.Г.Ишханов. Настольная книга по термическим методам добычи нефти. Краснодар: «Советская Кубань», 2000.
20. С.Г.Вольпин, В.А.Юдин, Р.М.Кац. и др. Применение суперкомпьютерных технологий – ключ к решению проблем повышения нефтеотдачи на месторождениях России // Наука и общество: Новые технологии для новой экономики России: С.-Петербургский научный форум. СПб., 30.09 – 04.10.2014.
21. В.Б.Бетелин. Экзафлопные вычисления и энергетическая безопасность США в период 2010–2020–2030 гг. // «Энергия», № 3, 2011.
22. В.Б.Бетелин. «Цифровое месторождение» — путь к трудноизвлекаемым запасам углеводородов // Наука и общество: Новые технологии для новой экономики России: С.-Петербургский научный форум. СПб., 2013.
23. В.Б.Бетелин, А.А.Боксерман, В.Е.Костюков. и др. Проблемы управления процессами повышения нефтеотдачи на основе моделирования на супер-ЭВМ // НефтеГазоПромысловый Инжиниринг, 3 кв., 2010.
24. ECLIPSE 2011.2 Technical Description.
25. http://www.roxar.ru/solutions/tempest/
26. http://www.timezyx.ru/mkt.php
27. tNavigator 3.0 Техническое руководство, 2010.
28. CMG STARS 2010 Users Guide
29. Теоретические вопросы термической переработки сланца с твёрдым теплоносителем. http://www.e-ope.ee/_download/euni_repository/file/2064/Enefit2.zip/_____.html
30. Н.С.Балушкина, Г.А.Калмыков, В.С.Белохин. и др. Кремнистые коллекторы баженовского горизонта Средне-Назымского месторождения и структура их пустотного пространства // Вестник Московского университета. Серия 4. Геология, том 4, № 2, 2014. - с. 35-43.
31. Р.А.Хамидуллин, Г.А.Калмыков, Д.В.Корост и др. «Фильтрационно-емкостные свойства пород баженовской свиты // Вестник Московского университета. Серия 4. Геология, №5, 2013. - с. 57-64.
32. Т.А.Кирюхина, Н.И.Коробова, Д.В.Корост и др. Закономерности строения баженовского горизонта и верхов абалакской свиты в связи с перспективами добычи нефти из них // Геология нефти и газа, № 3, 2013. - с. 48-60.
33. Н.С.Балушкина, Г.А.Калмыков, Р.А.Хамидуллин и др. Комплексная литофизическая типизация пород баженовской свиты по данным керна и комплексу ГИС // SPE-171168-RU, 2014.
34. Г.А.Калмыков, Н.С.Балушкина, И.С.Афанасьев и др. Баженовская свита. Общий обзор, нерешенные проблемы // Rogtec Российские нефегазовые технологии, №25. - с. 24-36.
35. Р.А.Хамидуллин, Г.А.Калмыков, Д.В.Корост. Емкостные свойства пород Баженовской свиты // Российская техническая нефтегазовая конференция и выставка SPE 2012.
36. И.С.Афанасьев, Е.В.Гаврилова, Е.М.Бирун. Баженовская свита. Общий обзор, нерешенные проблемы // Russian oil and gas technologies. 25 мая 2011. http://www.rogtecmagazine.com/ru-blog/%D0%B1%D0%B0%D0%B6%D0%B5%D0%BD%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F-%D1%81%D0%B2%D0%B8%D1%82%D0%B0-%D0%BE%D0%B1%D1%89%D0%B8%D0%B9-%D0%BE%D0%B1%D0%B7%D0%BE%D1%80-%D0%BD%D0%B5%D1%80%D0%B5%D1%88%D0%B5
37. Н.С.Балушкина. Литофизическая типизация и нефтеносность пород баженовского горизонта в зоне сочленения Сургутского и Красноленинского сводов. Дис. на соиск. уч. степ. канд. геол.-мин. наук. М.: МГУ, 2012.
38. А.Г.Замирайлова. Литология баженовской и георгиевской свит центральной и северной частей Западно-Сибирской плиты. Дис. на соиск. уч. степ. канд. геол.-мин. наук. Новосибирск: ИНГГ, 2004.
39. Ю.А.Кузьмин, Н.В.Судат. Особенности геологического строения, оценки и учета в госбалансе запасов углеводородов в отложениях баженовской свиты месторождений Ханты-Мансийского автономного округа-Югры // Вестник недропользователя Ханты-Мансийского автономного округа, № 24, 2011.
40. П.С.Куляпин, Т.Ф.Соколова. Использование статистического моделирования при интерпретации данных ГИС в нефтематеринских породах баженовской свиты Западно-Сибирской нефтегазоносной провинции // Технология сейсморазведки, №3, 2013. - с. 28–42.
41. А.А.Боксерман, В.И.Грайфер, В.И.Кокорев. и др. Термогазовый метод увеличения нефтеотдачи // «Интервал», 2008, №7. - с. 6-33.
42. В.И.Грайфер, А.А.Боксерман, Н.М.Николаев и др. Интеграция тепловых и газовых методов увеличения нефтеотдачи – основа технико-технологического комплекса разработки месторождений нетрадиционных ресурсов и трудноизвлекаемых запасов нефти // Международный форум по нанотехнологиям «Rusnanotech». М., 2010.

43. В.И. Кокорев. Основы управления термогазовым воздействием на породы баженовскои свиты применительно к геологическим условиям Средне-Назымского и Галяновского месторождений // Нефтепромысловое дело, № 6. 2010. - с. 29-32.
44. И.В.Афанаскин. Повышение технологической эффективности метода направленной закачки воздуха в нефтяные пласты на основе численного моделирования и результатов гидродинамических исследований скважин. Дис. на соиск. уч. стп. канд. техн. наук. М.: ВНИИнефть, 2013.
45. Д.Хант. Геохимия и геология нефти и газа. М.: Мир, 1982.
46. Ж.Бурже, П.Сурио, М.Комбарну. Термические методы повышения нефтеотдачи пластов. М.: «Недра», 1988.
47. Б.П.Никольский, О.Н.Григоров, М.Е.Позин. Справочник химика. Том 6. Сырьё и продукты промышленности органических веществ. Л.: «Химия», 1967.
48. Н.И.Зеленин, И.М.Озеров. Справочник по горючим сланцам. Л.: «Недра», 1983.
49. В.И.Кокорев, С.А.Власов, Н.Г.Судобин и др. Исследование процесса термического воздействия на образцы пород баженовской свиты // Нефтепромысловое дело, 2010, №3. – с. 12-19.
50. К.А.Щеколдин. Исследование возможностей регулирования технологии термогазового воздействия на залежи баженовской свиты // «Территория нефтегаз», №9, 2012.
51. A.Zolotukhin, A.Bokserman, V.Kokorev, K.Shchekoldin. New Upstream and Downstream Technologies for Extra Heavy Oils. SPE Heavy Oil Conference, Calgary, Alberta, Canada, 2012.
52. Ф.Д.Овчаренко. Гидрофильность глин и глинистых материалов. Киев: Изд. АН УССР, 1961.
53. В.В.Ржевский, Г.Г.Новик. Основы физики горных пород. М.: «Недра», 1978.
54. В.Г.Савельев, Н.Ф.Федоров. Физическая химия силикатов и других тугоплавких соединений. М.: 1988.
55. Электронная строительная энциклопедия. http://www.stfa.ru/povedenie-slyud-pri-nagrevanii/
56. С.Ц.Ханхасаева, Э.Ц.Дашинамжилова, В.В.Рампилова. Влияние термообработки на текстурные свойства монтмориллонита // Вестник Бурятского государственного университета, № 3, 2011. http://cyberleninka.ru/article/n/vliyanie-termoobrabotki-na-teksturnye-svoystva-montmorillonita
57. В.С.Бесков, В.С.Сафронов. Общая химическая технология и основы промышленной экологии. М.: Химия, 1999.
58. Г.И.Капаев. Физико-химические основы процесса термического разложения солей угольной кислоты. Дис. на соиск. уч. степ. канд. хим. наук. М, 2009.
59. Д.Д.Мотыль. Некоторые особенности нагрева и терморазложения кальцита. Дис. на соиск. уч. степ. канд. хим. наук. М., 2000.
60. Т.Н.Акимова. Минеральные вяжущие вещества. Учебное пособие. М.: МАДИ, 2007.
61. Н.И.Ерёмин. Неметаллические полезные ископаемые. М.: Изд. МГУ, 2004.
62. В.С.Попов. Как образуются граниты // Соросовский образовательный журнал, Науки о Земле, 1997. http://www.pereplet.ru/obrazovanie/stsoros/344.html
63. Н.Ф.Солодкий, А.С.Шамриков, В.М.Погребенков. Минерально-сырьевая база Урала для керамической, огнеупорной и стекольной промышленности. Томск: Издательство ТПУ, 2009.
64. В.П.Прянишников. Система кремнезёма. Л.: Издательство литературы по строительству, 1971.

# Анализ подходов к математическому моделированию разработки нефтяных месторождений вертикальными скважинами с трещинами гидроразрыва пласта

**И.В. Афанаскин[1], А.В. Королёв[1]**

*1 – кандидат технических наук.*

**Аннотация:** В работе анализируется эффективность ряда упрощенных подходов к моделированию разработки нефтяных месторождений с применением трещин гидроразрыва пласта.

**Ключевые слова:** Гидроразрыв пласта, разработка нефтяных месторождений, гидродинамическое моделирование разработки.

## Введение

Истощение запасов нефти на старых месторождениях и открытие большого числа месторождений с низкопроницаемыми коллекторами потребовало развития методов интенсификации добычи нефти. Проведение гидравлического разрыва пласта (ГРП) позволило ввести в промышленную разработку нефтяные месторождения в низкопроницаемых коллекторах Западной Сибири. В настоящее время применение ГРП имеет массовый характер. В год по всей стране проводятся тысячи операций по ГРП. Трещины ГРП могут быть весьма протяженными, их длина может быть сравнима с расстоянием между скважинами. Поэтому очень важным является правильный учет трещин ГРП при гидродинамическом моделировании разработки нефтяных месторождений.

## 1. Подходы к моделированию

МакКракен определил, что при длине трещин меньше четверти расстояния между скважинами, скважины с трещинами гидроразрыва в гидродинамической модели можно представить в виде радиальных скважин с отрицательным скин-фактором (то есть, в виде скважин с обработанной призабойной зоной, приведенный радиус которых больше фактического) [7]. Автор [7] не уточняет, насколько этот критерий корректен для применения такого подхода. Однако он отмечает, что во многих случаях подобное упрощение работает очень хорошо.

В некоторых случаях необходимы более сложные методы учета трещины ГРП при гидродинамическом моделировании:

1. когда длина трещины составляет существенную долю расстояния между скважинами;
2. когда существует опасность прорыва газа из газовой шапки и (или) воды из водоносной области в подошве продуктивного пласта;
3. проявление совместного влияния слоистости и трещины;
4. выпадение конденсата в призабойной зоне газовой скважины и снижение проницаемости.

Прямое моделирование трещины - трудная задача для численного симулятора, и ее необходимо выполнять осторожно. При моделировании трещин гидроразрыва приходится решать ряд проблем. Главная проблема связана с устойчивостью счета. Для ее обеспечение необходимо использовать очень маленький размер шага по времени – часы и даже минуты. Для расчета десятилетий разработки месторождения с сотнями скважин и десятками трещин ГРП это неприемлемо.

Как правило, необходимо использование упрощенных моделей. Использование же в полномасштабном моделировании измельчения сетки с описанием прискважинных эффектов представляется роскошью даже для современного уровня производительности компьютеров. Размеры сеточных блоков современных гидродинамических моделей в горизонтальной плоскости составляют 25-100 м. Ширина (раскрытость) трещин по оценкам разных специалистов может колебаться в пределах 3-13 мм. [6, 7, 10]. Проницаемость низкопроницаемого пласта может составлять 1-10 мД, а проницаемость трещины - от 50 000 до 150 000 мД. Такая разница в размерах и проницаемости делает прямое моделирование крайне затруднительным.

Одним из способов решения данной проблемы является введение локального измельчения сетки с плавным изменением размеров ячеек. При этом считается, что минимальный практический размер ячейки для построения модели составляет около 0,3 м. [7]. Соответственно необходимо корректировать и проводимость (произведение ширины трещины на проницаемость). Сразу становится очевидным, что объем трещины будет определен неправильно. В практическом смысле объем не так уж важен. Однако будет завышен начальный дебит скважины после гидроразрыва по сравнению с реальным дебитом [7] при управления скважиной по забойному давлению. С

другой стороны, обычно симулятор не учитывает объем ствола скважины, с которым в действительности необходимо считаться.

Кроме того, функции относительных фазовых проницаемостей (ОФП) в трещине принимаются линейно зависящими от насыщенности. При этом насыщенность трещины связанной водой и остаточная нефтенасыщенность часто полагается равной нулю. Следовательно, для ячеек модели, содержащих и матрицу пласта и трещину ГРП, необходимо рассчитывать некие эффективные функции ОФП.

В работах [4, 6] предлагается следующий способ расчета эффективной абсолютной проницаемости и ОФП для ячеек модели, содержащих и матрицу пласта, и трещину ГРП (трещина перпендикулярна оси X в декартовой системе координат, тензор абсолютной проницаемости диагональный):

$$\bar{k}_x = k_x,\ \bar{k}_{y,z} = k_{y,z} + \frac{\omega_f k_f}{\Delta x}, \qquad (1)$$

$$\bar{k}_{r\alpha x} = k_{r\alpha},\ \bar{k}_{r\alpha y,z} = \frac{k_{y,z} k_{r\alpha}}{\bar{k}_{y,z}} + \frac{\omega_f k_f k_{rf\alpha}}{\bar{k}_{y,z}\Delta x}, \qquad (2)$$

$$\alpha = o, w, g,$$

где $k_x$, $k_y$, $k_z$ - абсолютные проницаемости пласта вдоль соответствующих осей; $\Delta x$ - размер сеточного блока вдоль оси X, $k_{r\alpha}$ и $k_{rf\alpha}$ - ОФП для фазы $\alpha = o, w, g$ пласта и трещины ГРП соответственно; $\omega_f$ - ширина (раскрытость) трещины; $\bar{k}_x$, $\bar{k}_y$, $\bar{k}_z$ - эффективные абсолютные проницаемости пласта вдоль соответствующих осей; $\bar{k}_{r\alpha x}$, $\bar{k}_{r\alpha y}$, $\bar{k}_{r\alpha z}$ - эффективные ОФП для фазы $\alpha = o, w, g$ вдоль соответствующих направлений. В результате такого определения ОФП становятся зависящими от направления фильтрации.

Существуют другие подходы к моделированию трещины с помощью введения эффективной проницаемости ячеек, содержащих трещину ГРП, например [2].

При моделировании трещины в виде совокупности стоков, расположенных в разных ячейках, можно записать следующие выражения для притока к трещине [5]:
для ячейки, в которой расположен центр трещины (сток)

$$q = \frac{k\Delta z}{\mu} \frac{2\left[\pi - \Psi(r_1) - \Psi(r_2)\right]\sum_{i=1}^{4} a_i (p_0 - p_w)}{\sum_{i=1}^{4} a_i P(Z_i) - 2\left[\pi - \Psi(r_1) - \Psi(r_2)\right]}, \qquad (3)$$

для любой другой ячейки, через которую проходит трещина

$$q = \frac{k\Delta z}{\mu} \frac{2\left[\Psi(r_1) - \Psi(r_2)\right]\sum_{i=1}^{4} a_i (p_0 - p_w)}{\sum_{i=1}^{4} a_i P(Z_i) - 2\left[\Psi(r_1) - \Psi(r_2)\right]}, \qquad (4)$$

где

$$\Psi(r_i) = arctg\left(\frac{\sqrt{f^2 - r_i^2}}{r_i}\right),$$

$$P(Z) = Re\left[ln\left(\frac{Z}{f} + \sqrt{\frac{Z^2}{f^2} - 1}\right)\right], \qquad (5)$$

$$a_i = \frac{\Delta y}{\Delta x_i} \text{ для } i = 1, 3, \qquad (6)$$

$$a_i = \frac{\Delta x}{\Delta y_i} \text{ для } i = 2, 4, \qquad (7)$$

$\Delta x$, $\Delta y$, $\Delta z$ - размеры ячейки; $\Delta x_i$, $\Delta y_i$ - расстояние от узла, находящегося в данной ячейке, до соседних узлов; $p_0$ - давление в ячейке; $p_w$ - давление в скважине; $k$ - проницаемость; $\mu$ - вязкость; $Z_i$ - комплексная координата *i*-ого узла в системе координат, связанной с трещиной; $f$ - полудлина трещины в ячейке; $r_1$, $r_2$ - расстояния точек пересечения трещины с границами ячейки от центра трещины, рис. 1. Если трещина заканчивается внутри ячейки, то $r_2 = f$. Данный подход основан на решении И.А. Чарного задачи о притоке несжимаемой жидкости в изотропном несжимаемом пласте к вертикальной трещине ГРП бесконечной проницаемости, когда длина трещины много больше диаметра скважины.

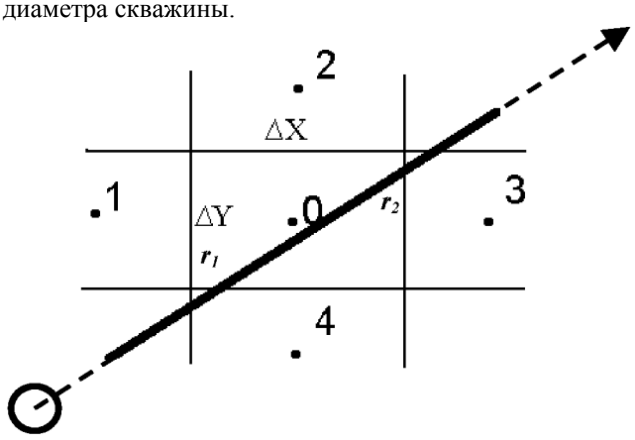



**Рис. 1.** Трещина ГРП в сеточной модели [5]

Возможны другие подходы к моделированию трещины в виде совокупности стоков, расположенных в разных ячейках, например, так называемая «модель скважины с обобщенными участками перфорации» [3].

Существуют и другие подходы к моделированию трещины, например с помощью недиагонального тензора проницаемости [1].

## 2. Численные эксперименты

Для рассмотрения эффективности различных подходов к моделированию трещин ГРП в симуляторе dz10 [8] построена двухфазная (нефть-вода) секторная гидродинамическая модель элемента девятиточечной системы разработки, рис. 2.

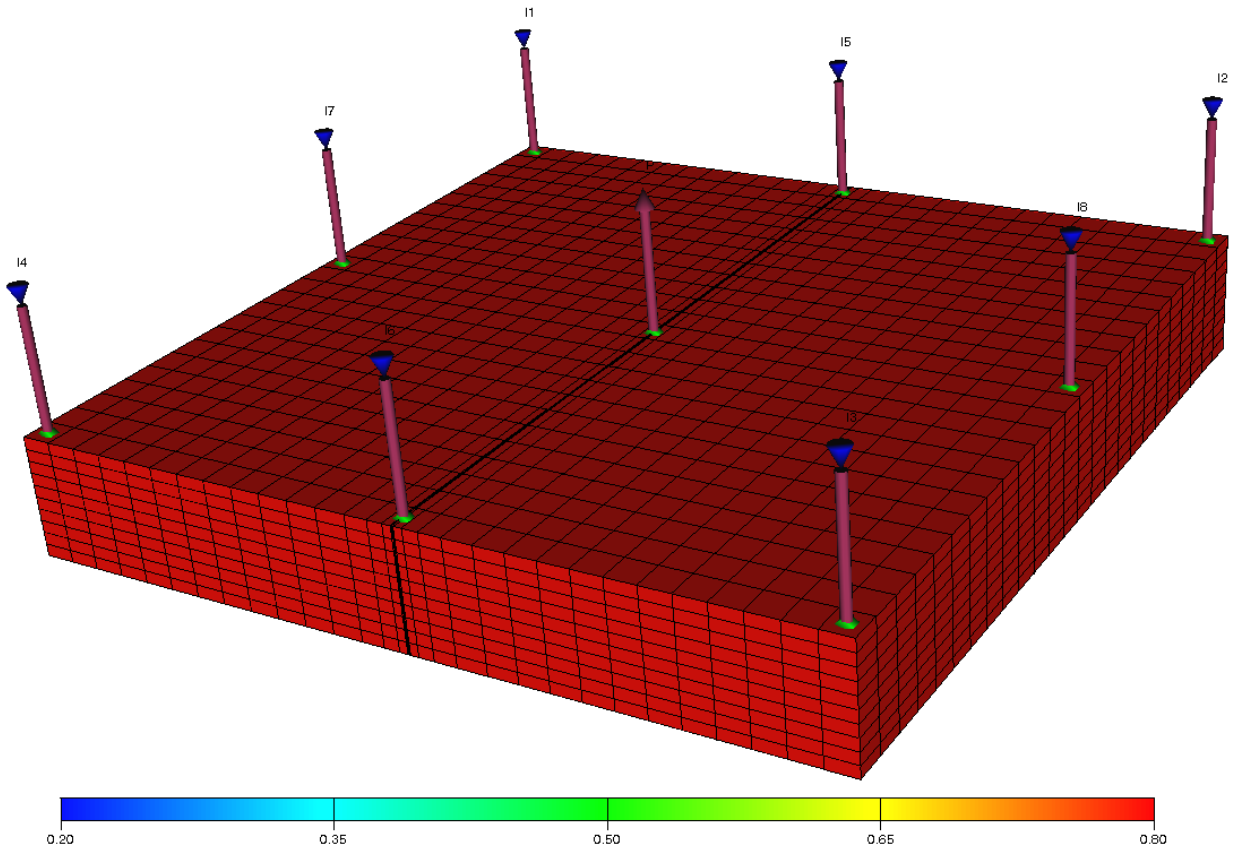



**Рис. 2.** Секторная модель. Поле нефтенасыщенности на начальный момент времени

Параметры модели следующие:

1. Сетка прямоугольная, блочноцентрированная.
2. Размеры модели 25*25*10 ячеек.
3. Размеры ячеек 27*27*1 м.
4. Проницаемость пласта 10 мД.
5. Пористость пласта 18 %.
6. Доля коллектора (песчанистость) 100 %.
7. Глубина кровли 3000 м.
8. Коэффициент анизотропии вертикальной проницаемости 0,1 д.ед.
9. Начальное пластовое давление 300 атм.
10. Начальная водонасыщенность 0,2 д.ед.
11. Свойства нефти при начальном пластовом давлении: объемный коэффициент 1,55 $м^3/м^3$, сжимаемость $1{,}01 \cdot 10^{-4}$ 1/атм, вязкость 0,397 мПа·с.
12. Растворимость газа в нефти при начальном пластовом давлении 203,5 $м^3/м^3$. Давление насыщения нефти газом 125 атм.
13. Свойства воды при начальном пластовом давлении: объемный коэффициент 1,02 $м^3/м^3$, сжимаемость $4{,}7 \cdot 10^{-5}$ 1/атм, вязкость 0,36 мПа·с.
14. Плотности нефти, воды и газа при нормальных условиях 815; 1056 и 1,130 $кг/м^3$ соответственно.
15. Сжимаемость пласта $4{,}7 \cdot 10^{-5}$ 1/атм.
16. ОФП и капиллярное давление для матрицы пласта – рис. 3.
17. Одна добывающая скважина с трещиной ГРП в центре модели. Скважина работает с постоянным дебитом жидкости 200 $м^3$/сут. Ограничение по минимальному забойному давлению 125 атм.
18. Восемь нагнетательных скважин, расположенных в углах и на сторонах элемента. Скважины работают с постоянным забойным давлением 350 атм.

Параметры трещины ГРП следующие:

1. Длина трещины 513 м. Трещина вертикальная, перпендикулярна оси X.
2. Ширина (раскрытость) 5 мм.
3. Проницаемость 150 000 мД.
4. Пористость трещины 0,35 д.ед.
5. ОФП для трещины ГРП – рис. 4. Капиллярное давление в трещине ГРП равно нулю.

Было рассмотрено 7 вариантов.

Вариант 1 – прямое моделирование трещины ГРП с помощью локального измельчения сетки с плавным изменением размеров ячеек, рис 5. Ячейки, моделирующие трещину, имеют ширину 5 мм и фильтрационно-емкостные параметры, соответствующие трещине.

Вариант 2 - моделирование трещины ГРП с помощью эффективной проницаемости и эффективных функций ОФП без измельчения сетки. Фильтрационные параметры этих ячеек рассчитывались по формулам (1) и (2). Полученные в результате расчета ОФП показаны на рис. 4.

Вариант 3 – моделирование трещины ГРП в виде совокупности стоков, расположенных в разных ячейках, рис. 6. Проводимость трещины бесконечна.

Вариант 3' – моделирование трещины ГРП в виде совокупности стоков, расположенных в разных ячейках. Подбирая параметры модели трещины ГРП в

виде совокупности стоков можно получить совпадение результатов расчетов с прямым моделированием трещины.

Вариант 3" – моделирование трещины ГРП в виде совокупности стоков, расположенных в разных ячейках. Увеличена длина трещины.

Вариант 4 – моделирование трещины ГРП с помощью изменения коэффициента продуктивности (в соответствии с коэффициентом продуктивности по варианту 1).

Вариант 5 – моделирование притока к вертикальной скважине без ГРП.

Прямое моделирование трещины ГРП с помощью локального измельчения сетки с плавным изменением размеров ячеек (вариант 1) является эталонным расчетом. Корректность расчета подтверждается правильным с общефизической точки зрения распределением нефтенасыщенности в процессе обводнения, рис. 7 и 8.

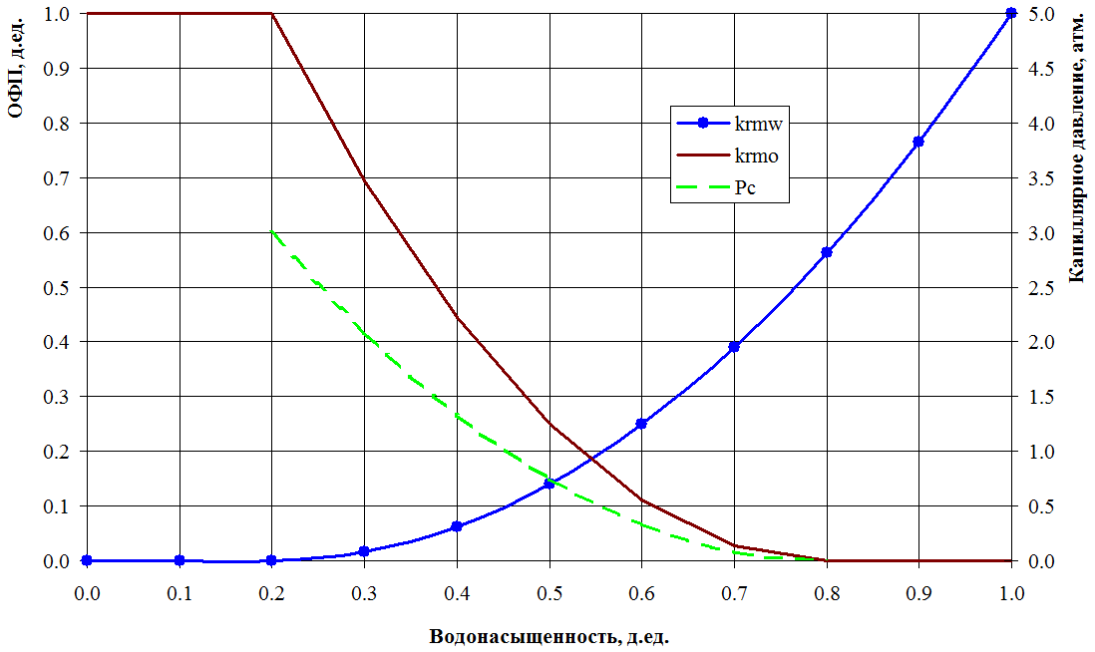



**Рис. 3.** ОФП и капиллярное давление для матрицы пласта
krmw – ОФП для воды в матрице, krmo – ОФП для нефти в матрице,
Pc – капиллярное давление в системе вода-нефть в матрице породы.

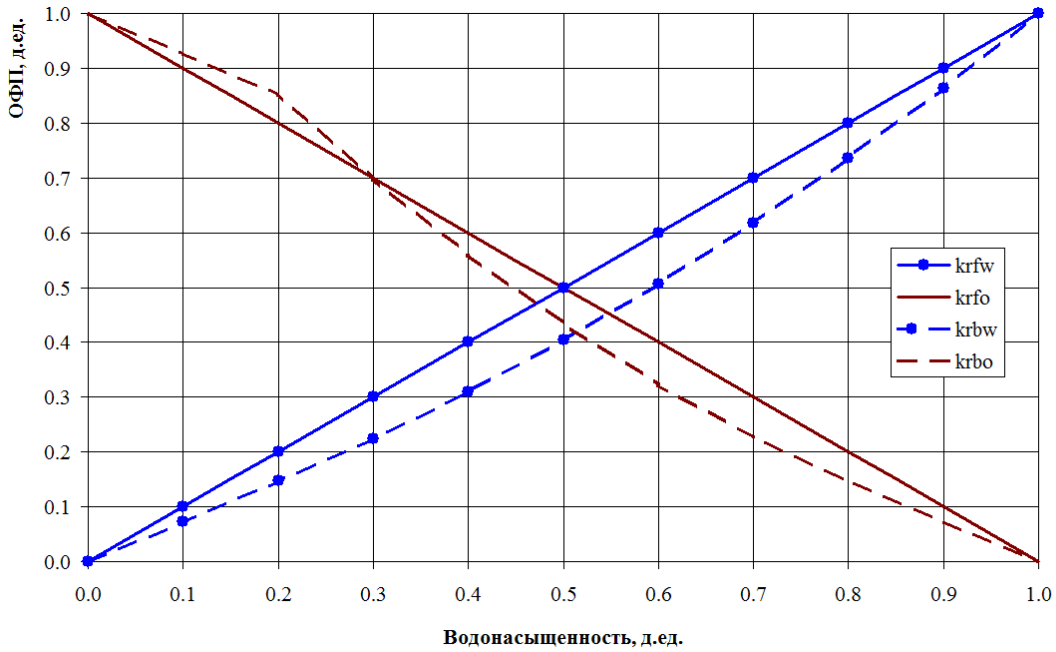



**Рис. 4.** ОФП для трещины ГРП
krfw – ОФП для воды в трещине, krfo – ОФП для нефти в трещине,
krbw – ОФП для воды в блоке модели с трещиной, krbo – ОФП для нефти в блоке модели с трещиной.

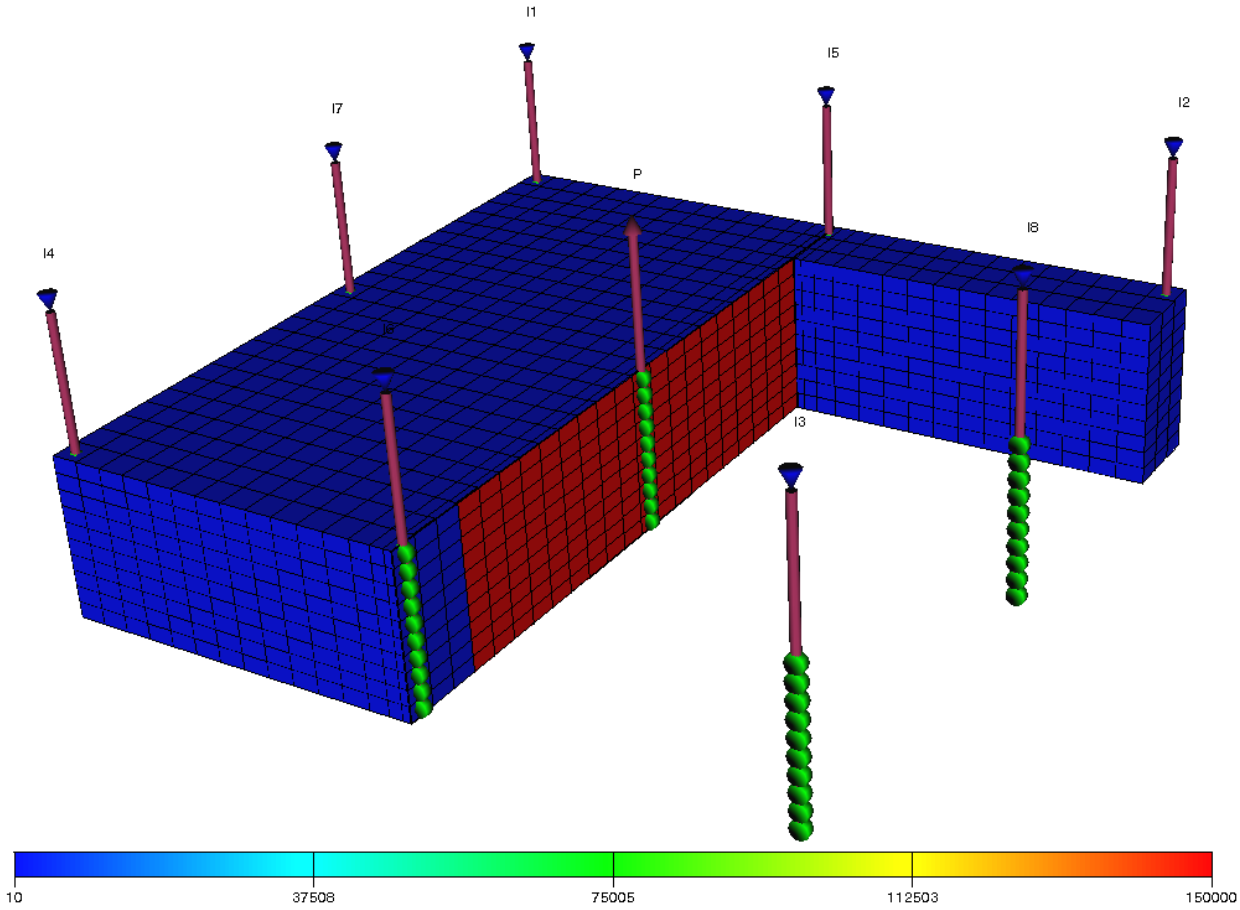



**Рис. 5.** Распределение абсолютной проницаемости (мД)
Прямое моделирование трещины ГРП. Вариант 1

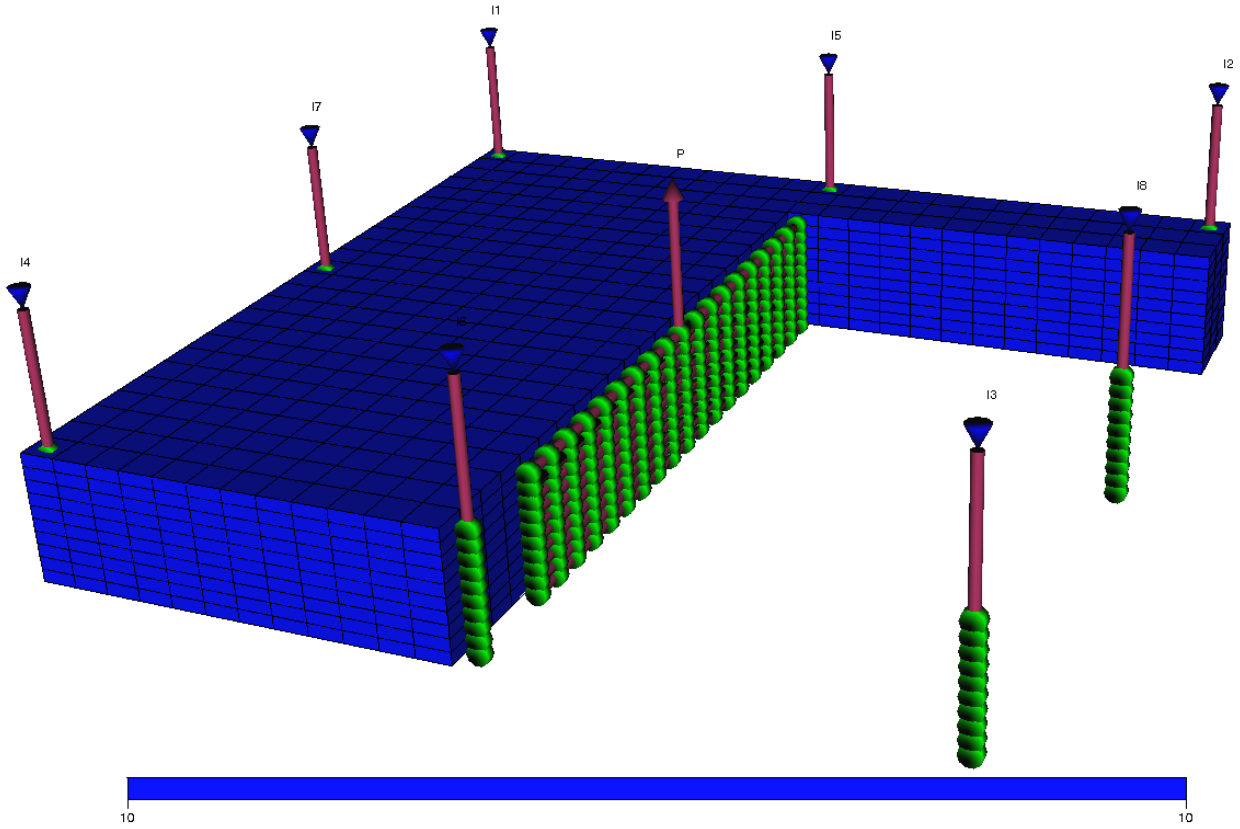



**Рис. 6.** Распределение абсолютной проницаемости (мД)
Моделирование трещины ГРП в виде совокупности стоков. Вариант 3

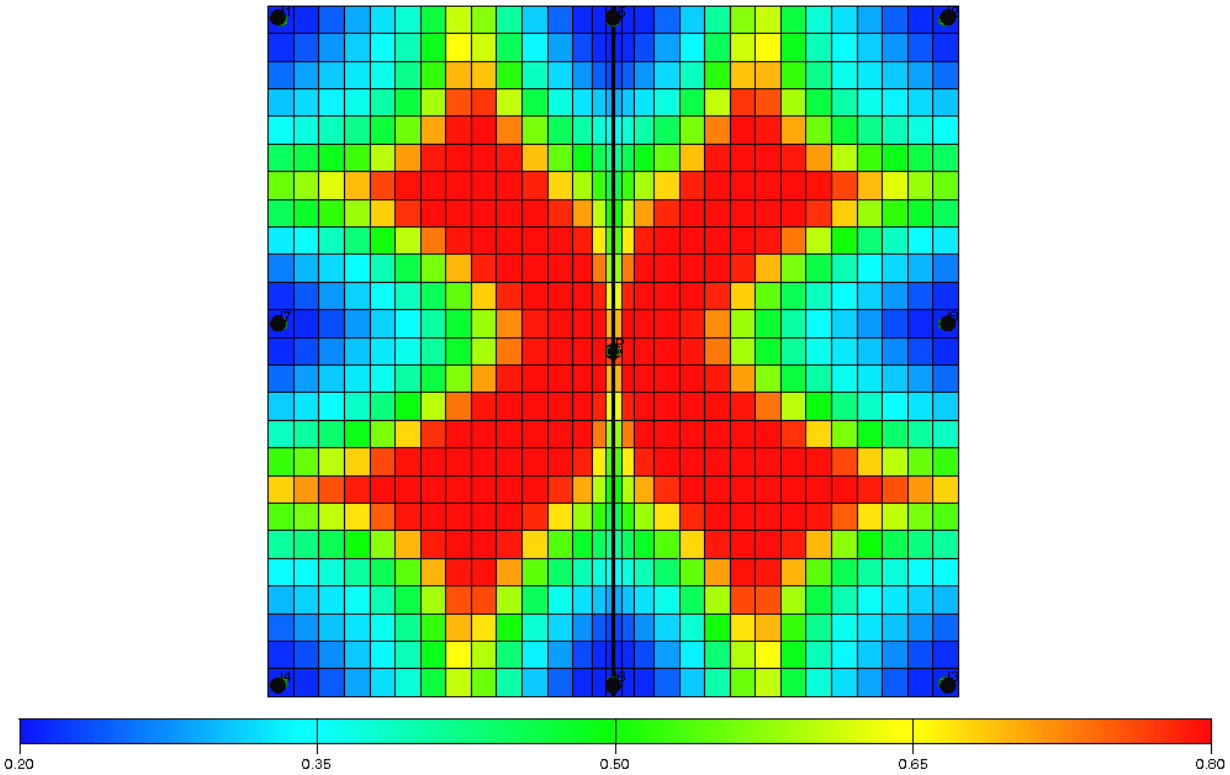



**Рис. 7.** Распределение нефтенасыщенности через 2,5 года (д.ед.).
Прямое моделирование трещины ГРП. Вариант 1.

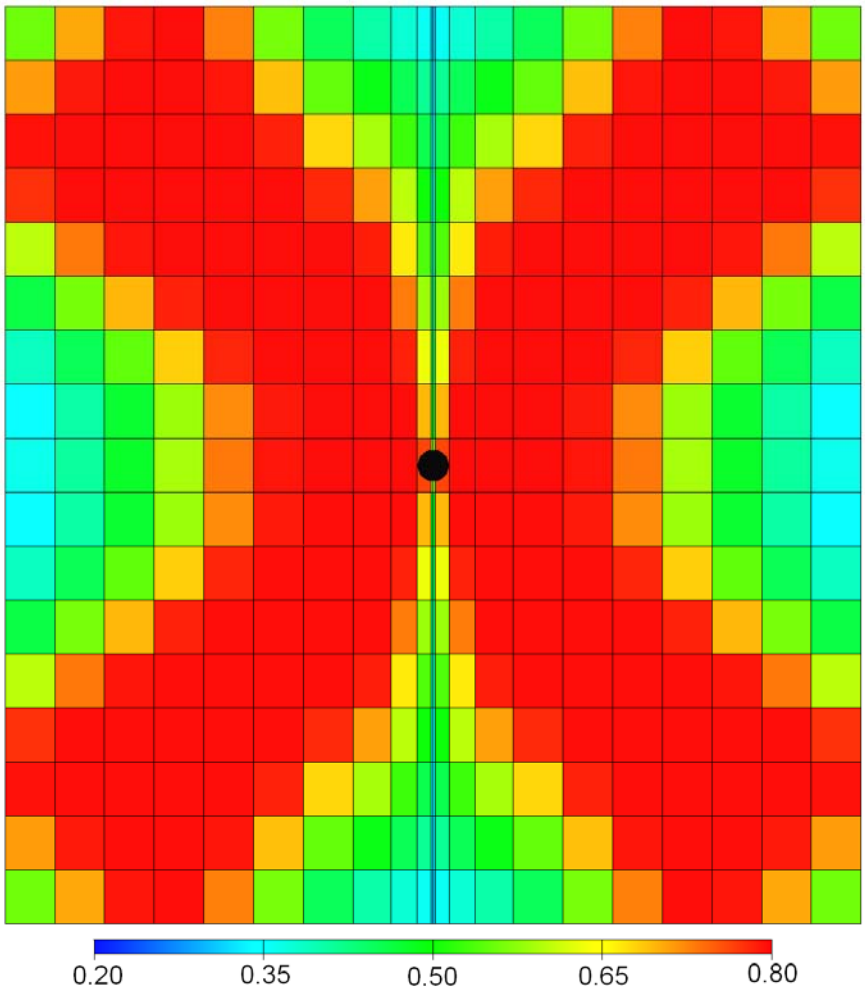



**Рис. 8.** Распределение нефтенасыщенности вблизи трещины через 2,5 года (д.ед.).
Прямое моделирование трещины ГРП. Вариант 1.

На рис. 9-11 показаны результаты расчетов по вариантам – изменение во времени обводненности, забойного давления и давления в скважинной ячейке. Варианты 1, 2 и 3' близки между собой. Они незначительно отличаются временем начала обводнения скважины.

Вариант 3 - моделирование трещины ГРП в виде совокупности стоков, расположенных в разных ячейках, без ручной подгонки коэффициентов продуктивности стоков - показывает более раннее и более интенсивное обводнение скважины. В случае необходимости моделирования экстремального обводнения скважины по трещине ГРП (которое иногда наблюдается на практике) также можно легко использовать данный подход, меняя параметры трещины – вариант 3".

Вариант 4 - моделирование трещины ГРП с помощью изменения коэффициента продуктивности – не позволяет моделировать обводнение скважины по трещине. Кривая обводнения для этого варианта идентична кривой для варианта 5 – без трещины. Некоторое расхождение кривых обводнения для вариантов 4 и 5 объясняется тем, что по варианту 5 в процессе обводнения скважины забойное давление падает до своего ограничения 125 атм. и скважина переключается на управление по забойному давлению, рис. 10. В результате дебит по жидкости несколько падает и обводнение скважины замедляется.

Даже при условии равенства коэффициентов продуктивности добывающей скважины в вариантах 4 и 1 забойное давление по варианту 4 существенно отличается от забойного давления по варианту 1. Это объясняется тем, что наличие трещины ГРП влияет на распределение давления в пласте. При заданном дебите забойное давление определяется, как:

$$p_w = p_b - \frac{q}{PI}, \qquad (8)$$

где $q$ - дебит скважины; $PI$ - коэффициент продуктивности, зависящий от размеров ячейки, вязкости и фильтрационных параметров; $p_b$ - давление в ячейке сетки, содержащей скважину. Так как давление в скважинной ячейке по вариантам 1 и 4 существенно отличается (в варианте 4 не учитывается геометрии трещины ГРП), рис. 11, то отличается и забойное давление.

Это еще раз доказывает то, что следует из простой логики: нельзя с помощью изменения коэффициента продуктивности моделировать трещину ГРП, размеры которой многократно превышают размеры скважинной ячейки.

В табл. 1 приведено время счета по вариантам (по сравнению с вариантом 1). Видно, что использование упрощенных моделей трещины ГРП приводит к существенному ускорению счета.

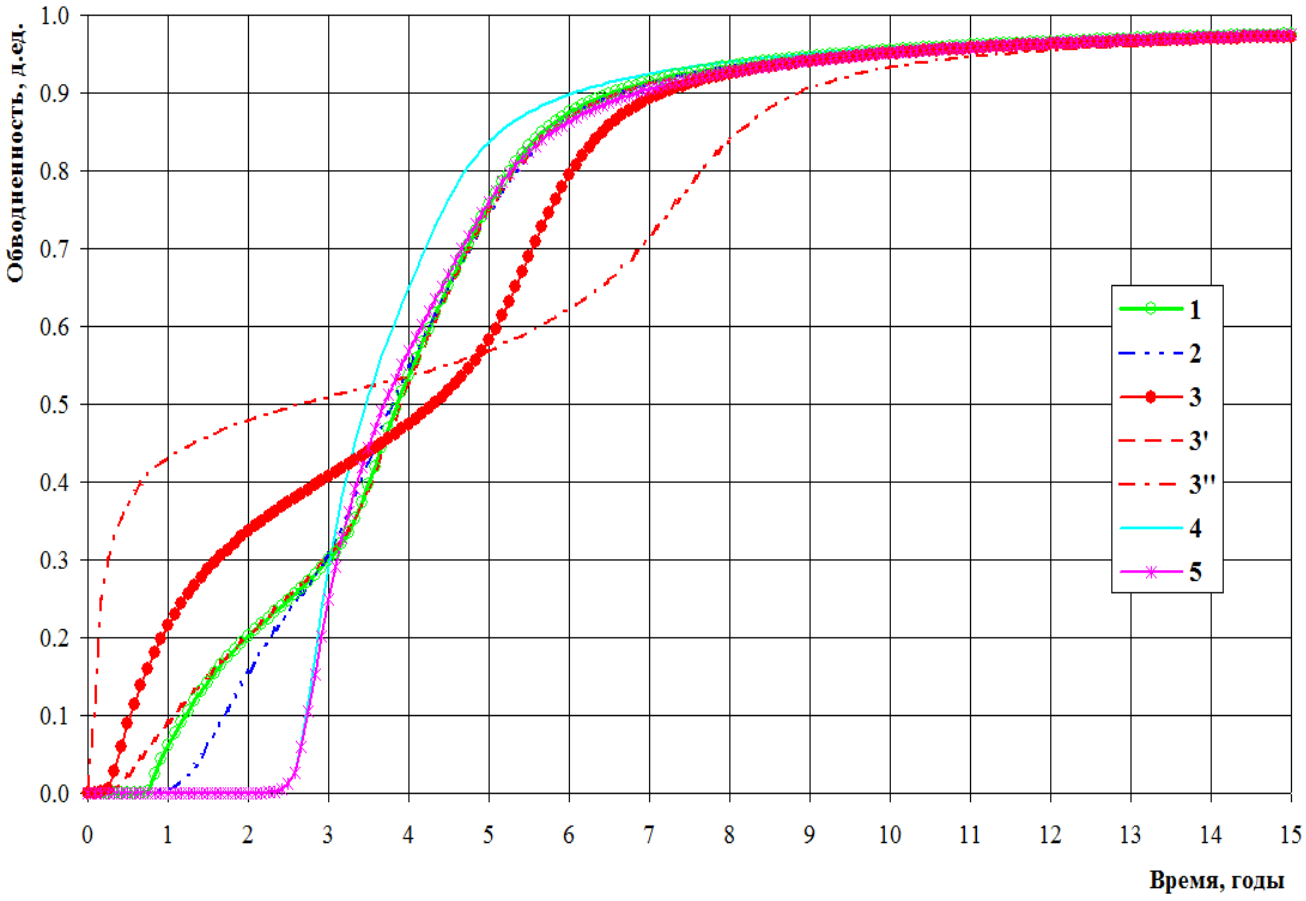



**Рис. 9.** Изменение обводненности во времени по вариантам.

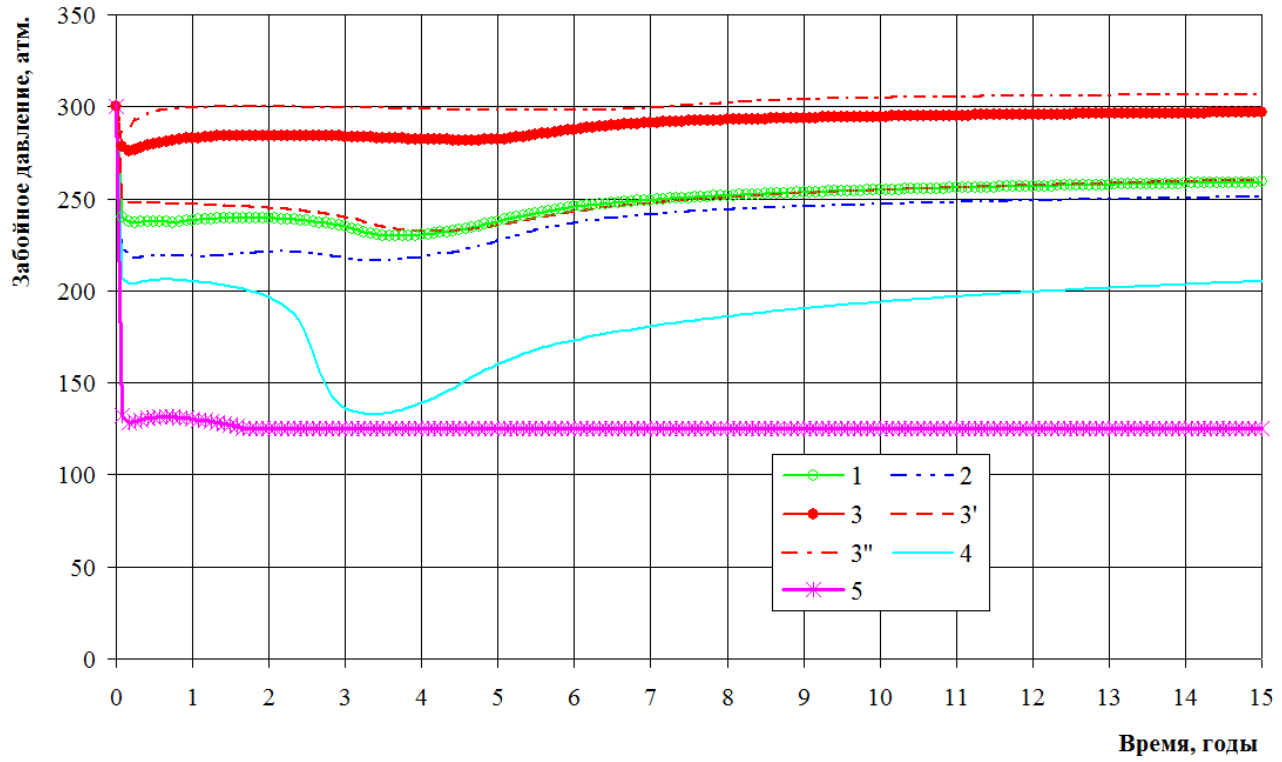



**Рис. 10.** Изменение забойного давления во времени по вариантам.

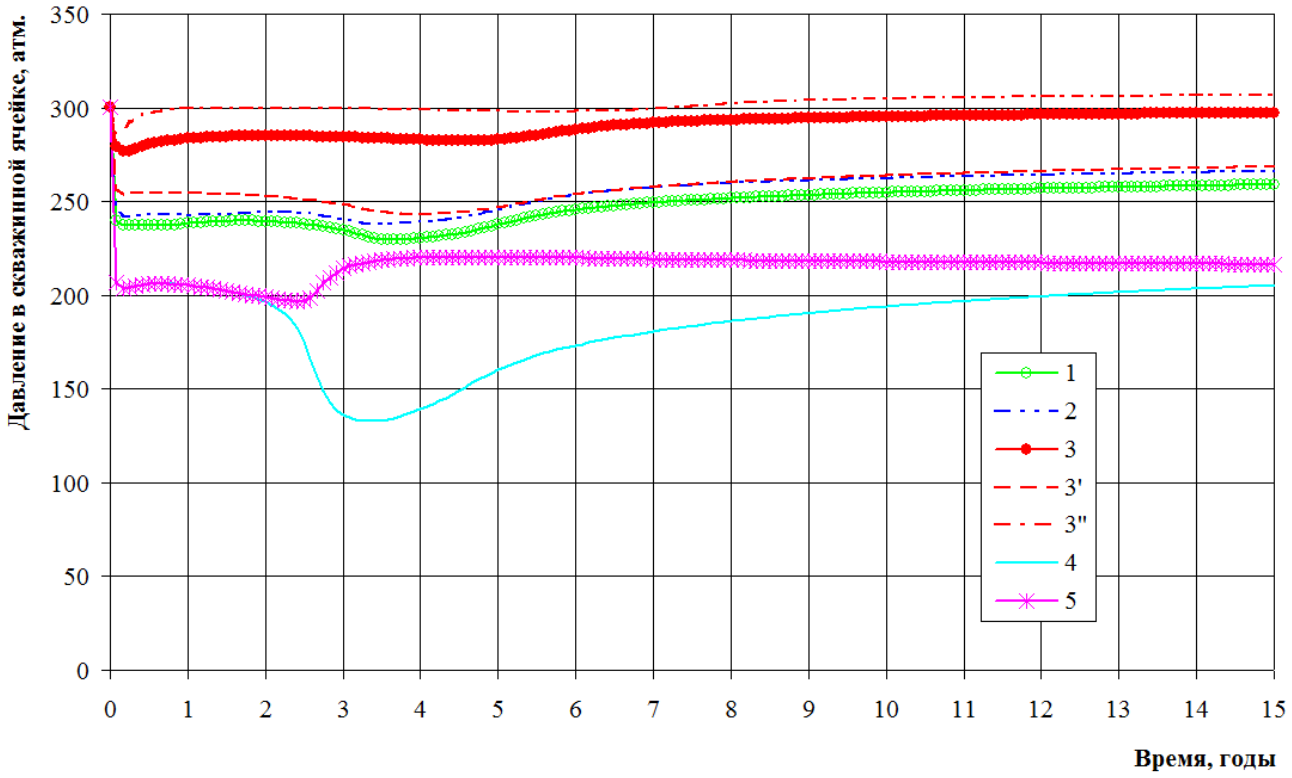



**Рис. 11.** Изменение давления в скважинной ячейке во времени по вариантам.

**Таблица 1**

Сравнение времени счета по вариантам

| Вариант | 1 | 2 | 3 | 3' | 3" | 4 | 5 |
|---|---|---|---|---|---|---|---|
| Уменьшение времени счета, раз | 1 | 4,4 | 1,7 | 4,4 | 1,4 | 2,4 | 4,4 |

## Выводы

1. Прямое моделирование трещины ГРП затруднительно (но возможно) ввиду необходимости построения нерегулярной сетки и вычислительных проблем, возникающих из-за большой разницы размеров и проницаемости сеточных блоков пласта и трещины.
2. Приемлемые по точности результаты могут быть получены:
   a. При использовании эффективной абсолютной проницаемости и эффективных функций ОФП для ячеек, содержащих трещину ГРП.
   b. При моделировании трещины ГРП в виде совокупности стоков, расположенных в ячейках, содержащих трещину.
3. Прямое моделирование трещин ГРП может использоваться в небольших исследовательских моделях, например для обоснования параметров упрощенных моделей.
4. С точки зрения простоты и гибкости использования оптимальным является моделирование трещины ГРП в виде совокупности стоков в ячейках, содержащих трещину.
5. Если длина трещины существенно превышает размеры расчетных ячеек и сопоставима с расстоянием между скважинами, то подход с моделированием таких трещин с помощью изменения только коэффициентов продуктивности скважины в ячейках, через которые проходит ствол скважины, является неприемлемым.

# Oil field development using fractured vertical wells - mathematical modelling aspects

**I.V. Afanaskin, A.V. Korolev**

**Abstract:** Some aspects of fractured vertical wells modelling are analyzed; the efficiency of approximate approaches is studied. Some cases of oil field development using fractured wells modelling are presented.

**Keywords:** Formation fracturing, oil field development, fractured wells modelling.

## Литература

1. А.Р.Ахметсафина, И.Р.Минниахметов, А.Х.Пергамент. Фильтрация в анизотропной трещиноватой среде // Вестник ЦКР Роснедра, №3, 2010. – с. 36-52.
2. Н.С.Бахтий. Некоторые аспекты моделирования многофазной многокомпонентной фильтрации и тестирования вычислительных алгоритмов, индуцированные программным комплексом «Техсхема». Дис. на соиск. уч. ст. канд. техн. наук. Тюмень: ФГБОУ ВПО ТГУ, 2012. – 136 с.
3. К.Ю.Богачев. Эффективное решение задач фильтрации вязкой сжимаемой многофазной многокомпонентной смеси на параллельных ЭВМ. Дис. на соиск. уч. ст. докт. физ.-мат. наук. М.: МГУ, 2012. – 201 с.
4. С.Ю.Жучков, Р.Д.Каневская. О моделировании многостадийного гидроразрыва пласта // Теория и практика применения методов увеличения нефтеотдачи пластов: Мат. IV Междунар. науч. симпозиума. – В 2 т. – Т. 1. – М.: ОАО «Всерос. нефтагаз. науч.-исслед. ин-т», 2013. – С. 163-167.
5. Р.Д.Каневская. Математическое моделирование гидродинамических процессов разработки месторождений углеводородов. М.-Ижевск: Институт компьютерных исследований, 2002. – 140 с.
6. Р.Д.Каневская. Математическое моделирование разработки месторождений нефти и газа с применением гидравлического разрыва пласта. М.: ООО «Недра-Бизнесцентр», 1999. - 212 с.
7. М.Р.Карлсон. Практическое моделирование нефтегазовых пластов. М.-Ижевск: Институт компьютерных исследований, 2012. – 944 с.
8. Р.М.Кац, Е.Р.Волгин, И.В.Афанаскин. Численное моделирование двухфазной фильтрации нефти и воды // Труды НИИСИ РАН, том 4, № 2, 2014. – с. 141-148.
9. РД 153-39.2-032-98 Методическое руководство по проектированию разработки нефтяных месторождений с применением гидроразрыва пласта (ГРП) на основе современных компьютерных технологий.
10. М.Экономидес, Р.Олини, П.Валько. Унифицированный дизайн гидроразрыва пласта: от теории к практике. М.-Ижевск: Институт компьютерных исследований, 2007. – 236 с.

# Методы моделирования на GPU снега и дождя в имитационно-тренажерных комплексах

**А.В. Мальцев**

*кандидат физико-математических наук*

**Аннотация.** При обеспечении реалистичности визуализируемых на компьютере трехмерных виртуальных сцен, которые представляют собой открытые пространства вне помещений, важным фактором является моделирование в них таких природных явлений, как дождь и снег. В работе предлагаются эффективные методы реализации этих явлений в трехмерных сценах, основанные на использовании систем частиц и параллельных вычислений на современных многоядерных GPU.

**Ключевые слова**: система частиц, снег, дождь, моделирование, визуализация, графический процессор, шейдеры, CUDA

## Введение

Широкой областью в трехмерном компьютерном моделировании является создание и визуализация виртуальных сцен, представляющих собой открытые пространства вне помещений. Такие сцены используются, в том числе, в имитационно-тренажерных комплексах управления сложными техническими системами. Эти комплексы позволяют осуществлять подготовку специалистов для работы с различными техническими средствами [1-3]. В процессе обучения на тренажерах у человека вырабатываются базовые навыки управления техникой (автомобили, самолеты, корабли, роботы и т.д.) и навыки принятия решений в различных нештатных ситуациях, что в дальнейшем сводит к минимуму затраты на испорченное дорогостоящее оборудование. Для формирования правильных навыков окружающая оператора виртуальная среда, синтезируемая на компьютере, должна как можно больше соответствовать своему реальному прототипу. При обеспечении реалистичности открытых трехмерных сцен (вне помещений), важным фактором является моделирование в них таких природных явлений, как дождь и снег.

В данной статье будут предложены методы моделирования в трехмерных виртуальных сценах снега и дождя с использованием систем частиц. Общие принципы реализации таких систем в реальном времени на современных многоядерных графических процессорах были подробно рассмотрены в [4]. Однако при моделировании интенсивного дождя и снега в трехмерных сценах, представляющих весьма протяженные открытые пространства, число частиц в таких системах может достигать количества, превышающего возможности даже самого современного GPU. Для решения этой проблемы в данной работе предлагаются методы и подходы, позволяющие ускорить расчет параметров и визуализацию систем частиц в случае крайне высокого числа элементов в них.

## 1. Методы ускорения обработки систем частиц с большим числом элементов

### 1.1. Отсечение невидимых частиц

Как было изложено в [4], после выполнения этапа расчета на GPU параметров всех частиц системы на текущий момент времени мы получаем некоторый массив $M$, в котором записаны эти параметры. Далее $M$ передается на стадию визуализации, выполняющуюся с помощью трех типов шейдеров: вершинного, геометрического и фрагментного. Вершинный шейдер не несет никакой вычислительной нагрузки, но поскольку он является обязательной частью графического конвейера, то он должен лишь передать полученные входные данные на следующую ступень – геометрический шейдер, который обеспечивает синтез полигональной модели частиц. Во фрагментном шейдере для каждого фрагмента (точки) $F$ подготовленной модели частицы определяется освещенность от источников света, размещенных в виртуальной сцене, с учетом применяемых к системе частиц материалов и текстур.

Проблема данной последовательности действий заключается в том, что модели в геометрическом шейдере строятся для всех частиц системы, в том числе, и для тех, которые находятся вне поля

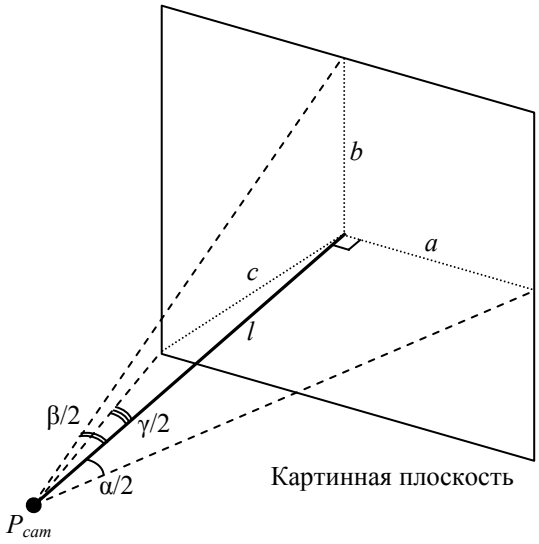



**Рис. 1.** Вычисление угла γ для конуса

зрения наблюдателя. А вот аппаратное отсечение таких невидимых частиц происходит уже после выполнения геометрического шейдера.

Чтобы ускорить этап визуализации, в данном случае необходимо реализовать собственное раннее отсечение невидимых наблюдателем частиц. Для этого будем строить в геометрическом шейдере модели только тех частиц, которые попадают в полубесконечный конус, описывающий пирамиду видимости виртуальной камеры (вершина конуса совпадает с точкой $P_{cam}$ размещения камеры). Пусть угол раствора конуса равен γ. Тогда условием принадлежности некоторой частицы с центром в точке $P$ конусу будет выполнение неравенства $\varphi \le \frac{\gamma}{2}$, где φ – угол между вектором $\boldsymbol{d} = P - P_{cam}$ и единичным вектором $\boldsymbol{v}$ направления взгляда виртуальной камеры. Отметим, что координаты всех точек и векторов должны быть представлены в одной СК. В нашем случае используется мировая система координат WCS, так как в ней производился расчет положений частиц.

Поскольку графический процессор эффективно выполняет операции скалярного произведения, то указанное выше неравенство можно заменить на

$$\left(\frac{\boldsymbol{d}}{|\boldsymbol{d}|}, \boldsymbol{v}\right) \ge \cos\frac{\gamma}{2}. \qquad (1)$$

Косинус половины угла γ раствора конуса определим, исходя из горизонтального α и вертикального β углов раствора камеры. Пусть $l$ – расстояние до картинной плоскости, $a$, $b$ и $c$ – половины соответственно ширины, высоты и диагонали прямоугольника пересечения данной плоскости с пирамидой видимости камеры (рис. 1). Тогда:

$$\cos\frac{\gamma}{2} = \frac{l}{\sqrt{l^2 + c^2}} = \frac{l}{\sqrt{l^2 + a^2 + b^2}}.$$

Так как $a = l\,\mathrm{tg}\frac{\alpha}{2}$ и $b = l\,\mathrm{tg}\frac{\beta}{2}$, то

$$\cos\frac{\gamma}{2} = \left(1 + \mathrm{tg}^2\frac{\alpha}{2} + \mathrm{tg}^2\frac{\beta}{2}\right)^{-1/2}.$$

Подставив вычисленное значение в (1), получим:

$$\left(\frac{\boldsymbol{d}}{|\boldsymbol{d}|}, \boldsymbol{v}\right) \ge \left(1 + \mathrm{tg}^2\frac{\alpha}{2} + \mathrm{tg}^2\frac{\beta}{2}\right)^{-1/2}. \qquad (2)$$

Правая часть этого неравенства вычисляется один раз перед выполнением этапа синтеза и визуализации системы частиц, левая – в геометрическом шейдере для каждой частицы. Построение полигональных моделей частиц, для которых не соблюдается неравенство (2), не производится, а их обработка прекращается.

### 1.2. Ограничение области моделирования

Если необходимо визуализировать виртуальную сцену, представляющую очень протяженное открытое пространство, с моделированием в ней атмосферных осадков в виде интенсивного дождя или снега, то количество элементов в системе частиц может стать чрезмерно высоким, вплоть до десятков и сотен миллионов. При этом сохранить визуализацию такой сцены в масштабе реального времени, учитывая современные параметры GPU, будет весьма затруднительно.

Решить эту проблему можно, основываясь на том факте, что капли дождя (как и снежинки) имеют маленький размер. Поэтому, начиная с некоторого небольшого расстояния $l$ от наблюдателя (зависящего от размера частиц и параметров виртуальной камеры), они будут занимать в кадре порядка одного пиксела. Понятно, что такие частицы не будут заметны на изображении, и тратить вычислительные мощности на их обработку неэффективно. В данном случае целесообразно ограничить область моделирования атмосферных осадков, например, поместив эмиттер системы частиц размером $2l$ x $2l$ на расстоянии не менее $l$ над наблюдателем так, чтобы точка

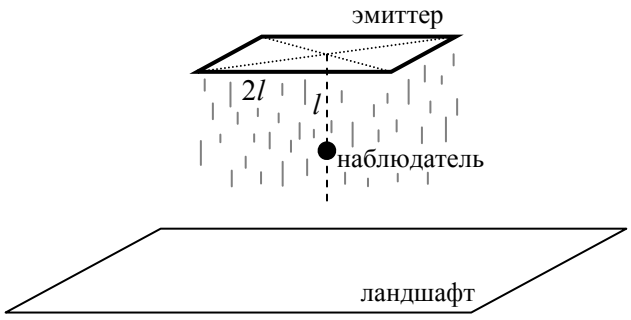



**Рис. 2.** Ограничение области моделирования осадков

расположения самого наблюдателя находилась на оси перпендикулярной эмиттеру и проходящей через его центр (рис. 2). Значение $l$ в среднем составляет 50-100 метров. При движении виртуальной камеры по ландшафту в горизонтальной плоскости будем пересчитывать положение эмиттера и всех уже сгенерированных частиц в мировой системе координат WCS виртуального пространства, сдвигая их координаты аналогичным образом. Тогда наблюдатель всегда будет находиться в области дождя или снега.

Единственным недостатком данного подхода является то, что при достижении большой скорости движения наблюдателя становится слегка заметным совместное перемещение виртуальной камеры и области моделирования осадков. Но при любых поворотах камеры, а также смещениях ее в вертикальном направлении визуально все остается, как и в случае со статическим эмиттером, покрывающим весь ландшафт целиком.

### 1.3. Частота изменения данных VBO буфера вывода

При реализации стадии расчета текущих параметров частиц с использованием архитектуры параллельных вычислений CUDA результирующий массив данных находится в CUDA-контексте. Для осуществления визуализации системы частиц в таком случае возникает необходимость организации доступа к этому массиву из контекста OpenGL. Обычно данная проблема решается с использованием механизма *CUDA OpenGL interoperability* [5], который позволяет отобразить данные из памяти CUDA в вершинный буфер VBO [6], доступный в шейдерах. Однако чем больше элементов содержит система частиц, тем дольше процесс отображения памяти. Поэтому целесообразно производить эту операцию не каждый кадр, а ограничить частоту обновления буфера VBO до значения, позволяющего обеспечить рендеринг плавной динамики частиц. Так при визуализации трехмерной сцены в масштабе реального времени вполне достаточно производить отображение данных из CUDA в VBO с частотой 25 раз в секунду.

## 2. Моделирование дождя

### 2.1. Модель капли дождя

Моделирование капель дождя выполним с помощью полупрозрачных частиц, имеющих форму тетраэдра, в основании которого лежит равносторонний треугольник с центром в $P'$ (рис. 3) - рассчитанной точке расположения частицы [4]. Пусть ось $V_0P'$ тетраэдра параллельна вектору $\boldsymbol{v}'$ текущей скорости частицы, а высота вычисляется как $h = ks|\boldsymbol{v}'|$, где $s$ – размер частицы,

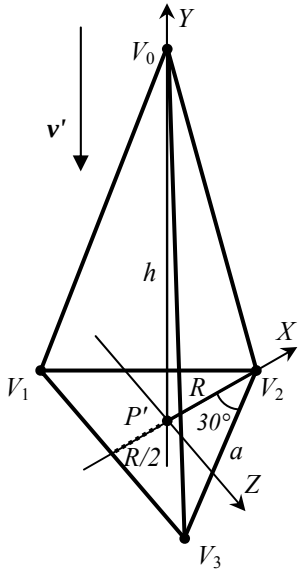



**Рис. 3.** Модель капли дождя

устанавливаемый дизайнером при подготовке виртуальной сцены, $k$ – коэффициент, регулирующий зависимость вычисляемого размера частицы от модуля ее скорости $\boldsymbol{v}'$. Для придания формируемой капле вытянутой формы зададим длину $a$ стороны основания равной $0.1h$.

Единичные направляющие векторы $\boldsymbol{e_x}$, $\boldsymbol{e_y}$, $\boldsymbol{e_z}$ осей $X$, $Y$, $Z$ локальной системы координат (с центром в $P'$) тетраэдра в СК WCS вычислим по формулам:

$$\boldsymbol{e}_y = -\frac{\boldsymbol{v}'}{|\boldsymbol{v}'|},\ \boldsymbol{e}_z = \frac{[\boldsymbol{X}_{WCS}, \boldsymbol{e}_y]}{\|[\boldsymbol{X}_{WCS}, \boldsymbol{e}_y]\|},\ \boldsymbol{e}_x = \frac{[\boldsymbol{e}_y, \boldsymbol{e}_z]}{\|[\boldsymbol{e}_y, \boldsymbol{e}_z]\|},$$

где $X_{WCS}$ – направляющий вектор (1,0,0) оси абсцисс системы WCS. Расположим одну из вершин ($V_2$) тетраэдра на оси $X$ его СК. Тогда вершины тетраэдра будут определяться как

$$V_0 = P' + h\boldsymbol{e}_y, \quad V_1 = P' - (R/2)\boldsymbol{e}_x - (a/2)\,\boldsymbol{e}_z,$$
$$V_2 = P' + R\boldsymbol{e}_x, \quad V_3 = V_1 + a\boldsymbol{e}_z,$$

где $R = \frac{\sqrt{3}}{3}a$ – радиус описанной вокруг основания тетраэдра окружности. Нормали в этих вершинах рассчитаем по формулам:

$$\boldsymbol{N}_0 = \boldsymbol{e}_y, \qquad \boldsymbol{N}_1 = \frac{V_1 - P'}{|V_1 - P'|} + \frac{V_1 - V_0}{|V_1 - V_0|},$$

$$\boldsymbol{N}_2 = \boldsymbol{e}_x + \frac{V_2 - V_0}{|V_2 - V_0|}, \quad \boldsymbol{N}_3 = \frac{V_3 - P'}{|V_3 - P'|} + \frac{V_3 - V_0}{|V_3 - V_0|}.$$

Отметим, что прежде чем передавать положения вершин на выход геометрического шейдера, необходимо умножить их на матрицу, равную произведению $M_{proj} \cdot M_V$, где $M_{proj}$ – матрица заданного для виртуальной камеры проекционного преобразования, $M_V$ – матрица перехода из мировой СК WCS в видовую СК VCS.

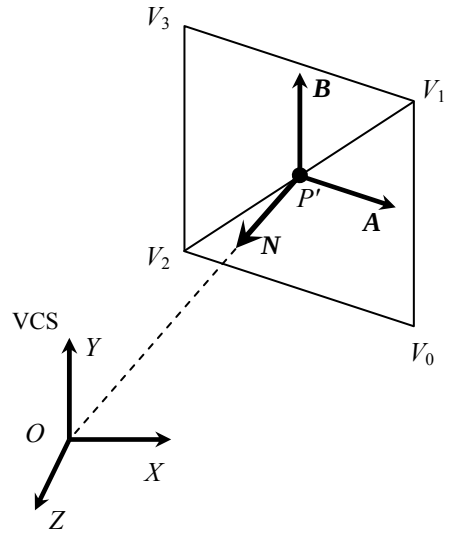



**Рис. 4.** Построение «спрайта» для снега

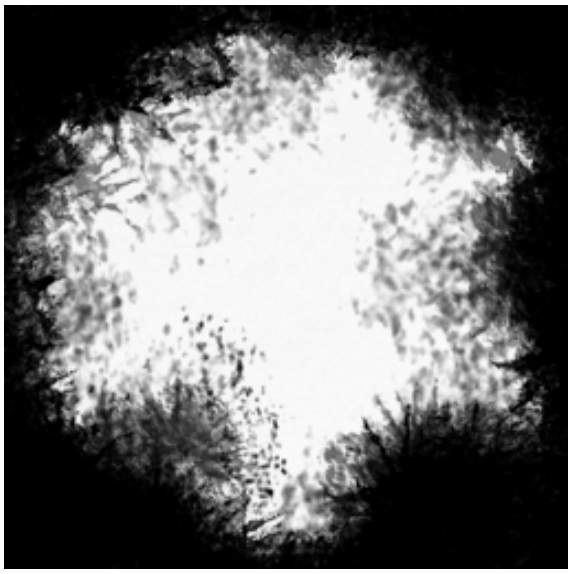


**Рис. 5.** Текстура снежинки

### 2.2. Визуализация полупрозрачных частиц

Поскольку капли дождя обладают прозрачностью, то, учитывая принципы работы технологии смешивания в графическом конвейере (по значению альфа канала), для правильного их отображения на синтезируемом кадре необходимо визуализировать трехмерные модели частиц в порядке уменьшения расстояния (глубины) от наблюдателя. Для этого кроме массива $M$ (он подробно описан в [4]) параметров частиц, введем массив индексов $I$ и массив глубин $D$ частиц. Число элементов в этих массивах должно совпадать с количеством $n$ ячеек в $M$ и, соответственно, с максимальным числом частиц в системе. Пусть после вычисления текущих параметров частиц и записи их в $M$, каждый элемент $I[k]$, где $k \in [0, n\text{-}1]$, содержит индекс $k$, указывающий на ячейку $M[k]$, а $D[k]$ – глубину рассчитанной точки $P'$ частицы из $M[k]$ относительно виртуальной камеры.

Для сортировки частиц по глубине от наблюдателя воспользуемся функцией *sort_by_key* параллельной сортировки массивов на GPU из библиотеки *Thrust*, включенной NVIDIA в пакет средств разработки CUDA-программ. На вход функции подаются массив ключей и массив данных, которые содержат одинаковое количество элементов. При этом считается, что оба массива взаимосвязаны, а именно: $i$-ый элемент первого массива образует с $i$-ым элементом второго массива пару (ключ, информация). Функция производит сортировку ключевого массива и аналогично изменяет индексы элементов массива данных. На выходе имеем пару массивов, отсортированных по ключам.

В нашем случае используем массив $D$ в качестве ключевого, а массив $I$ – в качестве массива данных. В результате применения функции *sort_by_key* получим массив $I$, в котором последовательно записаны индексы частиц (из $M$), отсортированных в нужном нам порядке, т.е. по глубине от наблюдателя. Далее произведем визуализацию частиц в той последовательности, в которой они записаны в $I$.

## 3. Моделирование снега

При синтезе снежинок в виртуальных сценах будем использовать модель частицы в виде текстурированного «спрайта» – квадрата, постоянно повернутого лицевой стороной к виртуальной камере. Для каждой снежинки сформируем в геометрическом шейдере квадрат так, чтобы его центр находился в точке $P'$ размещения соответствующей частицы, а нормаль $\boldsymbol{N}$ к поверхности квадрата из этой точки смотрела в начало $O$ видовой системы координат VCS (рис. 4). Тогда $\boldsymbol{N} = | O_{WCS} - P' |$, где $O_{WCS}$ – точка $O$, координаты которой представлены в мировой СК.

Генерируемый спрайт включает два треугольных полигона с вершинами $V_0$, $V_1$, $V_2$ и $V_1$, $V_2$, $V_3$, которые для оптимизации работы графического конвейера целесообразно представлять как треугольный стрип $V_0V_1V_2V_3$. Чтобы найти координаты вершин, вычислим вначале пару перпендикулярных векторов $\boldsymbol{A}$ и $\boldsymbol{B}$ таких, что $\boldsymbol{A} \parallel V_0V_2$, $\boldsymbol{B} \parallel V_0V_1$:

$$\boldsymbol{A} = [\boldsymbol{Y}_{\text{WCS}}, \boldsymbol{N}] \;/\; |\, [\boldsymbol{Y}_{\text{WCS}}, \boldsymbol{N}] \,|, \quad \boldsymbol{B} = [\boldsymbol{N}, \boldsymbol{A}] \;/\; |\, [\boldsymbol{N}, \boldsymbol{A}] \,|,$$

где $\boldsymbol{Y}_{\text{WCS}}$ – вектор $Y$ из базиса системы VCS с координатами в СК WCS, а квадратные скобки обозначают векторное произведение. Тогда, если $s$ – размер стороны спрайта, то:

$$V_0 = P' + 0.5s(\boldsymbol{A} - \boldsymbol{B}), \qquad V_1 = V_0 + s\boldsymbol{B},$$
$$V_2 = V_0 - s\boldsymbol{A}, \qquad V_3 = V_1 - s\boldsymbol{A}.$$

Отметим, что, как и в случае с тетраэдром (п. 2.1), перед передачей положения вершин на выход геометрического шейдера, необходимо умножить их на произведение матриц $M_{proj} \cdot M_V$ .

На созданный спрайт во фрагментном шейдере наложим текстуру прозрачности, изображенную на рисунке 5, задав основной цвет материала белым. Текстура прозрачности – это двумерная текстура в градациях серого цвета, в которой белые участки указывают на полную непрозрачность, а черные – на абсолютную прозрачность. Все остальные оттенки серого цвета в текстуре соответствуют разной степени полупрозрачности (чем ближе цвет к черному, тем больше прозрачность и наоборот). Текстурные координаты для вершин $V_0$, $V_1$, $V_2$ и $V_3$ нашего спрайта задаются соответственно как $T_0$ (1.0, 0.0), $T_1$ (1.0, 1.0), $T_2$ (0.0, 0.0) и $T_3$ (0.0, 1.0).

Данная работа выполняется при поддержке РФФИ, грант № 13-07-00674.

# Methods for GPU modeling of snow and rain in simulation-training complexes

**A.V. Maltsev**

**Abstract.** To provide a realism of three-dimensional virtual scenes rendered by a computer and represented open (outdoor) spaces, the important factor is a simulation of natural phenomena in them such as rain and snow. At this article methods of such phenomena realization at 3D scenes are proposed, based on using particle systems and parallel computations by modern multi-core GPU.

**Keywords**: particle system, snow, rain, simulation, visualization, graphics processor, shaders, CUDA

## Литература

1. М.В.Михайлюк. Видеотренажерный комплекс управления роботами и манипуляторами // Труды Международного симпозиума «Инновационные технологии в исследовании окружающей среды», Ларнака-Москва, 2013, с. 84-85.

2. М.В.Михайлюк, М.А.Торгашев. Использование технологий виртуальной реальности для моделирования безопасного управления антропоморфными робототехническими средствами // Труды XXI Международной конференции «Проблемы управления безопасностью сложных систем», Москва, 2013, с. 290-293.

3. М.В.Михайлюк, В.И.Брагин. Технологии виртуальной реальности в имитационно-тренажерных комплексах подготовки космонавтов // Пилотируемые полеты в космос, № 2, 2013, с. 82-93.

4. А.В.Мальцев. Реализация системы частиц в реальном времени на GPU // Программные продукты и системы, № 4, 2014, с. 57-62.

5. CUDA C Programming Guide. NVIDIA Corporation, 2015. URL: http:// docs.nvidia.com/cuda/ pdf/CUDA_C_Programming_Guide.pdf (дата обращения: 01.09.2015).

6. А.В.Боресков. Расширения OpenGL. СПб.: БХВ-Петербург, 2005, 667 с.

# Моделирование копирующего режима управления антропоморфным роботом в виртуальной среде

**М.А. Торгашев**

*кандидат физико-математических наук*

**Аннотация:** В статье предлагается технология копирующего режима управления антропоморфными роботами в виртуальной среде. Она включает реализацию схемы ПИД регулирования двигателей в системе управления и расчет динамики виртуальных объектов с учетом коллизий и внешних сил. Технология внедрена в комплексе виртуального моделирования робототехнических средств и апробирована на примере моделирования нескольких операций внекорабельной и внутрикорабельной деятельности на космической станции.

**Ключевые слова**: антропоморфные робототехнические системы, виртуальное моделирование, копирующий режим управления, ПИД регулирование.

## 1. Введение

Антропоморфные робототехнические системы (далее АРТС) имеют кинематическую схему, близкую к человеческой. Важное преимущество таких систем состоит в том, что они позволяют эффективно решать те задачи, которые изначально ориентированы на человека, например, выполнение операций с рабочими инструментами, адаптированными под человеческую руку. Такие АРТС могут успешно использоваться в космической отрасли при выполнении внекорабельной и внутрикорабельной деятельности [1]. Для экспериментальной отработки операций, выполняемых антропоморфным роботом, могут найти эффективное применение технологии виртуального окружения. Их использование позволяет проводить исследования на виртуальных интерактивных 3D моделях без проведения натурных экспериментов и построения дорогостоящих макетов. При этом на текущем уровне развития технологий можно обеспечить высокую достоверность виртуального моделировании как в части визуальных, так и динамических характеристик.

Для управления АРТС существует несколько различных методов: управление с помощью виртуальных пультов, командный режим и копирующий режим. Рассмотрим эти режимы управления более подробно.

Наиболее простая технология заключается в управлении АРТС от виртуального пульта [2]. Сам виртуальный пульт представляет собой схематическое изображение робота, на котором задано расположение осей вращения его двигателей (рис. 1). Пользователь с помощью щелчка мыши на соответствующих элементах может выбрать один или несколько шарниров, после чего с помощью перемещения виртуального элемента управления типа «Джойстик» управлять движением робота в данных суставах. Преимуществом этой технологии является простота реализации и наглядность процесса. Однако значительным недостатком является неудобство управления и невозможность задания сложных траекторий движения, реализуемых с помощью одновременного вращения нескольких шарниров.

В командном режиме управление осуществляется с помощью набора команд, каждая из которых задает движение робота по некоторой заранее определенной траектории. Этот режим предполагает, что робот заранее запрограммирован для выполнения нескольких операций. Достоинством данного способа является то, что он удобен и вполне достаточен для таких применений, где использование робота предопределено. Недостатком же является очевидное отсутствие гибкости – расширение списка выполняемых операций требует переделки системы управления, этого же требует изменение конструкции робота или изменение внешней обстановки.

Наиболее естественным режимом управления антропоморфными роботами является копирующий. В этом случае управление роботом выполняет человек с помощью специального костюма – экзоскелета (рис. 2). Встроенные в экзоскелет датчики фиксируют значения углов, задающих положение головы, торса, рук и ног оператора (используются также неполные экзоске-

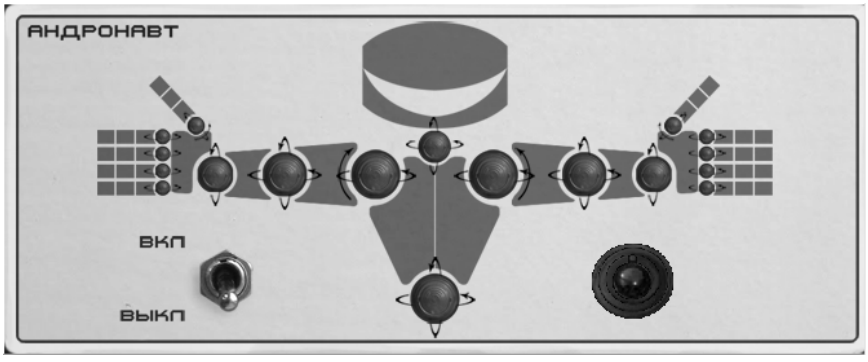



**Рис. 1.** Виртуальный пульт управления роботом

**Рис. 2.** Оператор в экзоскелете

леты, которые фиксируют, например, только положение рук). Зафиксированное экзоскелетом движение оператора передается роботу, который их повторяет (копирует). Отметим, что передача данных может осуществляться по беспроводным каналам связи. Оператор при этом может находиться в безопасном месте, удаленном от агрессивной и опасной среды, в которой может использоваться робот. Таким образом, копирующий режим является одним из наиболее естественных и удобных для управления АРТС. При его моделировании в виртуальной среде необходимо обеспечить точное соответствие поведения виртуальной модели ее реальному прототипу, как в части управления, так и в части динамических и визуальных характеристик робота и окружающей среды. Рассмотрим предлагаемую схему моделирования копирующего режима управления АРТС в виртуальной среде (рис. 3).

## 2. Схема моделирования копирующего режима

Модель робота и окружающей обстановки создается в системе трехмерного моделирования, в которую дополнительно интегрируется подключаемый модуль (plug-in). Этот модуль позволяет создавать специальные объекты робототехнических систем, в частности двигатели, шарниры, аппроксимирующие контейнеры для расчета коллизий, а также задавать массо-инерционные характеристики объектов и параметры трения. Файл, содержащий визуальные характеристики сцены, загружается в подсистему визуализации, а файл с динамическими параметрами – в подсистему динамики. В схему комплекса также входит подсистема управления, где производится расчет функциональных схем управления. Сами схемы создаются в специальном редакторе функциональных схем.

В процессе тренировки выполняется передача данных от экзоскелета в подсистему управления в рамках локальной вычислительной сети по сетевому протоколу UDP, входящему в семейство TCP/IP. Данные передаются по согласованному информационному протоколу, регламентирующему их формат. Для каждого двигателя робота задается его имя и абсолютный угол вращения от начального положения локальной системы координат двигателя. Управляющие сигналы от экзоскелета поступают в подсистему управления, где в соответствии с текущими углами поворота шарниров экзоскелета вычисляются управляющие напряжения на двигатели робота, необходимые для повторения роботом движения оператора. Вычисленные напряжения передаются в подсистему динамики, где вычисляются новые матрицы объектов, задающие их положение и ориентацию. Результирующие матрицы поступают в подсистему визуализации, которая синтезирует в моно или стерео режиме изображение робота с новыми положениями его звеньев, а также окружающую обстановку. Ориентируясь на полученное изображение, оператор выполняет необходимые действия, которые в реальном масштабе времени (т.е. с частотой не менее 25 кадров в секунду) отображаются либо на мониторе, либо на большом экране или в шлеме виртуальной реальности. Таким образом, у оператора возникает ощущение непрерывного движения модели робота, повторяющей движения оператора. Далее рассмотрим отдельные компоненты комплекса моделирования АРТС.

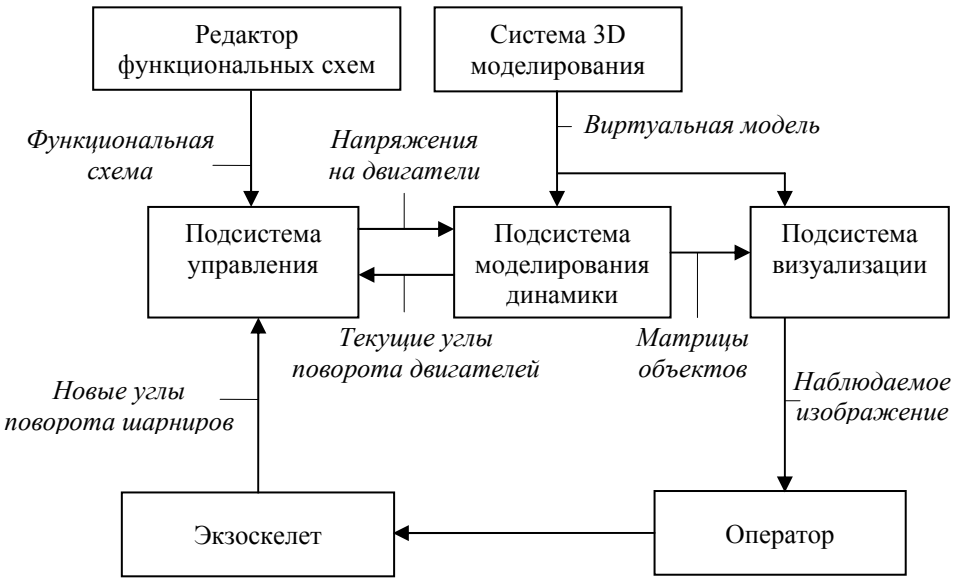



**Рис. 3.** Схема моделирования копирующего режима

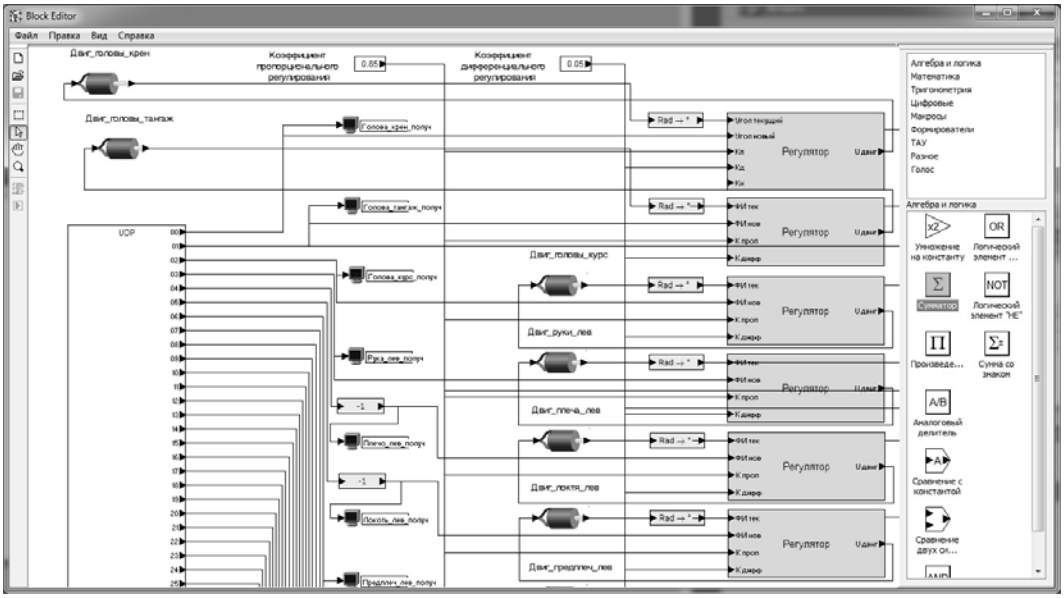



**Рис. 4.** Визуальный редактор функциональных схем

## 3. Подсистема управления

В реальных АРТС для реализации копирующего режима управления используются специальные регуляторы, либо встроенные в исполнительные устройства робота, либо реализованные отдельным блоком. Задачей каждого из регуляторов является сведение текущего рассогласования (разницы между значением, поступившим от экзоскелета, и текущим значением в шарнире) к минимуму за минимальное время. Для решения подобной задачи хорошо себя зарекомендовали и широко используются так называемые ПИД (пропорционально – интегрально - дифференциальные) регуляторы. Управляющий сигнал на исполнительное устройство формируется как взвешенная сумма пропорциональной, интегральной и дифференциальной составляющих с некоторыми коэффициентами. Коэффициенты регулятора подбираются исходя из характеристик двигателя, возможной нагрузки и максимально допустимой величины задержки. При моделировании АРТС в виртуальной среде требуется полностью симитировать поведение виртуальной модели АРТС, аналогичное реальному роботу, что требует реализации схемы ПИД регулирования.

Для построения и моделирования сложных систем управления в ФГУ ФНЦ НИИСИ РАН был разработан редактор функциональных схем [4], имеющий широкий набор функциональных блоков и позволяющий моделировать сложные схемы управления для реальных механических систем. Функциональные схемы создаются из набора блоков, имеющих определенное число входов и выходов, которые можно соединять связями. Пример построения функциональной схемы в редакторе показан на рисунке 4.

Рассмотрим функциональную схему для реализации копирующего режима на примере одного из двигателей (рис. 5). Для съема данных с экзоскелета используется специальный блок, который принимает данные по согласованному информационному протоколу, содержащему значения углов поворота всех шарниров.

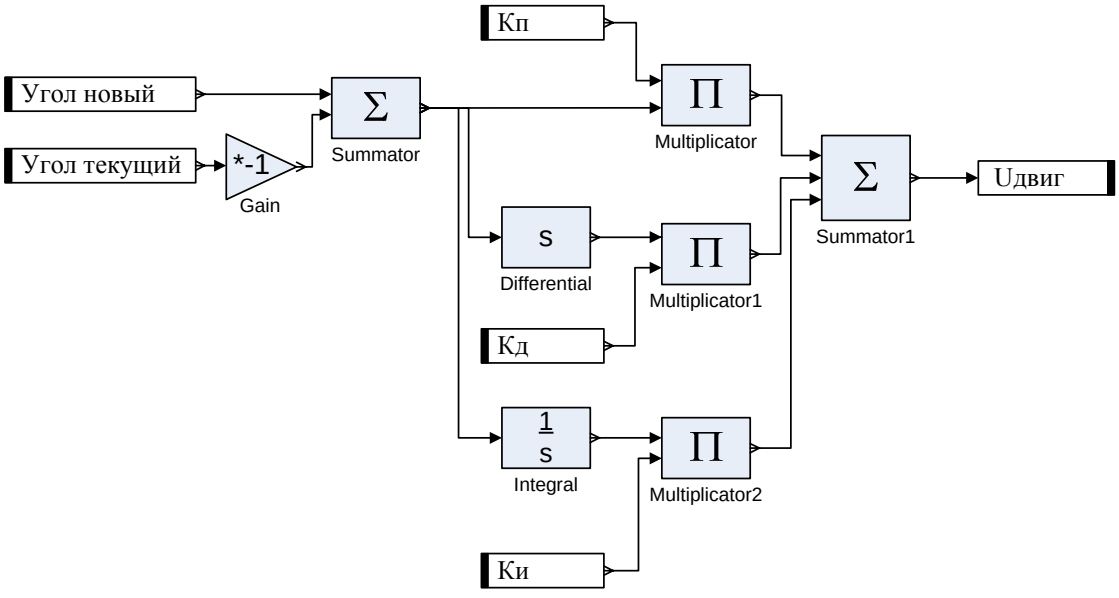



**Рис. 5.** Функциональная схема ПИД регулирования

Принятые значения поступают на вход соответствующих регуляторов. Для реализации требуемой схемы управления используются следующие функциональные блоки: сумматор, умножение, константа, дифференциатор и интегратор.

Значения текущего угла поворота двигателя передаются в подсистему управления из подсистемы моделирования динамики. Новое значение угла поворота, к которому необходимо привести двигатель, поступает от экзоскелета через подключаемый модуль. Значение рассогласования вычисляется с помощью блоков умножения на константу (-1) и сумматора. Далее значение рассогласования поступает на входы блоков дифференциатора и интегратора. Значения пропорционального, дифференциального и интегрального регулирования умножаются на соответствующие коэффициенты Кп, Кд и Ки, а затем складываются с помощью блока «Сумматор». Результирующее значение интерпретируется как напряжение и передается на вход двигателя в подсистему расчета динамики.

## 4. Подсистема расчета динамики робота и объектов внешней среды

Для повышения достоверности моделирования очень важно обеспечить адекватное динамическое поведение виртуальной модели АРТС и ее взаимодействие с окружающей обстановкой. В первую очередь сюда относится точное повторение динамики двигателей и шарниров робота, а также расчет коллизий (столкновений) робота с другими объектами.

Моделирование двигателей АРТС было реализовано на основе модели вращательных двигателей постоянного тока [5]. Такие двигатели обладает следующим набором характеристик:

- скорость холостого хода;
- постоянная времени;
- пусковой момент;
- максимальный момент трения вала.

Суммарный момент силы двигателя вычисляется из трех составляющих: момента силы двигателя без внешней нагрузки, момента силы, создаваемого нагрузкой двигателя, а также текущего момента трения вала. Текущий момент силы двигателя без нагрузки зависит от текущей угловой скорости вращения и текущего входного напряжения. Эта зависимость называется механической характеристикой двигателя, которая моделируется с помощью линейного приближения.

Одной из важнейших задач при расчете динамики виртуальных объектов является расчет коллизий, то есть ситуаций, при которых возникают столкновения объектов между собой. Эта задача включает в себя две составляющих – определение коллизий, включая определение самого факта столкновения и вычисление его параметров, и их обработку, то есть вычисление результирующий динамики тел после удара. Для определений столкновений между объектами в вычислительной динамике широко распространена технология упрощенного описания тел с помощью так называемых аппроксимирующих контейнеров. Вместо реальной формы объекта используется его аппроксимация в виде набора контейнеров простой формы. Это позволяет существенно ускорить расчет столкновений, что особенно актуально для сцен, содержащих большое количество объектов. В разработанной в ФГУ ФНЦ НИИСИ РАН подсистеме расчета динамики используются аппроксимирующие контейнеры типа «сфера» и «прямоугольный параллелепипед» (бокс). Как результат алгоритм определения коллизии [6] возвращает тройку параметров: точку пересечения, нормаль в точке контакта и глубину проникновения.

Разрешением коллизий на основании полученных параметров занимается модуль динамической библиотеки, называемый «солвером» (от аглийского to solve – решать). В рамках исследований был разработан собственный «солвер», основанный на методе последовательных импульсов [6]. Его суть состоит в том, что на каждый объект накладываются ограничения на скорости движения вдоль определенных осей с учетом его массо-инерционных характеристик и параметров трения. Ограничения решаются последовательно и независимо друг от друга с помощью вычисления таких импульсов, которые необходимо приложить к телам, чтобы выполнить текущее ограничение. По мере повторения такого процесса для всей системы ограничений получаемые скорости объектов приближаются к искомым значениям, удовлетворяющим решению всей системы (такой процесс называют сходящимся). Условием завершения итерационного процесса является обеспечение заданной погрешности для всех изменений скоростей, вносимых текущими применяемыми импульсами.

Важной составляющей задачи расчета динамики является моделирование систем тел, связанных шарнирами. Шарниры, соединяющие тела, могут иметь несколько типов: сферические, осевые, призматические и т.д. Отдельную сложность представляют замкнутые кинематические цепи, в которых любое звено входит в состав как минимум двух кинематических пар. Для расчета динамики связанных тел формируются ограничения на их относительное движение. Разрешение сформированных ограничений также выполняется с помощью метода последовательных импульсов [7]. Для улучшения сходимости применяется технология «горячего старта», использующая высокую временную когерентность между шагами моделирования.

Отдельной подзадачей при моделировании копирующего режима является реализация силомоментного очувствления. Весьма желательно, чтобы оператор чувствовал отдачу и давление при взаимодействии с объектами окружающей виртуальной обстановки. Такое очувствление позволяет значительно повысить точность управления и минимизировать возможные ошибки. Для реализации такого управления требуется реализация обратной связи от системы моделирования динамики к экзоскелету. Необходимо рассчитать текущие силомоментные характеристики, воздействующие на шарниры робота, передать их на силовой каркас экзоскелета и обеспечить их отработку. Необходимые силомоментные характеристики рассчитываются в системе моделирования динамики и могут быть переданы на экзоскелет по согласованному информационному протоколу.

## 5. Подсистема визуализации виртуальных сцен

Подсистема визуализации предназначена для высококачественного синтеза изображений виртуальной сцены, содержащей робота и окружающую визуальную обстановку, в режиме реального времени. В ФГУ ФНЦ НИИСИ РАН разработана собственная система визуализации «GLView» [8], обладающая широким списком возможностей. В число ключевых особенностей системы, наиболее актуальных для задачи моделирования робототехнических средств, входят синтез теней, поддержка стереорежима, многопортовый режим наблюдения и синтез специальных эффектов.

Моделирование теней и поддержка стереорежима особенно важны для точной оценки расстояний до объектов и их взаимного расположения, что позволяет добиться необходимой точности и повысить безопасность управления. Система визуализации поддерживает стереорежим в нескольких форматах – для анаглифических очков, для затворных очков, для шлемов виртуальной реальности и для больших экранов с использованием поляризационной технологии. Режим многопортового отображения позволяет одновременно наблюдать на экране изображение с нескольких ракурсов (например, с камер, установленных на голове и руке робота), что также служит цели повышения точности и надежности управления.

Система также позволяет имитировать специальные эффекты, которые наблюдаются в реальности. В число поддерживаемых эффектов входят: черно/белый режим для имитации монохромных средств наблюдения, белый шум и рассинхронизация изображения для имитации помех в каналах вязи, а также расфокусировка и ограниченная глубина резкости, возникающие в оптической системе объективов.

В системе визуализации также реализованы сервисные возможности по записи и воспроизведению тренировок. Это позволяет по команде выполнить полную запись тренировки, а затем в ходе анализа и изучения воспроизвести ее в точном соответствии с ходом процесса в момент записи. Из файлов записи также могут быть синтезированы высококачественные видеоролики высокого разрешения для демонстрационных целей.

## 6. Заключение

В исследовании предложена технология моделирования копирующего режима управления виртуальной моделью антропоморфного робота от экзоскелета. Движения оператора передаются в подсистему управления, где аналогично с реальной схемой управления робота происходит вычисление входных напряжений двигателей с помощью схемы ПИД регулирования. В подсистеме динамики выполняется расчет динамической модели в соответствии с управляющими напряжениями и с учетом действующих на шарниры робота сил и моментов, в том числе возникающих при коллизиях звеньев между собой и с объектами окружающей обстановки. Предложенная схема позволяет достоверно имитировать выполнение сложных операций, включая захват и перемещение объектов, нажатие на кнопки и т.д. Технология допускает реализацию силомоментной обратной связи, когда оператор в экзоскелете чувствует отдачу при взаимодействии с объектами внешней среды. Разработанная технология была реализована в рамках программного комплекса моделирования робототехнических средств и была успешно апробирована для реализации копирующего режима управления антропоморфным роботом SAR-401 (рис. 6).

С помощью построенного комплекса было смоделировано выполнение несколько возможных операций перспективного антропоморфного робота SAR-401 на космической станции:

- Инспектирование внешней поверхности МКС (робот закреплен на манипуляторе ERA, что позволяет подвести его к интересующей точке поверхности) (рис. 7).
- Извлечение научного блока «Биориск» на внешней поверхности модуля СО-1 «Пирс», входящего в состав МКС (рис. 8).
- Работа с пультом управления системы шлюзования в интерьере модуля СО-1 «Пирс» (нажатие на кнопки) (рис. 9).

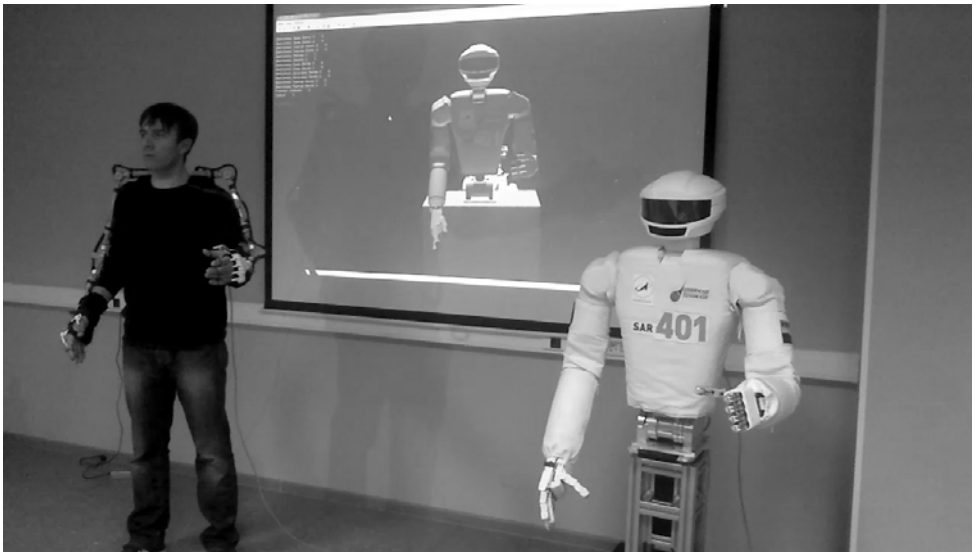



**Рис. 6.** Управление с помощью экзоскелета реальной и виртуальной моделями АРТС

Построенная система виртуального моделирования позволила исследовать возможности реального использования антропоморфного робота в космосе и отработать на виртуальной модели возможный перечень операций по внекорабельной и внутрикорабельной деятельности.

Данная работа выполняется при поддержке РФФИ (грант № 13-07-00708).

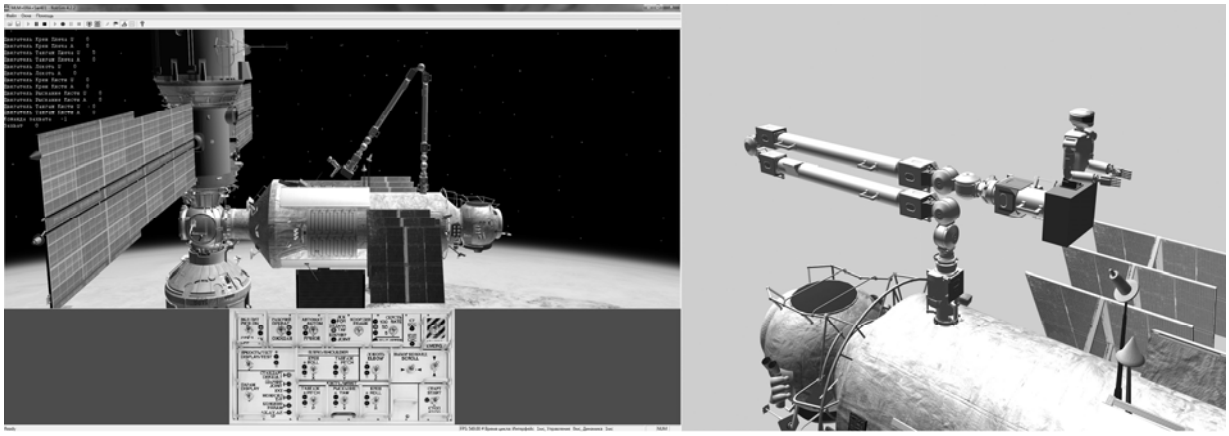


**Рис. 7.** Операция инспектирования внешней поверхности МКС

**Рис. 8.** Операция извлечения научного блока «Биориск»

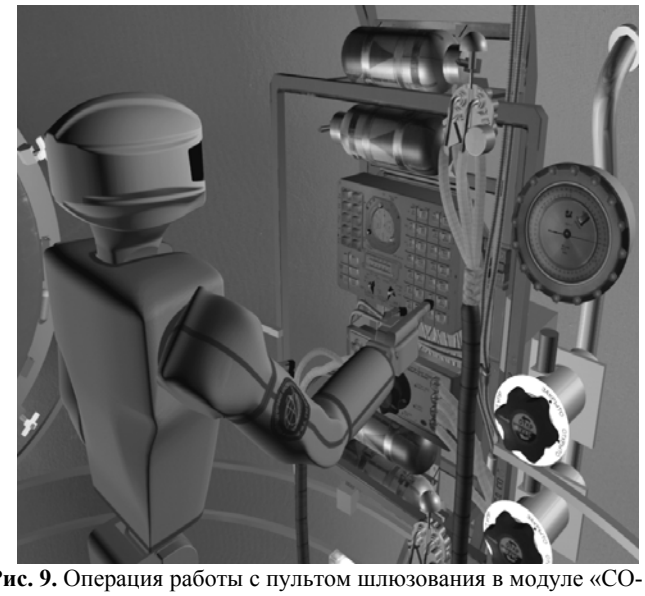


**Рис. 9.** Операция работы с пультом шлюзования в модуле «СО-1»

# Modeling of anthropomorphic robots' copying control in a virtual environment

**M.A.Torgashev**

**Abstract:** The article describes the implementation of the copying operation mode of anthropomorphic robots in a virtual environment. The technology includes a scheme of motor's PID regulation and the calculation of the virtual objects' dynamics including collisions and external forces. The technology is implemented in a complex of robotic tools' virtual simulation and approved on the example of several operations of extra- and intra-vehicular activity on the space station.

**Keywords**: anthropomorphic robotic systems, virtual simulation, copying mode of control, PID regulation.

## Литература

1. Б.В. Бурдин, М.В. Михайлюк, И.Г. Сохин, М.А. Торгашев. Использование виртуальных 3D-моделей для экспериментальной отработки бортовых полетных операций, выполняемых с помощью антропоморфных роботов // Робототехника и техническая кибернетика. - № 1, 2013. С. 42 - 46
2. М.В. Михайлюк, М.А. Торгашев. Моделирование и визуализация трехмерных виртуальных пультов управления в тренажерах // Научная визуализация. - Том 6, № 4, 2014. С. 50 - 60. http://sv-journal.org/2014-4/012139.html?lang=ru
3. М.В. Михайлюк, М.А. Торгашев. Визуализация динамики объектов управления в реальном времени. Научная визуализация. - № 5, 2014. С. 69 - 80. http://sv-journal.org/2014-5/index2139.html?lang=ru
4. М. В. Михайлюк, М. А. Торгашев. Визуальный редактор и модуль расчета функциональных схем для имитационно-тренажерных комплексов // Программные продукты и системы. - № 4, 2014. С. 10 - 15
5. М.В. Михайлюк, А.М. Трушин. Моделирование динамики двигателей в виртуальных сценах // Моделирование и визуализация. Многопроцессорные системы. Инструментальные средства разработки ПО. Сборник статей под редакцией академика РАН В.Б. Бетелина. НИИСИ РАН. 2010. С. 48 - 55
6. А.М. Трушин. Обработка коллизий виртуальных объектов с помощью метода последовательных импульсов // Труды НИИСИ РАН. - Том. 4, № 2, 2014. С. 95 - 105
7. М.В. Михайлюк, Е.В. Страшнов. Моделирование системы связанных тел методом последовательных импульсов // Труды НИИСИ РАН. - Том. 4, № 2, 2014. С. 52 - 60
8. М.В. Михайлюк, М.А. Торгашев. Система «GLVIEW» визуализации для моделирующих комплексов и систем виртуальной реальности // Вестник Российской академии естественных наук. - №2, 2011. С. 20 - 28

# Метод начального тестирования RapidIO систем

**Г.А. Лавринов**

**Аннотация:** в статье рассматривается метод начального тестирования интерфейса RapidIO в многопроцессорных системах на основе протокола управления тестированием и обменом данных, использующего служебные операции. Приведена схема организации тестируемой системы, которая позволяет снижать затраты на дополнительное оборудование. Схематично представлен протокол управления тестированием и обменом данными на базе служебных операций RapidIO. Описана синхронизация распределенного тестирования многопроцессорных систем. Приведены результаты тестирования RapidIO систем с применением рассматриваемого метода.

**Ключевые слова:** RapidIO, тестирование, многопроцессорная система, метод, синхронизация.

Одним из этапов подготовки к полноценной работе многопроцессорной системы является первоначальное тестирование, главная задача которого заключается в функциональной проверке всех программно доступных компонентов системы при первом включении. Одним из таких компонентов является коммуникационный интерфейс. В статье рассматриваются системы на базе коммуникационного интерфейса RapidIO [1, 2]. Основным способом тестирования интерфейса RapidIO на оконечном устройстве (ОУ) является проверка приема/передачи пакетов с помощью петли (loopback), которая настраивается на подключенном коммутаторе или при помощи тестовой заглушки. Исходя из этого, предлагается рассмотреть метод первоначального тестирования устройств RapidIO в системе с возможными вариантами настройки петель.

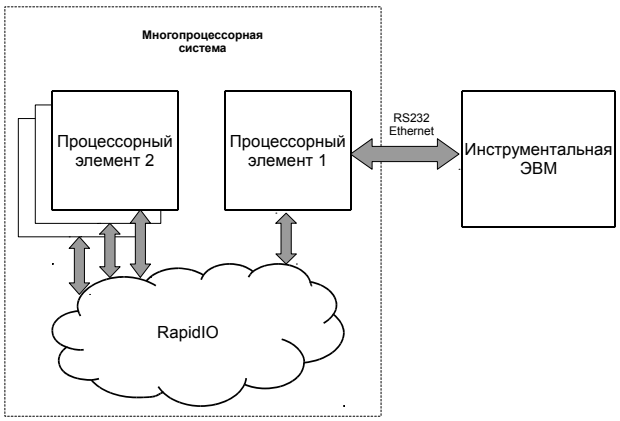



**Рис. 1.** Схема многопроцессорной системы

## 1. Организация тестируемой системы

К многопроцессорной системе на базе RapidIO целесообразно подключаться к одному процессорному элементу (процессор + память), посредством которого, по RapidIO, осуществлять взаимосвязь с другими процессорными элементами (см. рис. 1), используя программную поддержку обмена данными в Программе ПЗУ на каждом процессорном элементе. Данный тип подключения снижает затраты на дополнительное оборудование для тестирования функциональных узлов отдельных процессоров, а также снижается время на подготовку тестируемого стенда.

## 2. Протокол управления тестированием и обменом данных

Для того чтобы сделать процесс начального тестирования многопроцессорной системы управляемым, необходимо реализовать протокол обмена между устройствами. В спецификации RapidIO предложен протокол управления сессией [3]. Данный протокол имеет ряд особенностей, затрудняющих его использование на практике: используются потоки и идентификатор виртуального потока, обмен осуществляется операциями логического уровня и др. Для первоначального тестирования рассматривается система, приведенная на рисунке 1, в которой инициатором является Master-устройство, а все остальные оконечные устройства – Slave-устройства. Master-устройство в ходе тестирования многопроцессорной системы выполняет следующий перечень этапов:

1. Инициализация среды, используя метод инициализации на основе базовых блоков системы [4].
2. Запуск тестового программного обеспечения (ПО) на других оконечных устройствах RapidIO для проверки процессоров, системных контроллеров и контроллеров периферии.
3. Тестирование каждого коммутатора RapidIO в отдельности, для определения целостности сети обмена.
4. Получение информации о результатах тестирования от всех оконечных устройств.

Основные требования к протоколу управления передачей данных для первоначального тестирования:

- открыт один логический канал во время передачи данных;
- операции RapidIO для передачи данных должны быть максимально надежными;

- операции для передачи данных должны поддерживаться любым устройством RapidIO, для универсальности протокола;
- протокол должен поддерживать команды:
  - запуск тестов на Slave-устройствах;
  - ожидание окончания тестирования;
  - получение результата проверки и информации о ходе тестирования на удаленном узле.

В случае протокола управления сессией, операции RapidIO логического уровня нельзя считать наиболее отлаженными, по сравнению со служебными операциями чтения MAINTENANCE READ и записи MAINTENANCE WRITE [5]. Помимо этого, служебные операции MAINTENANCE поддерживаются всеми устройствами RapidIO. Если в контроллере RapidIO операции MAINTENANCE READ и WRITE работать не будут, то невозможно функционирование сети RapidIO на данном узле.

Предлагается рассмотреть протокол управления передачей данных, основанный на служебных операциях MAINTENANCE READ и WRITE [6]. Для синхронизации протокола управления обменом данных по RapidIO необходимо найти в служебном адресном пространстве RapidIO свободный регистр. К примеру, регистр «Component Tag» блока регистров логического уровня RapidIO CSR, который обычно используется при инициализации сети RapidIO в качестве метки и может быть изменен программно [7]. Выделяются поля в регистре для области данных и области команд. На рисунке 2 показан протокол управления обменом данных на базе операций MAINTENANCE READ и WRITE. Синхронизация требуется для того, чтобы согласовать запуск тестового ПО. До начала работы тестового ПО регистр «Component Tag» может быть изменен другим встроенным программным обеспечением, причем значение в регистре может совпадать со значением команды «SYNC», используемой для синхронизации, поэтому процедура синхронизация представляет собой две команды «SYNC1» и «SYNC2», чтобы зафиксировать изменение. Для гарантированного получения Slave-устройством команды запроса, им отправляется команда подтверждения «ACK», иначе устройство считается неисправным. В таблице 1 представлены команды протокола.

Таблица 1. Команды протокола управления тестированием и обменом данных

| Тип команды | Название команд | | Описание |
|---|---|---|---|
| Запрос | SYNC | SYNC1 | Синхронизация выполнения распределенного тестового программного обеспечения |
| | | SYNC2 | |
| | START | | Запуск тестов |
| | READY | | Готовность к передаче |
| | RESULT | | Получение результата тестирования |
| | LOG | | Получение информации о ходе тестирования |
| Ответ | ACK | | Подтверждение о получении запроса |
| | DATA | | Передача данных |
| | END | | Завершение передачи данных |

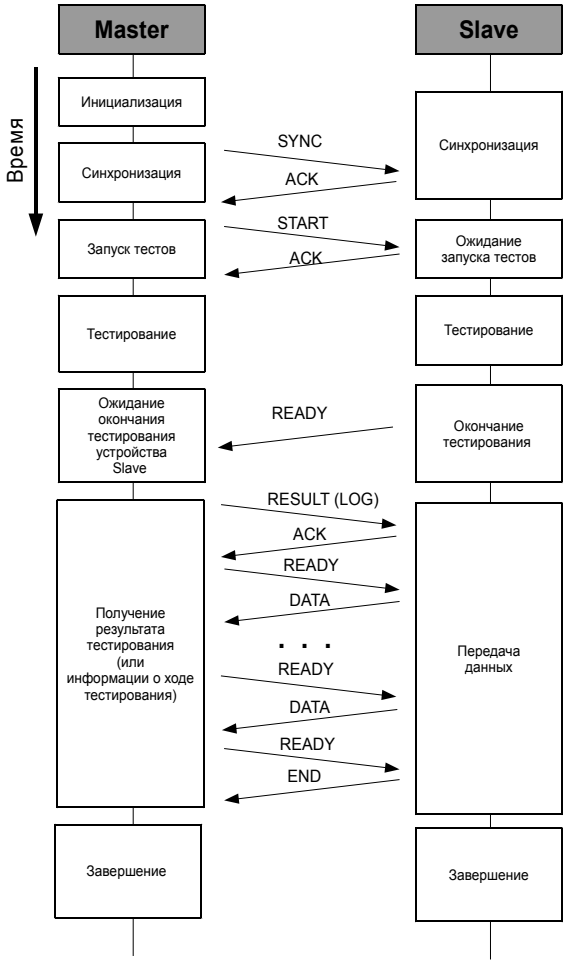



**Рис. 2.** Схема протокола управления тестированием и обменом данных

## 3. Синхронизация распределенного тестирования

В отличие от тестирования отдельно взятого устройства в системе (так называемое *standalone-тестирование*), в многопроцессорной системе для правильной работы тестов контроллера RapidIO по петле, необходимо синхронизовать работу тестового ПО на каждом устройстве. После применения метода инициализации с помощью конфигуратора базовых блоков, Master-устройство хранит набор маршрутов к каждому Slave-устройству. При локальном тестировании могут меняться и собственные ID и маршрут в соседнем коммутаторе, поэтому надо с помощью протокола изолировать тестирование, а после завершения тестирования Master-устройству необходимо восстановить все маршруты и работоспособность протокола. Стоит обратить внимание, что при настройке маршрутизации пакетов возможны пересечения записей в таблицах коммутаторов, а также, при организации петли во встроенном коммутаторе гибридного устройства (ГУ - устройство, объединяющее в себе оконечное устройство и встроенный коммутатор), возможны

совпадение идентификатора (ID) гибридного устройства, присвоенного при инициализации системы, с тестовым ID устройства, которое производит локальное тестирование. Особенностью встроенного коммутатора ГУ является отсутствие возможности настройки оконечным устройством петли через порт, соединяющий их между собой (например, микросхема 1890ВМ6Я), поэтому локальное тестирование подразумевает также тестирование встроенного коммутатора через петлю в ближайшем коммутаторе. Все остальные коммутаторы в системе тестируются Master-устройством. Таким образом, следует рассмотреть возможные конфигурации соединений устройств RapidIO, имеющие перечисленные выше проблемы (см. рис. 3).

Рассмотрим возможные ситуации, затрудняющие тестирование, для каждого случая отдельно:

*Вариант 1.*
*(Гибридное устройство – оконечное устройство)*

1. Гибридному устройству во время инициализации системы может быть присвоен идентификатор, который не должен совпадать с тестовым идентификатором устройства, производящего локальное тестирование контроллера RapidIO с помощью петли, организованной через встроенный коммутатор данного гибридного устройства;
2. Настройка таблицы маршрутизации встроенного коммутатора гибридного устройства может производиться несколькими оконечными (или гибридными) устройствами, так как коммутатор RapidIO является многопортовым устройством и может быть доступен разным оконечным (или гибридным) устройствам;
3. В момент настройки петли через встроенный коммутатор гибридного устройства, само гибридное устройство может производить тестирование через другой порт, подключенный к коммутатору или другому гибридному устройству.

*Вариант 2.*
*(Гибридное устройство – гибридное устройство)*

1. Все ситуации действительны, как и в варианте 1;
2. В один и тот же момент, может быть обращение гибридных устройств друг к другу для настройки петли;
3. Гибридное устройство может обратиться к другому гибридному устройству в момент тестирования по петле последнего через первое устройство.

*Вариант 3.*
*(ГУ/ОУ – коммутатор – ГУ/ОУ)*

Настройка таблицы маршрутизации коммутатора может производиться несколькими оконечными (или гибридными) устройствами, так как оно является многопортовым и может быть доступно разным оконечным (или гибридными) устройствам.

Для разрешения проблемных ситуаций, рассмотрим состояния, в которые может переходит оконечное (далее будет пониматься и гибридное устройство) устройство на RapidIO во время тестирования (см. рис. 4). Каждое состояние устройства RapidIO будет фиксироваться в первом байте регистра «Host Base Device ID Lock» блока регистров логического уровня RapidIO CSR, у которого для записи доступны первые 2 байта [7]. Значение регистра после сброса блока RapidIO равно 0x0000FFFF. Если была произведена запись в регистр, тогда, чтобы изменить записанные данные, необходимо сначала повторить запись тех же данных, а только потом записать новые данные. Разделим состояния на уровни: состояние 1 и 4 – это *уровень протокола*, а состояние 2 и 3 – это *уровень тестирование.* Уровнем протокола управляет Master-устройство, а уровнем тестирования все устройства производящие локальное тестирование.

Master-устройство переводит в состояние 1 все устройства, для того чтобы синхронизовать получение команды, например, START - начала тестирования. Иначе, первые Slave-устройства, получившие команду, начнут тестирование, не дожидаясь окончания передачи команд остальным Slave-устройствам. Это приведет к ошибочным ситуациям, связанным с перенастройкой таблиц маршрутизации.

После передачи команды, происходит переход из состояния 1 в состояние 2 уровня тестирования, который осуществляет каждое оконечное устройство в системе для настройки петли. Чтобы согласовать доступ к устройствам RapidIO для настройки тестовых петель используется механизм приоритетов. Каждому устройству назначается во время инициализации собственный ID, который является уникальным в системе и соответствует приоритету. Чем выше ID у устройства, тем выше приоритет. Другими словами, уровень тестирования характеризуется приоритетностью в системе. Во время настройки петли, устройство с меньшим приоритетом уступает устройству с большим приоритетом. Приоритет соседнего устройства определяется по чтению регистра «Base Device ID» блока регистров логического уровня RapidIO CSR. Оконечные устройства инициализируется, начиная с ID равное 3. ID равное 0 назначается после сброса системы всем оконечным устройствам, поэтому при прохождении служебных пакетов через гибридное устройство, выполняющего роль, исключительно, коммутатора, необходимо переназначить его ID, таким образом, усложняется процедура инициализации системы. ID равное 1 назначается оконечному устройству, которое

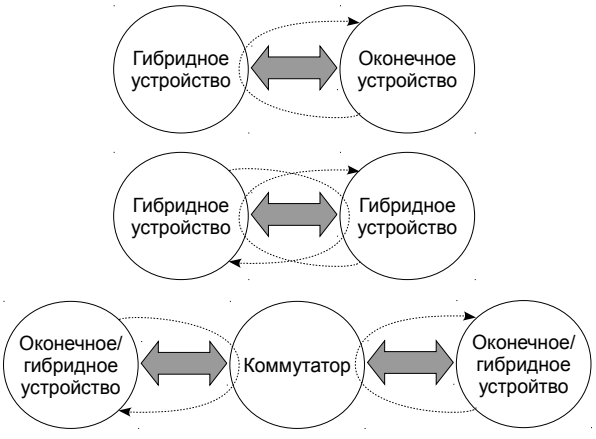



**Рис. 3.** Конфигурация соединений RapidIO устройств с пересечением петель

является Master-устройством в RapidIO системе. Для прокладки маршрута от Master-устройства ко всем остальным устройствам используется ID приемника равный 2. Во второй байт регистра «Host Base Device ID Lock» будет записываться номер порта, через который будет настраиваться петля в гибридном устройстве. Это делается для того, чтобы во время инициализации петли, соседнее гибридное устройство определяло: если тестирование будет производиться через его встроенный коммутатор, то в порядке приоритетности оно, либо настроит через данное устройство петлю, либо уступит свой встроенный коммутатор.


**Рис. 4.** Состояния устройств RapidIO

После настройки петли в коммутаторе, оконечное устройство переходит из состояния 2 в состояние 3 и начинает локальное тестирование. После этого во втором байте регистра «Host Base Device ID Lock» настроенного коммутатора, прописывается количество устройств, которое произвело настройку петель через данный коммутатор. То есть, первое устройство, настроив петлю, прописывает во втором байте регистра коммутатора значение 1, второе устройство, инкрементировав второй байт регистра, прописывает значение 2, и т.д. Это делается для того, чтобы несколько оконечных устройств могло проводить локальное тестирование через один коммутатор одновременно. После окончания тестирования, устройство декрементирует второй байт регистра коммутатора. После окончания тестирования всех устройств через данный коммутатор, значение 2-х байтного слова возвращается в первоначальное состояние равное 0xFFFF. В случае встроенного коммутатора гибридного устройства, значение 0xFFFF активизирует данное устройство для последующей настройки петли.

Для перехода в состояние 4 уровня протокола (восстановления протокола) после локального тестирования, Slave-устройство прописывает команду READY в регистр «Component Tag». Когда Master-устройство заканчивает локальное тестирование, оно начинает опрос Slave-устройств на наличие команды READY. Далее Master-устройство переводит систему в состояние 1.

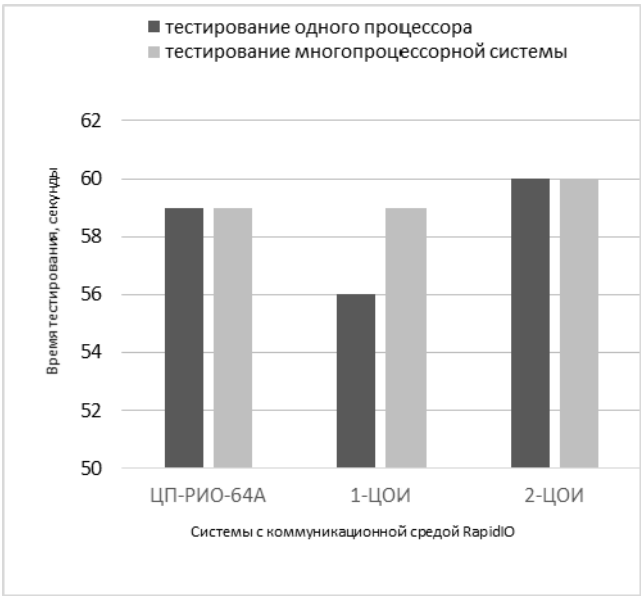



**Рис. 5.** Сравнение времени тестирования интерфейса RapidIO на одном оконечном устройстве и одновременно на каждом оконечном устройстве системы с применением предложенного метода

## 4. Результаты применения метода

Предлагается рассмотреть тестирование трех процессорных систем:

1. Двухпроцессорный модуль ЦП-РИО-64А [2];
2. Многопроцессорная система 1-ЦОИ, содержащая 2 процессора 1890ВМ6Я и 18 процессоров 1890ВМ7Я с архитектурой КОМДИВ [8];
3. Многопроцессорная система 2-ЦОИ, содержащая 8 процессоров 1890ВМ6Я.

С помощью предложенного метода тестирования RapdiIO систем можно распараллеливать процедуру тестирования процессорных элементов. Исходя из того, что одни и те же оконечные устройства тестируют схожие узлы процессорного элемента, время тестирования этих узлов практически совпадает. Поэтому рассматривается время тестирования, исключительно, блока RapidIO оконечных устройств для двух случаев:

- тестирование с применением предложенного метода на основе протокола управления тестированием и передачей данными, используя служебные операции MAINTENANCE READ и WRITE;
- тестирование системы, исходя из тестирования оконечных устройств параллельно без механизма

приоритетности и протокола управления тестированием, т.е. берется максимальное время тестирования оконечного устройства в системе.

На рисунке 5 приведены сравнения времени тестирования многопроцессорных систем. В случае двухпроцессорного модуля и системы 2-ЦОИ, время тестирования совпадает, так как содержит малое (меньше 8) количество процессоров. В случае системы 1-ЦОИ, есть незначительное изменение во времени. Все системы реализуют третий вариант конфигурации устройств с пересечением петель, поэтому устройства проводят тестирование параллельно друг другу, не ожидая освобождения коммутатора для настройки петли.

Рассматриваемый метод позволяет эффективно производить первоначальное тестирование многопроцессорных систем на базе интерфейса RapidIO, распределяя тестирование на каждом процессорном элементе. Несмотря на то, что данный протокол обмена по RapidIO не согласуется с протоколом, представленным в спецификации, его применение упрощает работу по выявлению ошибок на RapidIO и предъявляет минимальные требования к отлаженности коммуникационной среды.

# Method of initial RapidIO network testing

**G.A. Lavrinov**

**Abstract.** The article considers the method of initial RapidIO interface testing in multiprocessor systems based on the protocol of test management and data exchange using maintenance operations. A scheme of organization of the system under test described which reduces the cost of additional hardware. A protocol of test management and data exchange based on the maintenance operations of RapidIO schematically represented. Described synchronization of the distributed testing of multiprocessor systems. The results of testing RapidIO system using the specified method disclosed.

**Keywords:** RapidIO, testing, multiprocessor system, method, synchronization.

## Литература

1. S.Fuller. RapidIO. The Embedded Systems Interconnect. John Wiley & Sons, Ltd, 2005.
2. О.В. Сердин, С.Г. Бобков, Н.В. Кондратьева, А.А. Еремин. Разработка высоконадежных многопроцессорных модулей на базе высокоскоростных каналов rapidio. «Программные продукты и системы». 2013, №4, 49-55.
3. RapidIO Interconnect Specification. Annex 2: Session Management Protocol Specification. Revision 3.0. RapidIO Trade Association, 2013.
4. Г.А. Лавринов. Способы инициализации многопроцессорной системы. «Программные продукты и системы», 2014, №4, 37-40.
5. RapidIO Interconnect Specification. Part 1: Input/Output Logical Specification. Revision 3.0. RapidIO Trade Association, 2013.
6. Г.А. Лавринов. Способы повышения эффективности отладки и тестирования многопроцессорных систем. «Программные продукты и системы», 2012, №3, 86-89.
7. RapidIO Interconnect Specification. Part 3: Common Transport Specification. Revision 3.0. RapidIO Trade Association, 2013.
8. С.Г. Бобков, С.И. Аряшев, М.Е. Барских, К.С. Бычков, П.С. Зубковский. Микропроцессор гибридный. Патент № 2359315 от 20.06.2009.

# Методы оценки качества новых конструктивных решений современных СБИС

**А.В. Амирханов[1],С.И. Волков[2], С.А. Кизиев, В.Ю. Троицкий[1]**

*1-кандидат физико-математических наук, 2 – кандидат технических наук*

**Аннотация:** В статье рассмотрены вопросы разработки методов оценки качества новых конструктивных решений современных высокопроизводительных СБИС. Описаны основные задачи, поставленные при разработке этих методов. Определены характеристики продукции, использующей новые конструктивные решения. Предложены методы оценки качества микросхем в многовыводных матричных корпусах с шариковыми выводами для поверхностного монтажа и микросхем, использующих монтаж методом «перевернутого кристалла».

**Ключевые слова:** высокопроизводительные СБИС, оценка качества микросхем, матричные корпуса с шариковыми выводами, монтаж методом «перевернутого кристалла»

## Введение

Увеличение функциональной сложности микросхем, особенно таких, как СБИС современных высокопроизводительные микропроцессоров для суперЭВМ эксаплопсного диапазона с большим количеством внешних терминалов [1, 2], а также необходимость обеспечения работоспособности этих СБИС в жестких климатических условиях и после воздействия специальных факторов [3], толкают разработчиков на поиск новых конструктивных решений.

В течение длительного времени (с 1966 по 1976 годы) повышение функциональных характеристик микросхем обходилось без резкого увеличения количества выводов в традиционных 14-ти выводных плоских корпусах типа 4 по ГОСТ Р 54844-2011 [4]. С 1980 по 2000 годы количество выводов в таких корпусах изменялось, хотя и не резко, однако, принципиально конструкция корпуса не изменялась, хотя и появились новые корпуса такие, например, как типа 5, 6 и 7 по ГОСТ Р 54844-2011. Все это весьма характерно с точки зрения примера консервативности конструктива корпуса по сравнению с совершенствованием кристалла интегральной микросхемы. Причина такой консервативности – несовершенство кристалла, которое проявляется очень быстро (при испытаниях и в эксплуатации). Несовершенство корпуса может проявить себя только через несколько лет в связи с механизмами реального старения, связанными с экспоненциальными во времени процессами, определяемыми энергиями активации.

Сегодня же появление нового кристалла СБИС обращается, как правило, к новому конструктивному решению корпуса. Основной причиной этого, является стремление разработчиков микросхем добиться предельных функциональных характеристик СБИС в ущерб надежности, что не приемлемо для микросхем предназначенных для работы в аппаратуре специального назначения и создает ограничение их преминения.

Так, в США общими техническими условиями для микросхем MIL-PRF-38535K (подпункт H.3.4.4.1) [5] ограничено применение недостаточно проверенных конструктивных решений микросхем. К ним относят использование матричной конструкции корпуса с внешними выводами, изготовленными на основе легкоплавкого припоя, в том числе, шариковыми (англ. Ball grid array – BGA) или столбиковыми (англ. Column grid array – CGA) – корпуса типа 8 по ГОСТ Р 54844-2011. Кроме того, в MIL-PRF-38535K ограничено применение микросхем с использованием монтажа методом «перевернутого кристалла» (англ. Flip-Chip) [6, 7]. Применение таких, недостаточно зарекомендовавших себя с точки зрения надежности, новых конструктивных решений требует разработки специальной программы демонстрации целостности корпуса микросхем (англ. Package integrity demonstration test plan – PIDTP) для микросхем, комплектующих радиоэлектронную аппаратуру космического назначения, согласованной с разработчиками и изготовителями такой аппаратуры и дополняющей стандартные требования этого стандарта к программам контрольных испытаний микросхем.

Вкратце остановимся на причинах появления и основных особенностях таких новых конструктивных решений. Традиционный метод сборки микросхем представляет собой монтаж кристалла на основание корпуса (или промежуточную плату) с последующим соединением (разваркой) проволочными проводниками контактных площадок кристалла и контактных площадок корпуса [7, 8]. Однако, с увеличением частоты функционирования микросхемы начинают существенно сказываться паразитные индуктивность и сопротивление проволочных проводников. При современных требованиях к рабочим частотам СБИС влияние этих факторов все более возрастает. При использовании технологии монтажа методом «перевернутого кристалла», когда на контактные площадки кристалла наносят короткий (размером с саму контактную площадку) проводник-шарик, влияние паразитных связей значительно уменьшается. На корпусе контактные площадки

располагают зеркальным образом. Кристалл переворачивают и, после совмещения контактных площадок кристалла и корпуса, монтируют на основание корпуса. Такая технология обеспечивает минимальную длину электрического соединения между кристаллом и корпусом и, соответственно минимальные значения указанных негативных факторов (рис. 1). При этом дополнительным преимуществом монтажа методом «перевернутого кристалла» является возможность располагать контактные площадки на кристалле не только по его периметру, но и матричным способом. Количество контактных площадок может быть резко увеличено, что позволяет реализовать на кристалле все необходимые интерфейсы с большим количеством управляющих сигналов, а так же равномерно распределить по площади кристалла контактные площадки. Всё это даёт снижение падения напряжения на шинах в 2-3 раза по сравнению с обычными проволочными соединениями, а значит, позволяет дополнительно и существенно увеличить быстродействие СБИС.

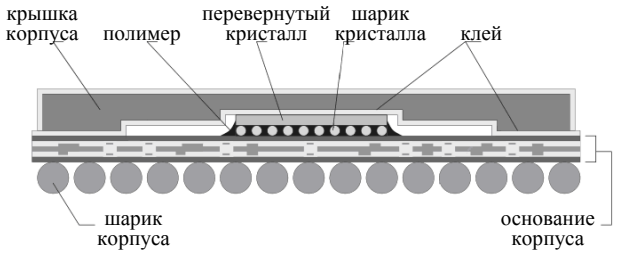



**Рис. 1.** Смонтированный кристалл (в разрезе)

Тип корпуса матричной конструкции типа 8 по ГОСТ Р 54844-2011 – это корпус интегральных микросхем, предназначенных для поверхностного монтажа, . внешние выводы которых представляют собой шарики или столбики, изготовленные на основе легкоплавкого припоя, как правило на основе сплава «олово-свинец» (рис. 2). Все внешние выводы микросхемы находятся на одной плоскости, поэтому их длина получается короче, чем у микросхем других конструктивных исполнений. Такое расположение выводов приводит к значительному снижению паразитных связей, а значит, также приводит к увеличению быстродействия СБИС.

**Рис. 2.** Матричный корпус с шариковыми выводами (вид снизу)

Благодаря шарикам или столбикам в корпусах этого типа длина выводов, уровень паразитных связей и наводок существенно ниже , чем в матричных корпусах со штырьковыми выводами, предназначенных для монтажа в отверстия печатной платы (корпусах типа 6 по ГОСТ Р 54844-2011.

Также к преимуществом корпуса с шариковыми выводами перед традиционными микросхемами с вертикальным и планарным расположением выводов является лучший тепловой контакт между микросхемой и печатной платой, что в ряде случаях избавляет от необходимости устанавливать дополнительные теплоотводы, поскольку тепло уходит от кристалла на плату более эффективно.

Кроме того, внешние терминалы корпусов с шариковыми и столбиковыми выводами располагаются матричным способом, что позволяет оптимальным образом разместить очень большое число выводов на ограниченной площади с сохранением достаточного зазора между ними.

Таким образом, микросхемы, использующие конструктивные решения на основе монтажа в корпус по методу «перевернутого кристалла» и применения корпусов с шариковыми и столбиковыми внешними выводами имеют несомненные преимущества над существующими типами корпусов в части обеспечения повышенных функциональных характеристик, и их использование для производства современных СБИС в настоящее время является актуальным. Наиболее перспективным является совмещение этих двух конструктивно-технологических решений (англ. Flip-Chip BGA – FCBGA) и оно уже востребовано многими разработчиками аппаратуры. Однако, даже по утверждению американских специалистов, как уже указывалось выше, такие решения находится «в зоне риска» в части обеспечения надежностных характеристик [9-11], и требуют разработки эффективных методов проверки их качества . Поэтому разработка соответствующих методов испытаний, дополняющих и развивающих методы ОСТ В 11 0998-99 [12] и ГОСТ РВ 5962-004-2012 (ОСТ 11 073.013-2012) [13, 14] также весьма актуальна.

ОСТ В 11 0998-99 распространяется на интегральные полупроводниковые микросхемы в корпусном исполнении, предназначенные для применения их в радиоэлектронной аппаратуре специального назначения всех климатических исполнений. Стандарт определяет систему требований к микросхемам, их разработке, производству и поставке, гарантии выполнения этих требований (гарантии качества) и устанавливает общие требования и условия, соблюдение которых разработчиками и изготовителями микросхем необходимо для поставок микросхем на комплектование радиоэлектронной аппаратуры специального назначения. Требования к конкретным типам микросхем устанавливают в технических условиях, разрабатываемых и утверждаемых в соответствии с требованиями настоящего стандарта.

Данный стандарт, в том числе определяет состав испытаний, деление состава испытаний на подгруппы испытаний и последовательность их проведения в пределах каждой подгруппы, метод испытаний, условия испытаний и планы контроля для соответствующих подгрупп испытаний. В свою очередь методы и условия испытаний определяются отраслевым стандартом ОСТ 11 073.013-2012 и аутентичным ему государственным стандартом ГОСТ РВ 5962-004-2012.

ГОСТ РВ 5962-004-2012 и ОСТ 11 073.013-2012 распространяется на интегральные микросхемы и корпуса микросхем и устанавливает основные положения методов испытаний на воздействие механических, климатических, биологических внешних воздействующих факторов и специальных сред, а также методов определения конструктивно-технологических запасов, электрических испытаний, электротермотренировок, испытаний на безотказность и сохраняемость, на стойкость к воздействию специальных факторов и импульсную электрическую прочность.

Таким образом для применения микросхем, изготовленных с использованием монтажа методом «перевернутого кристалла» и (или) с использованием корпуса с шариковыми или столбиковыми внешними выводами, в радиоэлектронной аппаратуре специального назначения (например категории качества «ВП») необходимо обеспечить их соответствие системе требований указанных документов.

Анализ применения методик проведения испытаний указанных стандартов для подтверждения качества микросхем, изготовленных с использованием монтажа методом «перевернутого кристалла» и (или) с использованием корпуса с шариковыми или столбиковыми внешними выводами показал, что для ряда задач по оценке качества таких конструктивных решений отсутствуют соответствующие методы испытаний.

## 1. Ограничение в применимости существующих методик проведения испытаний

Стандартные методы, изложенные в ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012, не позволяют провести испытание на способность к пайке у микросхем в матричном корпусе с шариковыми или столбиковыми внешними выводами, изготовленными из легкоплавкого припоя, так как при применении этих методов шариковые выводы могут разрушаться, и говорить о качестве их облуживания невозможно. Понятно, что оценка способности к пайке микросхемы в матричном корпусе типа 8 по ГОСТ Р 54844-2011 требует поиска других подходов к методам испытаний.

Аналогичная ситуация и с испытанием на теплостойкость при пайке.

В соответствии с ОСТ В 11 0998-99 проверку качества монтажа кристалла согласно контрольным испытаниям подгруппы К6 (последовательность 3) и подгруппы В4 (последовательность 4) проводят по методу 115-1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012, предусматривающим воздействие на кристалл усилия, воздействующего в тангенциальном направлении (испытанием «на сдвиг»). Указанный метод, как правило, не пригоден для проверки качества монтажа, проведенного методом «перевернутого кристалла», и требует уточнения как в части собственно методики испытания, так и в части критериев оценки его результатов, учитывающих многообразие конструктивных решений, используемых при таком подходе к монтажу кристалла в корпус.

Учитывая то, что контрольными испытаниями подгруппы К5 ОСТ В 11 0998-99 предусмотрены проверки целостности и стойкости к механическим воздействиям внешних выводов микросхем (последовательность 1 – испытание выводов на воздействие растягивающей силы по методу 109-1, последовательность 2 – испытание гибких проволочных и ленточных выводов на изгиб по методу 110-3, а также последовательность 3 – испытание гибких лепестковых выводов на изгиб по методу 111-1), существует необходимость разработки методики проведения проверки целостности конструктивного узла «шарик (столбик) – терминал матричного корпуса».

Из вышеизложенного следует, что применение указанных конструктивных решений в радиоэлектронной аппаратуре специального назначения ограничено отсутствием в действующей нормативно-технической документации методов оценки ряда их технических характеристик и требований к ним.

Следовательно, чрезвычайно актуальным является разработка дополнительных, либо доработка существующих методов оценки качества (испытаний) новых конструктивных решений, используемых для производства современных СБИС.

На основе проведенного анализа предлагаются нижеследующие уточнения в существующие методики проведения испытаний ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012.

## 2. Проверка на пригодность к пайке микросхем в матричных корпусах с шариковыми и столбиковыми выводами

Существующие методы испытаний на способность к пайке 402-1, 402-2 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012 предлагается дополнить проверкой на пригодность к пайке микросхем в матричных корпусах с шариковыми и столбиковыми выводами.

Проверку следует проводить с целью оценки способности шариковых и столбиковых припойных выводов микросхем обеспечивать качественное соединение при пайке.

Проверку следует проводить в испытательной камере, имеющей объем, достаточный для размещения керамической подложки с установленной на ней испытываемой микросхемой (микросхемами), и способной поддерживать испытательный режим, в том числе:

- температуру на корпусе испытываемой микросхемы 235°С ± 5°С;
- скорость нагрева и охлаждения испытательной платы с испытываемой микросхемой (микросхемами) от 1°С/с до 4°С/с.

Если иное не установлено в технических условиях (ТУ) на микросхему и (или) программе испытаний (ПИ) микросхемы.

Испытываемую микросхему следует устанавливать на керамическую подложку с нанесенной на нее, с помощью трафарета, паяльной пастой. Подложка должна иметь площадь не менее 650 мм$^2$ и толщину 1,6 мм ± 0,2 мм.

Испытательное оборудование и приспособления должны обеспечивать возможность контроля температуры испытания на корпусе испытываемой микросхемы.

Применение припоя и флюса при проведении испытания должно соответствовать требованиям к монтажу микросхемы, установленным в ТУ и (или) ПИ.

Проверку следует проводить с учетом требований п.4.1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012. Перед испытанием следует проводить проверку внешнего вида микросхем. Для определения характера дефектов необходимо применять лупу с кратностью увеличения до 10× или микроскоп.

Микросхему следует тщательно и аккуратно устанавливить на подложку, избегая размазывания паяльной пасты, и помещают в испытательную камеру.

Микросхему следует подвергнуть температурной обработке в режимах и в течение времени, указанных в ТУ для проведения операции монтажа в аппаратуре.

После проведения температурной обработки, подложку и микросхему следует охладить до нормальной температуры, после чего снимают микросхему с подложки с помощью пинцета и проводят визуальный осмотр припойных терминалов при увеличении от 10× до 100×. Конкретное значение кратности увеличения указывают в ТУ и (или) ПИ.

Микросхемы можно считать выдержавшими проведение проверки, если все припойные терминалы демонстрируют ровную гладкую бездефектную поверхность не менее чем на 95% площади каждого из терминалов, если иное не установлено в ТУ и (или) ПИ.

При проведении визуального осмотра дефектами следует считать:

- поры в припое внешнего вывода;
- участки несмоченной поверхности.

# 3. Проверка на теплостойкость при пайке микросхем в матричных корпусах с шариковыми и столбиковыми выводами

Существующие методы испытаний на теплостойкость при пайке 403-1, 403-2 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012 целесообразно дополнить проверкой на теплостойкость при пайке микросхем в матричных корпусах с шариковыми и столбиковыми выводами.

Проверку следует проводить с целью определения способности микросхем в матричных корпусах с шариковыми и столбиковыми выводами выдерживать воздействие тепла, возникающего при пайке.

Проверку следует проводить в испытательной камере, имеющей объем, достаточный для размещения испытательной платы с установленной на ней испытываемой микросхемой (микросхемами), и способной поддерживать испытательный режим, в том числе:

- температуру на корпусе испытываемой микросхемы 235°С ± 5°С;
- скорость нагрева и охлаждения испытательной платы с испытываемой микросхемой (микросхемами) от 1°С/с до 4°С/с;

Если иное не установлено в ТУ и (или) ПИ.

Испытываемую микросхему следует устанавливать на испытательную плату. Испытательная плата должна быть изготовлена из стеклопластика и иметь площадь не менее 650 мм$^2$ и толщину 1,6 мм ± 0,2 мм, если иное не установлено в ТУ и (или) ПИ. Контактные площадки испытательной платы должны иметь конфигурацию и размеры, соответствующие испытываемой микросхеме, а сама испытательная плата должна обеспечивать возможность контроля электрических характеристик микросхемы по завершении испытания.

Испытательное оборудование и приспособления должны обеспечивать возможность контроля температуры испытания на корпусе испытываемой микросхемы.

Применение паяльной пасты при проведении испытания должно соответствовать требованиям к монтажу микросхемы, установленным в ТУ и (или) ПИ.

Проверку следует проводить с учетом требований п.4.1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012.

Перед испытанием необходимо провести проверку внешнего вида микросхем и контроль электрических параметров – критериев годности, установленных в ТУ или ПИ для этого вида испытания, а так же функциональный контроль (если он предусмотрен ТУ). Для определения характера дефектов следует применять лупу с кратностью увеличения до 10х или микроскоп.

Испытательную плату с микросхемой (микросхемами) следует помещать в испытательную камеру, и температуру корпуса микросхемы повышают со скоростью от 1°С/с до 4°С/с до температуры 235°С ± 5°С. Время выдержки микросхемы при этой температуре должно составлять 30 с ± 5 с, при этом общее время нахождения микросхемы при температуре выше 183°С должно составлять от 90 с до 120 с. Затем испытательную плату с микросхемой (микросхемами) следует остудить до нормальной температуры, что составляет один цикл испытания. Затем следует последовательно провести еще два цикла испытания.

По окончании проверки испытательную плату с микросхемой (микросхемами) следует выдержать при нормальной температуре в течение 2 ч, если иное время не установлено в ТУ и (или) ПИ, после чего, не снимая микросхем с испытательной платы, необходимо провести проверку внешнего вида (кроме внешнего вида внешних выводов) и контроль электрических параметров-критериев годности микросхем, установленных в ТУ и (или) ПИ, а также функциональный контроль (если он предусмотрен в ТУ).

Микросхемы можно считать выдержавшими проведения проверки, если внешний вид, электрические параметры и функционирование соответствуют требованиям, установленным в ТУ и (или) ПИ.

# 4. Проверка прочности монтажа «перевернутого кристалла»

Существующий метод испытания прочности крепления кристалла на сдвиг 115-1 ОСТ 11 073.013-2012
и ГОСТ РВ 5962-004-2012 предлагается дополнить нижеследующими проверками.

## 4.1. Проверка прочности монтажа «перевернутого кристалла»

Проверку следует проводить с целью определения прочности межсоединений кристалла и основания корпуса (подложки) при монтаже методом «перевернутого кристалла»).

Проверку целесообразно проводить испытанием «на сдвиг» либо, если конструктивные решения микросхемы не позволяют провести испытание воздействием сдвигового усилия, – испытанием на воздействие разрывного усилия. При этом, методики испытаний «на сдвиг» и на воздействие разрывного усилия различаются в зависимости от конструктивного решения микросхемы в части заполнения полимером зазора между кристаллом и основанием корпуса (платой). Конкретные методы (методики) испытания следует указывать в ТУ и, при необходимости, в технологической документации.

## 4.2. Проверка на отрыв «перевернутого кристалла» после заполнения полимером

Для проведения проверки необходимо следующее оборудование и приспособления:

- испытательное оборудование, способное обеспечивать приложение к кристаллу и контролировать значение удельного разрывного усилия до 7 МПа ±5%;

- набор сменных инструментов для непосредственного контактирования с кристаллом, обеспечивающих площадь соприкосновения с ним не менее 60% от его площади.

Проверку следует проводить с учетом требований п.4.1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012.

Перед началом проверки на контактирующий инструмент наносят небольшое количество быстро схватывающегося клея (например, цианакрилата), который прижимают к поверхности кристалла. По прошествии времени, достаточного для полимеризации клея, образец следует подвергнуть вертикальной растягивающей нагрузке в соответствии с Таблицей 1.

**Таблица 1.** Минимальные значения растягивающей нагрузки, не приводящие к разрушению столбиковых выводов

| Диаметр столбика, мм | Минимальное значение разрушающей нагрузки, н |
|---|---|
| 0,55 | 6,7 |
| 0,51 | 5,75 |
| 0,41 | 3,7 |
| 0,33 | 2,4 |

Примечания.

1. Если диаметр столбикового вывода отличается от значений, указанных в таблице, минимальную величину разрушающей нагрузки рассчитывают, исходя из удельного значения 28 н на квадратный миллиметр поперечного сечения столбикового вывода.
2. Конкретное минимальное значение растягивающей нагрузки на столбиковый вывод устанавливают в ТУ или ПИ.

С помощью испытательного оборудования к кристаллу прикладывают усилие, достаточное для его отрыва или равное удвоенному минимальному установленному значению прочности на разрыв согласно второму столбцу таблицы 1, в зависимости от того, какое событие наступит первым. Микросхемы считают выдержавшими испытания, если:

- на кристалле и основании корпуса присутствуют признаки проведенного монтажа в виде остатков межсоединений или их следов, а отделение «перевернутого кристалла» от основания корпуса произошло при усилии, превышающем минимально допустимое (второй столбец таблицы 2);

- на кристалле или основании корпуса отсутствуют признаки проведенного монтажа в виде остатков межсоединений или их следов, но отделение «перевернутого кристалла» от основания корпуса или инструмента от кристалла произошло при усилии, превышающем удвоенное значение минимально допустимого (третий столбец таблицы 2).

**Таблица 2.** Критерии оценки результатов испытания «перевернутого кристалла», после заполнения полимером, на отрыв

| Площадь кристалла, мм$^2$ | Минимально допустимое усилие отрыва, н | |
|---|---|---|
| | при наличии остатков межсоединений на кристалле и подложке | при отсутствии остатков межсоединений на кристалле или подложке |
| менее 5 | 25 | 50 |
| от 5 до 10 включительно | 30 | 60 |
| от 10 до 50 включительно | 37 | 74 |
| от 50 до 100 включительно | 45 | 90 |
| от 100 до 200 включительно | 49 | 98 |
| более 200 | 52 | 104 |

Отрыв инструмента от кристалла при усилии, уступающем удвоенному значению минимально допустимого, следует рассматривать, как несоответствие процедуры проведения испытания установленным требованиям, результаты испытания следует аннулировать, причины несоответствия следует устранить и испытание необходимо повторить на новом образце (или выборке).

## 4.3. Проверка на сдвиг «перевернутого кристалла» без заполнения полимером

Для проведения проверки необходимо оборудование и приспособления, обеспечивающие приложение необходимой нагрузки и контроль приложенной нагрузки в момент разрыва соединения.

Проверку следует проводить с учетом требований п.4.1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012.

Нагрузку (равномерно по площади) необходимо прикладывать к одной из боковых граней кристалла под углом, близким к 90º, и регистрировать величину сдвигающей силы.

Микросхемы могут считаться выдержавшими проведение проверки, если удельное значение сдвигающего усилия, в пересчете на одно межсоединение, не уступает 0,05 н, а разрушение межсоединений имеет один из трех видов:

- разрушение межсоединения в его объеме или вблизи его основания;
- разрушение кристалла или основания корпуса (утрата части кристалла или подложки непосредственно под межсоединением);
- отслоение металлизации (отделение металлизации или основания межсоединения от кристалла или основания корпуса).

## 4.4. Проверка на отрыв «перевернутого кристалла» без заполнения полимером

Для проведения проверки необходимо следующее оборудование и приспособления:

- испытательное оборудование, способное обеспечивать приложение к кристаллу и контролировать значение удельного разрывного усилия до величины, не менее чем в два раза, превышающей минимально допустимое значение, с точностью ±5% или ±0,25 гс, в зависимости от того, какая из этих величин меньше.;
- набор сменных инструментов для непосредственного контактирования с кристаллом, обеспечивающих площадь соприкосновения с ним 75% +3% -5% от его площади.

Проверку следует проводить с учетом требований п.4.1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012.

Перед началом проверки на контактирующий инструмент следует нанести небольшое количество быстро схватывающегося клея (например, цианакрилата), который прижимают к поверхности кристалла. По прошествии времени, достаточного для полимеризации клея, образец необходимо подвергнуть вертикальной растягивающей нагрузке с отклонением от направления, перпендикулярного к плоскости кристалла, не превышающим 5 градусов.

С помощью испытательного оборудования к кристаллу следует приложить усилие, плавно, без толчков, увеличивающееся со скоростью 500 г ± 100 г в секунду до его отрыва от основания корпуса.

Микросхемы могут считаться выдержавшими проведение проверки, если удельное значение усилия отрыва, в пересчете на одно межсоединение, не уступает значению 5 н/мм$^2$, умноженному на среднее значение площади поперечного сечения одного межсоединения, а разрушение межсоединений может быть идентифицировано, как один из следующих видов:

- разрушение кристалла;
- отслоение металлизации контактной площадки от кристалла;
- отделение по границе «межсоединение – кристалл»;
- разрушение межсоединения в его объеме;
- отделение по границе «межсоединение – основание корпуса»;
- отслоение металлизации от основания корпуса;
- разрушение основания корпуса.

# 5. Проверка на сдвиг шариковых выводов микросхем в матричных корпусах

В развитие методов испытаний выводов микросхем 109-1, 110-3, 111-1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012 предлагается проверка на сдвиг шариковых и столбиковых выводов микросхем в матричных корпусах.

Проверку следует проводить с целью оценки способности шариковых и столбиковых выводов микросхем в матричных корпусах, сформированных на основе припоя, противостоять механическим сдвиговым нагрузкам.

Для проведения проверки необходимо оборудование и приспособления, обеспечивающие приложение необходимой нагрузки и контроль приложенной нагрузки в момент разрушения шарикового вывода. Оборудование должно быть способно сдвигать шарики припоя с заданной постоянной скоростью перемещения (испытательной скоростью) и должно иметь возможность регистрировать значения создаваемой сдвиговой нагрузки, как функции перемещения и времени. Для проведения проверки при высоких и низких испытательных скоростях можно допустить использование различных единиц оборудования.

Испытательное оборудование и приспособления должны исключать возможность любого смещения или деформации испытываемой микросхемы.

Проверку следует проводить с учетом требований п.4.1 ОСТ 11 073.013-2012 и ГОСТ РВ 5962-004-2012.

Перед проведением проверки на используемом испытательном оборудовании и с помощью используемых приспособлений следует провести достаточный объем предварительных испытаний на сдвиг аналогов шариковых выводов с целью получения статистически значимого количества результатов проверки сдвиговых усилий, позволяющего рассчитать границы естественной изменчивости их значений (через определение их среднего арифметического значения и утроенного значения среднего квадратичного отклонения).

Если перед проведением проверки должна быть проведена операция дополнительного оплавления шариков, режимы и длительность этой операции не подлежат изменению и должны быть указаны в ТУ и (или) ПИ. Время от завершения операции оплавления до начала испытания не должно быть менее одного часа и не должно превышать четыре часа, если иное не установлено в ТУ и (или) ПИ.

Перед началом проверки необходимо произвести визуальный осмотр исполнительного органа оборудования (толкателя) на предмет наличия следов его износа (зазубрин, выемок, скруглений), имеющих размеры более 2% высоты шарикового вывода. При обнаружении таких дефектов толкатель следует заменить.

Толкатель не должен одновременно касаться нескольких шариковых выводов и должен быть ориентирован под углом 90°± 2 относительно базовой плоскости микросхемы. Расстояние толкателя от базовой плоскости корпуса микросхемы должно не превышать 25% высоты шарика припоя, при этом, это расстояние должно исключать возможность касания толкателем элементов корпуса микросхемы на всем маршруте его перемещения. Если особенности конструкции испытательного оборудования и (или) приспособлений не позволяют проводить испытания, не касаясь шарикового вывода, включенного в испытательную выборку, шариковые выводы, не включенные в испытательную выборку следует удалить с корпуса испытываемой микросхемы. Не должно допускаться касание толкателем шарикового вывода, включенного в испытательную выборку, до начала его перемещения.

Проверку следует проводить при одной (низкой или высокой) или двух (низкой и высокой) скоростях перемещения толкателя.

Низкая скорость перемещения толкателя должна составлять от 0,1 мм/с до 0,8 мм/с, высокая скорость – более 50 мм/с. Конкретную процедуру испытания устанавливают в ТУ и (или) ПИ, в зависимости от особенностей конструкции и материала шарикового вывода, а также конкретных задач, решаемых при проведении испытания.

Проверку качества припойного шарика и наличия загрязнений или иных причин несмачивания контактной площадки под шариком следует проводить при низкой скорости перемещения толкателя. Проверку качества сцепления контактной площадки с корпусом микросхемы, качества основания корпуса в области монтажа шариков, качества шарика в области, граничащей с контактной площадкой корпуса, следует проводить при высокой скорости перемещения толкателя.

Планы контроля микросхем (количество микросхем в испытательной выборке, количество шариковых выводов в каждой микросхеме испытательной выборки и их приемочные числа) следует устанавливать в ТУ и (или) ПИ. При проведении проверки при двух скоростях перемещения толкателя объемы испытательных выборок следует устанавливать независимо для каждой скорости перемещения толкателя.

Толкатель необходимо поочередно прикладывать к шариковым выводам, включенным в испытательную выборку и перемещают его с заданной (низкой или высокой) скоростью на расстояние, равное диаметру шарикового вывода, если его разрушение не наступило ранее. После разрушения шарикового вывода следует регистрировать значение сдвигового усилия, достигнутого при разрушении, и подвергать визуальному анализу внешний вид разрушенного вывода при увеличении не менее 100×. Конкретную кратность увеличения следует устанавливать в ТУ и (или) в ПИ.

Шариковый вывод микросхемы можно считать выдержавшим проверку, если, в общем случае:

- начение сдвигового усилия, зарегистрированное при испытании, находится в пределах значений его естественной изменчивости, определенных исходя из того, что температуру окружающей среды при электротермотренировке микросхем, обеспеченных встроенными схемами блокировки при перегреве кристалла, определяют таким образом, чтобы функционирование микросхемы проходило в штатном режиме, а длительность электротермотренировки устанавливают в соответствии со значением этой, вновь определенной температуры.
- разрушение вывода произошло по объему припойного шарика либо в объеме основания корпуса;
- после проведения испытания с низкой скоростью – внешний вид среза шарика не имеет характера хрупкого разрушения;
- после проведения испытания с высокой скоростью – разрушение вывода не имеет характера

отслоения контактной площадки от основания корпуса, отсоединения шарика от контактной площадки, а внешний вид остатков припоя на поверхности контактной площадки не имеет характера хрупкого разрушения.

В ТУ и (или) ПИ критерии оценки результатов испытания могут быть уточнены, исходя из особенностей конструкции и материалов корпуса микросхемы.

## 6. Выводы

Вывод из всего сказанного следующий: необходима доработка существующих методов, введение новых методов и требований к ним. Это даст возможность оценить соответствие применяемых микросхемы, изготовленных с использованием монтажа методом «перевернутого кристалла» и (или) с использованием корпуса с шариковыми или столбиковыми внешними выводами, системе требований ОСТ В 11 0998-99 и ГОСТ РВ 5962-004-2012 (ОСТ 11 073.013-2012). И, как следствие, станет возможным применение новых конструктивных решений в радиоэлектронной аппаратуре специального назначения.

Статья поступила в редакцию 25.08.2015.

# Methods of the estimation of quality new constructive decisions of modern VLSI

**A.V. Amirkhanov, S.I. Volkov, C.A. Kiziev, V.Y. Troitskiy**

**Abstract:** The article considers development of methods of an estimation of quality of new constructive decisions, modern exascale VLSI. The basic goals put by development of these methods are described. The characteristics of production using qualities of new constructive decisions are determined. A number of new methods of an estimation of quality of cases such as BGA for superficial – mounted VLSI and installation VLSI by a method of «the turned crystal» such as Flip-Chip is offered.

**Keywords:** exascale VLSI, estimation of quality VLSI, BGA, Flip-Chip.

## Литература

1 С.Г.Бобков. Высокопроизводительные микропроцессоры для суперЭВМ эксафлопсного диапазона. Сборник научных трудов под ред. В.Я. Стенина. // М.: Изд. «НИЯУ МИФИ», 2012 – с. 129-141.

2 С.Г.Бобков. Импортозамещение элементной базы вычислительных систем // М.: Изд. «НИИСИ РАН», 2014. (http://www.pcweek.ru/gover/it-independence/materials/13-bobkov.pdf)

3 Е.С.Темников. Проблемы проведения контрольных испытаний высокосложных микросхем в условиях мелкосерийного производства // Труды НИИСИ РАН. Том 4, вып. 1, М.: Изд. «НИИСИ РАН», 2014 – с. 45-51.

4 ГОСТ Р 54844-2011 «Микросхемы интегральные. Основные размеры».

5 MIL-PRF-38535K «Performance Specification: Integrated Circuits (Microcircuits) Manufacturing, General Specification for».

6 R.C.Marrs et al. Ball Grid Array Technology // McGraw-Hill Book Company Inc. USA, 1995.

7 G.Harman. Wire bonding in microelectronics. Third Edition // McGraw-Hill Companies, Inc. USA, 2010.

8 Сборка интегральных микросхем в металлокерамические корпуса. Сборник под ред. Захарова А.Ю. // М.: Изд. «НИИСИ РАН», 2008.

9 И.Лейтес. Реболдинг и проблема обеспечения надежности паяных соединений // Производство электроники: Технологии, оборудование, материалы, № 8, 2008 – с. 34-37.

10 Чин-Май Ко и др. Надежность тестирования BGA-компонентов // Технологии в электронной промышленности, № 4, 2009 – с. 38-42.

11 К.К.Смирнов, А.Г.Сухов, А.С.Цимбалов.. Проблемы применения металлополимерных корпусов BGA // Труды НИИСИ РАН. Том 3, вып. 2, М.: Изд. «НИИСИ РАН», 2013 – с. 99-103.

12 ОСТ В 11 0998-99 «Микросхемы интегральные. Общие технические условия».

13 ГОСТ РВ 5962-004-2012 «Изделия электронной техники. Микросхемы интегральные. Методы испытаний. Основные положения»

14 ОСТ 11 073.013-2012 «Микросхемы интегральные. Методы испытаний»

# К вопросу учета диэлектрических свойств кремния при схемотехническом моделировании

**Е.Н. Епихин[1], Н.В. Масальский[1]**

*1 - кандидат физико-математических наук*

**Аннотация.** При помощи разработанной квазидвумерной аналитической модели симметричного двух затворного полностью обедненного КНИ КМОП нанотранзистора численно анализируется влияние температурной зависимости диэлектрической проницаемости кремния на вольт-амперные характеристики. Отмечено, что качественный сдвиг характеристик, который приводит к их еще большей деградации, наблюдается при температурах выше 400 К. Наибольшее количественное отклонение свойственно величине подпорогового тока.

**Ключевые слова**: кремний на изоляторе, двух затворный нанотранзистор, температурная зависимость характеристик транзистора, аналитическая модель

## Введение

В настоящее время наблюдается увеличение интереса к проблеме, посвященной применению электронных приборов в приложениях, где условия их эксплуатации отличаются от нормальных. Одними из таких устройств являются микросхемы на структурах «кремний на изоляторе» (КНИ). Они в полной мере являются идеальными высокотемпературными устройствами, которые широко применяются в вооруженных силах, нефтегазовой, ядерной и других отраслях, где необходимо их функционирование в высокотемпературной среде.

К перспективным представителям элементной базы КНИ микросхем по праву относят двух затворные полевые транзисторы (см., например, [1, 2]). Данная архитектура характеризуется уникальными возможностями для масштабирования параметров в наноразмерной области [3, 4]. Для наноэлектронных устройств проведение макетных исследований в высоко температурной области становится не постижимо дорогостоящей задачей [1]. С целью сокращения расходов разрабатываются методы схемотехнического моделирования, с помощью которых получают прогноз, сообразуясь с которым и разрабатывают микросхемы. В связи с этим важно оценить влияние температурной зависимости диэлектрических свойств кремния – одного из главных компонентов КНИ структуры - на ключевые характеристики транзистора. При этом, несмотря на большое количество экспериментальных работ в литературе практически нет соответствующих данных по оценке воздействия изменения диэлектрической проницаемости, например, на уровень тока транзистора.

Цель данной работы заключается в разработке подхода для исследования влияния температурной зависимости диэлектрической проницаемости кремния на ряд основных параметров транзистора, таких как пороговое напряжение и ток транзистора. В данной работе на основе экспериментальных данных и 2D аналитического моделирования и решается поставленная задача.

## 1. Экспериментальные данные

В таблице 1 приведены экспериментальные данные (первая и вторая колонки), заимствованные из [5] по диэлектрической проницаемости кремния для разных температур, где $T$ – температура, $\varepsilon_{S_{\{\exp;cor\}}}$ - диэлектрическая проницаемость кремния: экспериментальные и скорректированные значения, соответственно, $E_g(T)$ - ширина запрещенной зоны. При этом во всей температурной области пределы точности эксперимента совпадают.

**Таблица 1.** Экспериментальные и вычисленные значения диэлектрической проницаемости кремния

| $T$, К | $\varepsilon_{S\ \exp}$ | $E_g(T)$ | $\varepsilon_{S_{\exp}}(T)^2 E_g(T)$ | $\varepsilon_{S\ cor}$ |
|---|---|---|---|---|
| 200 | 11.5 | 1.15 | 151.7 | 11.52 |
| 296 | 11.6 | 1.23 | 151.2 | 11.63 |
| 373 | 11.8 | 1.11 | 153.8 | 11.74 |
| 473 | 11.9 | 1.08 | 152.2 | 11.90 |

Температурная зависимость диэлектрической проницаемости кремния $\varepsilon_S(T)$ хорошо описывается известным соотношением Мосса:

$$\varepsilon_S(T)^2 E_g(T) - const \qquad (1)$$

Исходя из данных таблицы для температур от 200 К вычислялось произведение $\varepsilon_{S_{\exp}}(T)^2 E_g(T)$ с учетом температурной зависимости ширины запрещенной зоны (третья колонка). Усредненное значение полученных произведений составляет 152.2 эВ. Отклонение усредненной величины от значения величины $\varepsilon_{S_{\exp}}(T)^2 E_g(T)$ для отдельной температуры не превышает 2 эВ. Используя усредненное значение, корректировались значения диэлектрической проницаемости (пятая колонка). При этом максимальное отклонение составляет 0.06, что в процентах равно 0.5%. Все проведенные расчеты иллюстрируются таблицей 1 и рисунком 1.

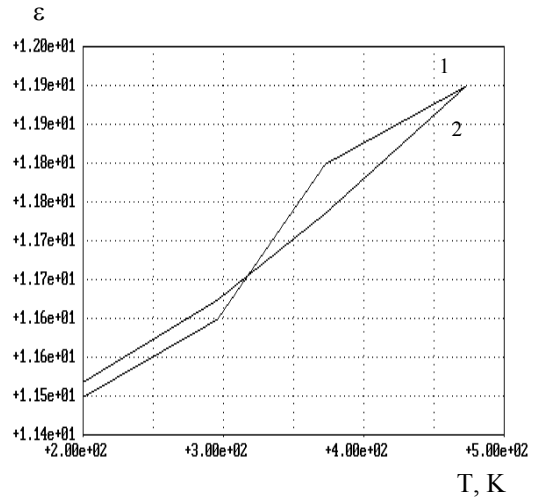



**Рис. 1.** Зависимость диэлектрической проницаемости кремния от температуры
1 - экспериментальные точки, соединенные прямыми линиями , 2 - кривая по закону (1)

## 2. Поведение потенциала

В рамках квазиклассической концепции зарядового разделения рассмотрим решение задачи температурной зависимости 2D распределения потенциала $\varphi(x,y,T)$ в рабочей области двух затворного симметричного полностью обедненного КНИ нанотранзистора. В общем случае 2D уравнение Пуассона для рабочей области рассматриваемого транзистора имеет вид [6, 7]:

$$\frac{\partial^2\varphi(x,y,T)}{\partial x^2}+\frac{\partial\varphi(x,y,T)}{\partial y^2}=-\frac{q}{\varepsilon_S(T)}N_A, \qquad (2)$$

где $q$ – заряд электрона, $N_A$ – концентрация легирования рабочей области и оси x и y выбраны так: x – перпендикулярно вглубь рабочей области, y – вдоль рабочей области.

Учет температурной зависимости параметра $\varepsilon_S(T)$ не меняет внешнего вида выражения для поверхностного потенциала

$$\varphi_s(y,T)=\frac{(u_{bi}(T)+l^2A(T))\left(e^{L_{g_{eff}}/l}-1\right)-U_{ds}}{2sh\left(L_{g_{eff}}/l\right)}e^{-y/l}+$$
$$+\frac{U_{ds}-(u_{bi}(T)+l^2A(T))\left(e^{-L_{g_{eff}}/l}-1\right)}{2sh\left(L_{g_{eff}}/l\right)}e^{y/l}-l^2A(T) \qquad (3)$$

где все обозначения сохранены в соответствии с [7]. Изменения претерпевают только выражения для характеристической длины $l$ и коэффициента $A(T)$, а также выражение для общего вида распределения потенциала. Любой желающий может сделать необходимые замены самостоятельно. Например, вид зависимости *l(T)* с учетом зависимости $\varepsilon_S(T)$ аналогичен кривой 2 на рис. 1. Их взаимосвязь определяется выражением $l_{norm}=\sqrt{\varepsilon_{norm}}$, где $l_{norm}=l(T)/l(T_0)$ и $\varepsilon_{norm}=\varepsilon_S(T)/\varepsilon_S(T_0)$, $l(T)$ $\varepsilon_S(T)$ - некие текущие значения, $l(T_0)$ $\varepsilon_S(T_0)$ - значения характеристической длины и диэлектрической проницаемости при базовой температуре, например, при T= 300 К.

Для получения оценки влияния температурной зависимости диэлектрической проницаемости на характеристики транзистора численно анализировались характеристики прототипа двух затворного симметричного полностью обедненного КНИ транзистора n-типа с параметрами. Длина затвора $L_g$ = 45 нм, $t_s$ = 10 нм, $t_f$ = 2.5 нм, $N_A = 5.5\text{x}10^{15}$ см$^{-3}$, $N_{ds} = 5\text{x}10^{19}$ см$^{-3}$. Для малосигнального случая при $U_{ds}=U_{gf}=0.1$ В при помощи выражения (3) вычислены значения минимума поверхностного потенциала с учетом и без учета температурной зависимости диэлектрической проницаемости. Результаты моделирования представлены на рис. 2.

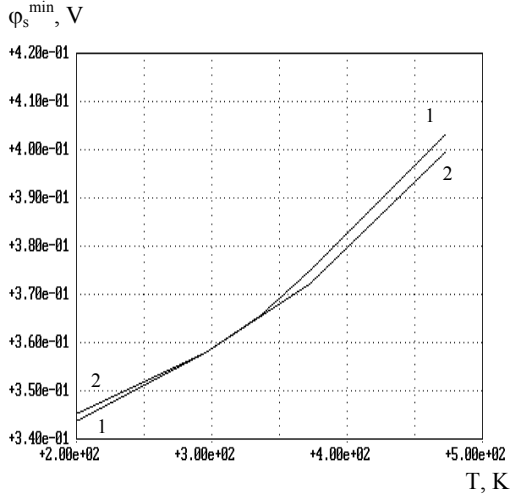



**Рис. 2.** Температурная зависимость минимума поверхностного потенциала
с учетом кривая 1 и без учета кривая 2 температурной зависимости диэлектрической проницаемости

С ростом температуры, а, следовательно, и диэлектрической проницаемости, увеличивается отклонение между значениями минимума поверхностного потенциала соответствующими расчетам с учетом и без учета температурной зависимости диэлектрической проницаемости. Однако, в пределах температурного диапазона 250…350 К значения минимума поверхностного потенциала одинаковы. Максимальное отклонение 0.004 В наблюдается в высоко температурной области. Такое же отклонение сохраняется и при увеличении концентрации легирования рабочей области. Результаты моделирования представлены на рис. 3.

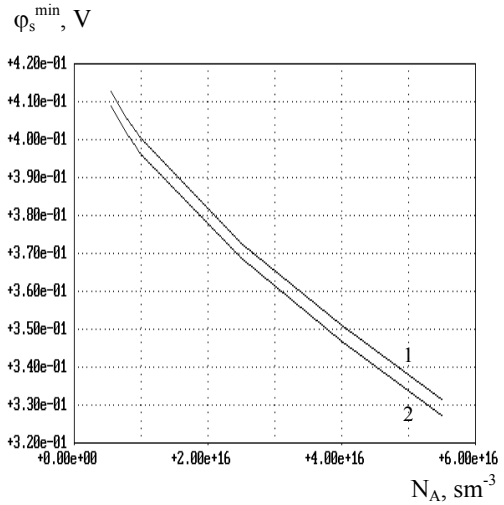



**Рис. 3.** Зависимость минимума поверхностного потенциала от концентрации легирования рабочей области с учетом кривая 1 и без учета кривая 2 температурной зависимости диэлектрической проницаемости T=500 К

Полученные результаты позволяют сделать следующий вывод. Учет температурной зависимости диэлектрической проницаемости кремния приводит к несущественному изменению минимума поверхностного потенциала. Из этого следует, что при расчетах порогового напряжения, в соответствие с общепринятыми методиками, нет необходимости учитывать температурную зависимость диэлектрической проницаемости кремния. И в данном случае к этой физической характеристике можно относиться как к «эффектам второго порядка» Однако, для оценки поведения тока транзистора, который является интегральной относительно потенциала характеристикой, необходим более детальный анализ.

## 3. Аналитическая модель тока транзистора

Выражение для тока двух затворного КНИ транзистора можно получить интегрированием уравнения общего вида по всей рабочей области. При этом концентрация носителей существенным образом зависит от распределения потенциала в рабочей области. Подвижность носителей также определяется данным распределением. Пренебрегая влиянием фиксированного окисного заряда в режиме сильной инверсии, ток в рабочей области, анализируемой двух затворной транзисторной архитектуры можно свести к выражению [8]:

$$I_{ds} = W\mu(y)Q_n(y)\frac{d\varphi_c(y)}{dy}, \qquad (4)$$

где $I_{ds}$ - ток транзистора, W - ширина рабочей области, $\mu(y) = \frac{\mu_0}{(1+(E(y)/E_{cr})^2)^{1/2}}$ - подвижность электронов, где $\mu_0$ - низко полевая подвижность, *E(y)* и $E_{cr}$ - продольное и критическое поле соответственно, $Q_n(y)$ - инверсионный заряд, $\varphi_c(y)$ - центральный потенциал. Для оценки влияния температурной зависимости диэлектрической проницаемости рабочей области необходимо отметить параметры, входящие в (4), которые зависят от температуры. Продольное поле $E(y,T) = \frac{C_f(T)(U_{FB_f}(T) - U_{gf} + \varphi_c(y,T))}{\varepsilon_S(T)}$, критическое поле $E_{cr} = 6.01x10^2 T^{3/2}$, низко полевая подвижность $\mu_0(T)$ (все остальные обозначения сохранены в соответствии с [9]). Взаимосвязь между поверхностным потенциалом $\varphi_s(y,T)$ и центром потенциала $\varphi_c(y,T)$ определяется соотношением

$$\varphi_c(y,T) = (H(T)+1)\varphi_s(y,T) - H(T)\times \times(U_{gf} - U_{FB_f}(T)),$$

где $H(T) = \varepsilon_f t_S / 4\varepsilon_S(T)t_f$. Значение тока $I_{ds}$ будет результатом перехода от двойного интегрирования по координатам к интегрированию по потенциалу пределах от

$$\varphi_{s_1} = \tan^{-1}(\sqrt{A_2(T)})(u_{bi}(T) - (U_{gf} - U_{FB_f}(T))$$

до $\varphi_{s_2} = \tan^{-1}(\sqrt{A_2(T)})(u_{bi}(T) + U_{ds} - -(U_{gf} - U_{FB_f}(T)))$.

Тогда выражения (4) можно записать в следующем виде:

$$I_{ds}(T) = \int_{\phi_{s_1}}^{\phi_{s_2}} d\varphi_s(T)$$

$$\frac{A_3(T)[A_4(T)\varphi_s^{'}(T) - A_1(T)(\varphi_s(y,T))^{1/2}]}{(1 + A_2(T)(\varphi_s^{'}(T))^2)^{1/2}}$$

где $A_1(T) = (2qN_A\varepsilon_S(T))^{1/2}$,

$$A_2(T) = \frac{C_f^2(T)(1+H(T))^2}{\varepsilon_S(T)^2 E_{cr}^2},$$

$$A_3(T) = \frac{2W\mu_0(T)(1+H(T))}{L_g},$$

$$A_4(T) = C_f(T)(1+H(T)),$$

$$\varphi_s^{'}(T) = \varphi_s(T) - (U_{gf} - U_{FB_f}(T))$$

Оценку влияния температурной зависимости диэлектрической проницаемости кремния $\varepsilon_S(T)$ на ВАХ нанотранзистора можно получить из результатов моделирования прототипа, представленного выше. Рассчитанные ВАХ при T=200 и 500 К, представленные на рис. 4, отражают общую тенденцию снижения тока транзистора с ростом температуры, а также иллюстрируют, что при высоких температурах учет зависимости $\varepsilon_S(T)$ приводит к большему сдвигу величины тока транзистора.

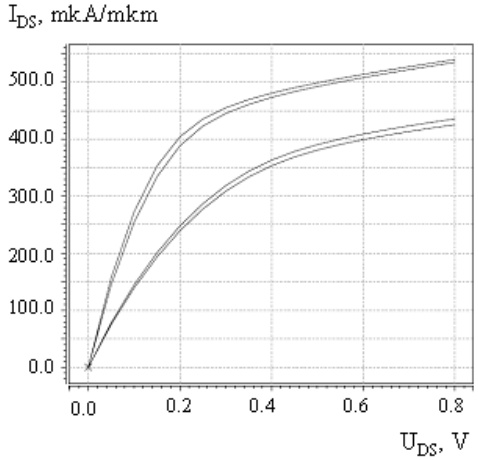



**Рис. 4.** ВАХ транзистора, где верхнее семейство при T=200 К, нижнее при T=500 К

Результаты моделирования, приведенные на рис. 5, подтверждают ранее сделанное заключение об учете температурной зависимости диэлектрической проницаемости кремния при расчетах порогового напряжения. В данном случае обе модельные зависимости практически совпадают.

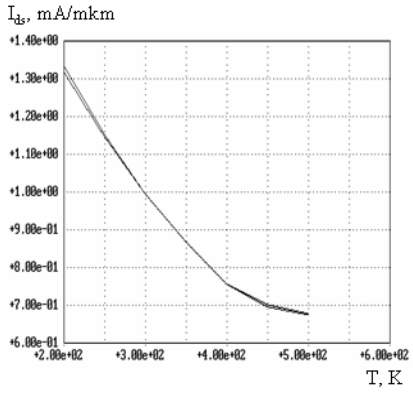



**Рис. 5.** Нормированная зависимость $I_{ds}$(T) при $U_{ds}$=0.05 В и $U_{gf}$=0.8 В для двух случаев

Напротив, ток насыщения, что было отмечено выше, более чувствителен к значению диэлектрической проницаемости рабочей области при данной температуре (см. рис. 6). При этом, чем выше температура, тем больше отклонение. На верхней границе температурного диапазона оно составляет 13 мА\мкм или примерно 3% .

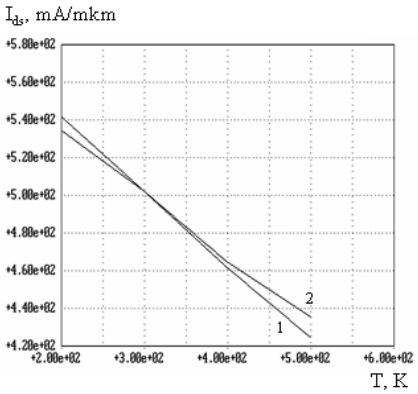



**Рис. 6.** Зависимость $I_{ds}$(T) при $U_{ds}$=0.8 В и $U_{gf}$=0.8 В, где 1 - с учетом зависимости $\varepsilon_S(T)$, 2 – $\varepsilon_S(T) - const$

Наиболее существенный сдвиг температурная зависимость диэлектрической проницаемости кремния привносит в подпороговый ток (см. рис. 7). В данном случае на верхней границе температурного диапазона отклонение составляет 40.3 pA\мкм.

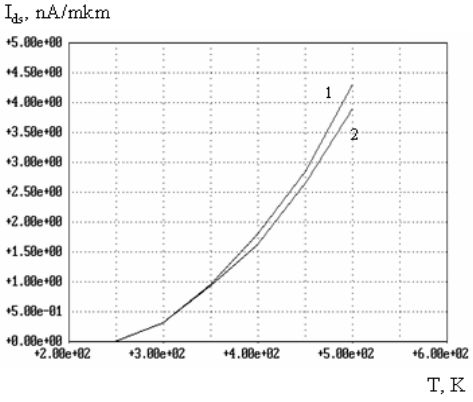



**Рис. 7.** Зависимость $I_{ds}$(T) при $U_{ds}$=0.8 В и $U_{gf}$=0.1 В, где 1 - с учетом зависимости $\varepsilon_S(T)$, 2 – $\varepsilon_S(T) - const$

Для нанотранзисторных структур «германий на изоляторе» учет температурной зависимости диэлектрической проницаемости германия (см. рис. 8) приводит к противоположным результатам: ток насыщения увеличивается, подпороговый ток уменьшается.

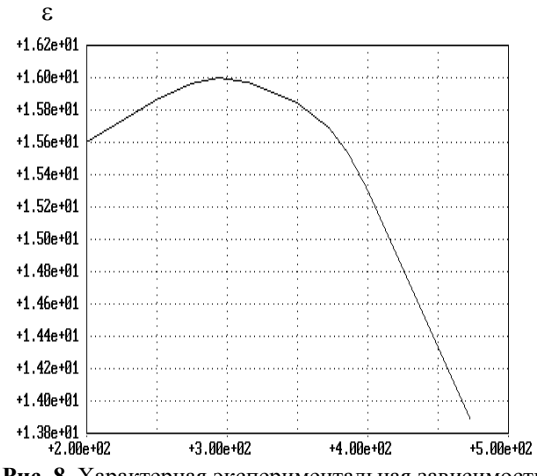



**Рис. 8.** Характерная экспериментальная зависимость диэлектрической проницаемости германия от температуры

Из результатов моделирования приведенных на рис. 9, следует что максимальное отклонение тока насыщения составляет 75.2 мкА\мкм и подпорогового тока 769 рА/мкм.

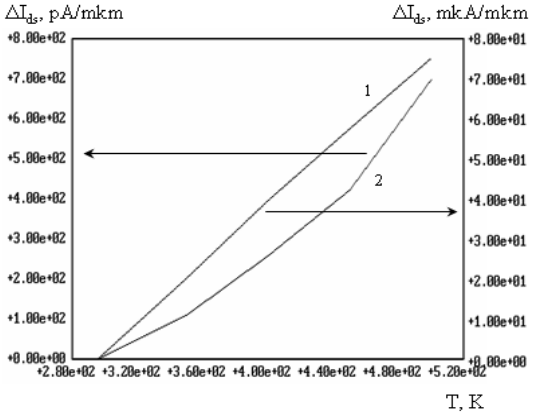



**Рис. 9.** Зависимость сдвига $I_{ds}(T)$ с учетом зависимости $\varepsilon_S(T)$ для транзисторной структуры «германий на изоляторе», где 1- $U_{ds}$=0.8 В и $U_{gf}$=0.8 В, 2 - $U_{ds}$=0.8 В и $U_{gf}$=0.1 В

## Заключение

Рассмотрена аналитическая квазидвумерная модель симметричного двух затворного полностью обедненного КНИ КМОП нанотранзистора. При помощи разработанной модели численно исследовано влияние температурной зависимости диэлектрической проницаемости кремния на распределение потенциала в рабочей области транзистора и на его вольт-амперные характеристики. Учет температурной зависимости диэлектрической проницаемости кремния приводит к несущественному изменению минимума поверхностного потенциала и ей можно пренебрегать при расчетах порогового напряжения. Качественный сдвиг вольт-амперных характеристик, который приводит к их еще большей деградации, наблюдается при температурах выше 400 К. В наибольшей степени это проявляется в уровне подпороговыго тока. В модельных расчетах для транзистора с длиной канала 45 нм в процентном отношении сдвиг составляет примерно 10%. Для тока насыщения - 3%. Для нанотранзисторных структур «германий на изоляторе» учет температурной зависимости диэлектрической проницаемости германия приводит к противоположным результатам: ток насыщения увеличивается на 2%, подпороговый ток уменьшается на 10%.

# The accounting of silicon dielectric properties for circuitry simulation

**E.N. Epikhine, N.V. Masalsky**

**Abstract.** By means of the developed quasi 2D analytical model of the symmetric double gate completely grown poor nanotransistor SOI CMOS influence of temperature dependence of silicon dielectric permittivity on volt ampere characteristics numerically is analyzed. It is marked that high-quality shift of characteristics which brings to them still bigger degradation, is watched at temperatures over 400 T. The greatest quantitative deviation is peculiar to value of subthreshold current.

**Keywords**: silicon on insulator, double gate nanotransistor, temperature dependence of the transistor characteristics, analytical model

## Литература

1. URL: http://public.itrs.net/International technology roadmap for semiconductor 2014 edition. (дата обращения 17.11.2014).
2. J.-P. Colinge. Silicon Insulator Technology: Materials to VLSI. - Boston, Dordrecht, London: Kluwer Acad. Publ., 1997
3. A. Kranti, G. A. Armstrong. Engineering source/drain extension regions in nanoscale double gate (DG) SOI MOSFETs: Analytical model and design considerations // Solid-State Electronics. - №3(50), 2006. P. 437 - 447.
4. Н.В.Масальский. Характеристики двух затворных КНИ КМОП нанотранзисторов для перспективных технологий с низким уровнем потребляемой мощности // Микроэлектроника. – №6(41), 2012, С. 436-444.
5. Б.П. Коршунов, В.П. Калинушкин, Г.В. Козлов, О.И. Сиротинский. Диэлектрические свойства кремния, германия и арсенида галлия в диапазоне субмиллиметровых волн // ФТТ.-№1(31), 1989.С.101-105
6. A. K. Goel, T. H. Tan. High temperature and self-heating effects in fully depleted SOI MOSFETs // Microelectronics Journal. - №4(37), 2006. P. 963-971
7. Н.В. Масальский. Новый подход аналитического моделирования температурных характеристик полевых нанотранзисторов // Евразийский союз ученых.-№ 4(13), 2015. С. 87-91
8. M. K. Pandey, S. Sen, R. S. Gupta. Thermal characterization of double-gate silicon-on-insulator MOSFET// J Phys D: Appl Phys.-№1(32), 1999. P. 344-349
9. F. Balestra, S. Cristoloveanu, M. Benachir, J. Brini, T. Elewa. Double-gate silicon on insulator transistor with volume inversion: A new device with greatly enhanced performance // IEEE Electron Devices Lett.-№2(8), 1987. P. 410-414.

# Подавление эффекта низкочастотного отклонения постоянной составляющей сигнала в приемном тракте последовательного канала, реализованного по технологии 65 нм

**В.В. Мастеров**

**Аннотация:** В статье представлен обзор схем подавления эффекта низкочастотного отклонения постоянной составляющей сигнала, обоснован выбор одной из них для реализации, а также представлены результаты моделирования.

**Ключевые слова**: последовательный канал, постоянная составляющая, джиттер, волновое сопротивление.

## Введение

С увеличением скорости передачи данных в последовательном канале одной из ключевых проблем становится отклонение волнового сопротивления, что ведет к увеличению потерь и отражений сигнала. Одним из источников отклонения волнового сопротивления являются контакты барьерной емкости, размещенной на плате.

В настоящей статье представлены методы решения данной проблемы, а также реализация одного из этих методов и результаты ее моделирования.

## 1. Особенности применения барьерных емкостей

Барьерную емкость размещают между выходом передатчика и входом приемника. Это дает возможность выбрать постоянную составляющую сигнала для этих устройств независимо друг от друга.

Однако, размещение емкости на плате влечет за собой увеличение коэффициента потерь канала, а с увеличением скорости передачи этот параметр становится более критичным. Источниками отклонения волнового сопротивления являются контакты между емкостью и линией передачи.

Для решения этой проблемы барьерные емкости реализуют на кристалле. Однако, на кристалле невозможно сформировать конденсатор с номиналом, сопоставимым с внешней емкостью. Вслед за уменьшением значения емкости уменьшается постоянная времени, что приводит к низкочастотным отклонениям постоянной составляющей сигнала. Этот эффект появляется, когда число «нулей» и «единиц» сильно различается в низкочастотной выборке времени. Пример приведен на рисунке 1. На вход приемника подается сигнал с генератора случайной последовательности PRBS23 на скорости 10Гбит/с.

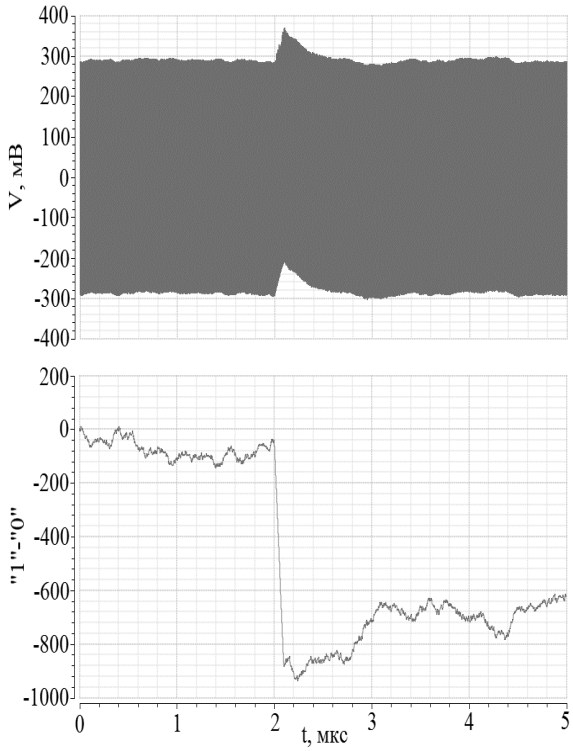



**Рис.1.** а – временная диаграмма сигнала, прошедшего через барьерную емкость; б – разность в количестве между «1» и «0»

## 2. Способы подавления эффекта

Известны два способа решения данной проблемы. Первый способ предполагает использование цепи обратной связи. Идея заключается в том, что цифровая схема считает разницу в количестве между «нулями» и «единицами» за каждый определенный промежуток времени и посылает управляющие сигналы в корректирующую цепь. Концептуально схема представлена на рисунке 2.

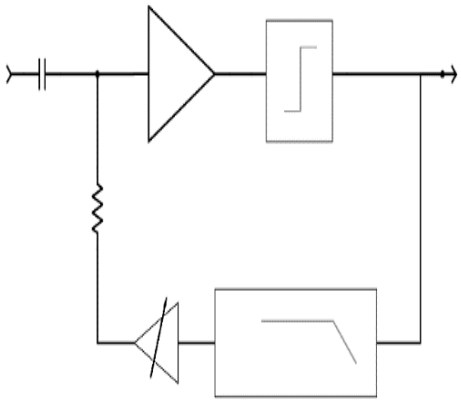


**Рис. 2.** Концептуальное представление корректирующей схемы с использованием обратной связи

Недостатками применения этой схемы являются высокое энергопотребление и большая площадь, занимаемая на кристалле [1].

Вторым методом является применение цепи, состоящей только из пассивных компонентов (рисунок 3).

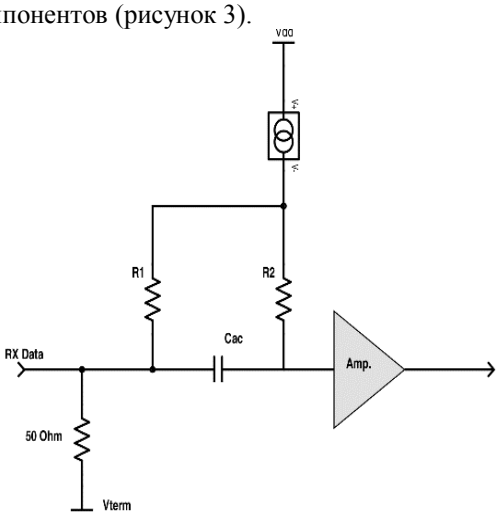



**Рис. 3.** Схема корректирующей цепи с использованием пассивных компонентов

Конструкция включает в себя барьерную емкость (Cac), два резистора, разделяющие вход приемника от входа усилителя (R1 и R2) и источник постоянного тока [2].

Последний способ выбран для реализации по причине малого энергопотребления вследствие использования только пассивных элементов.

Анализ цепи показывает, что передаточная характеристика имеет один полюс и один ноль на частотах

$$f_{pole} = \frac{1}{2\pi (C_{AC}+C_L)(R1+R2)},$$

$$f_{zero} = \frac{1}{2\pi C_{AC}(R1+R2)},$$

где $C_L$ – нагрузочная емкость.

При заданных параметрах R1=8 кОм, R2=132 кОм, $C_{AC}$=1,77 пФ, $C_L$=0,2 пФ, полюс и ноль будут на частотах 570 кГц и 640 кГц соответственно. На рисунке 4 представлена передаточная характеристика, полученная в результате моделирования.

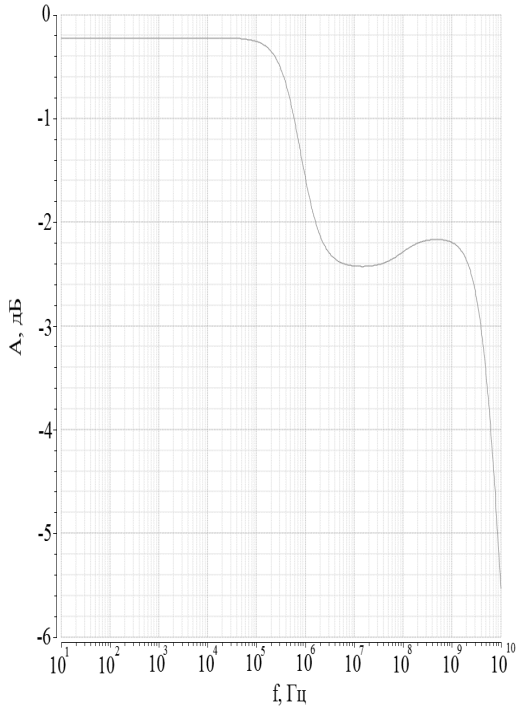



**Рис. 4.** Передаточная характеристика корректирующей цепи с применением пассивных компонентов

В дифференциальных цепях узел между резисторами R1 и R2 должен иметь одинаковый потенциал. Так как после изготовления кристалла номиналы резисторов могут отличаться, необходимо сделать устройство, которое вычислит нужное значение тока в процессе калибровки.

## 3. Моделирование и результаты

Приемник с цепью коррекции реализован по технологии 65 нм.

На рисунке 5 представлены глазковые диаграммы сигналов, полученные с выхода приемного тракта в случаях с корректирующей схемой и без. Входной сигнал подан с генератора PRBS23 на скорости 10Гбит/с. В случае канала без затухания джиттер выходного сигнала уменьшился на 6,48% от единичного интервала.

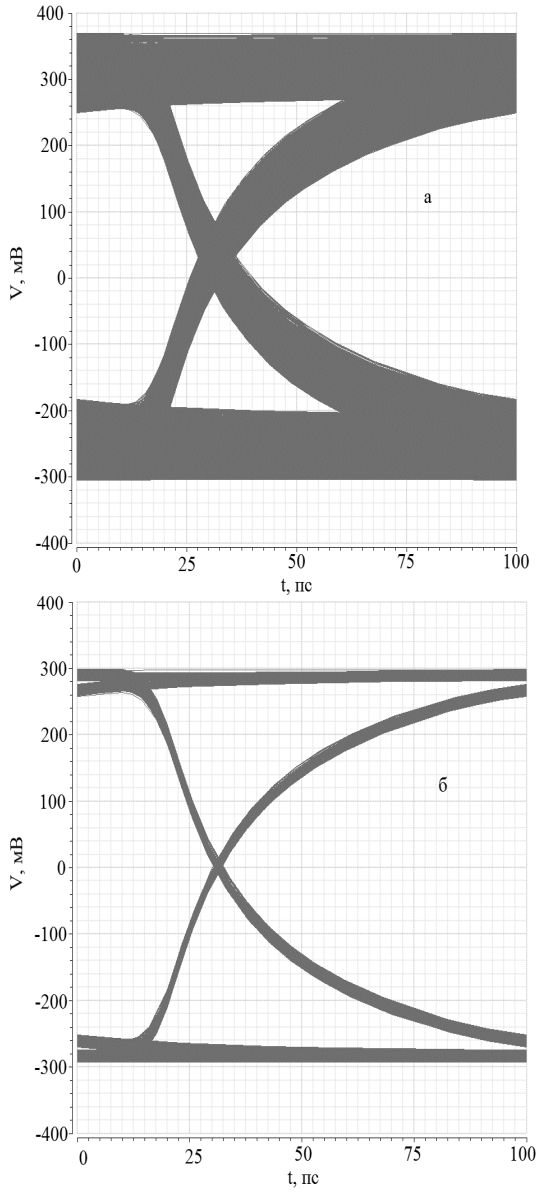



**Рис. 5.** Глазковые диаграммы на выходе приемного тракта. Затухание канала 0 дБ. а – без корректирующей схемы; б – с корректирующей схемой

Аналогично измерен джиттер для случая канала с затуханием 8 дБ (рис. 6). В данном случае он уменьшился на 21% от единичного интервала.

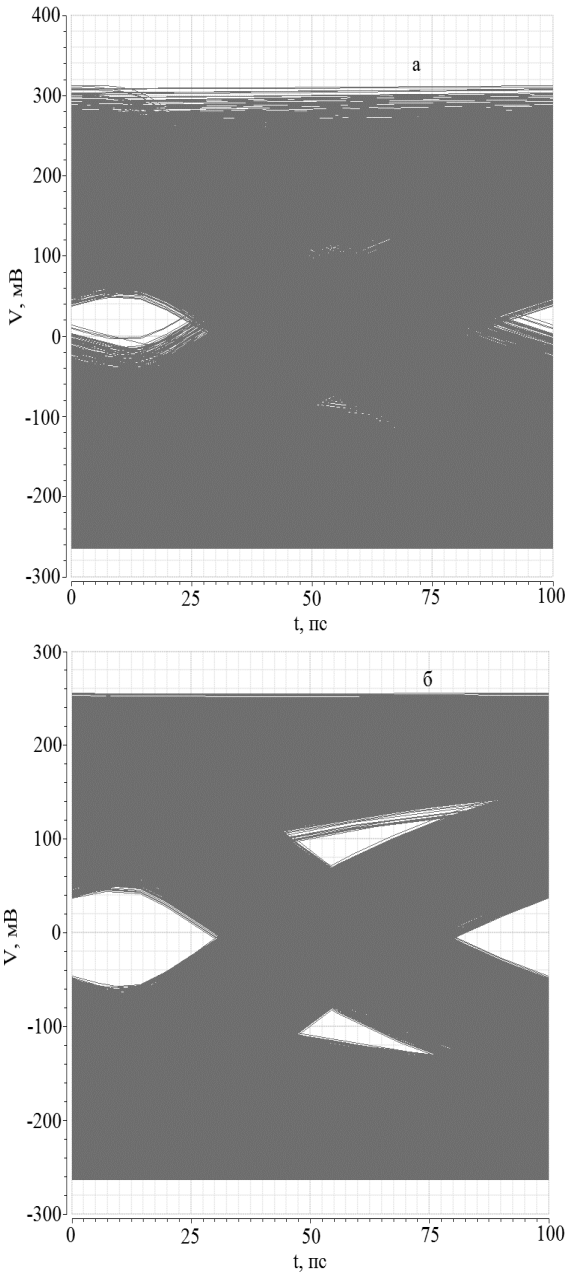



**Рис. 6.** Глазковые диаграммы на выходе приемного тракта. Канал с затуханием 8 дБ. а – без корректирующей схемы; б – с корректирующей схемой

## Заключение

В статье представлен обзор способов решения задачи подавления эффекта низкочастотного отклонения постоянной составляющей сигнала, обоснован выбор одного из них. Представленные результаты моделирования подтверждают улучшения параметров системы в результате использования данной схемы.

# Baseline Wander Correction Circuit for Serial Link Receiver in 65nm CMOS

**V.V. Masterov**

**Abstract:** This paper describes baseline wander correction techniques. One of them is chosen to be implemented in serial link receiver. Simulation results are presented.

**Keywords**: serial link, baseline wander correction, jitter

## Литература

1. Freeman Zhong, Shaolei Quan, A 1.065 ~ 14.025 Gb/s Multi-Media Transceiver With Full-Rate Source-Series-Terminated Transmit Driver and Floating-Tap Decision-Feedback Equalizer in 40 nm CMOS // IEEE Journal of Solid-State Circuits. - Vol. 46, 2011. p. 3126-3139

2. R. Gangasani Guatam, Chun-Ming Hsu, A 32 Gb/s Backplane Transceiver With On-Chip AC-Coupling and Low Latency CDR in 32 nm SOI CMOS Technology // IEEE Journal of Solid-State Circuits. – Vol. 49, 2014. p. 2474-2489

# Верификация мостовой микросхемы PCI-VME с использованием методики UVM

**Н.А. Козлов**

**Аннотация:** В статье поставлена задача произвести выбор метода верификации микросхемы, обосновать выбор метода верификации микросхемы и сравнить выбранный метод с другими способами отладки. Приводится краткая характеристика универсальной методологии верификации UVM, затем описываются основные характеристики мостовой микросхемы PCI-VME и применение методологии UVM для неё. В заключении даётся отчёт о результатах верификации и сравнение с другим способом отладки.

**Ключевые слова:** верификация, микросхема, UVM

В последнее время всё чаще можно наблюдать, что наибольшие усилия по верификации микросхем производятся на уровне RTL [1]. На этом уровне наиболее часто применяется 2 типа тестирования: направленное и случайное [2]. В направленном тестировании проверяется работоспособность при определённой подаче входных воздействий. Достоинством такого способа является простота реализации, недостаток - малая степень покрытия тестовых случаев. В случайном тестировании на объект тестирования подаются случайные воздействия, что даёт покрытие всех входных воздействий в рамках протокола. Наиболее часто используемая методология, которая подразумевает подачу таких воздействий - это UVM.

UVM – это методология тестирования [3], при которой создаётся унифицированное тестовое окружение. Эта унификация даёт ряд преимуществ: тестовое окружение, выполненное по упомянутой методологии для любого проекта понятно всем. Другое преимущество – это возможность повторно использовать имеющееся тестовое окружение для другого проекта. Ещё одно свойство UVM – описание тестов на уровне транзакций. Из-за абстрагирования от уровня сигналов и перехода к транзакциям, написание тестов заметно упрощается. В настоящее время большинство интерфейсов являются стандартными, поэтому UVM особенно актуальна.

В текущей работе стоит цель произвести тестирование микросхемы. Поскольку тестируемая микросхема обладает стандартными распространёнными интерфейсами, методология UVM хорошо подходит к ней.

В данной статье описывается как методология UVM была применена к микросхеме моста. Микросхема является мостом между двумя интерфейсами - PCI и VME. Она может выполнять несколько ролей:

- задатчик
- исполнитель
- инициатор и обработчик прерываний
- арбитр

Все описанные функции относятся как к PCI так и к VME. Трансляция данных может происходить и в режиме одиночных посылок, и в режиме прямого доступа памяти. Специфика тестирования моста заключается в необходимости создания стресс теста, при котором происходят все возможные транзакции одновременно. Создание такого теста заметно упрощается, если перейти от уровня сигналов уровню транзакций, что и обеспечивает UVM.

До начала написания тестового окружения на уровне RTL, микросхема тестировалась на стендовом оборудовании с ПЛИС [4]. В процессе тестирования задача запускалась на управляющем модуле, после чего шел обмен денными. Контроль правильности производился на системном уровне. При возникновении ошибки вручную производился анализ исходного текста микросхемы, после чего на интегрированный в ПЛИС логический анализатор выводились интересующие сигналы. Далее корректировался исходный текст, проект переразводился и снова загружался на ПЛИС. Такая итерация повторялась в процессе поиска и определения ошибок. К достоинствам описанного метода можно отнести сравнительную простоту написания тестов и сложного тестового окружения, высокую скорость тестирования (количество произведённых операций в секунду). Недостатком такого подхода является аппаратная зависимость (отсутствие стандартизации аппаратного тестирования на системном уровне), сложность поиска ошибки при значительном количестве сигналов, ограниченные средства проверки (большие затраты при отслеживании внутренних сигналов), низкая скорость каждой итерации, отсутствие оценки покрытия. Данный метод лучше использовать как финальный

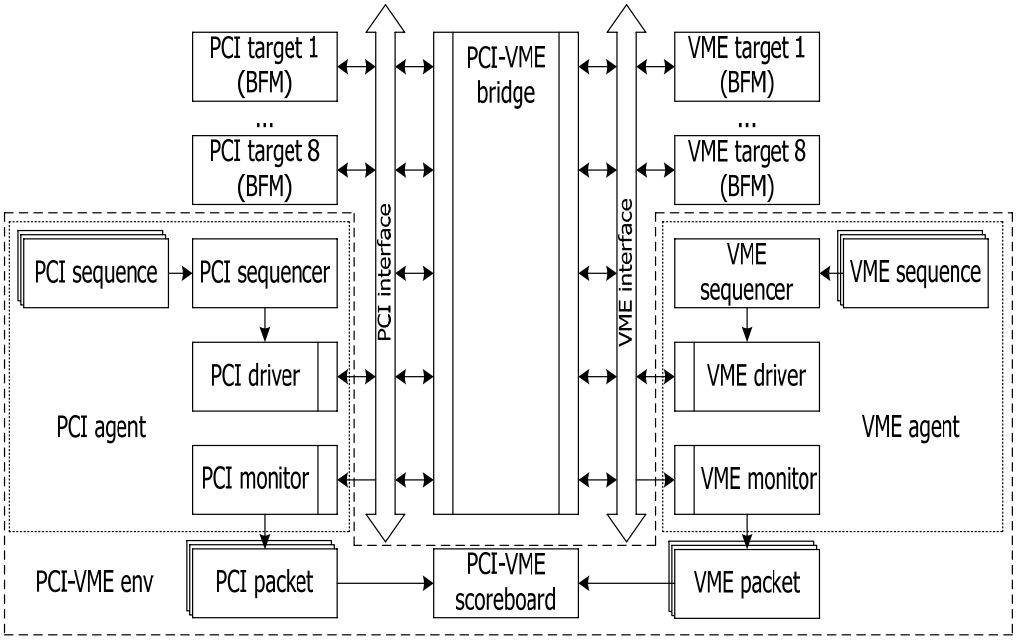



**Рис. 1.** Схема тестирования мостовой микросхемы PCI-VME по методологии UVM

этап проверки. Для тестирования на уровне RTL была создана схема тестового окружения микросхемы моста PCI-VME по методологии UVM, изображенная на рисунке 1. PCI транзакция начинается с формирования последовательности (блок PCI sequence). Последовательность может быть как конфигурационной, так и предназначенной для трансляции PCI-VME. После формирования последовательности, она передаётся в драйвер (блок PCI driver) через блок управления последовательностями (PCI sequencer). После получения последовательности, драйвер начинает вырабатывать сигналы в соответствии с протоколом PCI интерфейса и посылать их на шину (PCI interface). Если был послан пакет трансляции PCI-VME, мост передаст данные на интерфейс VME и обратится к модели исполнителя VME (VME target). Таким способом вырабатываются входные данные для тестируемого устройства. При формировании обращения VME-PCI, цепочка действий полностью аналогична описанной.

В процессе работы тестируемого устройства необходимо проверить корректность его функционирования. По методологии UVM эта проверка должна выполняться автоматизировано в два этапа: сначала фиксируется значение состояний сигналов интерфейса и значения преобразуются в пакеты, затем происходит проверка пакетов. Функцию преобразования сигналов в пакеты (PCI и VME packet) выполняет монитор (блоки PCI и VME monitor). Полученные пакеты отправляются в блок сравнения пакетов (PCI-VME scoreboard). По результатам сравнения ведётся журнал о соответствии пакетов. Таким образом, выполняется автоматизированная проверка работы микросхемы.

После запуска всех тестов было найдено 6 критических ошибок. Эти ошибки не были выявлены ранее при тестировании на стенде. Особенность проведённого тестирования состоит в генерации входных воздействий со случайными параметрами и автоматизированная проверка результатов сравнения. Созданные тесты позволяют максимально загрузить устройство, таким образом, создав необходимый стресс-тест.

В результате работы, кроме выявленных ошибок мы получили набор тестов и тестовое окружение для интерфейсов PCI и VME, написанное на языке SystemVerilog. Поскольку окружение является стандартизованным, его можно использовать в других проектах с этими интерфейсами.

# Verification of chip PCI-VME by UVM methodology

**N.A. Kozlov**

**Abstract:** The purpose of this article is to select a method of verification of the chip, to substantiate the choice of method of verification of a chip and to compare the chosen method with other methods of debugging. The article gives a brief description of the universal verification methodology UVM, and then describes the main characteristics of the bridge chip PCI-VME and the application of the methodology UVM for her. In conclusion, given the report on results of verification and comparison with other method of debugging.

**Keywords:** Verification, integrated circuit, UVM

## Литература

1. J.Bergeron. Writing Testbenches Functional, feb. 2003
2. K.U.Bhaskar. A universal random test generator for functional verification of microprocessors and system-on-chip, jan. 2005
3. Группа Acellerra Systems Initiative http://accellera.org/community/uvm ссылка активна на 21 июня 2014
4. W.Harald. http://www.newelectronics.co.uk/electronics-technology/debugging-methods-for-fpgas/18546/ may 2009, ссылка активна на 21 июня 2014

# Программное обеспечение для автоматизации учетно-управленческой деятельности подразделения по изготовлению единичной продукции

**А.Б. Бетелин**

*кандидат физико-математических наук*

**Аннотация.** В статье приводится общее описание программного обеспечения, предназначенного для автоматизации решения учетных и управленческих задач в подразделении, производящем единичную кабельную продукцию.

**Ключевые слова:** единичное производство, автоматизация, учет и управление, программное обеспечение

## 1. Введение

В НИИСИ РАН ведется разработка высокопроизводительных микропроцессоров и вычислительных систем на их основе [1]. Выполнение подобных работ подразумевает создание экспериментальных образцов аппаратуры, число которых обычно невелико - один или нескольких экземпляров. Эти образцы, как правило, уникальны, и конструкторская документация (КД) на них в ходе разработки постоянно изменяется. Однако применяемые при их создании кабели, заглушки, переходники, адаптеры и т.п. являются вполне серийными изделиями, КД на которые уже устоялась и изменениям практически не подвергается.

Изготовление экспериментальных образцов аппаратуры и кабельной продукции для них осуществляет по заявкам разработчиков специальное производственное подразделение. Оно состоит из нескольких сотрудников, часть которых выполняет организационно-управленческие функции, остальные же заняты собственно выполнением производственных операций по изготовлению требуемых изделий. Характер работы этого подразделения наиболее близок к единичному (штучному) производству [5].

Средства автоматизации, применяемые в подразделении для решения учетных и управленческих задач, представлены в основном офисными программами, работающими на персональных ЭВМ и сохраняющими данные в файлах текстового процессора и электронных таблицах. Однако с увеличением ассортимента выпускаемых изделий и ростом количества заявок на их изготовление возможностей имеющихся аппаратных и программных средств становится недостаточно для оперативного контроля производственного процесса. В связи с этим возникает необходимость повысить степень автоматизации работы подразделения в решении задач учетного характера.

## 2. Производственный процесс и возможности его автоматизации

Процесс выпуска изделия в производственном подразделении состоит из ряда этапов. Начинается он с того, что в подразделение поступает заявка на изготовление изделия с указанием его наименования, обозначения, требуемого количества экземпляров.

Из архива конструкторской документации извлекается КД на заказанное изделие и на основании его спецификации определяется потребность в комплектующих. При этом возможно, что часть комплектующих уже имеется в наличии, а часть необходимо купить дополнительно. Тогда ответственный за закупки сотрудник подразделения организует приобретение недостающих комплектующих, вместе с которыми поступают первичные бухгалтерские документы (товарные чеки, накладные и т.п.).

Далее непосредственному исполнителю выдается задание на изготовление изделия вместе с набором необходимых для этого комплектующих. По окончании работы непосредственный исполнитель сообщает о завершении выполнения задания.

Наконец, готовое изделие передается заказчику по акту приема-передачи. Одновременно составляется отчет об изготовлении изделия с указанием цен на комплектующие и реквизитов первичных документов.

Повысить эффективность решения учетно-управленческих задач на каждом этапе этого процесса возможно, если сократить ручной набор текста, уменьшить объем работы с бумажными документами, предоставить сотрудникам подразделения дополнительные средства для хранения и поиска нужной информации. При этом затрагиваются такие участки деятельности как регистрация заявок на изготовление, хранение спецификаций, учет поступления и расходования комплектующих, хранение отчетности и т.п.

Для автоматизации производственных предприятий предлагается значительное число

готовых программных продуктов [2], многофункциональных и весьма дорогостоящих. Однако для небольшого подразделения и достаточно узкого круга задач их возможности избыточны. К тому же, ввиду ориентации на типовые ситуации, в них не учитываются специфические особенности конкретного производства. Поэтому было принято решение о самостоятельной реализации программного обеспечения (ПО) для автоматизации учетно-управленческой деятельности в рамках описанного выше производственного процесса. Данное ПО реализовано на базе инструментария [4] и входит в состав АСУ НИИСИ РАН в качестве подсистемы.

## 3. Предметная область и ее представление в базе данных

Описываемая подсистема предназначена для автоматизации следующих участков деятельности производственного подразделения:

– регистрация заявок на изготовление изделий, контроль за их выполнением, доступ к отчетности по заявкам;

– доступ к спецификациям изделий и номенклатуре комплектующих;

– учет поступления и расхода комплектующих;

– доступ к справочной информации о комплектующих, их наименованиях и организациях-поставщиках.

Вся информация по указанным темам размещается в таблицах реляционной базы данных, что обусловлено применяемыми инструментальными средствами.

Предметная область представлена моделью, при описании которой употребляются нижеперечисленные понятия.

***Изделие*** – продукт, изготовленный производственным подразделением путем выполнения операций сборки. Для выпуска изделия необходима КД на него.

***Комплектующие*** – используемые при сборке материалы и покупные комплектующие изделия (ПКИ), КД на которые не требуется при их применении.

***Элемент комплектации изделия*** – это либо комплектующее, либо другое изделие, для применения которого необходимо его вначале изготовить в соответствии с КД на него.

***Спецификация изделия*** – список элементов комплектации. Отметим, что понятие "спецификация изделия" не является копией конструкторского документа "спецификация" [3]. Для решения учетно-управленческих задач по организации изготовления изделия из всей информации, содержащейся в конструкторской спецификации, необходим лишь перечень входящих в изделие компонентов.

***Склад комплектующих*** – список имеющихся в наличии комплектующих. На складе фиксируется поступление комплектующих и выделение их для изготовления изделий. Физическое устройство склада никакого значения не имеет.

Модель предметной области включает следующие основные типы объектов:

– заявка на изготовление;

– спецификация изделия;

– элемент комплектации;

– номенклатурное наименование;

– комплектующее;

– складская позиция.

Имеется также ряд второстепенных типов объектов, играющих вспомогательную роль (например, элементы справочников). Каждый объект модели представляется строкой таблицы базы данных. Объект может входить в иерархическую структуру, тогда соответствующая ему строка имеет ссылку на строку объекта, находящегося на ступень выше по иерархии.

Объекты типа "*заявка на изготовление*" хранятся в таблице заявок (рис. 1). Каждая строка этой таблицы содержит идентифицирующие данные: номер заявки и дату ее поступления, наименование и обозначение изделия, нужное количество экземпляров, идентификатор работы, для которой требуется изделие, желательную дату изготовления.

Кроме того, заявка включает еще и организационно-технические данные, такие как номер и дата служебной записки об изготовлении изделия, состояние изготовления, дата фактического завершения работ, даты составления отчета об изготовлении и акта приема-передачи.

Поле "*состояние изготовления*" позволяет контролировать процесс выпуска изделий. Оно может иметь следующие значения:

– "закупка комплектующих";

– "готово к изготовлению";

– "передано исполнителю";

– "изготовлено".


Главное меню → Заявки

Заявки | Отчетность | Спецификации | Номенклатура | Склад | Закупки | Справочники | Иниц. работы | Настройки | Выход

Полный список

| № | Номер заявки | Наименование и обозначение изделия | Кол-во | Инициативная работа | НИОКР | Заказ/этап | Срок исполнения | Сл. записка номер | Сл. записка дата | Состояние изготовления | Дата завершения изготовления | Дата акта приема-передачи | Дат отче |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 910 | ЭН-ОИ.685611.086. Кабель разветвительный CBL_RS50VCI_U... | 2 | | Обработка-10 (зак. 1212) | | | 01-22/1141 | 29.04.14 | Изготовлено | | 30.07.14 | 13 |
| 22 | 905 | ЭН-ОИ.685611.076. Кабель разветвительный CBL_ADP_RS50V... | 2 | | Обработка-10 (зак. 1212) | | | 01-22/1141 | 29.04.14 | Изготовлено | | 17.07.14 | 15 |
| 23 | 904 | ЭН-ОИ.685611.075-02. Кабель разветвительный CBL_ADP_RS... | 2 | | Обработка-10 (зак. 1212) | | | 01-22/1141 | 29.04.14 | Изготовлено | | 17.07.14 | 17 |
| 24 | 901 | ЭН-ОИ.685611.066. Кабель CBL_SWIR_CO8_UD9F | 3 | | Обработка-10 (зак. 1212) | | | 01-22/1141 | 29.04.14 | Изготовлено | | 28.07.14 | 17 |
| 25 | 902 | ЭН-ОИ.685611.067. Кабель корпусной CBL_MKIO_FB4_DB9M | 3 | | Обработка-10 (зак. 1212) | | | 01-22/1141 | 29.04.14 | Изготовлено | | 07.07.14 | 14 |
| 26 | 908 | ЭН-ОИ.685611.018-05. Кабель-переходник K2 | 4 | | Процессор-6 (зак. 1126) | | | 01-22/1153 | 30.04.14 | Изготовлено | | 23.07.14 | 03 |

**Рис. 1.** Таблица заявок

Объекты типа "*спецификация изделия*" располагаются в таблице спецификаций (рис. 2), строки которой содержат наименование и обозначение изделия в соответствии с КД. С объектом типа "спецификация" связаны несколько подобъектов типа "элемент комплектации",

находящиеся в таблице комплектации (рис. 3). Строка этой таблицы может содержать либо наименование комплектующего, либо указание на другую (вложенную) спецификацию. При этом комплектующее представляется ссылкой на номенклатурное наименование, а вложенная спецификация - ссылкой на строку таблицы спецификаций. Также строка таблицы комплектации содержит количество и единицу измерения элементов комплектации.

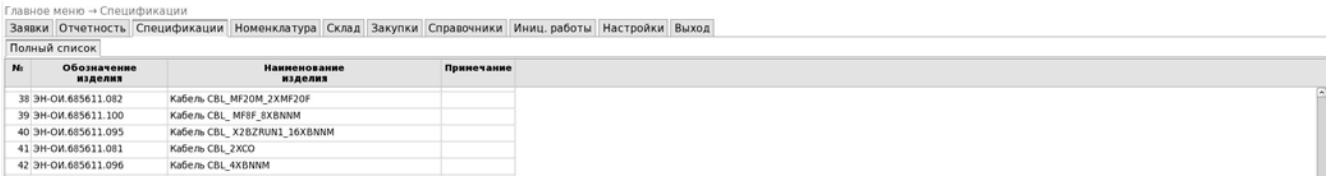



**Рис. 2.** Таблица спецификаций


**Рис. 3.** Таблица комплектации

Множество объектов типа "*номенклатурное наименование*" по сути представляет собой словарь используемых наименований, который хранится в таблице номенклатуры в виде единого списка уникальных строк (рис. 4). Уникальность их обеспечивается средствами СУБД и позволяет автоматически препятствовать повторному вводу наименований комплектующих при внесении новых данных (например, при вводе новых спецификаций).

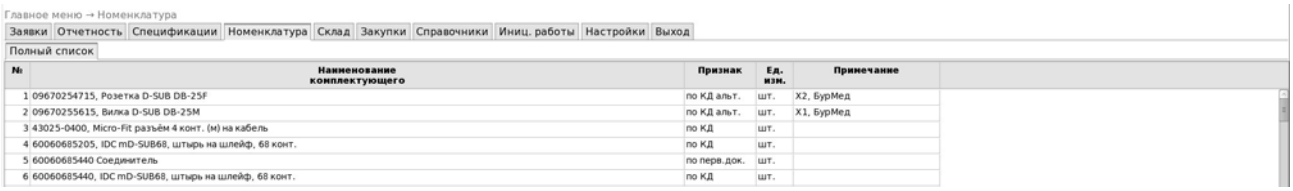



**Рис. 4.** Таблица номенклатуры

В таблицу номенклатуры попадают абсолютно все наименования, когда-либо встретившиеся в КД или же в первичных бухгалтерских документах. При этом возможны ситуации, когда одному и тому же комплектующему соответствуют несколько наименований. Например, термоусадочная трубка может в одной спецификации именоваться как "термоусадочная трубка", в другой - как "трубка термоусадочная", а в товарной накладной - как "кембрик термоусадочный". Формально все эти наименования различны, хотя обозначают одно и то же комплектующее.

В целях упорядочивания наименований были приняты следующие соглашения. Среди всех имеющихся наименований комплектующего наименование по КД считается основным, остальные же (как правило, происходящие из бухгалтерского учета) - его синонимами. Если наименований по КД оказалось несколько, то одно из них выбирается в качестве основного, а прочие считаются альтернативными. При такой ситуации в дальнейшем желательно проведение организационных мероприятий по замене в КД альтернативных наименований на основное.

Указанные взаимоотношения между наименованиями фиксируются в специальной информационной таблице "Справочник наименований" (рис. 5).


**Рис. 5.** Справочник наименований

Каждая строка этой таблицы содержит ссылку на строку таблицы номенклатуры. Кроме того, в справочнике поддерживается древовидная структура, позволяющая хранить и отображать отношения между наименованиями. В этой структуре строка со ссылкой на основное наименование по КД всегда является корнем дерева, а строки со ссылками на синонимы - терминальными узлами этого дерева.

При ручном добавлении наименования в таблицу номенклатуры всегда считается, что это основное наименование по КД. Поэтому введенное вручную альтернативное наименование будет автоматически помечено как основное. Но затем необходимо выполнить операцию привязки этого наименования к основному в качестве альтернативного.

Наименования, происходящие из бухгалтерского учета, автоматически попадают в таблицу номенклатуры и в справочник при вводе первичных бухгалтерских документов. При этом происходит привязка наименования-синонима к основному наименованию по КД.

Объекты типа "*комплектующее*" хранятся в таблице фактической комплектации, где указываются все комплектующие, необходимые для сборки требуемого количества изделий по заявке (рис. 6). Каждая строка этой таблицы содержит ссылку на соответствующую строку таблицы заявок.


Главное меню → Заявки → Заявка 1005

Комплектация по заявке | Спецификация | Дав. сырье | Отчетность

| № | Наименование по КД | Ед. изм. | Кол-во требуемое | Кол-во в резерве | Сколько не хватает | Остаток на складе |
|---|---|---|---|---|---|---|
| 1 | Штекер TS-1R (banana), красный | шт. | 12.00 | 0.00 | 12.00 | 0.00 |
| 2 | Штекер TS-1B (banana), чёрный | шт. | 4.00 | 0.00 | 4.00 | 0.00 |
| 3 | Mini-Fit 8 конт., гнездо (м) на кабель (2660-08M), MF-8F | шт. | 2.00 | 0.00 | 2.00 | 0.00 |
| 4 | Провод МГТФ 1.0 ТУ 16-505.185-71 | м | 6.40 | 0.00 | 6.40 | 0.00 |
| 5 | Кембрик термоусадочный F32-12.0 прозрачный | м | 0.20 | 0.00 | 0.20 | 0.00 |

**Рис. 6.** Таблица фактической комплектации по заявке

Объекты типа "*складская позиция*" располагаются в таблице склада, строки которой содержат наименование комплектующего, его цену, количество, доступное для использования, и количество, уже зарезервированное для работ по заявкам (рис. 7). Таблица склада позволяет учитывать имеющиеся в наличии комплектующие и распределять их для выполнения заявок. В таблице поддерживается древовидная структура, в которой строка с наименованием по КД всегда является корнем дерева, а строки с наименованием по первичному документу - терминальными узлами этого дерева. Это дает возможность пользователю видеть содержимое склада по каждой позиции, включая наименование комплектующего, его количество и цену. Учитывать цену комплектующих приходится из-за того, что использование в составе одного изделия одноименных комплектующих с разной ценой (например, закупленных в разное время или у разных поставщиков) нежелательно.

Следует отметить, что при разработке подсистемы не ставилась задача реализовать полноценный складской учет, сопровождаемый генерацией соответствующих документов, т.к. потребности состояли только в фиксации поступающих комплектующих, оценки их запасов и выделения необходимого их количества для нужд производства.


Главное меню → Склад

Заявки | Отчетность | Спецификации | Номенклатура | Склад | Закупки | Справочники | Иниц. работы | Настройки | Выход

Полный список

| Наименование | Ед. изм. | Кол-во доступное | Кол-во в резерве | Цена, руб. | Цена с НДС, руб. |
|---|---|---|---|---|---|
| кембрик термоусадочный F32-1.0 черн. | м | 0.00 | 0.00 | | |
| Кембрик термоусадочный F32-11.0 бел. | м | 0.00 | 0.00 | | |
| ⊟ Кембрик термоусадочный F32-11.0 чёрн. | м | 2.65 | 0.00 | | |
| Кембрик термоусадочный K32-11 черный (K32-11) | м | 2.65 | 0.00 | 45.08 | 53.19 |
| Кембрик термоусадочный F32-12.0 бел. | м | 0.00 | 0.00 | | |
| ⊟ Кембрик термоусадочный F32-12.0 прозрачный | м | 0.80 | 0.20 | | |
| Кембрик термоусадочный F32-12.0 прозрачный | м | 0.80 | 0.20 | 42.37 | 50.00 |
| Кембрик термоусадочный F32-15.0 бел. | м | 0.00 | 0.00 | | |
| ⊟ Кембрик термоусадочный F32-2.0 бел. | м | 1.00 | 0.00 | | |
| Кембрик термоусадочный F32-2.0 белый (F32-2.0-W) | м | 1.00 | 0.00 | 12.24 | 14.44 |

**Рис. 7.** Таблица склада

## 4. Некоторые особенности изображения объектов модели предметной области

Инструментальные средства предоставляют возможность хранения древовидных структур в таблицах базы данных. Узлами такой структуры являются строки таблицы. Данные об иерархическом положении строки содержатся в ее служебных полях и используются для обеспечения корректного выполнения операций вставки или удаления строк. Кроме того, эти поля обрабатываются специальным образом для наглядного изображения структуры при выводе таблицы на экран.

Например, рассмотрим таблицу склада. Ее основное содержимое - строки с данными из первичных документов, соответствующие реальным закупкам. Добавим к ним строки, которые никаким закупкам не соответствуют, но содержат наименование комплектующего по КД и общее количество таких комплектующих. Сделаем строку с наименованием по КД корнем дерева, а строки с наименованием того же комплектующего по первичному документу - терминальными узлами этого дерева. Тогда изображение таблицы склада на экране позволит в удобной форме видеть, какие

реальные складские позиции соответствуют абстрактной позиции с наименованием по КД.

Практический опыт показывает, что изображение на экране в виде древовидной структуры таких объектов модели предметной области как складские позиции (рис. 7), а также элементы справочников наименований (рис. 5) и комплектующих (рис. 8), является наиболее наглядным для конечного пользователя.

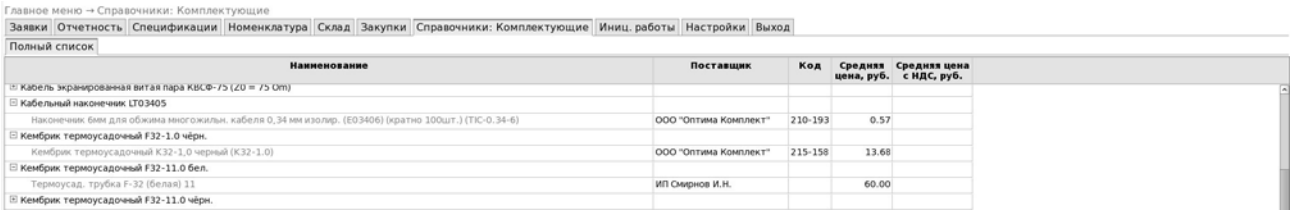



**Рис. 8.** Справочник комплектующих

## 5. Как происходит работа с подсистемой

Конечному пользователю описываемой подсистемы доступны следующие функциональные возможности, предоставляемые в режиме диалога:

– просмотр, модификация и ввод заявок на изготовление изделий;

– просмотр, модификация и ввод спецификаций изделий, включая комплектацию;

– просмотр, модификация и ввод номенклатурных наименований;

– просмотр и модификация справочника наименований;

– просмотр и ввод сканированных образов отчетных документов по заявкам;

– просмотр и модификация состояния склада;

– просмотр справочников комплектующих и организаций-поставщиков.

Часть функций реализована в форме генерируемых отчетов:

– формирование списка комплектующих для выполнения заявки;

– выпуск акта приема-передачи изделия;

– выпуск отчета об изготовлении изделия.

Конечным пользователем подсистемы может быть разработчик аппаратуры, подающий заявки на изготовление изделий, или же сотрудник производственного подразделения, участвующий в выполнении этих заявок. Поэтому в зависимости от круга обязанностей каждому пользователю предоставляется тот или иной набор функциональных возможностей, который регулируется при помощи обеспечиваемого инструментарием механизма ролевого доступа.

Последовательность действий пользователей и соответствующих реакций подсистемы от подачи заявки до выхода готового изделия может быть, например, следующей.

Заявитель (разработчик аппаратуры) регистрирует новую заявку в таблице заявок. Он заполняет поля, позволяющие идентифицировать саму заявку, заявителя, какое изделие необходимо изготовить, в каком количестве, в рамках какой работы. При этом номер заявки и дата ее поступления могут заполняться автоматически.

Обнаружив поступление очередной заявки, ответственный сотрудник производственного подразделения (назовем его "диспетчер") запускает функцию формирования списка необходимых комплектующих, где в том числе указывается наличие их на складе. Если все комплектующие имеются в наличии, диспетчер проставляет в поле "состояние изготовления" значение "готово к изготовлению". В противном случае проставляется значение "закупка комплектующих" и инициируется процесс закупки. Остальные поля заявки заполняются диспетчером по мере появления актуальной информации.

После того, как необходимые комплектующие закуплены, их необходимо учесть путем ввода информации с первичных бухгалтерских документов, таких как товарные чеки, товарные накладные и т.п.

Разумеется, впоследствии эти комплектующие будут зафиксированы в бухгалтерской программной системе. Но при бухучете нередко происходит замена наименования из первичного документа на наименование из справочника "номенклатура" бухгалтерской системы. Это неприемлемо при составлении отчета об изготовлении, где в обязательном порядке перечисляются использованные комплектующие с наименованиями, указанными в первичном документе, а также реквизиты этого первичного документа. Еще одна особенность бухгалтерского учета - данные об использовании комплектующих в систему бухучета вносятся с задержкой, что не позволяет иметь оперативную информация о состоянии запасов комплектующих.

Ввиду перечисленных обстоятельств в подсистеме реализован собственный учет комплектующих, который устроен следующим образом.

Ввод информации с первичных документов происходит путем заполнения экранной формы-бланка с полями, обычно присутствующими в таких документах (рис. 9). Это наименование комплектующего, его количество и цена, номер и дата первичного документа, название организации-поставщика. Все эти данные должны быть введены так, как они указаны в документе. Кроме того,

обязательно должна быть введена дополнительная информация - наименование данного комплектующего в соответствии с КД. Тем самым устанавливается соответствие наименований по КД и по первичному документу. Введенные данные проверяются на корректность и поступают на обработку.

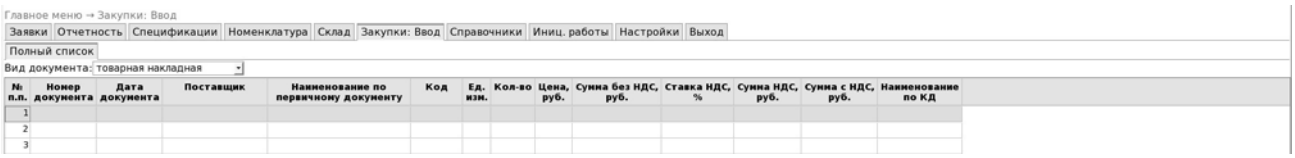



**Рис. 9.** Экранная форма для ввода первичных документов

Если введенное наименование по первичному документу отсутствует в номенклатуре, то оно добавляется как в номенклатуру, так и в справочник наименований. Точно так же новое наименование добавляется в справочник комплектующих вместе с данными о поставщике и о цене закупки. Если же такое наименование уже присутствует в справочнике комплектующих, то перевычисляется лишь средняя цена комплектующего.

Также информация из первичных документов записывается еще в две таблицы - таблицу склада и таблицу закупок (рис. 10). В таблице склада создаются строки с наименованием по первичному документу, количеством и ценой приобретенных комплектующих. Таблица закупок содержит записи о закупках комплектующих в хронологическом порядке и выполняет информационную функцию - в любой момент можно узнать, что именно было куплено, когда, по какой цене, у какого поставщика.

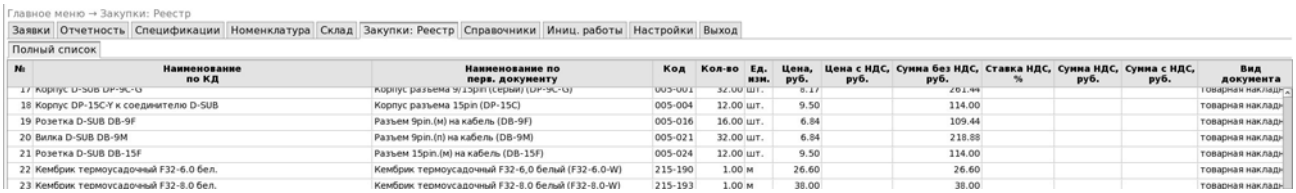



**Рис. 10.** Таблица закупок

Распределение комплектующих со склада по заявкам происходит следующим образом. С таблицей заявок связана таблица фактической комплектации, где перечисляются все комплектующие, необходимые для выполнения заявки. Пользователю необходимо открыть таблицу фактической комплектации по конкретной заявке, а затем для каждого из комплектующих из этой таблицы выбрать подходящую складскую позицию (рис. 11).


**Рис. 11.** Выбор складской позиции

Как только складская позиция выбрана, количество комплектующих, требуемое для выполнения заявки, переходит из категории доступных в категорию зарезервированных, а количество доступных комплектующих на этой позиции уменьшается на величину резерва.

Если на складской позиции недостаточно комплектующих для выполнения заявки, то резервируется все, что там есть, после чего необходимо выбрать другую подходящую складскую позицию и там зарезервировать недостающие комплектующие.

Когда распределение всех необходимых комплектующих закончено, можно выполнить операцию отправки в производство, которая приводит к обнулению всего, что было зарезервировано на складе по данной заявке. Попутно генерируется перечень комплектующих для передачи непосредственному исполнителю работ.

Если непосредственный исполнитель готов к выполнению заявки, ему выдается набор комплектующих. В поле "состояние изготовления" диспетчер проставляет значение "передано исполнителю". Значение "изготовлено" может проставить только сам исполнитель, сигнализируя о том, что он закончил работу над заявкой.

## 6. Заключение

В настоящее время описываемое ПО находится в опытной эксплуатации в производственном подразделении НИИСИ РАН. На ряде учетных операций отмечено сокращение ручного ввода текста. Повысилась скорость работы сотрудников

благодаря централизации хранения информации и наличия развитых средств поиска. В будущем планируется дальнейшее увеличение производительности труда в подразделении за счет расширенного применения в работе ПО автоматических алгоритмов обработки данных.

# Software for automation of accounting and management in subdivision for the single-unit production

**A.B. Betelin**

**Abstract.** This article gives a general description of the software for automation accounting and management tasks in subdivision manufacturing the single-unit cable production.

**Keywords:** single-unit production, automation, accounting and management, software.

## Литература

1. В.Б.Бетелин. Отечественные суперкомпьютерные технологии экзафлопсного класса - необходимое условие обеспечения технологической конкурентоспособности России в XXI веке. "Программные продукты и системы", 2013, №4, 4-9.

2. Б.Гайфуллин, И.Обухов. Современные системы управления предприятием. "Компьютер Пресс", 2001, №9. http://compress.ru/article.aspx?id=11760 (15.09.2015)

3. ГОСТ 2.106-96 "Единая система конструкторской документации. Текстовые документы". Минск, Стандартинформ, 2007. http://www.internet-law.ru/gosts/gost/4669/ (15.09.2015)

4. И.Б.Егорычев. Инструментарий для построения автоматизированных учетных систем с web-интерфейсом. "Программные продукты и системы", 2008, №4, 49-52.

5. URL: https://ru.wikipedia.org/wiki/Единичное_производство (15.09.2015)

# Проблемы мобильности данных числового формата в отчетных документах

**Г.Л.Левченкова**

**Аннотация:** Исследуются особенности объединения информации из нескольких отчетных документов, полученных путем экспорта из баз данных, или изготовленных вручную в виде файлов, содержащих таблицы данных.

**Ключевые слова:** информация, база данных, табличные данные.

## 1. Введение

С развитием информационных технологий всё большее число предприятий переходит на компьютерные системы учета разнообразной информации. Современные базы данных успешно справляются с задачами формирования различных отчетов и сводок и, в большинстве случаев, позволяют сохранять результаты оформленных требуемым образом документов в виде файлов, содержащих таблицы. Кроме того, наличие на рабочих местах большинства сотрудников компьютеров способствует тому, что практически любые заметки, записи, сведения и данные теперь не записываются на листе бумаги или в Книги учета, а вводятся в электронном виде в файлы (преимущественно форматов Microsoft Word и Microsoft Excel).

При работе с электронными документами может возникнуть необходимость в подготовке различных сводок для представления на анализ. Зачастую при этом требуется дополнительная обработка созданных инструментами баз данных файлов с целью их объединения и оформления окончательного отчета в требуемой форме. Для составления отчетов приходится также использовать данные из "внешних" файлов, не являющихся результатом экспорта из баз данных. Типы и форматы данных во всех этих документах не согласованы между собой, а в документах, созданных "вручную", встречаются однотипные данные, намеренно занесенные в ячейки с различными форматами. Особые сложности возникают, когда информация должна быть объединена из всех вышеперечисленных источников в один итоговый файл. В таких ситуациях приходится решать вопросы мобильности данных, которым и посвящена предлагаемая статья.

Рассматриваются процедуры объединения информации из файлов Microsoft Word (*.doc, *.docx) и Microsoft Excel (*.xls, *.xlsx). Важной задачей является обеспечение корректности переноса информации, в частности, числовых данных, которые могут представлять, например, денежные суммы, потому что из-за больших объемов данных провести проверку итога можно будет только выборочно либо визуально.

Опыт объединения сведений из разнородных файлов в сводки показал, что копирование информации через буфер обмена Microsoft Windows не обеспечивает мобильность данных, поскольку в него заносятся как сами данные, так и их тип. Если тип данных в исходном документе не совпадает с типом данных в соответствующей области итогового (выходного) документа, то при копировании может произойти их искажение. Некоторые аспекты мобильности информации уже были рассмотрены ранее (см. [3] и [4]). В настоящей статье предлагается простая методика, обеспечивающая корректный перенос данных при подготовке сводных документов.

Наиболее информативным в качестве примера изменяющихся "типов данных" в нашем случае является, конечно, Microsoft Excel.

Рассмотрим особенности копирования информации, которые помогут предотвратить некорректный перенос данных в сводку и исключат необходимость их проверки вручную.

## 2 Задача

На примере составления сводки из информации, содержащейся в разных файлах, покажем "подводные камни", выявляющиеся при копировании данных.

Допустим, у нас есть три файла, содержащие данные о закупках неких соединителей. Данные в этих файлах занесены в таблицы, колонки которых содержат необходимые для сводки сведения. Требуется объединить информацию в один "лист Microsoft Excel", чтобы можно было производить необходимый отбор сведений, сортировку данных, или вычислять итоговые суммы получений.

## 3 Имеющаяся информация

Обзор исходных данных.

Первый файл (файл_1) содержит информацию в виде таблицы Microsoft Excel (см. Рис. 1).

| | Наименование соединителя | Кол-во | Цена | Стоим. | Организация | курс $ | дата получения |
|---|---|---|---|---|---|---|---|
| 1 | 083614-9012 | 10 | 243,26 | 2432,6 | ООО Страт | 33,0001 | 01.10.2013 |
| 2 | 083614-9012 | 4 | 315 | 1260 | ОАО Имидж | 32,9998 | 16.01.2014 |
| 3 | 1725 095 2102 | 12 | 228,81 | 2745,72 | ОАО Имидж | 35,0002 | 02.07.2014 |
| 4 | 083614-9012 | 2 | 419,96 | 839,92 | ООО Пират | 34,9999 | 25.09.2014 |
| 5 | 1725 095 2102 | 6 | 549,15 | 3294,9 | ООО Пират | 34,9998 | 25.11.2014 |
| 6 | 083614-9012 | 3 | 338,98 | 1016,94 | АНО Пламя | 39,9995 | 15.12.2014 |
| 7 | 083614-9012 | 2 | 243,26 | 486,52 | АНО Пламя | 54,9989 | 16.02.2015 |

**Рис.1.** Вид первого файла исходных данных

| Организация | дата получения | курс $ | Наименование соединителя | Кол-во | Цена | Стоимость |
|---|---|---|---|---|---|---|
| ООО Страт | 04 октября 2013 | 33 | 17 250 952 102 | 4 | 158,51 | 634,04 |
| ОАО Имидж | 22 апреля 2014 | 35 | 1725 095 2102 | 4 | 263,55 | 1054,2 |
| АНО Пламя | 15 декабря 2014 | 39,99 | 1725 095 2102 | 6 | 297,03 | 1782,18 |
| ООО Пират | 25 декабря 2014 | 35 | 083614-9012 | 3 | 419,96 | 1259,88 |
| ООО Томас | 06 мая 2015 | 52,99 | 1725 095 2102 | 3 | 364,41 | 1093,23 |
| ООО Пират | 29 мая 2015 | 62 | 1725 095 2102 | 3 | 440,68 | 1322,04 |

**Рис.2.** Вид второго файла исходных данных

| дата получения | курс $ | Стоим. | Наименование соединителя | Кол-во | Цена | Организация |
|---|---|---|---|---|---|---|
| 20.06.2014 | 35,0002 | 620,00 | 083614-9012 | 2 | 310,00 | ООО Томас |
| 16.01.2014 | 32,9999 | 2925,00 | 1725 095 2102 | 13 | 225,00 | ОАО Имидж |
| 11.06.2013 | 31,6387 | 317,02 | 1725 095 2102 | 2 | 158,51 | ООО Страт |

**Рис.3.** Вид третьего файла исходных данных

| Организация | Наименование соединителя | Общая Стоимость | дата получения | курс $ | Кол-во |
|---|---|---|---|---|---|
| ООО Пират | 1725 095 2102 | 549,15 | 25.09.14 | 35,0000 | 1 |
| ООО Томас | 083614-9012 | 3305,10 | 06.05.15 | 53,0000 | 3 |
| ООО Пират | 083614-9012 | 10025,40 | 29.05.15 | 62,0000 | 12 |

**Рис.4.** Первоначальный вид итогового файла данных

Второй файл (файл_2) также имеет формат таблицы Microsoft Excel и выглядит, как изображено на Рис.2.

Третий файл (файл_3) содержит данные, структурированные в таблице Microsoft Word (см. Рис. 3).

Четвертый файл формата Microsoft Excel (файл_4) имеет информацию, которую нужно дополнить сведениями из первых трех таблиц (см. Рис. 4).

Очевидно, что указанные файлы имеют столбцы одинакового содержания, поэтому легко можно будет скопировать нужные данные выделением определенной колонки, и перенести всю информацию в один итоговый файл, расположив однотипные данные должным образом в требуемом порядке.

# 4 Результат переноса данных

Откроем файл_4 (который и будет итоговым) и перенесем в него копированием столбцы из исходных трех файлов данных (см. Рис. 5).

Поскольку необходимо ввести дополнительные и проверочные вычисления (формулы в ячейки) для проверки сумм "Всего" по файлам и для вычисления долларового эквивалента цены соединителя в каждой партии (за штуку), попробуем добавить соответствующие формулы. Однако, при вводе этих вычислений возникают сложности, которые связаны, к нашему удивлению, с поведением преобразования при начальном вводе наименования соединителя в итоговый файл. А именно: создав файл_4 сотрудник ввел наименование:

1725 095 2102

нажал Enter и в ячейке увидел запись

17 250 952 102

Другими словами, вместо обозначения соединителя ввелось (отобразилось) числовое значение. Мало того, если бы наименование начиналось с нуля, то он бы (ноль) вообще бы пропал, как незначащий символ, что есть совсем не правильно. Поэтому сотрудник, не долго думая, установил на всем листе "формат ячейки - текстовый" и продолжил вводить данные. Теперь они не искажались. Но и формулы перестали вводиться. При попытке просуммировать значения столбца "Общая Стоимость" для файла_2 в ячейке D19 вместо

Проанализируем, что у нас получилось (см. Рис. 5) после копирования информации в итоговый файл.

Сразу обращаем внимание на то (на Рис. 5 такие места окрашены в темный цвет), что вместо чисел в некоторых ячейках отображаются совсем не числовые величины или явно неправильные числа (например, сумма в ячейке D4). И наименования соединителей в ячейках C21 и C22 превратились в числовые значения.

| A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| | Организация | Наименование соединителя | Общая Стоимость | дата получения | курс $ | Кол-во | Примечание | | $ эквивалент цены |
| 1 | ООО Пират | 1725 095 2102 | 549,15 | 25.09.14 | 35,0000 | 1 | было в файле 4 | | 15,69 |
| 2 | ООО Томас | 083614-9012 | 3305,10 | 06.05.15 | 53,0000 | 3 | было в файле 4 | | 20,78679245 |
| 3 | ООО Пират | 083614-9012 | 10025,40 | 29.05.15 | 62,0000 | 12 | было в файле 4 | | 13,475 |
| 4 | | Всего на сумму: | 0,00 | | | | | | |
| 5 | ООО Страт | 083614-9012 | #ЗНАЧ! | 01.10.2013 | 33,0001 | 10 | из файла 1 | | #ЗНАЧ! |
| 6 | ОАО Имидж | 083614-9012 | #ЗНАЧ! | 16.01.2014 | 32,9998 | 4 | из файла 1 | | #ЗНАЧ! |
| 7 | ОАО Имидж | 1725 095 2102 | #ЗНАЧ! | 02.07.2014 | 35,0002 | 12 | из файла 1 | | #ЗНАЧ! |
| 8 | ООО Пират | 083614-9012 | #ЗНАЧ! | 25.09.2014 | 34,9999 | 2 | из файла 1 | | #ЗНАЧ! |
| 9 | ООО Пират | 1725 095 2102 | #ЗНАЧ! | 25.11.2014 | 34,9998 | 6 | из файла 1 | | #ЗНАЧ! |
| 10 | АНО Пламя | 083614-9012 | #ЗНАЧ! | 15.12.2014 | 39,9995 | 3 | из файла 1 | | #ЗНАЧ! |
| 11 | АНО Пламя | 083614-9012 | #ЗНАЧ! | 16.02.2015 | 54,9989 | 2 | из файла 1 | | #ЗНАЧ! |
| 12 | | Всего на сумму: | #ЗНАЧ! | | | | | | |
| 13 | ООО Страт | 1725 095 2102 | 634,04 | 04 октября 2013 | 33 | 4 | из файла 2 | | 4,803333333 |
| 14 | ОАО Имидж | 1725 095 2102 | 1054,2 | 22 апреля 2014 | 35 | 4 | из файла 2 | | 7,53 |
| 15 | АНО Пламя | 1725 095 2102 | 1782,18 | 15 декабря 2014 | 39,99 | 6 | из файла 2 | | 7,427606902 |
| 16 | ООО Пират | 083614-9012 | 1259,88 | 25 декабря 2014 | 35 | 3 | из файла 2 | | 11,99885714 |
| 17 | ООО Томас | 1725 095 2102 | 1093,23 | 06 мая 2015 | 52,99 | 3 | из файла 2 | | 6,876957917 |
| 18 | ООО Пират | 1725 095 2102 | 1322,04 | 29 мая 2015 | 62 | 3 | из файла 2 | | 7,107741935 |
| 19 | | Всего на сумму: | 7145,57 | | | | | | |
| 20 | ООО Томас | 083614-9012 | 620 | 20.06.2014 | 35,0002 | 2 | из файла 3 | | 8,857092245 |
| 21 | ОАО Имидж | 17 250 952 102 | 2925 | 16.01.2014 | 32,9999 | 13 | из файла 3 | | 6,818202479 |
| 22 | ООО Страт | 17 250 952 102 | 317,02 | 11.06.2013 | 31,6387 | 2 | из файла 3 | | 5,010003572 |
| 23 | | Всего на сумму: | 3545,00 | | | | | | |

**Рис.5.** Результат переноса данных в итоговый файл

числового значения итоговой суммы мы увидим текст "=СУММ(D13:D18)". Поэтому в сводном файле нужно восстановить "формат ячейки – числовой" (или Общий) для тех мест, куда планируется вводить числовые значения или формулы. Отметим, что если в "текстовую ячейку" сначала ввести "вычисление по формуле/сумму" и только потом, спохватившись, что отображается нежелательный результат, назначить этой ячейке общий или числовой формат, то ничего не изменится в лучшую сторону. Сначала придется удалить все данные из ячейки, назначить ей общий/числовой формат и только потом вставлять сумму/формулу в неё.

В случае отображения "#ЗНАЧ!" вставлялась ячейка из файла_1, которая в другом файле была формулой, она так и перенеслась – формулой, только вычисления теперь ссылаются совсем не на те ячейки, откуда раньше брались исходные данные для формул. Чтобы скопировать числовые значения, а не формулы, придется использовать Microsoft Word (или любой другой текстовый редактор, в который можно перенести данные из таблицы через буфер "копировать – вставить"). Нужно открыть файл_1 (исходный), скопировать столбец с суммами "Стоим.", которые являются ячейками с вычислением по формуле, и "Вставить" этот столбец в файл Microsoft Word. Потом скопировать данные (получившийся столбец) из файла Microsoft Word и вставить вместо столбца в итоговом

файле_4 в соответствующее место. Поскольку формулы в строке "Всего на сумму (по файлу_1)" и "$ эквивалент цены" (ячейки колонки J) уже внесены, то и данные в ячейках D12 и J5–J11 в итоговом файле сразу примут корректный вид.

Теперь о сумме в ячейке D4. Как уже было сказано, в строках 1, 2 и 3 первоначального файла_4 была занесена текстовая информация. А формула подсчета суммы (∑) считает текст нулевым числовым значением. Отметим, что именно формула "∑", так как вычисления по любой другой формуле (например, вида "= D2/F2/G2" в случае подсчета $ эквивалента цены) работают с текстовыми данными в численном отображении безукоризненно. Чтобы "переделать" текстовые данные "Общей стоимости" из первых трех строк придется опять воспользоваться переносом столбца в файл Microsoft Word и обратно. Никакие другие ухищрения изменить текст на число в столбце/строке внутри файлов Microsoft Excel не помогут – ни изменение формата ячейки на числовой/общий, ни удаление информации с последующей установкой формата ячейки, ни копирование на другой лист. Можно, конечно, удалить данные и, установив нужный формат ячейки, ввести

| A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| 20 | ООО Томас | 083614-9012 | 620,00 | 20.06.2014 | 35,0002 | 2 | из файла 3 | | 8,857092245 |
| 21 | ОАО Имидж | 1725 095 2102 | 2925,00 | 16.01.2014 | 32,9999 | 13 | из файла 3 | | 6,818202479 |
| 22 | ООО Страт | 1725 095 2102 | 317,02 | 11.06.2013 | 31,6387 | 2 | из файла 3 | | 5,010003572 |
| 23 | | Всего на сумму: | 3545,00 | | | | | | |

**Рис.6.** Некорректная сумма по файлу_3

суммы вручную,… но…, а если у нас не три строки, а шестьдесят? Поэтому копируем часть столбца в Word-файл и переносим обратно полученные данные. Ячейки D1-D3 при этом становятся формата "Общий", а в ячейке "Всего на сумму" у файла_4 появляется результат правильного сложения.

Сложнее обстоит дело с устранением автоматического преобразования копируемого текста в числовое выражение (наименование соединителя, ячейки C21 и C22 на Рис. 5), которое произошло при вставке таблицы из Word-файла_3. После копирования в Excel-файл_4, ячейки с этими наименованиями стали числовыми. Нет никакой возможности автоматически привести их к виду с одним пробелом между группами цифр (к виду, как в других строках сводного файла – "1725 095 2102"). Придется полуручным образом делать соответствующие ячейки одинаковыми: сначала установим у всей колонки C текстовый формат. Потом заменим

"17250952102"
на "1725  095  2102"

(вставим по два пробела между группами чисел наименования, один пробел вставить не получится).

Затем в остальных "хороших" строках столбца C также заменим один пробел на два, чтобы окончательные наименования совпадали по всему файлу:

"1725 095 2102" меняем
на "1725  095  2102"

Теперь все соединители этого наименования можно будет выбрать по фильтру и т.д. – теперь они одинаковые.

Далее, прежде чем сформировать окончательную версию сводки для печати, приведем данные к более или менее одинаковому виду. А именно:

- отформатируем колонку "Наименований" по левому краю;
- в колонках, где проставлены числа (которые, как стоит отметить, не обязательно являлись числовыми ячейками), установим точность (два знака после запятой для сумм в рублях и четыре для курса $).

После всех этих "выравниваний" сразу бросается в глаза несоответствие в итоговой сумме по файлу_3 (см. Рис. 6).

Проверка формата ячеек (после того, как мы установили у этой колонки формат числовой, с точностью до двух знаков после запятой) нам ничего не даёт. Формула ∑ в ячейке D23 имеет правильное содержание – "=СУММ(D20:D22)". Но в итоговой сумме явно не хватает прибавленного числа 317,02. Ситуация возникла из-за того, что в процессе переносов (копирований) в текстовую ячейку D22 было записано текстовое значение 317,02. После изменения формата этой ячейки на "числовую - два знака после запятой" СУММА "∑" (ячейка D23) не сочла это изменение существенным, и продолжает считать бывшее текстовое значение 317,02 нулевым числом. Придется еще раз скопировать эту ячейку из Word-файла_3, чтобы она превратилась в "общий" формат, и затем повторить все необходимые форматирования. Теперь сумма D23 в порядке. Еще раз отметим, что значение $ эквивалента (J22) было корректным даже в случае текстового значения этой злополучной ячейки.

Добавим в итоговую сводку сведения о затратах на закупку в $ эквиваленте - столбцы с ориентировочной (вычисляемой) суммой для каждой партии. Для выявления еще одного "ляпа" нам нужно четыре столбца, а не один. Первый из столбцов – G (см. Рис. 7) - имеет формат "Общий" и вычисляется, другой – H – имеет формат "числовой, число десятичных знаков = 2" и также вычисляется, третий – I – содержит скопированные через промежуточный Word-файл ячейки столбца G с последующей установкой формата

ячеек "числовой, число десятичных знаков = 2", четвертый – J – имеет формат "числовой, число десятичных знаков = 2" и вычисляется функцией ОКГУГЛЕНИЕ от деления с точностью 2 знака после запятой.

Обратите внимание на суммы "Общих затрат", которые получаются при вычислении по столбцам (Рис. 7). Нас интересуют колонки, в которых два

I4 fx 44,565641

| A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| | Организация | Наименование соединителя | Общая Стоимость | курс $ | Кол-во | Формат ячейки "Общий", **вычисление: общ.стоим. / курс** | Формат ячейки "Числовой, с точностью 2 знака", **вычисление по формуле общ.стоим. / курс** | Формат ячейки "Числовой, с точностью 2 знака", **перенос** чисел из колонки G | Формат ячейки "Числовой, с точностью 2 знака", **вычисление по формуле ОКРУГЛЕНИЕ (общ.стоим. / курс) с точностью 2 знака** |
| 14 | ОАО Имидж | 1725 095 2102 | 1054,20 | 35,0000 | 4 | 30,12 | 30,12 | 30,12 | 30,12 |
| 15 | АНО Пламя | 1725 095 2102 | 1782,18 | 39,9900 | 6 | 44,565641 | 44,57 | 44,57 | 44,57 |
| 16 | ООО Пират | 083614-9012 | 1259,88 | 35,0000 | 3 | 35,996571 | 36,00 | 36,00 | 36,00 |
| | | Общие затраты в $ эквиваленте | | | | 110,68221 | 110,68 | 110,68 | 110,69 |

**Рис.7.** Потеря цента

десятичных знака, потому что для сводки нужны реальные денежные выражения стоимости. В колонках H и J потерялся цент. Имея три строки в результирующей сводке это легко отследить, но если строк достаточно много, то найти такую блуждающую неточность очень сложно. Заметим, что в ячейке I15 отображается число 44,57, курсор находится в этой ячейке, поэтому можно видеть, что в строке значений/формул (вверху рисунка) введено на самом деле число с шестью знаками после запятой. Сумма ∑ по столбцу I оперирует не отображаемыми числами ячеек, а именно введенными. Поэтому она оказалась некорректной. Таким образом, наиболее точно произведен подсчет в колонке J. Для вычисляемых значений нужно иметь это в виду.

## 5 Принципы переноса информации

В приведенном примере было рассмотрено ограниченное количество строк, содержащих обрабатываемую информацию, и только благодаря тому, что наше внимание было сконцентрировано на обозримом за раз объекте, нам удалось сразу обнаружить несоответствия в перенесенных в итоговый файл данных. Если бы в каждом из имеющихся исходных файлов было бы строк эдак по 100, к примеру, то, скорее всего, мы не заметили бы ни одной оплошности копирования, кроме самых очевидных – там, где числовое значение отображалось бы как "=СУММ(D13:D18)" или "#ЗНАЧ!".

Соблюдая определенные принципы оформления, можно гарантировать однородность и корректность данных результирующего файла после копирования информации:

1 Рекомендуется создавать новый "лист Microsoft Excel", а не добавлять информацию к уже имеющейся; или копировать предыдущую сводку, в которой форматирование уже было выставлено должным образом, и убивать (DEL'ом) старые данные.

2 Поскольку во вновь созданном Excel-файле форматы ячеек автоматически (по умолчанию) установлены как "общий формат", то для корректного переноса текстовых полей следует установить у колонок, где планируется вставка текстовых значений, "формат – текстовый".

3 Перенос текстовых значений из Excel-файла (колонкой) следует производить следующим образом:

- убедиться, что вся колонка в результирующем файле, куда будет переноситься информация, установлена в "текстовый формат";
- выделить нужный столбец в исходном файле;
- установить формат выделенных ячеек "текстовым";
- запомнить (скопировать) выделенные ячейки и вставить в нужное место в результирующий файл.

4 Перенос из Excel-файла колонок, где содержатся даты (исходные ячейки не обязательно должны быть форматом "дата"):

- убедиться, что вся колонка в результирующем файле, куда будет переноситься информация, установлена в "общий формат";
- выделить нужный столбец в исходном файле;
- запомнить (скопировать) выделенные ячейки и вставить в нужное место в результирующий файл.

5 Перенос всех остальных ("НЕ_текстовых" и "НЕ_дат") значений из Excel-файла производится следующим образом:

- сначала следует убедиться, что все колонки в результирующем файле, куда будет переноситься информация, установлены в "общий формат";
- надо создать Word-файл;
- запомнить (скопировать) переносимые из исходного Excel-файла ячейки и вставить в созданный Word-файл;
- запомнить (скопировать) колонку (или блок) из Word-файла и вставить в нужное место в результирующий файл;
- установить форматы каждой из перенесенных колонок (поверх вставленных значений) в результирующем файле нужным образом - "числовой с точностью X знаков" для столбцов, где нужно вывести числа, "денежный" и т.д., или "функцию ОКГУГЛЕНИЕ", например, где нельзя позволить потеряться копейке, и т.п.;
- особое внимание нужно уделить колонкам, содержащим количества, - установить (оставить) формат таких ячеек "Общим". Если поменять формат на "числовой", то точность может нарушить привычное восприятие целых количеств (вместо 4 мы увидим 4,00), а если установить точность "0 знаков после запятой", то дробные числа (метры, например, в количестве 4,56 или 0,5) будут отображаться некорректно.

6 При копировании "НЕ_текстовых" и "НЕ_дат" значений из Word-файла (если исходный файл был не Excel-файлом) сложностей и неожиданностей не возникает. Единственным ограничением является оформление колонки в Word-файле, как "таблицы", а не как текста – то есть при наличии в файле просто записанных в столбик слов (чисел) такая информация перенесется не совсем корректно. Нужно, чтобы эти слова (числа) были в "таблице Word". Следует также предварительно убедиться, что все колонки в результирующем Excel-файле, куда будет переноситься информация, установлены в "общий формат".

7 После переноса из Word-файла ячейки, содержащие ранее "Сумму ИТОГО" (в том числе вычисляемую по формуле в исходном Excel-файле), станут обычными числовыми выражениями. Их следует удалить (DEL'ом), чтобы не забыть потом вставить в должное место операции сложения (∑).

8 При копировании в "текстовые" ячейки из Word-файла (если исходный файл был не Excel-файлом) полей, которые должны стать текстом, следует придерживаться следующего порядка:

- выделить столбец, в котором содержится будущий "текстовый формат";
- с помощью меню "Найти и заменить" произвести по всему выделенному столбцу замену одного пробела на два пробела;
- запомнить (скопировать) полученную колонку и вставить в нужное место в результирующий файл.

9 При переносе из Word-файла (если исходный файл был не Excel-файлом) данных, которые несут в себе информацию о датах, нужно поступать следующим образом:

- столбец следует преобразовать в текст (каждая дата на отдельной строке). Для этого расположить в Word-файле столбец как таблицу, состоящую из одной колонки, и воспользоваться операцией "Преобразовать таблицу в текст", разделитель – знак абзаца;
- запомнить (скопировать) полученные строки и вставить в нужное место в результирующий файл.

После всех копирований предлагается "навести красоту" – выровнять колонки по нужному краю, высоте и т.п. Установить шрифты, размеры букв и т.д. Проставить подсчет сумм (∑).

# 6 Заключение

Разнообразие форматов и типов данных в документах Microsoft Office обеспечивает гибкие возможности для обработки и представления информации, но может приводить к проблемам мобильности данных. В частности, при переносе сведений между файлами через буфер копирования может происходить их искажение, которое не всегда легко обнаружить визуально. В работе предложена процедура, гарантирующая корректный перенос числовых данных при составлении сводных отчетов из разнородных файлов с табличной информацией. Процедура включает действия по предварительному оформлению итогового файла, а также форматированию копируемых данных, позволяющие избежать ошибок при слиянии информации и обеспечить создание корректного итогового файла, с которым впоследствии легко работать и преобразовывать в отчетные документы.

В результате исследований можно считать установленным, что простой перенос информации копированием в итоговый Excel-файл вероятнее всего приведет к искажению результирующих данных, поэтому следует соблюдать определенную аккуратность в части строгого соблюдения "типов и форматов" с которыми приходится оперировать.

# The Problems of Data Mobility in Accounting Documents

**G.L. Levchenkova**

**Abstract.** The article treats the issues of merging information from multiple documents exported from data bases or produced manually as files containing data tables.

**Keywords:** information, data bases, data tables.

## Литература

1. О.Б. Калугина, В.С. Люцарев Работа с электронными таблицами. Microsoft Office Excel 2003. – М.: ИНТУИТ, 2006.

2. Э. М. Берлинер, И. Б. Глазырина, Б. Э. Глазырин Microsoft Office 2003. Руководство пользователя // Москва. БИНОМ, 2004 http://bookre.org/reader?file=526016&pg=4 (14.07.15)

3. Г.Л. Левченкова, И.В. Холмов Система ДОКУМЕНТООБОРОТ. Организация документооборота и учета на рабочих местах и в подразделениях организаций // *под редакцией академика РАН В.Б.Бетелина*, Москва, НИИСИ РАН, 2006, С. 4-75

4. Г.Л. Левченкова, А.Г. Прилипко Перенос информации между базами данных, существующими в средах с разными операционными системами // Труды НИИСИ РАН. - Том 1 №1. Математическое и компьютерное моделирование систем. Теоретические и прикладные аспекты. ISSN 2225-7349, Москва, 2011, С. 77-85

# Выделение аффинно-инвариантных контуров с помощью обобщенного преобразования Хафа

**А.С. Куцаев**

*кандидат физико-математических наук*

**Аннотация:** Поиск аффинно-инвариантных контуров на изображении полезен в задачах распознавания образов, в том числе при сегментации. Существующие методы поиска дают плохие результаты, если часть контура объекта загорожена или искажена шумом. Предлагается развитие метода [1], основанного на обобщенном преобразовании Хафа (GHT). Преобразование применяется к парам граничных точек, в которых направление касательных совпадает. Для повышения точности вычисления параметров аффинного преобразования между этими касательными строится сетка секущих. Фильтр наименьших квадратов, оценивает пропорциональность длин отрезков секущих и эффективно исключает ложные соответствия. Отбор результатов GHT производится с помощью оценки качества, основанной на метрике Прэтта (FOM). Метод позволяет обнаруживать объекты со значительными дефектами, при частичном наложении и шуме.

**Ключевые слова:** аффинно-инвариантный поиск форм, вычисление касательных, обобщенное преобразование Хафа, фильтр наименьших квадратов.

## 1. Введение

При распознавании изображений плоских контуров по заданному образцу (шаблону) необходимо учитывать возможные искажения объектов, изменения точки и масштаба съемки, частичное загораживание объектов. Часто переход от шаблона к изображению контура может быть приближенно описан с помощью аффинного преобразования, и тогда его свойства можно использовать при решении задачи. Удобство этого подхода также в том, что для распознавания целого класса контуров достаточно задать один шаблон. Выделение на изображении произвольных аффинно-инвариантных контуров можно использовать как для непосредственного распознавания изображений, так и для предварительной сегментации при наличии разрывов контура, неоднородного фона и шума.

Источником данных при поиске контуров обычно служит изображение, на котором выделены границы всех объектов - как интересующих (при наличии), так и посторонних (фоновых). Шаблон задается при помощи изображения объекта или его контура, на однородном фоне. Задача состоит в том, чтобы найти все аффинные преобразования, которые переводят шаблонный контур в соответствующее подмножество граничных точек на изображении с требуемой точностью.

Мощным средством для обнаружения контуров и фигур, полученных в результате геометрических преобразований, является обобщенное преобразование Хафа (GHT). Первоначальная идея GHT [2] состоит в том, что выбранная область изменения параметров искомого геометрического преобразования разбивается сеткой, и с каждой дискретной точкой связывается счетчик. Затем рассматриваются все возможные преобразования, переводящие точку контура шаблона в граничную точку на изображении, и для каждой пары таких точек соответствующие счетчики увеличиваются на 1. Этот процесс называется голосованием. После его окончания локальные пики счетчиков указывают на наиболее вероятные параметры преобразования, переводящего шаблон в изображение объекта.

Поскольку здесь одна точка границ на изображении вызывает увеличение многих счетчиков, такой подход называют подходом "грубой силы" или "отображения 1 в n". Он показал хорошие качества при обработке зашумленных и частично загороженных изображений. Однако этот подход требует очень больших вычислительных ресурсов, если число параметров геометрического преобразования велико. Кроме того, изображение не может быть слишком сложным, иначе число пар точек, связанных искомым преобразованием, составит малую часть от всех пар, и полезная информация потеряется в шуме. Поэтому необходим отбор пар соответствующих точек.

Отбор соответствий, основанный на анализе направления градиента интенсивности [7], позволил повысить скорость обработки и точность определения параметров. Для поиска прямых линий [5] и аффинно-инвариантных объектов [4] использовалось усложнение структуры массивов счетчиков и итерационная стратегия уточнения. Это позволило снизить требования к памяти, сохраняя возможность получения решений с нужной точностью. Однако проблема ресурсоемкости этим полностью не решена.

Наряду с методами "грубой силы" были разработаны методы "отображения n в 1", в которых группа из n точек изображения вызывает увеличение одного счетчика. В нашем случае аффинные преобразования сохраняют параллельность прямых и касательные к границе. Поэтому если выбрать две граничные точки шаблона, в которых направление касательных совпадает, то искомое аффинное преобразование должно перевести их в граничные

точки изображения, в которых направления касательных должны совпадать. Преобразование Хафа для соответствий таких пар точек было применено в [6]. Там же дополнительные аффинно-инвариантные построения позволили вместо перебора вычислять параметры аффинных преобразований для каждого соответствия пар точек. Такой подход еще больше снизил ресурсоемкость метода и в ряде случаев показал хорошие результаты. Однако, как обнаружено в [4], использованные аффинно-инвариантные построения очень чувствительны к шуму и к наложению объектов. Точность решения также сильно зависит от кривизны контуров и их симметрии.

В этой статье рассматривается развитие метода работы [6], направленное на повышение устойчивости выделения контуров. При вычислении касательных используется одномерный метод Хафа [9], дающий более точные результаты для контуров с большой кривизной. При сопоставлении пар опорных точек дополнительно применяется пересечение контура с сеткой секущих и оценка соответствия на основе метода наименьших квадратов. Все это позволило повысить надежность результатов GHT за счет повышения качества входных данных. Также для каждого найденного варианта преобразования оценивается его качество, т. е. мера близости образа контура шаблона и границ на изображении. Отбор по качеству используется и при наложении образов преобразований с разными параметрами и/или от разных шаблонов.

Далее статья имеет следующую структуру. В Главе 2 описывается метод обнаружения аффинно-инвариантных объектов на основе обобщенного преобразования Хафа и фильтра наименьших квадратов. В Главе 3 кратко описываются методы подготовки шаблонов и изображений для исследования свойств метода. Глава 4 посвящена некоторым результатам применения метода. В Главе 5 приведены выводы.

# 2. Описание метода

## 2.1. Объекты для сопоставления

При голосовании в GHT происходит сопоставление объектов шаблона и изображения. Обычно объект представляет собой набор точек с дополнительной информацией. Список объектов называется R-таблицей и создается до голосования. В ранних примерах использования GHT запись в R-таблице включала одну точку границы и направление градиента интенсивности.

Запись R-таблицы в [6] включает две опорные граничные точки с одинаковыми направлениями касательных. Расстояние между касательными должно быть достаточным, чтобы искомые преобразования были невырожденными. Параллельно касательным через середину опорного отрезка проводится секущая. Точки пересечения секущей с границами и углы наклона касательных в этих точках также включаются в запись R-таблицы. Опорные точки и каждая из точек пересечения образуют тройку граничных точек, не лежащих на одной прямой.

В преобразовании Хафа попарно сопоставляются все записи R-таблиц шаблона и изображения. Ищутся допустимые аффинные преобразования, переводящие тройку точек шаблона в тройку соответствующих точек изображения, с точностью до перестановки опорных точек. Фильтр при отборе соответствий проверяет, что направления касательных в точках пересечения (не опорных) также переводятся друг в друга с заданной точностью. Это позволяет исключить точки, принадлежащие заведомо разным объектам. Счетчик, отвечающий найденным параметрам аффинного преобразования, увеличивается на 1.

Основной недостаток этой схемы связан с низкой точностью вычисления касательных. Она зависит от размера локального окна и кривизны исследуемых границ. Для дуг эллипсов ошибка составляет около 3°, однако тот же метод на менее гладких кривых может давать ошибку в 20° и выше. Низкая точность усложняет отбор пар опорных точек и делает фильтр при отборе соответствий неэффективным.

Основное отличие предлагаемого подхода в том, что вместо одной секущей используется несколько, а фильтр при отборе соответствий использует не совпадение двух касательных, а пропорциональность наборов длин отрезков секущих.

Для начала необходимо уточнить, как оценивается касательная к цифровой кривой. Например, если цифровая кривая служит представлением некоторой гладкой кривой, то можно использовать касательную к этой гладкой кривой в выбранной точке. При этом для оценок необходимо ограничить максимальную кривизну гладких кривых, так как иначе можно, вообще говоря, получить любое направление касательной. Однако для границ, выделенных на изображении, с искажениями и дефектами, подбор гладких кривых с минимальной кривизной может стать слишком сложным.

В этой работе направление касательной к цифровой кривой вычисляется с помощью одномерного метода Хафа, подробно описанного в [9]. Вычисление касательной в граничной точке происходит в локальном окне с центром в этой точке. Для каждой тройки граничных точек, одна из которых всегда берется в центре окна, рассматриваются все цифровые кривые окна с известными касательными в центре, проходящие через эти три точки. Интервал направлений этих касательных заранее затабулирован. Голосование в методе Хафа состоит в том, что все счетчики для этого интервала увеличиваются на 1.

В расчетах использовалось локальное окно, размер которого (11×11 пикселов) был подобран так, чтобы обеспечить наибольшую точность на тех изображениях, где не гарантируется высокая гладкость границ. Максимальная погрешность угла наклона касательной меняется при этом от 7º при малой кривизне до 13º при максимальной кривизне 0.5. Соответствующий радиус кривизны равен 2 шагам сетки, что означает довольно резкий поворот кривой.

Запись R-таблицы включает пару опорных точек границ, в которых направления касательных совпадают, а расстояние между касательными

достаточно велико. Одна из точек всегда берется в центре граничного пиксела и определяет угол касательной τ. Вторая точка может определяться с помощью линейной интерполяции. Ее можно расположить:

- В центре другого граничного пиксела, если угол касательной в нем совпадает с τ.
- На отрезке, соединяющем центры двух соседних граничных пикселов, если угол τ заключен между углами касательной в этих пикселах.

В последнем случае положение точки находится из условия, что интерполированное значение угла касательной в ней равно τ. Такой подход позволяет выбирать гораздо больше пар точек границы, чем при использовании одних только центров пикселов. Это дает более полную информацию о форме границы. Заметим, что здесь и далее построения инвариантны относительно аффинных преобразований.

Пример построения для записи R-таблицы показан на Рис. 1. Секущие $s_1$, $s_2$, $s_3$ параллельны касательным в опорных точках $p_0$ и $p_1$ и делят отрезок $p_0p_1$ на равные части. Для каждой секущей ищутся точки ее пересечения с линиями границ.

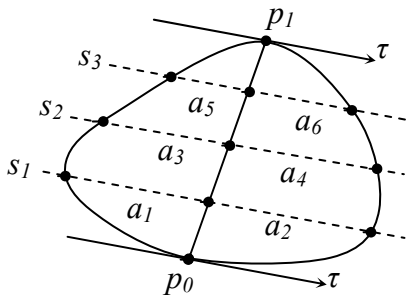



**Рис. 1.** Построение для записи R-таблицы: опорные точки, секущие и точки пересечения.

Для определенности выберем положительное направление секущих, при котором поворот от секущей к опорному отрезку $p_0p_1$ происходит против часовой стрелки. Положение точки пересечения на секущей определяется координатой точки вдоль секущей с началом отсчета на опорном отрезке, т. е. длиной и знаком отрезка $a_i$. Если опорные точки и границы, с которыми пересекаются секущие, принадлежат одному объекту, то при его аффинном преобразовании все длины отрезков секущих меняются пропорционально. Список длин включен в запись R-таблицы, на Рис. 1 это длины $a_1$, ..., $a_6$. Сейчас для шаблона сохраняется по одной длине отрезка на каждом направлении каждой секущей. Для изображения сохраняется до двух длин таких отрезков на каждом направлении, чтобы иметь возможность выбора в случае помех, ошибок дискретизации, частичного наложения и разрывов кривых.

Использование нескольких секущих вместо одной позволяет полнее описать форму границы и обеспечить эффективное применение фильтра наименьших квадратов (см. ниже) при сопоставлении R-таблиц. На каких-то секущих точки пересечения для изображения могут отсутствовать из-за помех и искажений, перечисленных выше.

## 2.2. Фильтр наименьших квадратов

При голосовании сопоставляются записи R-таблиц шаблона и изображения. Для вычисления параметров аффинного преобразования достаточно соответствия между тройками точек шаблона и изображения. Поскольку координаты точек в наборах известны приближенно, можно попытаться снизить погрешность определения параметров преобразования, используя сразу весь набор точек пересечения в записи R-таблицы вместо того, чтобы перебирать точки по одной. Для этого вычислим средний коэффициент пропорциональности наборов отрезков методом наименьших квадратов.

Пусть запись R-таблицы шаблона содержит набор длин отрезков пересечения $\{a_i\}$, $i = 1, ..., n$, а запись R-таблицы изображения - соответствующий набор $\{b_i\}$. Для простоты примем, что отрезки с одинаковым индексом лежат на соответствующих направлениях секущих, т. е. "лишние" отрезки, для которых нет соответствия, не включены в набор. Если аффинное преобразование переводит набор $\{a_i\}$ в набор $\{b_i\}$, то в отсутствие погрешностей координат соответствующие длины в этих наборах пропорциональны, $b_i = c \cdot a_i$. Отклонение наборов от пропорциональности, вызванное всевозможными погрешностями, можно характеризовать суммарной невязкой $S$ (1).

$$S = \sum_{i=1}^{n} (c \cdot a_i - b_i)^2 \qquad (1)$$

Минимум $S_{min}$ и соответствующее ему значение коэффициента пропорциональности $c_{min}$ находятся просто:

$$S_{\min} = \sum_{i=1}^{n} b_i^{\,2} - \left( \sum_{i=1}^{n} a_i b_i \right)^2 \Big/ \sum_{i=1}^{n} a_i^{\,2}, \qquad c_{\min} = \sum_{i=1}^{n} a_i b_i \Big/ \sum_{i=1}^{n} a_i^{\,2} \qquad (2)$$

Если разделить $S_{min}$ на сумму $b_i^2$, то полученная нормированная невязка $\overline{S}_{\min}$ не будет зависеть от линейного масштаба и количества значений длин в наборах $n$. Если рассматривать $\{a_i\}$ и $\{b_i\}$ как векторы в $n$-мерном пространстве, то невязка $\overline{S}_{\min}$ - это просто квадрат синуса угла между векторами. Ее можно использовать для отбора допустимых соответствий по порогу. Превышение порога обычно означает, что пары опорных точек не соответствуют друг другу, т. е. не связаны аффинным преобразованием. Если пары опорных точек все же связаны преобразованием, то причиной превышения порога может быть ошибочный выбор точек пересечения за счет наложения других объектов, разрывов границы, ошибок дискретизации и т. д. В этом случае можно попытаться снизить $\overline{S}_{\min}$ путем отбора точек пересечения на изображении или просто исключения точек, которые сильнее всего ухудшают оценку $\overline{S}_{\min}$. В наших экспериментах для нормированного отклонения от пропорциональности $\overline{S}_{\min}$ было принято значение порога от 0.004 до 0.006.

Теперь нужно заметить, что при поиске соответствии пар опорных точек шаблона ($p_0$, $p_1$) и изображения ($q_0$, $q_1$) возможны два варианта соответствия самих опорных точек: $p_0 \leftrightarrow q_0$ и $p_1 \leftrightarrow q_1$, а также $p_0 \leftrightarrow q_1$ и $p_1 \leftrightarrow q_0$. Необходимо рассматривать оба варианта, так как заранее неизвестно, какой из них реализуется. Кроме того, есть два варианта соответствия положительного направления секущих, которые получаются зеркальным отражением двух первых. Эти варианты показаны в примере на Рис. 2.

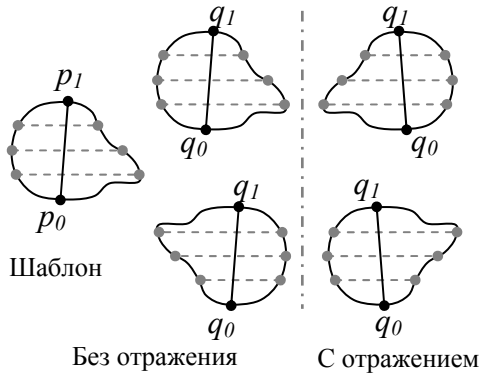



**Рис. 2.** Варианты соответствия опорных точек и положительного направления секущих на шаблоне и на изображении.

Из четырех возможных вариантов преобразования два будут положительно определенными. Если шаблон имеет ось симметрии или если зеркальное отражение не рассматривается, достаточно двух вариантов, показанных в примере на Рис. 2 слева. Иначе нужно рассмотреть еще два варианта, в которых меняется соответствие направления секущих на шаблоне и изображении, то есть соответствие длин отрезков в записях R-таблиц. В примере на Рис. 2 эти варианты изображены справа.

Описанный пороговый отбор соответствий по $\overline{S}_{\min}$ далее называется фильтром наименьших квадратов (LSF). Его эффективность основана на том, что при использовании нескольких точек вместо одной влияние отдельных ошибок уменьшается. Иначе говоря, действие этого фильтра основано на анализе контура объекта в целом, а не его локальных особенностей.

## 2.3. Параметры преобразования

Для каждого варианта соответствия опорных точек определено аффинное преобразование. Оно не вырождено в силу условий на опорные точки и точки пересечения. Параметры преобразования легко определить по исходным и преобразованным координатам трех точек. Две из них - это опорные точки на шаблоне $p_0$ и $p_1$, которым соответствуют еще две опорные точки на изображении $q_0$ и $q_1$ или $q_1$ и $q_0$, согласно выбранному варианту. Последние две точки, $p_2$ на шаблоне и $q_2$ на изображении, можно выбрать на соответствующей средней секущей согласно (3).

$$\begin{aligned} p_2 &= (p_0 + p_1)/2 + \tau_T \\ q_2 &= (q_0 + q_1)/2 \pm \tau_I \cdot c_{\min} \end{aligned} \qquad (3)$$

Здесь $\tau_T$ и $\tau_I$ - единичные направляющие векторы касательных в опорных точках шаблона и изображения, соответственно. Знак перед вектором $\tau_I$ также зависит от выбранного варианта соответствия.

## 2.4. Два этапа GHT

Определение параметров аффинного преобразования при помощи GHT удобно делать в два этапа. На первом этапе при помощи двумерного преобразования Хафа определяются возможные значения вектора перемещения. Для этого используется двумерный массив счетчиков, размеры которого совпадают с размерами изображения. Здесь важно правильно выбрать начало координат в шаблоне. В частности, если шаблон (или любой его аффинный образ) имеет ось симметрии, то начало координат должно лежать на этой оси, иначе пик счетчиков будет разделен на части, фактически отвечающие разным представлениям одного и того же преобразования.

На втором этапе для каждого найденного вектора перемещения производится уточнение матрицы. В данной работе использован тот же алгоритм уточнения, что в [6]. Массив счетчиков имеет размеры 9×9×9×9. Интервал изменения каждого параметра вычисляется и делится равномерно на 9 частей. В голосовании участвуют только матрицы преобразования, отвечающие выбранному вектору перемещения, затем определяются локальные пики в 4-мерном массиве счетчиков. Если пик достаточно высок, выполняются уточняющие итерации по тем осям параметров, где еще не достигнуто нужное разрешение. На этих осях интервал размером 3 шага сетки около пика разбивается на 9 частей, после чего повторяется голосование и выбор локальных пиков. Если высота пика недостаточна или нужное разрешение достигнуто по всем осям, вычисляется решение, в котором компоненты матрицы являются средними по матрицам, отвечающим последнему локальному пику. Выполнив уточнение по всем локальным пикам, можно для выбранного значения вектора перемещения получить список (дерево) возможных решений, каждое из которых определяет некоторое аффинное преобразование.

Далее в этом списке необходимо отобрать лучшие решения. Вообще говоря, возможно наличие двух объектов с одним и тем же вектором перемещения (например, концентрические окружности). Однако для анализа таких изображений надежнее использовать детектор, в котором изображения объектов удаляются по мере обнаружения. Критерии отбора решений рассмотрены ниже.

## 2.5. Критерии отбора решений

Для отбора решений в первую очередь полезно использовать ограничения на компоненты матрицы преобразования. В наших экспериментах

использовались достаточно слабые ограничения на образы ортов декартовой системы координат шаблона: длина вектора от 0.4 до 2.5, отношение длин не менее 0.3 и угол между векторами не менее 45°. В условиях помех недостаточные ограничения могут приводить к неожиданным неверным решениям, поэтому ограничения необходимо усиливать настолько, насколько это допускает прикладная задача.

Пусть $A$ - преобразование плоскости, множество точек $\{a_i\}$, $i$ = 1,..., $N$, образует контур шаблона, множество точек $\{x_i\}$ получено преобразованием множества $\{a_i\}$, $x_i = Aa_i$, а множество **B** состоит из всех точек границ на изображении. Используются оценки качества решений, основанные на вычислении расстояния $d(x_i, \mathbf{B})$, $i$ = 1,..., $N$. Первоначально для сравнения двух решений использовалось среднее расстояние, однако оказалось, что метрика Прэтта FOM (4) [1, 8] дает несколько лучшие результаты при наличии больших разрывов на контурах объектов.

$$FOM = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{1+k\cdot d(x_i,\mathrm{B})^2} \qquad (4)$$

где $k$ = 1/9. Удобным средством для получения оценки является карта расстояний, т. е. серотоновое изображение, в котором нулевые пикселы соответствуют множеству границ на исследуемом изображении. Интенсивность в любом другом пикселе пропорциональна его расстоянию до множества нулевых пикселов. Если использовать диагональную метрику и эффективный алгоритм вычисления карты расстояний [3], то вычисление оценки не представляет трудности.

Заведомо неверные решения отбрасываются при сравнении оценки качества с некоторым порогом. Для этой оценки в данной работе использовалось среднее расстояние, модифицированное для учета разрывов границ. Поскольку разрыв ухудшает оценку независимо от качества решения, была сделана попытка отделить точки, которые расположены в области разрыва, и не использовать их при вычислении оценки. Точками над разрывом здесь считаются преобразованные точки $x$, $d(x, \mathbf{B}) > 3$, причем число таких точек не может быть больше $N/4$. Это несколько меньше, чем предельный размер разрыва, при котором еще возможно обнаружение объекта. Такая оценка была выбрана главным образом из-за того, что смысл ее параметров очевиден.

Один и тот же набор линий на изображении может интерпретироваться как образ разных шаблонов либо одного шаблона, но с разными векторами перемещения. В этом случае можно ожидать наложения образов внутренней части соответствующих шаблонов. Степень такого наложения вычислялась приближенно с использованием минимального содержащего многоугольника для шаблона (в экспериментах использовался прямоугольник). Вычислялась площадь общей части образов двух содержащих многоугольников, отнесенная к площади каждого из них. Если хотя бы одно из отношений превышало порог (в экспериментах 0.75), то принималось, что обнаружен один и тот же объект. В этом случае из двух решений отбиралось одно с лучшей оценкой качества. При сравнении качества, как и выше, использовалась метрика Прэтта FOM (4).

## 2.6. Границы применения

Границы применения описываемого метода зависят от размеров и формы контура шаблонов и изображений объектов. Идеальной формой является окружность, поскольку ошибка вычисления направления касательных мала, а их направление изменяется монотонно. В вершинах углов, особенно острых, ошибка вычисления направления касательной растет, однако это отчасти компенсируется тем, что уменьшается ошибка локализации направления. Например, при шаге изменения угла наклона касательной 20° и ошибке по углу 10° ошибка локализации направления касательной составит всего 0.5 шага сетки. Длинные отрезки прямых неудобны тем, что на них локализация направления возможна только в характерных точках (начало, конец, середина), которые сами локализуются не слишком точно.

Для преобразования Хафа требуется достаточно много соответствий между парами точек контуров шаблона и объекта. Однако с ростом числа точек шаблона увеличивается потребность в вычислительных ресурсах. Кроме того, увеличение размера не имеет смысла, если R-таблица уже содержит необходимую информацию о форме контура. Оптимальное число точек шаблона можно оценить для выпуклого контура, близкого к окружности. Средняя погрешность вычисления направления касательных, определенная в [9], меняется от 2.5° до 5.5°, когда использованные при оценке кривые имеют максимальную кривизну 0.5 (единицей длины служит шаг сетки). Шаг изменения угла должен быть того же порядка, что и погрешность. Разделив интервал изменения направления касательной (360°) на характерную погрешность 5°, получим число точек контура $N$ = 72. Эксперименты показали, что шаблоны соответствующего размера (29×29) также позволяют достаточно полно представить форму контура.

Влияние изменения размеров объекта показано далее на примерах с преобразованием подобия. Можно заметить, что при линейных размерах объекта 10 пикселов и меньше его форма сильно искажается при преобразованиях, а вычисление касательных часто теряет смысл. Увеличение размеров объекта не препятствует обнаружению, но в этом случае нужны дополнительные меры по ограничению размеров R-таблицы.

# 3. Построение шаблонов и изображений

Необходимой частью разработки является исследование свойств метода в зависимости от различных условий. Здесь для такого исследования требуется техника построения шаблонов и достаточного количества модельных изображений с заданными параметрами. Изображения можно получать путем наложения изображений объектов под

действием заданных аффинных преобразований. Шаблоны проще всего создавать на основе исходных изображений объектов. В этой главе описаны некоторые приемы генерации изображений и шаблонов.

Изображения объектов не должны терять качество при аффинном преобразовании. Бинарное изображение имеет тот недостаток, что положение границы неустойчиво и определяется с точностью до одного пиксела. Например, граница прямоугольника размером m×n может иметь размер как $m \times n$, так и $(m+2)\times(n+2)$. Чтобы исключить неоднозначность, здесь пикселы 8-связного контура объекта изображаются интенсивностью $c_s = (f_s + b_s)/2$, где $f_s$ и $b_s$ - интенсивность объекта и фона, соответственно. Это также позволяет повысить устойчивость выделения границ после аффинного преобразования. Примеры шаблона и исходного изображения объекта показаны на Рис. 3a и 3b, соответственно.

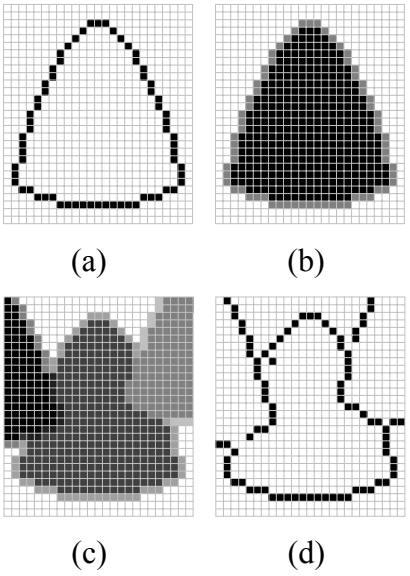



**Рис. 3.** Шаблон и изображение: (a) шаблон, (b) изображение объекта, (c) наложение объектов, (d) выделенные границы.

Аффинное преобразование одного или нескольких объектов без наложения при однородном фоне не вызывает сложностей. Сложность возникает при наложении, так как в преобразованном изображении объекта могут присутствовать все интенсивности от $b_s$ до $f_s$, (см. выше). В нашем частном случае исходное и преобразованное изображение объекта можно рассматривать как однородное с интенсивностью $f_s$ и переменной степенью непрозрачности θ. При наложении новая интенсивность $c_d$ в каждом пикселе дается выражением (5):

$$c_d = f_s \cdot \theta + c_{dp} \cdot (1-\theta), \quad \theta = \frac{c_s - b_s}{f_s - b_s} \qquad (5)$$

где $c_{dp}$ - интенсивность в данном пикселе перед наложением, а $c_s$ - интенсивность, вычисленная в результате преобразования изображения объекта. При использовании (5) наложение на однородный фон ($c_{dp} = b_s$) является простым копированием, а при неоднородном фоне оно почти не искажает границы объекта. Пример наложения объектов с использованием (5) показан на Рис. 3c, d.

# 4. Некоторые результаты

## 4.1. Обнаружение нескольких объектов

На Рис. 4 показаны результаты обработки изображения, которое моделирует вид сверху на группу одинаковых автомобилей.

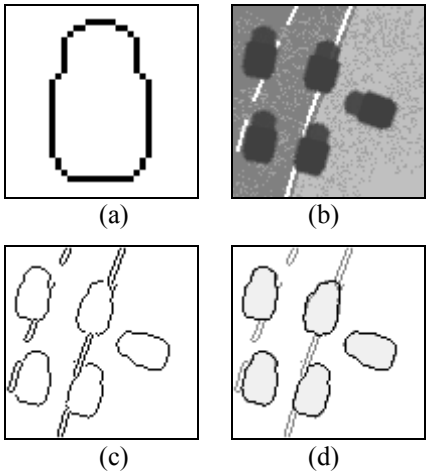



**Рис. 4.** Результаты обнаружения объектов на модели изображения дороги. (a) Увеличенный шаблон. (b) Исходное изображение. (c) Выделенные границы. (d) Результаты обнаружения.

Шаблон, увеличенный вдвое, показан на Рис. 4 a. Исходное серотоновое изображение (Рис. 4 b) включает неоднородный фон, изображающий дорогу с разметкой, и несколько наложенных на него аффинных образов одного и того же источника. Аффинное преобразование состоит из поворота и переноса. Картина выделенных границ (Рис 4 c) включает контуры объектов и границы, выделенные из фона. Результат обнаружения представлен на Рис. 4 d. Здесь серым показаны выделенные границы, поверх них черным нанесены результаты преобразований контура шаблона со всеми найденными наборами параметров преобразования. Видно, что, несмотря на помехи, все объекты обнаружены.

Помехи другого типа возникают при наложении объектов. Здесь возможны два случая, в зависимости от того, сливаются объекты при наложении или нет. Оба эти случая показаны в примере на Рис. 5. Исходное изображение (Рис. 5 b) содержит увеличенные и уменьшенные образы источника, соответствующего шаблону на Рис. 5 a (увеличен вдвое). Преобразование включает подобие, поворот и перенос. В нижнем ряду накладываются объекты разной интенсивности, из-за чего граница одного из объектов почти не искажается. В верхнем ряду интенсивность объектов одинакова, они сливаются и возникает один замкнутый контур вместо трех. Выделенные границы показаны на Рис. 5 c.

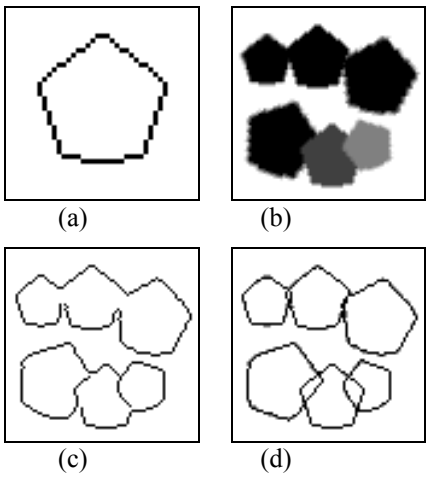



**Рис. 5.** Результаты обнаружения при наложении объектов одинаковой (верхний ряд) и различной (нижний ряд) интенсивности. (a) Увеличенный шаблон. (b) Исходное изображение. (c) Выделенные границы. (d) Результаты обнаружения

Результаты обнаружения показаны на Рис. 5 d. Серым показаны выделенные границы, поверх которых черным изображены результаты преобразований контура шаблона. Заметим, что при слиянии объектов метод обнаружения менее эффективен. Этот недостаток, возможно, удастся устранить путем совершенствования отбора соответствий.

## 4.2. Обнаружение одиночных объектов

Изображение аффинно-инвариантного объекта можно рассматривать как результат дискретизации аффинного преобразования некоторого точного (например, векторного) представления исходного объекта. Успех обнаружения объекта зависит от погрешностей дискретизации. Относительная погрешность растет при уменьшении размеров объекта, так что существует граница уменьшения, ниже которой метод обнаружения перестает работать. Для отдельной оценки влияния погрешностей дискретизации можно использовать промежуточный результат, получаемый в процессе обнаружения - высоту пика счетчиков для вектора перемещения.

Высота пика счетчиков в двумерном преобразовании Хафа на первом этапе GHT в точности равна числу соответствий пар точек шаблона и изображения, прошедших фильтр LSF и относящихся к данному значению вектора перемещения. Чем выше пик, тем больше шансов, что в пространстве параметров аффинного преобразования найдется кластер, отвечающий решению, которое удовлетворяет критериям качества. Высота пика для случаев успешного обнаружения меняется в широких пределах и зависит, кроме прочего, от относительных размеров объекта, т. е. от коэффициента сжатия в аффинном преобразовании. Поворот также может добавить искажения, однако оценить эту добавку гораздо сложнее. Как показывают эксперименты, наибольшие трудности при обнаружении возникают в случае анизотропного сжатия.

При обнаружении одиночных объектов характерные значения высоты пиков зависят от формы шаблона. Здесь в экспериментах шаблонами служили дискретные представления правильных многоугольников со скругленными вершинами. Два из них показаны выше: треугольник на Рис. 3 a, пятиугольник на Рис. 5 a. Шаблон квадрата очень прост и потому не приводится.

Число записей R-таблицы важно для обнаружения объектов, так как оно определяет число кандидатур для соответствия пар точек. Число точек не очень существенно, так как разрешение изображения и погрешность дискретизации в основном уже определены выбранными размерами шаблона. Число записей и число точек составляло 52 и 66 (треугольник), 30 и 96 (квадрат), 68 и 72 (пятиугольник).

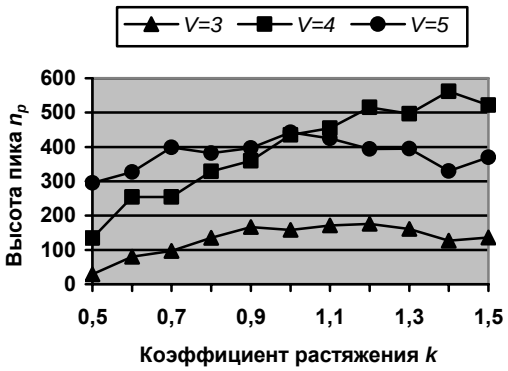



**Рис. 6.** Средняя по углу высота пика счетчиков для треугольника ($V$=3), квадрата ($V$=4) и пятиугольника ($V$=5) в зависимости от коэффициента растяжения $k$.

Для преобразований подобия и поворота (без зеркальных отражений) одного объекта на Рис. 6 показано изменение средней высоты пика счетчиков $n_p$ в зависимости от коэффициента сжатия $k$. Осреднение по углам поворота 0, 10°, 20° и 30° сделано для подавления несущественных колебаний. Расстояние от образа контура шаблона до границ на изображении $d(x_i, \mathbf{B})$ (4) нигде не превышает 1, оценка качества FOM колеблется около 1 без какой-либо закономерности и потому не приводится.

Несмотря на колебания кривых из-за погрешностей дискретизации, на Рис. 6 можно заметить влияние симметрии. Так как правильный многоугольник с числом вершин $V$ при повороте на угол $2\pi/V$ переходит в себя, в отсутствие погрешностей каждой паре точек контура объекта может соответствовать $V$ пар точек шаблона (если рассматривать только положительно определенные преобразования). Эффект проявляется в различной характерной высоте пиков для разных шаблонов на Рис. 6. Эффект ослабляют погрешности при построении шаблонов, которые для квадрата действуют в меньшей степени.

Погрешности при дискретизации исходного изображения, отнесенные к размеру объекта, растут обратно пропорционально этому размеру. Поэтому число соответствий и высота пиков при постоянном значении порога в фильтре LSF убывают с уменьшением размера объекта (на Рис. 6 ему соответствует коэффициент растяжения $k$). Это объясняет поведение кривых на Рис. 6 при малых $k$. При $k > 1$ их поведение имеет более сложную природу.

## 4.3. Затягивание разрывов

Методы обнаружения аффинно-инвариантных объектов допускают наличие незамкнутых контуров. Затягивание разрыва при обнаружении можно рассматривать как дополнительную возможность. Поскольку при этом используется информация о форме всех сохранившихся частей контура, эти методы способны затягивать разрывы довольно сложного вида.

Затягивание разрыва в качестве основной задачи требует критерия правильности решения. Таким критерием может быть оценка FOM (4), но только при условии использования в ней соответствующего изображения без разрыва. В противном случае оценка будет зависеть в основном от точек шаблона, отображаемых в окрестность разрыва, а не от качества аппроксимации границы. Поэтому в приводимых ниже экспериментах решается именно задача обнаружения, а качество затягивания разрыва определялось дополнительно визуальным анализом полученных изображений.

В наших экспериментах по затягиванию разрывов шаблонами, как и в предыдущей части, служили дискретные представления правильных многоугольников со скругленными вершинами. Исходные изображения строятся наложением результата преобразования изображения соответствующего объекта на однородный фон. Далее на изображениях выделялись границы, и в окрестности заданной точки контура часть пикселов границы удалялась. Затем к оставшимся границам применялся метод обнаружения аффинно-инвариантных форм. Такое искусственное создание разрыва позволяет создать все необходимые условия для исследования метода, но проще для контроля, чем попытки моделировать "естественное" возникновение разрыва на контуре.

На Рис. 7 представлены примеры затягивания разрыва на треугольном контуре, когда число точек в разрыве близко к критическому. Объект соответствует шаблону Рис. 3a, преобразование включает только поворот. Черным везде показан контур объекта с разрывом, серым - точки образа контура шаблона, не совпадающие с точками контура объекта. Середина разрыва расположена на середине стороны треугольника (Рис. 7a, b) либо в вершине (Рис. 7c, d). Успешное затягивание разрыва (Рис. 7a, c) происходит, когда удаленные точки составляют 23% от общего числа точек контура, удаление еще одной точки приводит к неудаче. При неудаче образ шаблона аппроксимирует не удаленную, а только оставшуюся часть контура объекта. Для улучшения ситуации в прикладной задаче можно использовать усиление ограничений на матрицу преобразования.

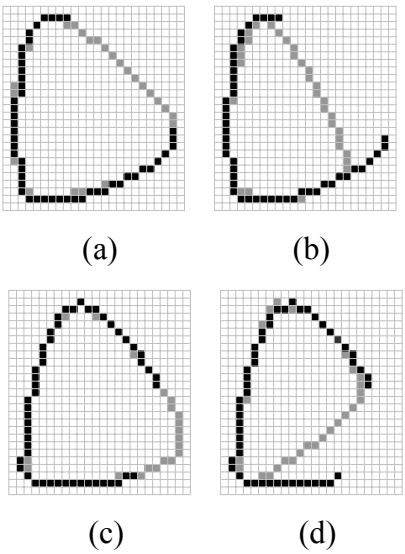



**Рис. 7.** Успешное и неудачное затягивание разрыва в середине стороны (a, b) и в вершине (c, d) треугольного контура.

Максимальная длина разрыва, который удается затянуть, зависит от местоположения разрыва на контуре, то есть от того, достаточно ли информации о форме контура содержит его оставшаяся часть. На Рис. 8 показано изменение максимально допустимой длины разрыва $N_h$ в зависимости от положения разрыва на контуре объекта. В качестве объекта был выбран треугольник, показанный на Рис. 3 a. К нему применялось преобразование подобия с коэффициентом $k$ = 0.8, 1.0 и 1.2. Положение средней точки разрыва на контуре задается ее полярным углом в системе координат, связанной с центром объекта. Под длиной разрыва $N_h$ понимается число удаленных пикселов в окрестности точки середины разрыва. Относительная длина разрыва берется в процентах от общего числа пикселов $N$, составляющих контур объекта.

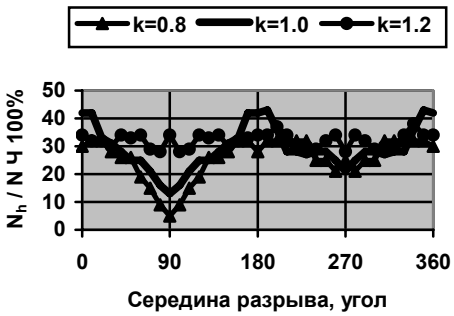



**Рис. 8.** Максимально допустимая относительная длина разрыва в зависимости от полярного угла точки середины разрыва.

На Рис. 8 видно, что при $k$ = 1 и 0.8 допустимая длина разрыва падает в окрестности верхней вершины треугольника. Это говорит о важности данного участка контура для обнаружения объекта. При $k$ = 1.2 картина улучшается, возможная причина - повышение

точности вычислений при растяжении. Погрешности дискретизации при преобразовании обычно снижают возможности обнаружения объекта, но также встречаются исключения из этого правила.

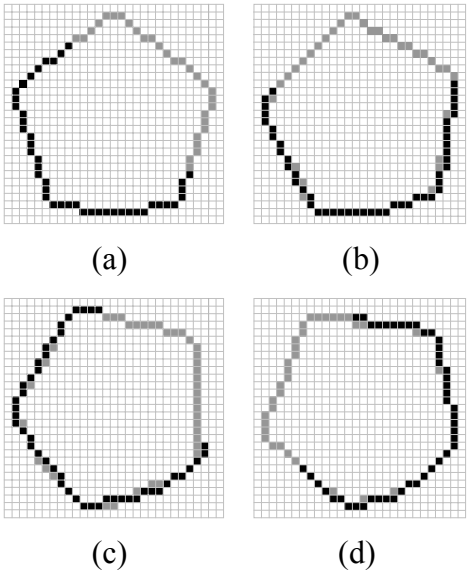



**Рис. 9.** Затягивание разрыва на одиночном пятиугольнике при разных углах поворота.

На Рис. 9 показаны примеры затягивания разрыва более сложной формы. Здесь шаблоном служит пятиугольник, изображенный на Рис. 5 а. Здесь видно, что максимальная длина разрыва больше при нулевом угле поворота, когда нет погрешностей, вызванных дискретизацией. Однако и при наличии поворота длина разрыва гораздо больше той, которую удается затянуть в случае нескольких объектов (Рис. 5). Это означает, что в обоих случаях R-таблица содержит достаточно информации о форме контура и что нужно совершенствовать отбор соответствий для GHT.

## 5. Заключение

В предлагаемом методе обнаружения аффинно-инвариантных форм, основанном на вычислении касательных к границам и применении GHT, используется информация не только о локальных свойствах контура, но и о его форме как целого. Метод вычисления касательных имеет удовлетворительную точность, позволяя в то же время обрабатывать границы достаточно большой кривизны. Фильтр наименьших квадратов позволяет с большой точностью определять соответствие данных шаблона и изображения. При отборе решений используется объективная оценка близости образа контура шаблона и границ на изображении. Определены также границы применения метода и наиболее подходящие размеры объектов.

Проведенные эксперименты показали, что предлагаемый метод позволяет обнаруживать объекты в условиях помех, со значительными дефектами и при наложении. Вместе с тем техника отбора соответствий нуждается в совершенствовании, что и должно быть предметом дальнейшей работы.

# Detection of affine-invariant shapes based on generalized Hough transform

**A.S. Koutsaev**

**Abstract:** The detection of affine-invariant shapes from an image is useful in image recognition, including image segmentation. The existing methods of detection show poor results if a part of the shape is occluded or corrupted by noise. In this paper we introduce a method that improves the approach proposed in [6], based on the generalized Hough transform (GHT). The transform is applied to pairs of reference edge points with parallel tangents. To improve the accuracy of the affine transform parameters calculation, a grid of secant lines is drawn between the tangents. To avoid false correspondences we use a least-squares filter that estimates the proportionality of corresponding segments of secants. The selection of the GHT results is performed via the estimate of quality based on Pratt figure of merit (FOM). The method can detect objects with occlusions and with considerable missing parts of the contour.

**Keywords:** affine-invariant shape detection, tangent estimator, generalized Hough transform, least squares filter.

## Литература

1. I.E. Abdou, W.K. Pratt. Quantitative design and evaluation of enhancement thresholding edge detectors. Proceedings of IEEE, Vol. 67, No 5, 1979, pp. 753-763.

2. D.H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition 13 (2), 1981, 111–122.

3. G. Borgefors. Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence, , V 10, N 6, 1988, pp 849-865.

4. O. Ecabert, J.-P. Thiran. Adaptive Hough transform for the detection of natural shapes under weak affine transformations. Pattern Recognition Letters 25 (2004) pp 1411–1419.

5. J. Illingworth, J. Kittler. The adaptive Hough transform. IEEE Transactions on Pattern Analysis and Machine Intelligence 9 (5), 1987, 690–698.

6. A. Kimura, T. Watanabe. Generalized Hough Transform to Be Extended as an Affine-Invariant Detector of Arbitrary Shapes. Electronics and Communications in Japan, Part 2, Vol. 87, No. 6, 2004, pp 58-68.

7. E. Montiel, A.S. Aguado, M.S. Nixon. Improving the Hough transform gathering process for affine transformations. Pattern Recognition Letters 22, 2001, pp 959–969.

8. W.K. Pratt. Digital Image Processing, New York: John Wiley & Sons, 1991.

9. A.V. Zakharov, P.P. Kol'tsov, N.V. Kotovich, A.A. Kravchenko, A.S. Kutsaev, and A.S. Osipov. A Method for Calculating the Tangent on the Basis of the Hough Transform. Pattern Recognition and Image Analysis, Vol. 23, No. 2, 2013, pp. 258–268.

# Математическое моделирование конвективного теплообмена тепловыделяющего элемента с воздушной средой в электронной системе

**М.Ж. Акжолов**

*кандидат физико-математических наук*

**Аннотация:** Приведены результаты математического и компьютерного моделирования ламинарного и турбулентного конвективного теплообмена тепловыделяющих элементов и воздушной среды в электронных системах.

**Ключевые слова:** электронные системы, тепловыделяющие элементы, конвективный теплообмен.

При тепловом проектировании электронных систем (ЭС), создании эффективных конструкций теплоотводов, устанавливаемых на тепловыделяющих элементах (ТВЭ) в ЭС, необходимо располагать адекватными методами моделирования теплообмена ТВЭ с воздушной средой внутри ЭС. Теплообмен между ТВЭ и средой носит сложный характер и осуществляется одновременно конвекцией и излучением [3], причем конвективный теплообмен в ЭС может протекать как в ламинарном, так и турбулентном режимах. Граница между двумя этими режимами является довольно неустойчивой, так что поначалу спокойный ламинарный поток внезапно и непредсказуемо сменяется турбулентным и обратно. Математическое моделирование конвективного теплообмена основано на уравнениях Навье-Стокса, решению которых посвящена огромная литература. Между тем тепловые процессы, протекающие в ЭС, носят специфический характер, существенно усложняющий их адекватное моделирование. Так, при тепловом проектировании ЭС необходимо учитывать такие факторы, как зависимость мощностей тепловыделения ТВЭ от температуры собственного разогрева, а температуры ТВЭ, в свою очередь, от изменившейся мощности (тепловая обратная связь); статистический интервальный характер тепловых процессов, вызванных интервальным характером определяющих параметров [6]; неоднородность и сложностью формы множества ТВЭ, входящих в конструкцию ЭС; сопряженный характер теплообмена и др. [3, 4].

В настоящей работе приведены результаты математического и компьютерного моделирования ламинарного и турбулентного конвективного теплообмена ТВЭ с воздушной средой внутри ЭС [1, 5]. При моделировании полагалось, что газовая среда идеальная, вязкость отсутствует, теплопроводность в газе пренебрежимо мала по сравнению с конвективным теплопереносом.

Математическая модель газодинамических процессов, протекающих при теплообмене ТВЭ и средой (система уравнений газовой динамики Эйлера) имеет вид [2]:

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u}{\partial x}+\frac{\partial \rho v}{\partial y}=0,$$

$$\frac{\partial \rho u}{\partial t}+\frac{\partial \rho u^2}{\partial x}+\frac{\partial \rho uv}{\partial y}+\frac{\partial p}{\partial x}=0,$$

$$\frac{\partial \rho v}{\partial t}+\frac{\partial \rho uv}{\partial x}+\frac{\partial \rho v^2}{\partial y}+\frac{\partial p}{\partial y}+\rho g_0=0,$$

$$\frac{\partial \rho E}{\partial t}+\frac{\partial (\rho uE)}{\partial x}+\frac{\partial (\rho vE)}{\partial y}+$$

$$+\frac{\partial (pu)}{\partial x}+\frac{\partial (pv)}{\partial y}+\rho g_0 \mathrm{v}=0. \qquad (1)$$

и замыкается уравнением состояния (2).

$$p = (\gamma - 1)\rho J = (\gamma - 1)\rho[E - (u^2 + v^2)/2], \qquad (2)$$

где $\vec{V} = (u, v)$ – вектор скорости, $J$ – удельная внутренняя энергия, $E$ – полная удельная энергия, с – плотность газа, $g_0$ – ускорение свободного падения, направленное строго вдоль оси OY, $\varepsilon$ – показатель адиабаты, $p$ – давление.

Уравнения (1) решались численно с помощью метода крупных частиц [2].

Размеры расчетной области равны $L_x$ = 0.18 м, $L_y$ = 0.09 м (рис.1). У основания области $y = 0$ в первой расчетной ячейке при $L_n \le x \le L_k$ находится ТВЭ, начальная температура воздуха $T_0$ = 300°К. Левая ($y$ = 0) и правая ($y = L_x$) границы теплоизолированы (поток

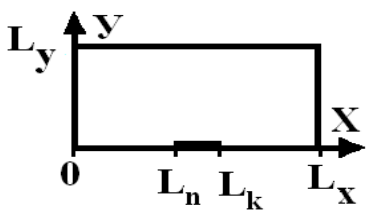



**Рис.1.** Схема расчетной области

равен нулю). На нижней границе участка [$L_k$ , $L_n$] задан источник тепла (ТВЭ) длиной 40 мм, имеющий постоянную температуру $T_{гр}$ = 500°К. На нижней границе области ($0 \le x \le L_x$) заданы условия непроницаемости. На верхней границе заданы условия свободного протекания и постоянная температура $T_0$ = 300°К. Расчетная область разбивалась на 90x45 квадратных ячеек со сторонами $d_x = d_y$ = 2 мм, шаг по времени равен $dt = 10^{-8}$ сек.

## Результаты и анализ

Результаты численного моделирования динамики развития газодинамических процессов под влиянием тепловыделяющего ТВЭ для различных моментов времени, приведены на рис. 2 – 6, на которых показаны температурные поля в воздушной среде.

Анализ полученных результатов показывает следующее.

При нагреве области, занятой ТВЭ, над ним развивается процесс конвекции, который происходит за счет действия как давления, так и гравитационных сил (максимальная скорость $\le \sqrt{2g_0H}$ , где H =0.09 м – высота расчетной области).

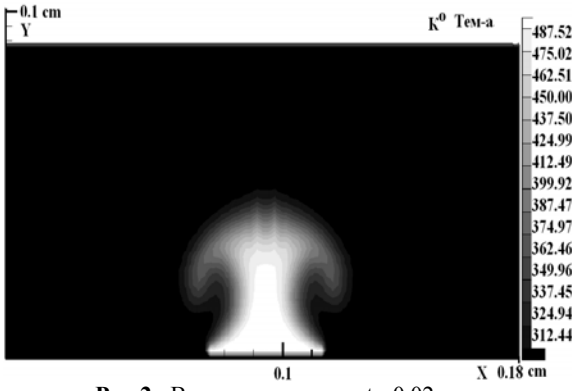



**Рис.2.** В момент времени t= 0.02 сек.

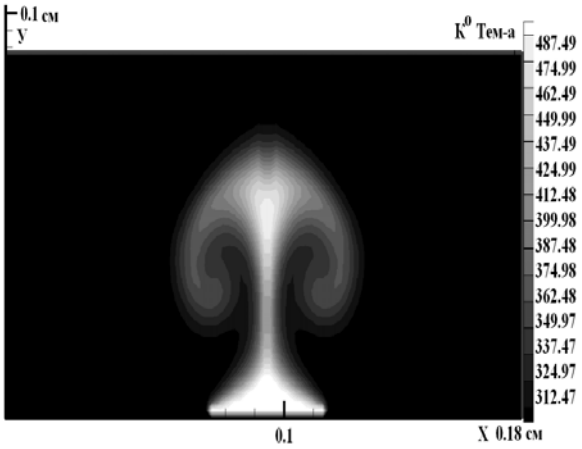



**Рис.3.** В момент времени t= 0.06 сек.

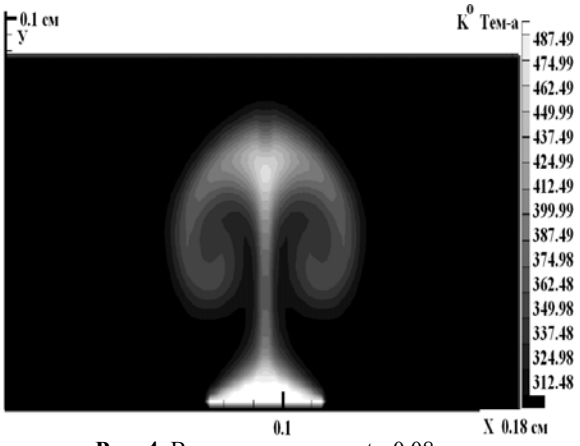



**Рис. 4.** В момент времени t= 0.08 сек.

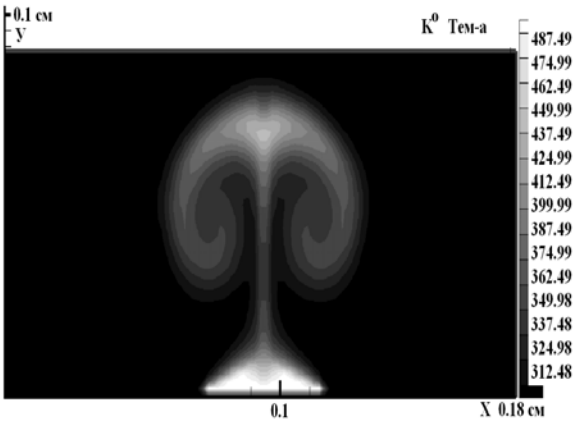



**Рис. 5.** В момент времени t= 0.10 сек.

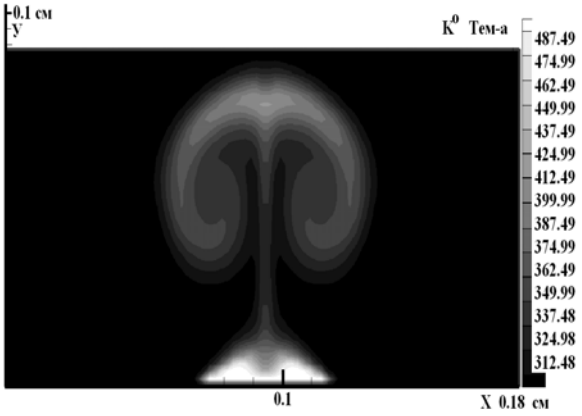



**Рис. 6.** В момент времени t= 0.12 сек.

Полученные изображения развития конвективных процессов в воздухе над ТВЭ достаточно хорошо отражают моделируемое физическое явление. Тепловые потоки, возникающие вследствие нагревания ТВЭ, развиваются плавно (монотонно) начиная с приповерхностного слоя; дальнейшие развитие процесса во времени показывает расширение области охвата конвекцией все большего объема воздуха. Отметим также симметрию профиля развития конвективного потока в процессе динамики развития. Полученные данные свидетельствует об правильности выбранных математической и сеточной моделей.

# Mathematical modeling of convective heat transfer from an heating element to air in electronic systems

**M.J. Akzholov**

**Abstract**: The results of mathematical and computer modeling of laminar and turbulent convective heat transfer from an heating element to environment in electronic systems.

**Keywords**: electronic systems, heating elements, convective heat transfer

## Литература

1. М.Ж.Акжолов, П.И.Кандалов, И.Г.Лебо, А.Г.Мадера. Компьютерное моделирование конвективных процессов в воздушной среде вблизи электронных устройств. // Труды НИИСИ РАН. Т. 1. №2. 2011. С. 44 – 46.

2. О.М.Белоцерковский, Ю.М.Давыдов. Метод крупных частиц в газовой динамике. – М.: Наука, 1982.

3. А.Г.Мадера. Моделирование теплообмена в технических системах. – М.: НФ им. акад. В.А. Мельникова, 2005.

4. А.Г.Мадера. Математическое моделирование свободного конвективного теплообмена в электронных системах // Труды НИИСИ РАН. Т. 1, № 1, 2011, с. 31 – 37.

5. А.Г.Мадера, М.Ж.Акжолов, И.Г.Лебо. Моделирование развития процессов «конвекция плюс теплопроводность» в воздухе вблизи процессора. // Труды НИИСИ РАН. Т. 3, № 1, 2013, с. 90 – 93.

6. A.G.Madera. Heat transfer from an extended surface at a stochastic heat-transfer coefficient and stochastic environmental temperature // International Journal of Engineering Science. 1996. т. 34, № 9, с. 1093-1099.

# К теории исчисления обыкновенных разделённых и конечных разностей

**В.Б.Демидович**

*кандидат физико-математических наук*

**Аннотация:** В работе выводятся формулы взаимосвязи обыкновенных разделённых и конечных разностей *при произвольном распределении узлов*, на основании которых исследуются некоторые свойства поведения этих разностей на классе гладких функций.

**Ключевые слова:** разделённые разности, конечные разности, произвольно распределённые узлы, равноотстоящие узлы, гладкие функции

## Введение

В классической *теории исчисления разностей* функций хорошо известны свойства обыкновенных *разделённых* и *конечных* разностей (см., например, [1]-[3]). При этом если *разделённые разности* рассматриваются, вообще говоря, относительно *произвольной* системы узлов, то для *конечных разностей узлы предполагаются равноотстоящими*. Правда, ещё Шарль Эрмит (1822-1901) указывал на возможность рассматривать конечные разности «с переменными сдвигами», но широкого распространения эта идея, насколько нам известно, не получила.

В данной работе мы, вводя обыкновенные конечные разности относительно произвольной системы узлов, устанавливаем их связь с соответствующими обыкновенными разделёнными разностями относительно этой системы узлов и устанавливаем некоторые свойства искомых разностей «на классах функций», важных в *теории аппроксимации*.

## $ 1. Предварительные сведения

Положим, что

$$f : X \subseteq R^1 \to R^1 \qquad (1)$$

- действительная функция от одной переменной.

**Определение 1.** Пусть выбраны и пронумерованы две *различные* точки $x_0, x_1 \in X$. Тогда величину

$$\delta^1 f(x_0, x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

называют (классической) *разделённой разностью 1 –го порядка функции f относительно узлов* $x_0$, $x_1$.

Вообще, если выбраны и пронумерованы (n+1)-на попарно различные точки $x_0, x_1, ..., x_n \in X$, то рекуррентно определяемая величина

$$\delta^n f(x_0, x_1, ..., x_n) = \frac{\delta^{n-1} f(x_1, ..., x_n) - \delta^{n-1} f(x_0, ..., x_{n-1})}{x_n - x_0},$$

$$\delta^0 f(x_0) \equiv f(x_0), \quad n \geq 1, \qquad (2)$$

называется (классической) *разделённой разностью n-го порядка функции f относительно узлов* $x_0$, $x_1, ..., x_n$.

Известно, что для (классических) разделённых разностей справедливы нижеследующие формулы, проверяемые, например по индукции.

**Лемма 1.** Для (классической) разделённой разности *n*-го порядка функции *f* относительно (попарно различных) узлов $x_0, x_1, ..., x_n$ верно представление

$$\delta^n f(x_0, ..., x_n) = \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}. \qquad (3)$$

Обратно, для значения $f(x_n)$ верна формула

$$f(x_n) = f(x_0) + \sum_{i=1}^{n} \prod_{j=0}^{i-1} (x_n - x_j)\, \delta^i f(x_0, ..., x_i), \qquad (4)$$

где $\delta^i f(x_0, ..., x_i)$ $(i = 1, 2, ..., n)$ - (классические) разделённые разности *i*-го порядка функции *f* относительно соответствующих узлов $x_0, ..., x_i$.

***Следствие 1.*** (Классическая) разделённая разность функции *симметрична по отношению к своим узлам*.

***Следствие 2.*** (Классическая) разделённая разность *линейной комбинации функций обладает свойством линейности* по отношению к функциям комбинации.

***Следствие 3.*** Формула (4) является некоторым *дискретным аналогом классической формулы разложения функций в степенной ряд,* установленной ещё Бруком Тейлором. (1685-1731).

**Определение 2.** Пусть выбраны и пронумерованы две точки $x_0, x_1 \in X$. Тогда величину

$$\Delta^1 f(x_0, x_1) = f(x_1) - f(x_0)$$

называют (классической) *конечной разностью 1 –го порядка функции f относительно узлов* $x_0$, $x_1$.

Вообще, если выбраны и пронумерованы (n + 1)-на точки $x_0, x_1, \ldots, x_n \in X$, то рекуррентно определяемая величина

$$\Delta^n f(x_0, \ldots, x_n) =$$
$$= \Delta^{n-1} f(x_1, \ldots, x_n) - \Delta^{n-1} f(x_0, \ldots, x_{n-1}), \quad (5)$$
$$\Delta^0 f(x_0) \equiv f(x_0), \quad n \geq 1,$$

называется (классической) *конечной разностью n-го порядка функции f относительно узлов* $x_0$, $x_1, \ldots, x_n$.

**Замечание 1.** Подчеркнём, что в данном определении *конечной разности узлы* (в принципе) *не обязаны быть различными*, *но важна их последовательная нумерация*, в то время как для вышеприведенного определения *разделённой разности,* наоборот, в силу её симметричности по отношению к своим узлам, *последовательная нумерация узлов не существенна*, *но требование их «попарной различности»* (без дополнительной информации *о гладкости* исходной функции $f$), *необходимо*.

Как и для рассматриваемой разделённой разности, на основании индукции легко проверяются нижеследующие формулы, обычно выписываемые лишь для равноотстоящих узлов.

**Лемма 2.** Для (классической) конечной разности *n*-го порядка функции $f$ относительно узлов $x_0, x_1, \ldots, x_n$ верно представление

$$\Delta^n f(x_0, \ldots, x_n) = \sum_{j=0}^{n} (-1)^{n-i} C_n^i f(x_i), \quad (6)$$

где $C_n^i$ $(i = 0, 1, \ldots, n)$ - соответствующие *биномиальные коэффициенты*.

Обратно, для значения $f(x_n)$ верна формула

$$f(x_n) = f(x_0) + \sum_{i=1}^{n} C_n^i \Delta^i f(x_0, \ldots, x_i) \quad (7)$$

где $\Delta^i f(x_0, \ldots, x_i)$ $(i = 1, 2, \ldots, n)$ - (классические) конечные разности *i*-го порядка функции $f$ относительно соответствующих узлов $x_0, \ldots, x_i$.

*Следствие.* Аналогично (классическим) разделённым разностям, *операция взятия* (классической) *конечной разности от линейной комбинации функций обладает свойством линейности* (по отношению к функциям комбинации), однако *свойство симметричности* (по отношению к своим узлам) *у* (классической) *конечной разности,* вообще говоря, *отсутствует*. Добавим, что формула (7) является ещё одной записью дискретного аналога формулы разложения функций в степенной ряд.

## $ 2. О некоторых свойствах обыкновенных разностей

Широко известна связь классических разделённых и конечных разностей *в случае системы равноотстоящих узлов*. Здесь мы, на основании простых рассуждений, выведем подобную связь *в случае произвольной системы узлов*, а затем приведём некоторые свойства рассматриваемых разностей *для гладких функций.* При этом, эпитет «классический» перед соответствующими разностями, для краткости, мы будем просто опускать.

**Теорема 1.** Пусть для функции

$$f : X \subseteq R^1 \to R^1 \quad (8)$$

выбраны и пронумерованы (n + 1)-на $(n \geq 1)$ попарно различные точки $x_0, x_1, \ldots, x_n \in X$,

Тогда:

**1.** Для выражения разделённой разности *n*-го порядка функции $f$ относительно узлов $x_0, x_1, \ldots, x_n$ справедлива формула

$$\delta^n f(x_0,...,x_n) = \tag{9}$$
$$= \sum_{j=0}^{n-1} [\sum_{i=j+1}^{n} \frac{C_i^{j+1}}{\prod_{\substack{k=0 \\ k\neq i}}^{n}(x_i - x_k)}]\, \Delta^{j+1} f(x_0,...,x_{j+1}),$$

где $\Delta^{j+1} f(x_0,...,x_{j+1})$ $(j=0,1,...,n-1)$ - конечные разности *(j+1)*-го порядка функции $f$ относительно узлов $x_0,...,x_{j+1}$.

**2.** В свою очередь, для выражения конечной разности *n*-го порядка функции $f$ относительно узлов $x_0, x_1,..., x_n$ справедлива формула

$$\Delta^n f(x_0,...,x_n) =$$
$$\sum_{j=0}^{n-1} [\sum_{i=j+1}^{n} (-1)^{n-i} C_n^i \prod_{k=0}^{j}(x_i - x_k)] \times \tag{10}$$
$$\times \delta^{j+1} f(x_0,...,x_{j+1}),$$

где $\delta^{j+1} f(x_0,...,x_{j+1})$ $(j=0,1,...,n-1)$ - разделённые разности *(j+1)*-го порядка функции $f$ относительно узлов $x_0, x_1,..., x_{j+1}$.

*Доказательство.*

**1.** Подставляя выражение для $f(x_i)$, получаемое из формулы (7) Леммы 2 в формулу (3) Леммы 1, имеем

$$\delta^n f(x_0,...,x_n) = \sum_{i=0}^{n} \frac{\sum_{j=0}^{i} C_i^j \Delta^j f(x_0,...,x_j)}{\prod_{k=0,\,k\neq i}^{n}(x_i - x_k)}$$
$$= \sum_{i=0}^{n}\sum_{j=0}^{i} \frac{C_i^j \Delta^j f(x_0,...,x_j)}{\prod_{k=0,\,k\neq i}^{n}(x_i - x_k)}. \tag{11}$$

После изменения порядка двойного суммирования в формуле (11), находим

$$\delta^n f(x_0,...,x_n) =$$
$$\sum_{j=0}^{n}\sum_{i=j}^{n} [\frac{C_i^j}{\prod_{k=0,\,k\neq i}^{n}(x_i - x_k)}]\, \Delta^j f(x_0,...,x_j) =$$
$$= [\sum_{i=0}^{n} \frac{1}{\prod_{k=0,\,k\neq i}^{n}(x_i - x_k)}]\, f(x_0) + \tag{12}$$
$$+ \sum_{j=1}^{n} [\sum_{i=j}^{n} \frac{C_i^j}{\prod_{k=0,\,k\neq i}^{n}(x_i - x_k)}]\, \Delta^j f(x_0,...,x_j).$$

Но ясно, что при $n \geq 1$

$$\sum_{i=0}^{n} \frac{1}{\prod_{k=0,\,k\neq i}^{n}(x_i - x_k)} = 0 \tag{13}$$

как выражение для разделённой разности *n*-го порядка от функции, равной *единице* в рассматриваемых узлах (см. формулу (3) Леммы 1). Поэтому формула (12), после очевидной переиндексации, примет требуемый вид (9).

**2.** В свою очередь, подставляя в формулу (6) Леммы 2 выражение для $f(x_i)$, получаемое из формулы (4) Леммы 1, находим

$$\Delta^n f(x_0,...,x_n) = \sum_{i=0}^{n} (-1)^{n-i} C_n^i \times$$
$$\times [f(x_0) + \sum_{j=1}^{i>0} \prod_{k=0}^{j-1}(x_i - x_k)\, \delta^j f(x_0,...,x_j)] =$$
$$= [\sum_{i=0}^{n} (-1)^{n-i} C_n^i]\, f(x_0) + \tag{14}$$
$$+ \sum_{i=1}^{n} (-1)^{n-i} C_n^i \sum_{j=0}^{i-1} \prod_{k=0}^{j}(x_i - x_k)\, \delta^{j+1} f(x_0,...,x_{j+1}).$$

Но, очевидно, при $n \geq 1$

$$\sum_{i=0}^{n} C_n^i (-1)^{n-i} = 0. \tag{15}$$

Поэтому формула (14), после изменения в ней порядка двойного суммирования, принимает требуемый вид (10).

Теорема полностью доказана.

***Следствие.*** В условиях данной Теоремы в случае *равноотстоящих* узлов

$$\{x_i = x_0 + ih \quad (i=0,1,...,n),$$
$$h > 0 - const, \quad x_i \in X\} \tag{16}$$

обе формулы (9) – (10) объединяются в единую (широко известную) формулу

$$\delta^n f(x_0, ..., x_n) = \frac{\Delta^n f(x_0, ..., x_n)}{n!\, h^n}. \tag{17}$$

Для проверки этого факта следует лишь упростить, в случае равноотстоящих узлов, какую-нибудь из двух данных формул.

В самом деле, например, из формулы (10), учитывая очевидное равенство

$$\prod_{k=0}^{j}(x_i - x_k) = h^{j+1}[i\cdot(i-1)\cdots(i-j)]$$
$$(0 \leq j \leq i-1), \tag{18}$$

а также равенства

$$\sum_{i=j+1}^{n}(-1)^{n-i}[i\cdot(i-1)\cdots(i-j)]C_n^i = \begin{cases}0 & при\ j=0,1,\ldots,n-2\\ n! & при\ j=n-1\end{cases}, \quad (19)$$

получаемые после последовательного дифференцирования по параметру $p$ тождества

$$\sum_{i=0}^{n} C_n^i\, p^i\cdot q^{n-i} = (p+q)^n \quad (20)$$

и подстановки $p=1,\ q=-1,$ сразу находим

$$\Delta^n f(x_0,\ldots,x_n) = \sum_{j=0}^{n}\{\sum_{i=j+1}^{n}(-1)^{n-i}\times[i\cdot(i-1)\cdots(i-j)]C_n^i\}\times \times h^{j+1}\cdot\delta^{j+1} f(x_0,\ldots,x_{j+1}) = n!\cdot h^n\cdot\delta^n f(x_0,\ldots,x_n), \quad (21)$$

что нам и требовалось.

При наличии дополнительной информации об исходной функции $f$ можно указать ряд полезных свойств её обыкновенных разностей. Здесь мы рассмотрим лишь случай, когда $f$ принадлежит одному из весьма распространённых в *теории аппроксимации* классу гладких функций, заданному на конечном отрезке.

**Определение 3.** Говорят (см., например, [5]), что непрерывная функция $y=f(x),\ x\in[a,b],$ принадлежит классу

$$W^{(n)}([a,b])\ \ (n\geq 1-целое),$$

если на заданном промежутке она имеет непрерывные производные до $(n-1)$-го порядка включительно, и кусочно-непрерывную производную $n$-го порядка.

Если же, сверх того, для данной функции на промежутке $[a, b]$ выполняется неравенство

$$|f^{(n)}(x)|\leq M_n, \quad (22)$$

где $M_n$ - некоторая фиксированная положительная константа, то говорят, что функция $f(x)$ принадлежит классу

$$W^{(n)}(M_n;[a,b])\subset W^{(n)}([a,b]).$$

**Замечание 2.** На заданном промежутке кусочная непрерывность функции здесь понимается в том смысле, что этот промежуток можно разбить на *конечное число* подпромежутков, внутри которых будет иметь место непрерывность функции, а на их концах *будут существовать* соответствующие *односторонние производные.*

**Теорема 2.** Пусть для функции $f(x)\in W^{(n)}([a,b])$ выбраны и пронумерованы (n + 1)-на точки $x_0,\ x_1,\ldots,x_n\in[a,b].$

Тогда:

**1.** Для выражения *разделённой* разности $n$-го порядка функции $f(x)$ относительно системы узлов $x_0,\ x_1,\ldots,x_n$ справедливо представление

$$\delta^n f(x_0,\ldots,x_n) = \int_0^1\int_0^{t_1}\cdots\int_0^{t_{n-1}} f^{(n)}[x_0+\sum_{i=1}^{n}(x_i-x_{i-1})t_i]\,dt_1\ldots dt_n. \quad (23)$$

**2.** Для выражения *конечной* разности $n$-го порядка функции $f(x)$ относительно системы узлов $x_0,\ x_1,\ldots,x_n$ справедливо представление

$$\Delta^n f(x_0,\ldots,x_n) = \sum_{j=0}^{n-2}\{\sum_{i=j+1}^{n}(-1)^{n-i}C_n^i\prod_{k=0}^{j}(x_i-x_k)\}\frac{f^{(j+1)}(\xi_{j+1})}{(j+1)!} + \prod_{k=0}^{n-1}(x_n-x_k)\times \times\int_0^1\int_0^{t_1}\cdots\int_0^{t_{n-1}} f^{(n)}[x_0+\sum_{l=1}^{n}(x_l-x_{l-1})t_l]\,dt_1\ldots dt_n, \quad (24)$$

где $\xi_{j+1}\ \ (j=0,1,\ldots,n-2)$ - некоторые «промежуточные точки» для точек $x_0,\ x_1,\ldots,x_n$ .

*Доказательство.*

**1.** Формула (23) известна в случае, когда $f(x)\in C^{(n)}[(a,b])$, то есть когда функция $f(x)$ принадлежит, на отрезке $[a, b]$ классу непрерывных функций с непрерывными производными вплоть до $n$-го порядка включительно (см., например, [1], [3]). В функциональном классе $W^{(n)}([a, b])$ она также остаётся в силе, что нетрудно понять из соображений индукции с учётом применимого в данном случае известного соотношения между взятием интеграла функции и вычислением её первообразной (корнями уходящего к классическим исследованиям Исаака Ньютона (1643-1727) и Готфрида Лейбница (1646-1716)), называемого в математическом анализе *правилом Ньютона-Лейбница.*

Заметим, что если $f(x)\in C^{(n)}[(a,b])$, то на основании *теоремы о среднем* для данного интеграла, область интегрирования которого является $n$-мерной пирамидой, из формулы (23), в частности, вытекает широко известная связь

$$\delta^n f(x_0,...,x_n)=\frac{f^{(n)}(\xi)}{n!}, \qquad (25)$$

где $\xi$ - некоторая «промежуточная точка» на отрезке $[a, b]$.

**2.** Формула (24) является прямым следствием подстановки выражений вида (23) для разделённых разностей в общую формулу (10) Теоремы 1 для конечной разности, если только учесть связи вида (25) для непрерывных (на соответствующих промежутках) производных $f^{(j+1)}(x)\ \ (j=0,1,...,n-2)$.

Тем самым, теорема доказана.

***Следствие 1.*** Как видно из доказательства данной Теоремы, в случае $f(x)\in C^{(n)}[(a,b])$ разделённая разность функции *f*(*x*) дополнительно характеризуется формулой (25), а её конечная разность – соответственно формулой

$$\Delta^n f(x_0,...,x_n)=\sum_{j=0}^{n-1}[\sum_{i=j+1}^{n}(-1)^{n-i}\times \\ \times C_n^i\prod_{k=0}^{j}(x_i-x_k)]\frac{f^{(j+1)}(\vartheta_{j+1})}{(j+1)!}, \qquad (26)$$

где $\vartheta_{j+1}\ \ (j=0,1,\ ...\ ,n-1)$ - некоторые «промежуточные точки» для точек $x_0, x_1,..., x_n$ .

***Следствие 2.*** Если исходные узлы *равноотстоящие*

$$\{x_i=x_0+ih\quad (i=0,1,...,n),\\ h>0-const,\quad x_i\in[a,b]\}, \qquad (27)$$

то формулу (23) можно ещё переписать в виде

$$\delta^n f(x_0,\ ...,x_n)=\\ =\frac{1}{n!\,h^n}\int_0^h...\int_0^h f^{(n)}[x_0+\sum_{i=1}^{n}\tau_i]\,d\tau_1...d\tau_n, \qquad (28)$$

а формулу (24) - соответственно, в виде

$$\Delta^n f(x_0,...,x_n)=\\ =\int_0^h...\int_0^h f^{(n)}[x_0+\sum_{i=1}^{n}\tau_i]\,d\tau_1...d\tau_n. \qquad (29)$$

Отсюда в случае $f(x)\in C^{(n)}([a,b])$ сразу вытекает другая, аналогичная формуле (25), широко известная связь

$$\Delta^n f(x_0,...,x_n)=h^n f^{(n)}(\xi), \qquad (30)$$

где $\xi$ - некоторая «промежуточная точка» на отрезке $[a, b]$.

В самом деле, для обоснования формул (28) - (29) (а, значит, и формулы (30)) заметим, что в случае равноотстоящих узлов аргумент подинтегральной функции в формуле (23) симметричен по отношению к переменным $t_1, t_2, \dots, t_n$. Следовательно, при всевозможных перестановках этих переменных (а их ровно $n!$) интеграл правой части этой формулы не меняется, причём их сумма даст, как легко понять, значение данного интеграла по *n*-мерному единичному кубу. Отсюда, учитывая замену переменной $\tau_i=h\,t_i$, получаем

$$\delta^n f(x_0,...,x_n)=\\ =\frac{1}{n!}\int_0^1...\int_0^1 f^{(n)}[x_0+h\sum_{i=1}^{n}t_i]\,dt_1...dt_n= \qquad (31)\\ =\frac{1}{n!h^n}\int_0^h...\int_0^h f^{(n)}[x_0+\sum_{i=1}^{n}\tau_i]\,d\tau_1...d\tau_n,$$

и формула (28) обоснована.

Формула же (29) сразу вытекает из формулы (28) на основании Следствия Теоремы 1.

***Следствие 3.*** Если

$$f(x)\in W^{(n)}(M_n;\ [a,\ b]),$$

то на данном классе функций справедлива оценка

$$|\delta^n f(x_0,\ ...,x_n)|\le\frac{M_n}{n!} \qquad (32)$$

и, соответственно, оценка

$$|\Delta^n f(x_0,\ ...\ ,x_n)|\le M_n(b-a)^n\alpha_n, \qquad (33)$$

где

$$\alpha_n=(\sum_{i=1}^{n}\sum_{j=0}^{i-1}\frac{1}{(j+1)!})\,C_n^i. \qquad (34)$$

Оценка (32) очевидным образом вытекает из формулы (23). Оценка (33)–(34) легко получается из формулы (24), если только учесть, что для $f(x)\in W^{(n)}(M_n;\ [a,\ b])$ справедливы вполне понятные соотношения

$$|f^{(j)}(x)|\le M_n(b-a)^{n-j}\quad (j=1,\ ...\ ,n), \qquad (35)$$

гарантирующие, в свою очередь, соотношения

$$
\begin{aligned}
&|\Delta^n f(x_0, \dots, x_n) \le \\
&\le \sum_{j=0}^{n-1} \{ \sum_{i=j+1}^{n} C_n^i (b-a)^{j+1} \} \frac{M_n (b-a)^{n-j-1}}{(j+1)!} = \\
&= M_n (b-a)^n \sum_{j=0}^{n-1} \frac{\sum_{i=j+1}^{n} C_n^i}{(j+1)!} = \\
&= M_n (b-a)^n \sum_{i=1}^{n} (\sum_{j=0}^{i-1} \frac{1}{(j+1)!}) C_n^i .
\end{aligned}
\tag{36}
$$

**Замечание 3.** Из вида (34) константы $\alpha_n$ ясно, что её *грубо* можно оценить простым неравенством

$$1 \le \alpha_n \le (2^n - 1)e, \tag{37}$$

где $e$ - основание натурального логарифма.

**Замечание 4.** Оценка (32) *неулучшаема* на классе функций $W^{(n)}(M_n; [a, b])$, так как, например, она достигается, когда $f(x)$ есть многочлен вида

$$f(x) = \frac{M_n}{n!} x^n . \tag{38}$$

Отметим, что неулучшаемость оценки вида (32) в функциональном классе $C^{(n)}(M_n; [a, b])$ (то есть на классе функций, являющимся подклассом для $C^{(n)}([a, b])$, характеризуемым, при этом, ограниченностью константой $M_n$ на отрезке [*a, b*] модуля *n*-ой производной входящих в него функций) указана в монографии [3].

Оценку же (33) - (34), за счёт более точного учёта распределения узлов, возможно, можно и улучшить.

**Замечание 5.** На основании Теоремы 2 и её Следствий вытекает, что обыкновенная конечная разность *n*-го порядка *гладкой* функции $f(x)$, относительно произвольно заданных узлов характеризуется, вообще говоря, не только *n*-ой производной от $f(x)$ (как это имеет место для соответствующей её разделённой разности *n*-го порядка и для её конечной разности *n*-го порядка относительно равноотстоящих узлов), но и её *производными всех предыдущих порядков*.

В частности, этим объясняется, почему конечная разность *n*-го порядка относительно произвольно заданных узлов, для алгебраического многочлена (*n-1*)-ой степени, вообще говоря, отлична от нуля (что, как вытекает из предыдущего, невозможно для соответствующей разделённой разности *n*-го порядка или для конечной разности *n*-го порядка относительно равноотстоящих узлов). Например, если на соответствующем промежутке

$$f(x) = 2x, \quad x_0 = 1, \; x_1 = 2, \; x_2 = 4, \tag{39}$$

то

$$
\begin{aligned}
&\Delta^2 f(x_0, x_1, x_2) = \\
&= f(x_0) - 2 f(x_1) + f(x_2) = 2 \ne 0,
\end{aligned}
\tag{40}
$$

хотя $f(x)$ есть многочлен 1-ой степени.

По нашему мнению благодаря этой более сложной природе связи конечных разностей относительно произвольной системы узлов с соответствующими производными, они будут полезны в вычислительной математике для более тонкого исследования дифференциальных свойств нетривиальных функций.

## Заключение

Выскажем теперь общие соображения по поводу развития данной работы.

1. Нам представляется, что введенное здесь понятие (классической) конечной разности относительно произвольной системы узлов допускает обобщение на *комплексный случай*, а также на случай *функций от многих переменных*.

2. Изучаемые здесь (классические) конечные разности относительно произвольной системы узлов могут пригодиться для исследования общих *импульсных* и *разностных* уравнений (в частности, в отношении *устойчивости* их решений), рассматриваемых *на произвольной сетке* (подобные уравнения исследуются, например, в [6] – [12]).

3. Как известно, (классические) *конечные разности относительно равноотстоящих узлов* лежат в основе введения соответствующих *модулей непрерывности,* активно используемых в *теории аппроксимации непрерывных функций.* Рассмотренные здесь конечные разности *относительно произвольной системы узлов* позволяют ввести в эту теорию «более тонкие» модули непрерывности для эффективного исследования непрерывных функций с «более хитрым поведением».

4. Наконец, напомним, что если в *теории аппроксимации* понятие *обобщённой разделённой разности относительно произвольных узлов* от заданной функции $f(x) \in C([a,b])$ *в силу рассматриваемой* (в пространстве $C([a,b])$) *чебышёвской системы функций* (такие системы были введены в математический обиход Пафнутием Львовичем Чебышёвым (1821-1894), а подробные сведения о чебышёвских системах функций изложены, например, в [13] – [17], причём в [16] изучаются соответствующие *обобщённые* разделённые разности) *известно*, то понятие

*обобщённой конечной разности* от заданной функции $f(x) \in C([a, b])$ *в силу рассматриваемой чебышёвской системы функций,* даже в случае *равноотстоящих узлов,* никем не определялось.

Представляется, что и в этом направлении можно получить «далеко идущие» обобщения изложенных в работе результатов.

# On the theory for calculus of ordinary finite and divided differences

**V.B.Demidovich**

**Abstract:** In the paper are derived the formulas of interconnection between ordinary divided differences and finite differences *for arbitrary distribution of the nodes*, serving as a basis for study of some properties of behavior these differences on the class of smooth functions.

**Keywords:** finite difference, divided difference, arbitrarily distributed nodes, equidistant nodes, smooth function.

## Литература

1. А.О.Гельфонд. Исчисление конечных разностей. М., «Наука», 1967.
2. Р.Хемминг. Численные методы. М., «Наука», 1968.
3. В.И.Крылов, В.В.Бобков, П.И.Монастырный. Вычислительные методы высшей математики. Минск, «Вышейшая школа», 1972.
4. R.Folsom, D.Boger, H. Millikin. Stability conditions for linear constant coefficient difference equations in generalized differenced form. «Econometrica», v. 44 (1976), № 3, 575-591.
5. С.М.Никольский. Квадратурные формулы. М., «Наука», 1974.
6. А.Халанай, Д.Векслер. Качественная теория импульсных систем. М., «Мир», 1971.
7. A.Halanay. Quelques questyions de la theorie de la stabilite pour les systemes aux differences finies. «Arch. Ration Mech. And Analysis», v. 12 (1963), № 2, 150-154.
8. R. Driver. Note on a paper of Halanay on stability for finite difference equations. «Arch. Ration Mech. And Analysis», v. 18 (1965), № 3, 241-243.
9. D.Rao. Stability of complex difference equations. «Proc. Nat. Acad. Sci India», v. A-36 (1966), № 4, 953-955.
10. L.Grujuc, D.Siljak. On a stability of discrete composite systems. «JEEE Trans., Automat. Contr.», v. 18 (1973), № 5, 522-524.
11. G.Bitsoris, G.Burgat. Stability analysis of complex discrete systems with locally and globally stable subsystems. «Int. J. Contr.», v. 255 (1977), № 3, 413-424.
12. T.Denkowski. On a criterion of uniqueness for periodic solutions of linear second order difference equations. «Ann. Pol. Math.», v. 27 (1972), № 1, 41-49.
13. С.Карлин, В.Стадден. Чебышёвские системы и их применение в анализе и статистике. М., «Наука», 1976.
14. М.Г.Крейн, А.А.Нудельман. Проблемы моментов Маркова и экстремальные задачи. М., «Наука», 1973.
15. В.Б.Демидович. Приближённые вычисления с помощью обобщённых полиномов из чебышёвских пространств: чебышёвские обобщённые полиномы. М., «Изд -во Моск. ун –та», 1990.
16. В.Б.Демидович. Приближённые вычисления с помощью обобщённых полиномов из чебышёвских пространств: простое интерполирование, кратное интерполирование, формулы тейлоровского типа. М., «Изд -во Моск. ун –та», 1994.
17. В.Б.Демидович, Г.Г.Магарил-Ильяев, В.М.Тихомиров. Экстремальные задачи для линейных функционалов на чебышёвских пространствах. «Фундаментальная и прикладная математика» т. 11 (2005), № 2, 87-100.

# Реализация принципа контролируемого выполнения для прикладных программ реального времени

**А.И. Грюнталь[1], К.Г.Нархов, А.М. Щегольков**

*1 - кандидат физико-математических наук*

**Аннотация:** Контролируемое выполнение представляет собой парадигму, нацеленную на гарантированное выполнение программой своей миссии в неблагоприятных условиях ее выполнения (программные ошибки, сбои аппаратуры, некорректные входные данные, и т. д.). В статье рассматривается методы и средства реализации принципа контролируемого выполнения для многопоточных прикладных программ реального времени. Вводится концепция монитора, который оценивает характеристики выполнения программы и генерирует команды управления выполнением потоков управления. Представлены некоторые особенности программной реализации монитора.

**Ключевые слова:** контролируемое выполнение, многопоточная программа, поток управления, очередь сообщений, сигнал, операционная система реального времени, команда управления.

## Введение

Принцип контролируемого выполнения [1, 2]представляет собой общий концептуальный поход к проектированию и разработке программ, направленный на безусловное обеспечение выполнения программой своей миссии (цели). Программа с контролируемым выполнением – это программа со встроенными механизмами, обеспечивающими выполнение программы, вообще говоря, при некорректных внешних данных, ошибках в среде исполнения, ошибках в самой прикладной программе, и др.

В настоящей статье рассматривается одна из реализаций принципа контролируемого выполнения для прикладных программ, особое внимание уделяется структурным, архитектурным и низкоуровневым реализационным особенностям. Приводится описание библиотеки мониторинга – программного средства, обеспечивающего стандартизованную генерацию объектов программы с контролируемым выполнением.

Средой выполнения программ с контролируемым выполнением была операционная система реального времени ОС РВ Багет 2.6 (далее называемая ОС РВ).

## 1 Структура прикладной программы с контролируемым выполнением

Основу средств контролируемого выполнения прикладной программы (в рассматриваемой реализации) составляют механизмы, обеспечивающие контроль состояния выполняемой программы, сравнение параметров состояния прикладной программы с эталоном, и, в случае отклонения поведения программы от эталона, генерацию управляющих воздействий, направленных на устранение выявленных отклонений.

Реализация изложенного подхода требует наличия в прикладной программе агентов, регистрирующих и собирающих данные о выполнении прикладной программы (контрольные параметры состояния), монитора, анализирующего в реальном масштабе времени получаемые от агентов контрольные параметры состояния и генерирующего управляющие воздействия (команды управления приложением), и средств выполнения сгенерированных монитором команд управления приложением.

Кроме того, в прикладной программе должны присутствовать средства передачи контрольных параметров состояния от агентов к монитору и средства, передающие команды управления приложением. Наконец, монитор должен содержать эталон, то есть модель корректного выполнения прикладной программы, содержащую прогнозируемые (эталонные) значения контрольных параметров.

В общем виде прикладная программа с реализованными средствами контролируемого выполнения включает следующие структурные компоненты:

- собственноприложение, то есть ту часть программы, которая реализует прикладную функциональность;
- внедрённые в приложение агенты;
- монитор;
- средства реализации управляющих воздействий;
- средства передачи контрольных данных состояния от агентов к монитору;
- средства передачи команд управления приложением от монитора к приложению;
- модель корректного выполнения приложения.

Прикладную программу, содержащую перечисленные структурные компоненты, будем далее называть программой с контролируемым выполнением. Заметим также, что для ясности формулировок понятие приложения и прикладной

программы различаются. Под приложением понимается та часть прикладной программы, которая непосредственно отвечает за прикладную функциональность. Прикладная программа в целом содержит приложение и фрагменты, обеспечивающие организацию вычислений и контролируемое выполнение. Концептуальная структура прикладной программы с контролируемым выполнением представлена на рисунке 1.

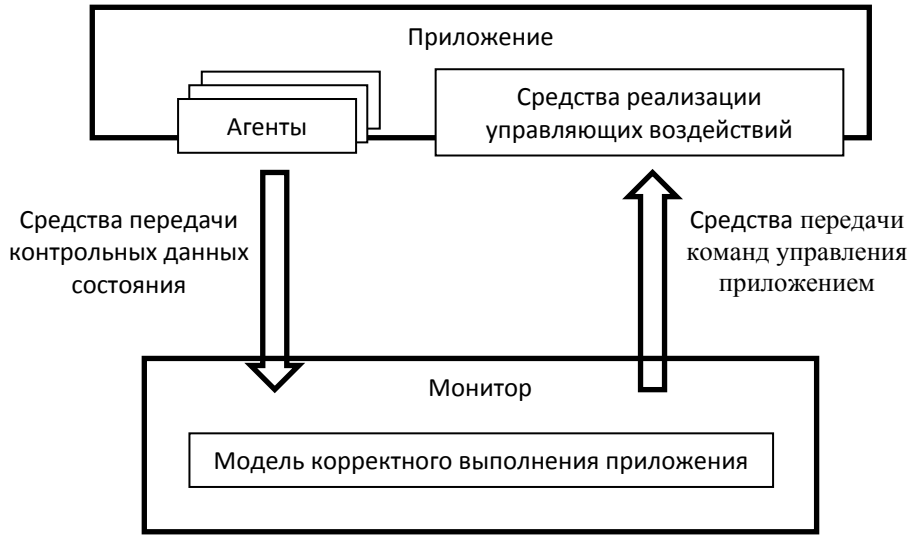



**Рис. 1.** Концептуальная структура прикладной программы с контролируемым выполнением

## 2 Архитектура программы с контролируемым выполнением в среде операционной системе реального времени

Приведём конкретизацию представленной выше структуры программы с контролируемым выполнением, относящуюся к программам реального времени. Существенной чертой программ реального времени является многопоточность, обеспечивающая быстрое (своевременное) переключение программы между различными прикладными функциями в зависимости от поступающих внешних воздействий [3].

Применение многопоточности вытекает из следующих характеристик, которые должны быть присущи программе реального времени:

- *Быстрая и своевременная реакция на прерывания (действия, выполняемые непосредственно процессором и операционной системой между появлением запроса прерывания и началом выполнения обработчика прерывания)*. Важной задачей управления является принятие управляющих решений и генерация управляющих сигналов в связи с изменением внешних воздействий – аппаратных прерываний, в том числе от внешних устройств, индицирующих определенные характеристики окружения, и программных прерываний. Для систем с критической миссией задержка в реакции на такие прерывания может носить катастрофический характер.
- *Обработка прерываний*. Обработка прерываний представляет собой логически отдельную задачу, которую целесообразно (с точки зрения простоты архитектуры программы) выполнять в контексте отдельных потоков управления (*групп функциональных потоков управления*).

Многопоточность предполагает использование в тексте программы объектов операционной системы, связанных с организацией взаимодействия потоков управления, передачей данных и синхронизацией. Операционные системы реального времени спроектированы так, чтобы минимизировать (при заданных аппаратных ресурсах) длительность интервала времени, затрачиваемого на переключение между различными потоками управления.

С учётом сказанного, прикладные программы реального времени целесообразно проектировать так, чтобы различные прикладные функции были локализованы в различных потоках управления. При такой организации программывремя переключения между различными прикладными функциями при изменении внешних условий может выполняться средствами операционной системы реального времени и, тем самым, будет минимальным.

Таким образом, прикладную программу мы рассматриваем как многопоточную программу, каждый поток управления которой выполняет определённую прикладную функцию, обусловленную миссией приложения.

Функционирование прикладной программы представляет собой псевдопараллельное выполнение потоков управления. Последовательность выполнения потоков управления определяется поступающими внешними данными, а при их отсутствии или в случае, когда реакция на внешние данные не должна быть неотложной, определяется выбранной для приложения дисциплиной планирования [4, 5, 6, 7].

Предложенная схема, согласно которой имеется взаимно однозначное соответствие между потоками управления и прикладными функциями, несколько идеализирована, поскольку наряду с потоками управления, выполняющими чисто прикладные функции, программа включает потоки управления, выполняющие вспомогательные функции, обеспечивающие организацию вычислительного процесса в целом. Например, программа может содержать потоки управления, создающие необходимые для функционирования прикладной программы программные ресурсы (очереди сообщений, таймеры, сигналы и др.) Кроме того, некоторые прикладные функции могут быть локализованы не в одном, а в нескольких потоках управления. Например, функциональность, обеспечивающая обмен данными с относительно медленными устройствами, может быть локализована в отдельных потоках управления, которые должны обладать меньшим приоритетом, чтобы их влияние на функциональную часть программы было минимальным. Примером может служить:

- *Использование асинхронного ввода-вывода*. Программы реального времени характеризуются необходимостью ввода-вывода данных с использованием специфических внешних интерфейсов. Программирование конкретных особенностей ввода-вывода может выполняться в специализированных потоках управления, что снижает логическую сложность программы.
- *Реализация диалога с оператором*. При необходимости программа реального времени должна работать в диалоговом режиме. По сути это частный случай асинхронного ввода-вывода.
- *Использование прикладных библиотек*. Целесообразно по возможности изолировать алгоритмы, использующие в своем теле функции (API) прикладных библиотек, в специализированные потоки управления. Это позволит снизить логическую сложность программы и увеличить ее общую масштабируемость.

В связи со сказанным, будем рассматривать более общую ситуацию: будем считать, что каждая прикладная функция может быть, вообще говоря, локализована не в одном, а в нескольких потоках управления. Совокупность потоков управления, реализующих одну прикладную функцию, будем называть группой функциональных потоков управления (ГФП). При этом реакция приложения на конкретное внешнее событие состоит в передаче управления конкретному (заранее определённому при проектировании программы) потоку управления. Такой поток управления будем называть ведущим потоком управления группы функциональных потоков управления.

С учётом того, что ОС РВ включает развитые средства взаимодействия между потоками управления и средства синхронизации [4, 5, 6, 7, 8], описание программы в терминах примитивов ОС РВ представляет собой функциональное описание вычислительного процесса, составляющего высокоуровневую алгоритмическую основу приложения.

Высокоуровневое описание приложение в терминах примитивов ОС РВ позволяет осуществить явную проекцию предъявляемых к приложению функциональных требований на структуру программы. Такой подход делает прозрачным трассировку функциональных требований на структуру программы, что соответствует, в частности, требования спецификации КТ-178 [9].

Отметим также, что излагаемый подход близок к методу послойного программирования, предложенному А.Л. Фуксманом [10].

Излагаемая ниже модель контролируемого выполнения базируется на многопоточном представлении приложения. А именно, взаимодействие между управляющим компонентом и группой функциональных потоков управления осуществляется путём передачи команд отдельным потокам управления, составляющим ГФП.

В рамках этой статьиуправление приложением со стороны монитора будем рассматривать как управление на уровне потоков управления. Это означает, что команды монитора, направляемые приложению, относятся к потоку управления в целом и не затрагивают внутреннюю программную структуру потока управления. Дополнительная детализация функционального наполнения потока управления напрямую зависит от специфики приложения. Внастоящей статье ограничимся общей схемойконтролируемого выполнения прикладной программы (рисунок 2), базирующейся на примитивах ОС РВ и действующей для многопоточных приложений реального времени, независимо от их специфики.

Агенты, регистрирующие и передающие контрольные параметры состояния, представляют собой функции, выполняемые в контексте потока управления. Параметры состояния содержат идентификационную информацию о потоке управления, в контексте которого исполняется агент, о ГФП, в состав которой входит поток управления, и контрольный идентификатор состояния программы. Один поток управления может содержать несколько агентов, и поэтому параметры состояния содержат также идентификационную информацию об агенте, сгенерировавшем параметры состояния. Конкретный формат параметров состояния будет приведён ниже.

Агенты помещают параметры состояния в очередь сообщений к главному потоку управления монитора – к потоку управления для сбора сообщений Mntr (рисунок 2). Текущая реализация предусматривает наличие одной очереди сообщений для каждой группы функциональных потоков управления.

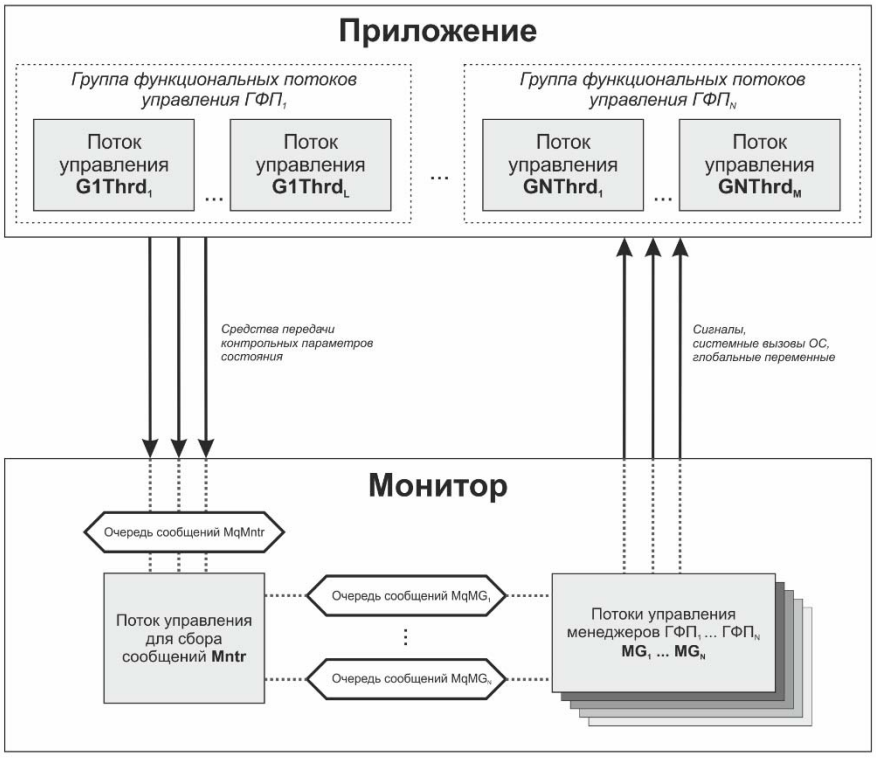
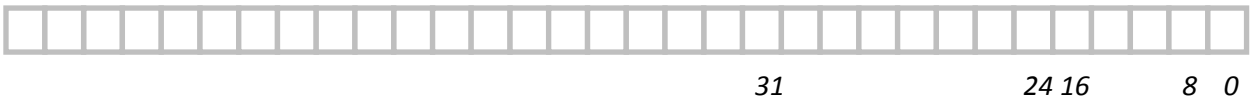



**Рис. 2**. Модель прикладной программы с контролируемым выполнением

В соответствии с рисунком 2 данные от потоков управления приложения поступают в монитор через входную очередь сообщений. Каждое сообщение в очереди сообщений имеет следующий формат:




- битами **0-7** кодируются данные (аргумент *userdata*– см. «Программная реализация агента»), передаваемые в монитор – это может быть команда, значение переменной *errno*(код ошибки) или значение переменной, анализируемой в данной контрольной точке;
- битами **8-15** кодируется идентификатор контрольной точки в потоке управления приложения;
- битами **16-23** кодируется тег потока управления приложения;
- битами **24-31** идентификатор группы функциональных потоков управления, в состав которой входит поток управления, передающий сообщение в монитор.

На основании сообщений, полученных от потока управления из некоторой группы функциональных потоков управления, монитор формирует реакцию, а именно – формирует и отправляет сообщение соответствующему менеджеру (см. «Взаимодействие групп функциональных потоков управления с монитором») группы функциональных потоков управления, который имеет доступ к атрибутам всех входящих в заданную группу функциональныхпотоков управления(см. «Дополнительные атрибуты потоков управления»).

# 3 Состав и функции монитора

Монитор считывает сообщения из очереди сообщений, анализирует их, сравнивает с моделью корректного выполнения приложения, и, в зависимости от результатов анализа, формирует и направляет потокам управления, входящим в составы групп функциональных потоков, команды управления.

Как было отмечено выше, в рамках предлагаемой архитектуры команды, которые генерирует монитор, относятся к потоку управления в целом. Будем рассматривать следующие команды, реализация которых обеспечивается средствами ОС РВ:

- завершить поток управления;
- приостановить поток управления;
- возобновить поток управления;
- приостановить поток управления на заданный промежуток времени (таймаут) и потом возобновить;
- перезапустить поток управления.

Необходимо отметить, что перезапуск потока управления требует специальных действий по удалению созданных потоком управления ресурсов, выполняемых в контексте перезапускаемого потока управления.

Предлагаемый список корректирующих действий для потоков управления аналогичен списку корректирующих действий для «процессов», предусматриваемых спецификацией ARINC 653 [11], выполняемых в результате корректирующих действий «монитора здоровья».

Характер ошибок, которые регистрируются агентами и передаются в монитор, в настоящей статьене рассматривается, поскольку цель статьи – изложение общей архитектуры программы реального времени с контролируемым выполнением, а не анализ конкретных алгоритмов, лежащих в основе вычислительного процесса приложения.

Поэтому логика, лежащая в основе формирования команд управления приложением, также не является предметом настоящего рассмотрения.

В качестве средства передачи команд управления приложением и средства реализации управляющих воздействий используется единый встроенный вОС РВ механизм, а именно, сигналы реального времени и системные вызовы (например, системный вызов порождения нового потока управления), а также глобальные переменные.

# 4 Библиотека мониторинга

Для автоматизированного создания программ реального времени с контролируемым выполнением разработана ииспользуется библиотека мониторинга. Эта библиотека предоставляет прикладному программисту набор функций, автоматизирующих порождение потоков управления, создание средств контроля их выполнения, средств взаимодействия потоков управления с монитором.

Использование библиотеки мониторинга совместно с прикладной программой предполагает вставку вызовов функций библиотеки в исходный текст программы, а также подготовку описания прикладной программы в терминах библиотеки мониторинга.

Библиотека мониторинга оптимизирует работу системного программиста, предоставляя ему инструментарий для контроля, тестирования и отладки многопоточной программы.

## 4.1 Взаимодействие групп функциональных потоков управления с монитором

Каждаягруппа функциональных потоков управления состоит из нескольких(или одного) потоков управления. В контексте этих потоков управления выполняются пользовательские функции, содержащие вызовы агентов контролируемого выполнения. Агент контролируемого выполнения – это функция, вызов которой может быть встроен в исходный код любого потока управления прикладной программы. Агенты, используя очередь сообщений к главному потоку управления монитора (на рисунке 2 – к потоку управления для сбора сообщений Mntr), передают ему информацию о текущем состоянии приложения.

Монитор– это многопоточнаяпрограмма, в которой с каждой группой функциональных потоков управления ассоциирован один поток управленияприложением (один из потоков управления $MG_1 \ldots MG_N$, см. рисунок 2), выполняющий передачу команд от главного потока управленияMntrмонитора конкретному функциональному потоку управления.

Особенностью работы монитора является двухэтапная система передачи информации от главного потока управления монитора (Mntr)к потокам управления приложения ($G1Thrd_1$ …$G1Thrd_L$, … $GNThrd_1$… $GNThrd_M$).На первом этапе информация от главного потока управления монитора (Mntr) передается потоку управления$MG_к$(где K – номер группы функциональных потоков управления) через соответствующую очередь сообщений, а на втором этапекоманды управленияпередаются из потока управления$MG_к$в поток управления приложения посредством механизма сигналов. Данная схема передачи информация используется для минимизации общего количествачередей сообщений между приложением и монитором, поскольку если бы на втором этапе вместо сигналов использовались очереди сообщений, то их общее количество ровнялось бы количеству потоков управления $MG_1 \ldots MG_N$.

Далее по тексту будем называть потоки управления$MG_1 \ldots MG_N$*менеджерами*групп функциональных потоков управления.

## 4.2 Использование библиотеки мониторинга

Использование библиотеки мониторинга предполагает, что программист выполнил проектирование приложения в терминах потоков управления, а именно: определил количество групп функциональных потоков управления, количество потоков управления, входящих в каждую группу, и разработал для каждого потока управления головную функцию на языке Си.

Для удобства интегрированиявходящих в библиотеку мониторинга функций в прикладную программу реального времени, спроектированную в терминах потоков управления, составляется файл конфигурации на языке Си. Файл конфигурации включает описание групп функциональных потоков управления (в виде массива структуртипа usergfp), иописание потоков управления, входящих в заданную группу (в виде массива структур типа userthreads). Структура типа usergfp содержит поле threads, являющееся указателем на массив типа userthreads. Подробное описание полей структур usergfp и userthreads представлено в разделе «Структуры данных библиотеки мониторинга».

На основании описания, выполненного при конфигурировании системы, библиотека мониторинга формирует программу, выполняющую запуск всех потоков управления прикладной программы и организующую мониторинг запущенных потоков управления. Таким образом, при использовании

библиотеки мониторинга прикладному программисту не требуется выполнять запуск потоков управления из прикладной программы – все описанные на этапе конфигурирования системы потоки управления будут запущены автоматически библиотекой мониторинга.

## 4.3 Дополнительные атрибуты потоков управления

В библиотеке мониторинга каждый запущенный поток управления описывается структурой данных типа gfp_thread_pool. С помощью этого описания потоку управления назначаются дополнительные атрибуты, необходимые для его идентификации и адресации в мониторе. Основные атрибуты– это:

- *Идентификатор группы функциональных потоков управления*. Идентификаторыназначаются группам прикладным программистом на этапе конфигурирования.
- *Количество потоков управления в группе функциональных потоков управления.* Размер группы, в которую входит данный поток управления.
- *Тег потока управления*. Номер потока управления в группе функциональных потоков управления.
- *Идентификатор потока управления типа pthread_t*. Используется значение, полученное посредством системного вызова OCPBpthread_self().
- *Флаг остановки потока управления is_frozen*. Используется для программной остановки потока управления.

При передачи команд от менеджера группы функциональных потоков управления заданному потоку управления приложения значение идентификатора потока управленияpthread_id извлекается из массива *gfp_thread_pool по тегу потока управления (извлекается из полученного от агента сообщения, см. «Архитектура программы с контролируемым выполнением в среде операционной системе реального времени»), а затем потоку управления созначениемpthread_idпосылается сигнал (SIGRT_EXIT,SIGRT_FREEZE и SIGRT_UNFREEZE, которым в ОС РВ [6]соответствуют SIGRTMIN+1, SIGRTMIN+2 и SIGRTMIN+3).

Отметим, что библиотека мониторинга автоматически маскирует указанные сигналы при порождении потока управления (UNBLOCK), а также при выполнении прерываемых сигналами системных вызовов (BLOCK), таких как mq_receive() или mq_send().

Доступ к очереди сообщений для передачи данных менеджеру группы функциональных потоков управления монитор получает с помощью системного вызова mq_open(), в который в качестве аргумента передается предопределенное имя, ассоциированное с заданной очередью сообщений: /gfp%d_monitor_in, где %d – идентификатор группы функциональных потоков управления, который извлекается из полученного от агента сообщения.

## 4.4 Команды управления

Как было отмечено ранее в библиотекемониторинга используются следующие команды управления:

- *Продолжить выполнение потока управления прикладной программы*. После приема сообщения с данной командой *менеджер* группы функциональных потоков управления сбрасывает атрибут is_frozen заданного потока управления в 0.
- *Завершить поток управления прикладной программы*. После приема сигнала потокуправления выполняет системный вызов pthread_exit(), в результате выполнения которого поток управления завершается.
- *Приостановить выполнение потока управления (в том числе и на предопределенный таймаут)*. После приема сигнала поток управления устанавливает атрибут is_frozen в 1 и переходит в ожидание сброса этого атрибута в 0.

Реализация команды перезапуска потокаможет быть выполненаследующими способами (предпочтительным к использованию является перезапуск потока управления с использованием функций свертывания, тем не менее выбор конкретного способа перезапуска потока управления остаётся за программистом):

- *Грубый перезапуск потока управления*. После приема сигнала поток управления выполняет системный вызов pthread_exit(). Далее*менеджер* группы функциональных потоков управления фиксирует завершение заданного потока управления и порождает его повторно с помощью вызова pthread_create().При этом не гарантируется, что вновь созданный поток управления будет работоспособным, так как завершение было принудительным и не выполнялось освобождение ресурсов (например, файловых дескрипторов), связанныхс потоком управления.
- *Перезапуск потока управления с использованием функций свертывания*. В библиотеке мониторинга вводится дополнительный атрибут is_restartable. Потоки управления с установленным при конфигурировании флагом is_restartable должны быть снабжены функциями свертывания. Реализация функций свертывания лежит на прикладном программисте. Библиотека мониторинга при запуске такого потока управления выполнит регистрацию пользовательских функций свертывания с помощью системных вызовов pthread_cleanup_push()ипthread_cleanup_pop(). Таким образом, при принудительном завершении заданного потока управления, ресурсы будут освобождены функциями свертывания.

## 4.5 Структуры данныхбиблиотеки мониторинга

В оставшейся части раздела 4 приведены некоторые реализационные аспекты библиотеки мониторинга, уточняющие характер взаимодействия библиотеки мониторинга с приложением.

В библиотеке мониторинга для описания, адресации и идентификации запущенных потоков управления используются следующие структуры данных:

- gfp_thread_pool – содержит описание дополнительных атрибутов потока управления (атрибуты, назначаемые и используемые библиотекой мониторинга, структура инициализируется библиотекой при запуске потоков управления);
- usergfp – содержит описание прикладной группы функциональных потоков управления (поля структуры должны быть проинициализированы программистом при конфигурировании прикладной программы);
- userthreads– содержит описание потоков управления, входящих в прикладную группу функциональных потоков управления (поля структуры должны быть проинициализированы программистом при конфигурировании прикладной программы).

В состав структуры usergfpвходят следующие поля:

- name– имя группы (тип char *);
- id– идентификатор группы (тип int);
- size– количество потоков управления, входящих в группу (тип int);
- threads– указатель на массив описаний потоков управления группы (тип userthreads*).

В состав структуры userthreadsвходят следующие поля:

- name– имя потока управления (тип char *);
- thrd_func– указатель на головную функцию потока управления (тип int *);
- _tpoolptr– указатель на структуру с описанием атрибутов потока управления (тип gfp_thread_pool*, при конфигурировании поле должно быть проинициализировано прикладным программистом в значение NULL).

## 4.6 Программная реализация агента

Прототип функции агента библиотеки мониторинга:

int checkpoint_agent(struct gfp_thread_pool *tpool, int usercheckpoint_id, int userdata);

Функции передаются следующие аргументы:

- tpool – указатель на структуру с атрибутами потока управления, в контексте которого выполняется агент;
- usercheckpoint_id – идентификатор агента (номер контрольной точки);
- userdata – данные для передачи в монитор.

Упрощенный алгоритм работы *агента* библиотеки мониторинга представлен на рисунке 3.

```
алг
нач
  если tpool <> NULL и usercheckpoint_id > 0
    то
      сформировать(сообщение, userdata);
      mq_send(сообщение);
  все
кон
```

**Рис. 3.** Алгоритм работы агента библиотеки мониторинга

## 4.7 Программная реализация менеджера группы функциональных потоков управления

Упрощенный алгоритм работы менеджера группы функциональных потоков управления библиотеки мониторинга представлен на рисунке 4.

```
алг
нач
  подсчитать(количество_потоков_управления_в_гфп);
  если количество_потоков_управления_в_гфп > 0
    то
      если mq_open("/монитор->менеджер") > 0
        то
          нц
            если mq_receive(сообщение) > 0
              то
                декодировать(сообщение);
                pthread_kill();
            все
          кц
      все
  все
кон
```

**Рис. 4.** Алгоритм работы менеджера группы функциональных программ

## 4.8 Программная реализация главного потока управления монитора

Упрощенный алгоритм работы главного потокауправления монитора представлен на рисунке 5.

```
алг
нач
  если mq_open("/агент->монитор") > 0
    то
      нц
        если mq_receive(сообщение) > 0
          то
            декодировать(сообщение);
            определить(действие);
            если mq_open("/монитор->менеджер") > 0
              то
                сформировать(сообщение, действие);
                mq_send(сообщение);
                mq_close("/монитор->менеджер");
            все
        все
      кц
  все
кон
```

**Рис. 5.** Алгоритм работы главного потока управления монитора

## 5 Результаты тестирования библиотеки мониторинга

Для тестирования библиотеки мониторинга была использована контрольная задача, состоящая из пяти групп функциональных потоков управления, каждая группа включаетпять потоков управления, и в каждом потоке управления было установлено пять контрольных точек (агентов мониторинга).

Каждый из потоков управления выполнил 1 млрд. итераций. При выполнении потоков управления монитор реагировал на искусственно порождаемые в потоках управления сбои и направлял в эти потоки управлениякоманды, соответствующие сбоям (см. «Команды управления»).

## 6 Выводы

В статье приведена типовая архитектура программ с контролируемым выполнением, функционирующих в среде операционной системы реального времени. Конкретно, в качестве операционной системы использовалась операционная система ОС РВ Багет 2.6. Возможно использование других версий операционных систем семейства ОС РВ Багет 2.0. Цель применения контролируемого выполнения – обеспечение устойчивости выполнения программы, достижения поставленной перед программой миссии, независимо от условий эксплуатации, а также от возможных ошибок при проектировании и разработке

программы. Устойчивость выполнения достигается путём встраивания в программу средств анализа состояния программы в процессе её выполнения, выработке управляющих воздействий на алгоритм работы программы при наличии признаков некорректного поведения и реализации управляющих воздействий.

Рассматриваются программы с прикладной многопоточностью, то есть такие программы, в которых различные прикладные функции распределены по различным потокам управления.

Распределение функциональности приложения по потокам управления позволяет осуществить явную проекцию функциональных требований на структуру программы, поскольку потокам управления соответствуют явно локализованные в программе фрагменты кода. Кроме того, локализация прикладной функциональности по потокам управления позволяет управлять функционированием прикладной программы «на уровне потоков управления». Такое управление может быть реализовано с использованием примитивов операционной системы.

В статье приведена типовая архитектурапрограммы, в соответствии с которой контролируемое выполнение осуществляется с использованием примитивовоперационной системы реального времени, а также некоторые реализационные особенности. Также коротко описаны инструментальные средства (библиотека мониторинга), предназначенные для разработки многопоточных программ со средствами контролируемого выполнения.

Дальнейшая работа связана с анализом типовых схем программ реального времени, учитывающих как особенности аппаратуры, так и приложений, и разработке средств контролируемого выполнения для таких программ.

# Realization of controlled executionprinciplefor realtime applications

**A.Gryuntal, K. Narkhov, A. Shchegolkov**

**Annotation:**Controlled execution is a paradigm that aimed at ensuring the program mission fulfillmentin adverse environment (program mistakes, hardware failures, incorrect input, so on). In this paper we consider some control execution implementation methods for multithread real-time applications. We introduce a concept of a monitor that evaluates program execution characteristics and generates commands for managingthreads execution. Some low level realization features are presented.

**Keywords:** controlled execution, multithread program, real-time applications, monitor library, queues, signals.

## Литература

1. В.А. Галатенко. Контролируемое выполнение / В.А. Галатенко, К.А. Костюхин, К.А., Н.В. Шмырев – М: НИИСИ РАН, 2012. – 157 с.
2. В.Б. Бетелин. Контролируемое выполнение с явной моделью / В.Б. Бетелин, В.А. Галатенко, К.А. Костюхин – ПРОГРАММИРОВАНИЕ, 2014, N- 6, с. 45-55.
3. В.А.Галатенко, Н.И.Вьюкова, С.В.Самборский, С.И.Трифонов. Программирование в среде VxWorks. - М.: Наука, 1997.

4. В.Л.Безруков, А.Н.Годунов, П.Е.Назаров, В.А.Солдатов, И.И.Хоменков. Введение в ос2000, Вопросы кибернетики. Информационная безопасность, Операционные системы реального времени, Базы данных/Под ред. чл.-корр. РАН В.Б.Бетелина. - Москва; НИИСИ РАН, 1999, –с. 76-106.
5. А.Н.Годунов, В.А.Солдатов. Особенности отладки встроенных систем – Программная инженерия. 2013, №3, с. 2-8
6. А.Н.Годунов, В.А.Солдатов. Операционные системы семейства Багет (сходство, отличия, перспективы) – Программирование, 2014, №5, с. 68-76
7. DavidR. Butenhof. ProgrammingwithPOSIXThreads, Addison-Wesley, 1997
8. International Standard ISO/IEC 9945-1 (ANSI/IEEE Std 1003.1) Second Edition. 1996-07-12. Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language].
9. КТ-178. Квалификационные требования. Требования к программному обеспечению бортовой аппаратуры и систем сертификации авиационной техники. – Межгосударственный авиационный комитет, Авиационный регистр, 1996.
10. А.Л. Фуксман. Технологические аспекты создания программных систем / А.Л. Фуксман. – М.: Статистика, 1979 – 184 с.
11. ARINC Specification 653P1-3: Avionics Application Software Standard Interface Part 1 - Required Services. Aeronautical Radio INC, Maryland, USA, 2010.

# Инструментальное программное обеспечение для подготовки запуска задач в мультипроцессорных комплексах реального времени

**Т.К. Грингауз**[1], **А.Н. Онин**

*1 – кандидат технических наук*

**Аннотация:** Описываются назначение и функциональность программного изделия «Утилиты поддержки запуска программ в среде RapidIO» (УПЗ-РИО), разработанного в НИИСИ РАН. Программное изделие обеспечивает автоматизацию выполнения на инструментальной ЭВМ следующих групп процедур: поддержка статической инициализации среды RapidIO, оптимизация распределения вычислительных задач по процессорам, создание сценария программы ПЗУ для загрузки программного обеспечения в целевую аппаратуру комплекса. Автоматизация основана на использовании формализованного описания аппаратуры комплекса и потоков данных.

**Ключевые слова:** коммуникационная среда RapidIO, статическая инициализация, оптимизация, распределение задач по процессорам, сценарий программы ПЗУ

## 1. Описание аппаратно-программной платформы. Основные понятия и определения

Ниже будут рассматриваться мультипроцессорные цифровые вычислительные комплексы с коммуникационной средой стандарта RapidIO[1] (далее - ЦВК), создаваемые на основе аппаратных средств и средств общего программного обеспечения (далее - ОПО), разрабатываемых в НИИСИ РАН. Для автоматизации подготовки запуска программ в таких комплексах разработано программное изделие «Утилиты поддержки запуска программ в среде RapidIO» (ПИ УПЗ-РИО). Ниже приведены сведения об аппаратных средствах, о средствах ОПО, о технологии разработки, загрузки и запуска функционального программного обеспечения (далее - ФПО) в объеме, достаточном для понимания назначения и функциональности УПЗ РИО.

### 1.1 Аппаратные средства. Операционная система. Кроссплатформенная разработка ФПО

ФПО ЦВК представляет собой совокупность программ, параллельно исполняющихся на вычислительных узлах комплекса и обменивающихся между собой потоками данных в среде RapidIO.

В качестве вычислительных узлов используются микропроцессоры КОМДИВ64-РИО (СБИС 1890ВМ6Я, [2]), КОМДИВ128-РИО (СБИС 1890ВМ7Я, [3]) (далее - «процессоры»). Перечисленные СБИС входят в состав процессорных модулей серий ЦП-РИО-64, ЦП-РИО-128, а также мезонинных модулей М-РИО, М128-РИО. В состав процессорного модуля могут входить один или два процессора, в состав мезонинного модуля входит один процессор. На один процессорный модуль можно установить не более одного процессорного мезонина. Коммуникации в среде RapidIO обеспечиваются посредством коммутаторов (СБИС 1890КП3Я) и гибридных устройств [1] (СБИС 1890ВГ18Я, 1890ВМ6Я), входящих в состав процессорных модулей, а также в модуль маршрутизатора МР-РИО. Поддерживаются протоколы RapidIO 1x/4x Serial (далее – SRIO), RapidIO 8 LP-LVDS (далее-PRIO) [1].

Примечание. СБИС 1890ВГ18Я может использоваться в двух режимах: в режиме коммутатора RapidIO, обеспечивающего преобразование форматов SRIO <-> PRIO, и в режиме контроллера ВСК [6], обеспечивающего коммутацию потоков высокоскоростного последовательного канала [6] и потока PRIO. В последнем случае СБИС 1890ВГ18Я играет роль источника или приемника потока данных в среде RapidIO.

На основе перечисленных модулей создаются ЭВМ (далее - «приборы»). Прибор – это объединительная плата с установленными в ее позициях модулями. Соединения портов коммутаторов RapidIO в рамках одного прибора определяются архитектурой модулей и разводкой объединительной платы. Возможны также соединения модулей прибора кабелями SRIO посредством интерфейса СБИС 1890ВГ18Я, 1890ВМ6Я. Наплатные процессоры взаимодействуют между собой и с процессором мезонинного модуля посредством коммутируемых

каналов по протоколу PRIO. Взаимодействие на межмодульном уровне в рамках одного прибора осуществляется по протоколу PRIO или по протоколу SRIO. Приборы объединяются в комплекс посредством соединений по протоколу SRIO.

Процессоры функционируют под управлением операционной системы ОС РВ Багет 3.3, принадлежащей семейству операционных систем реального времени ОС РВ Багет [5] (далее - ОС РВ). На каждый процессор загружается образ операционной системы с включенными в него прикладными программами [5]. Разработка и отладка программ, предназначенных для функционирования в среде ОС РВ, производится с использованием инструментальной Linux-ЭВМ (далее - ИЭВМ). На ИЭВМ программы на исходном языке программирования (язык Си) подвергаются кросс-компиляции под целевую платформу (КОМДИВ 64/128-РИО) и включаются в образ ОС РВ для соответствующей платформы.

Загрузка и запуск образов ОС РВ на микропроцессорах выполняется при старте приборов. По включении питания на каждом микропроцессоре запускается программа ПЗУ, которая, в зависимости от настроек, может ожидать внешних команд или передавать управление «пользовательскому сценарию». Пользовательский сценарий - это программа на интерпретируемом языке команд PSL (PMON2000 Script Language), выполняемая программой ПЗУ и предназначенная для управления стартом прибора. Пользовательский сценарий не входит в состав ОПО, он составляется с учетом требований конечного изделия разработчиком прикладной задачи.

Команды языка PSL позволяют осуществлять загрузку программ и данных в память модуля из РПЗУ модуля, из файловой системы семейства FAT, а также по интерфейсам RS-232, Ethernet (протокол tftp), RapidIO. Загруженная программа может быть записана в РПЗУ модуля, на жесткий диск с файловой системой FAT (мезонинный модуль электронного диска БТМРС4-301) или запущена на исполнение. Загрузку осуществляет «ведущий процессор». В качестве «ведущего» может быть выбран процессор, установленный на модуле, соединенном по Ethernet с ИЭВМ комплекса. В период отладки ФПО образы ОС РВ загружаются по Ethernet с ИЭВМ в память ведущего процессора, а затем по RapidIO - в остальные процессоры.

Для загрузки приборов, а также для осуществления межпроцессорных обменов во время исполнения программ необходимо, чтобы среда RapidIO была инициализирована. Это означает, что оконечным и гибридным устройствам [1] должны быть присвоены идентификаторы (далее - «РИО-идентификаторы»), а в коммутаторы должны быть записаны таблицы маршрутизации, определяющие маршруты передачи данных между устройствами [1]. Пользовательский сценарий ведущего процессора должен выполнять инициализацию RapidIO перед загрузкой программного обеспечения.

## 1.2 Средства программной поддержки межпроцессорных обменов данными

При описании межпроцессорного обмена будем использовать понятие «поток данных» (далее - «поток»). Под потоком будем подразумевать цикл передач фиксированного объема данных за равные промежутки времени от источника приемнику. В ЦВК источником или приемником потока может быть процессор или СБИС 1890ВГ18Я, используемая в режиме контроллера ВСК (далее процессоры и контроллеры ВСК, передающие или принимающие данные по RaidIO, будем называть «участниками обмена»). Поток можно характеризовать количественно средней скоростью передачи данных или парой {частота передачи, объем передаваемых данных за итерацию}.

В ОС РВ в качестве интерфейсов межпроцессорного взаимодействия поддерживаются каналы стандарта ARINC [5], а также специфические каналы передачи сообщений «точка-точка» mp (message passing). Кроме того, в состав ОПО входит специализированная «Библиотека параллельной обработки сигналов для микропроцессоров КОМДИВ64-РИО и КОМДИВ128-РИО» (БПОС), ЮКСУ.91003-01 [7], обеспечивающая более технологичный способ организации межпроцессорных обменов.

Задача, реализованная в архитектуре БПОС (далее - «задача БПОС»), представляется в виде последовательности стадий вычислительного конвейера, циклически обрабатывающего данные, поступающие на его вход с фиксированной частотой. Вычислительные стадии допускают распараллеливание: код стадии может исполняться на нескольких процессорах. Процессоры системы разбиваются на однородные группы: на всех процессорах группы выполняется один и тот же набор стадий (далее - «группа стадий»). Количество процессоров в разных группах может отличаться. В такой системе каждому процессору соответствует «логический номер» - сквозной номер в упорядоченном множестве {{<номер группы>, <номер процессора в группе>}}. В формализме БПОС «поток» определяется как объект, описывающий однонаправленную передачу трехмерных массивов фиксированной размерности от стадии-источника к стадии-приемнику на каждой итерации цикла с разбиением массива на источнике или на приемнике по указанной пользователем размерности. Описание потока в БПОС определяет множество параллельно исполняемых передач «точка-точка» на каждой итерации цикла, т.е. множество потоков между парами процессоров. Во избежание терминологической путаницы, там, где это необходимо, будем пользоваться словосочетанием «поток БПОС». С потоком БПОС

связываются библиотечные функции чтения данных из потока и отправки данных в поток. Функции реализованы с использованием библиотеки mp, но вызовы функций mp скрыты интерфейсом БПОС.

Средства ОС РВ обеспечивают соответствие логических номеров и РИО-идентификаторов процессоров при инициализации каналов mp посредством функции mp_init(), входящей в состав ОС РВ. На вход функции должен передаваться массив РИО-идентификаторов, упорядоченных в соответствии с логическими номерами процессоров (далее - «массив RIO_MAP»). Функция возвращает РИО-идентификатор и логический номер локального процессора.

Обмен данными между процессорами и контроллерами ВСК СБИС 1890ВГ18Я обеспечивается средствами «Библиотеки ВСК», ЮКСУ.90948 01 [8]. Потоки данных между процессорами и контроллерами ВСК не описываются в терминах БПОС и должны программироваться явно.

## 1.3 Особенности организации межпроцессорных обменов данными в среде RapidIO. Цель разработки УПЗ-РИО

В среде RapidIO оконечные устройства идентифицируются посредством РИО-идентификаторов. Соответствие РИО-идентификаторов конкретным процессорным элементам устанавливает статическая инициализация RapidIO. Именно по РИО-идентификаторам идентифицируют участников обмена механизмы межпроцессорного взаимодействия, реализованные в ОС РВ и в «Библиотеке ВСК». Если схема соединений оконечных устройств и коммутаторов (топология) среды RapidIO неизменна, а инициализация выполняется одной и той же процедурой, то числовое значение РИО-идентификатора однозначно определяет местоположение устройства в сети.

Разработчики прикладных программ идентифицируют участников обмена семантически, например, посредством символических имен или посредством логического номера процессора в технологии БПОС. «Семантический» идентификатор определяет функциональность участника обмена и не привязан к его топологическому расположению.

Для запуска программы на целевой аппаратуре необходимо установить соответствие символических имен числовым значениям РИО-идентификаторов. Соответствие устанавливается в тексте программы (например, посредством заголовочного файла с макроопределениями).

Для установления соответствия необходима информация о распределении РИО-идентификаторов по оконечным устройствам и о распределении задач по процессорам. Соответствие устанавливается через топологическую привязку символических имен и числовых значений к физическим процессорам. В результате установления соответствия числовое значение РИО-идентификатора несет семантическую нагрузку (определяет функциональность задачи). Семантика РИО-идентификаторов должна учитываться при разработке процедуры загрузки приборов комплекса.

При распределении задач по процессорам должны учитываться следующие факторы:

- тип процессора (например, задачи, предназначенные для КОМДИВ281-РИО, не всегда можно исполнить на КОМДИВ64-РИО),
- эффективность межпроцессорного обмена.

В среде RapidIO тракт данных, передаваемых от источника приемнику, определяется не только топологической привязкой участников обмена, но и тем, какая схема маршрутизации установлена при инициализации системы [1]. Эффективность обменов при фиксированной схеме маршрутизации может существенно различаться в зависимости от распределения участников обмена по оконечным устройствам. Данные потока передаются пакетами, проходящими последовательность коммутаторов в соответствии с предустановленным маршрутом. Если данные нескольких потоков проходят через один порт коммутатора, возможна их буферизация. Задержка пакетов в буферах снижает темп передачи данных и может привести к снижению производительности или даже к неработоспособности вычислительного комплекса в целом.

Если первоначально установленное распределение задач не обеспечивает требуемой производительности, то целесообразен поиск лучшего варианта. Смена распределения задач сопряжена с изменением соответствия символических имен и числовых значений РИО-идентификаторов, и, как следствие, может привести к необходимости изменения процедуры загрузки программ в приборы.

Таким образом, разработка, отладка и оптимизация программного обеспечения требуют многократного выполнения весьма нетехнологичных операций, связанных с обращением к числовым значениям РИО-идентификаторов, устанавливаемым при инициализации RapidIO. Целью разработки УПЗ-РИО была автоматизация цикла подготовки разработанных программ к запуску на целевой аппаратуре путем увязки в единую технологическую цепочку процедур подготовки данных для статической инициализации, распределения задач по процессорам, составления файлов описания РИО-идентификаторов, оптимизации распределения задач, создания пользовательского сценария. Автоматизация подготовки запуска задач посредством УПЗ-РИО позволяет сделать числовые значения РИО-идентификаторов прозрачными для разработчиков программного обеспечения. Радикальное сокращение времени подготовки запуска задачи за счет автоматизации облегчает отладку и оптимизацию

программ, а также их адаптацию к различным конфигурациям аппаратуры. Последнее свойство особенно важно в условиях опережающей разработки программного обеспечения, при работе на недоукомплектованном оборудовании.

# 2. Общая характеристика УПЗ-РИО

## 2.1 Принцип функционирования УПЗ-РИО

Программная реализация УПЗ РИО - программа rio-stat (далее - «утилита») - функционирует на ИЭВМ под управлением ОС Fedora NIISI 16x86_64.

Утилита автоматизирует совокупность процедур, выполняемых на ИЭВМ в следующих целях:

- поддержка статической инициализации среды RapidIO;
- оптимизация распределения задач по процессорам;
- создание пользовательских сценариев.

Выбор процедуры и ее входных данных определяется опциями командной строки запуска утилиты.

Автоматизация подготовки запуска программ осуществляется на основе обработки формализованного описания системы, составляемого пользователем в виде совокупности конфигурационных файлов.

Совокупность конфигурационных файлов составляет интерфейс для ввода следующей информации:

- формализованное описание аппаратной конфигурации комплекса;
- символические имена оконечных и гибридных устройств в составе комплекса;
- первоначальное распределение задач по процессорам (для последующей оптимизации этого распределения);
- состав и количественные характеристики потоков данных в системе;
- состав и параметры процедур пользовательского сценария.

Для описания перечисленных видов информации разработаны специфичные для УПЗ-РИО форматы. Кроме того, предусмотрена интеграция технологии УПЗ-РИО с технологией БПОС, позволяющая извлекать символические имена и информацию о потоках данных из текста программы. В программе, реализованной по архитектурным правилам БПОС, информация о стадиях, группах процессоров, потоках данных содержится в специфическом файле в формате языка программирования Си (далее - «файл конфигурации задачи»). Файл конфигурации задачи БПОС может включаться в состав входных данных УПЗ-РИО и обрабатываться как процедурами поддержки статической инициализации, так и процедурами оптимизации распределения задач по процессорам. В этом случае должна поддерживаться специальная дисциплина символического именования РИО-идентификаторов, в соответствии с которой символическое имя однозначно определяется именем группы и номером процессора в группе. Такой способ использования УПЗ-РИО будем называть режимом интеграции с БПОС.

Опционально в качестве входных данных могут вводиться значения или диапазоны значений РИО-идентификаторов в приборах комплекса, а также описание специфицированных пользователем маршрутов RapidIO (далее - «индивидуальные маршруты»).

Подготовка задачи к запуску включает следующие этапы:

1) составление формализованного описания аппаратной конфигурации комплекса (далее - «файл аппаратной конфигурации»),

2) указание в файле аппаратной конфигурации первоначального распределения задач по процессорам с использованием символических имен РИО-идентификаторов,

3) генерация сценария статической инициализации среды RapidIO,

4) оптимизация первоначального распределения задач по процессорам с учетом загруженности каналов RapidIO на основе использования формализованного описания потоков данных,

5) генерация файлов с определениями символических имён РИО-идентификаторов в форматах языков Си и PSL в соответствии с оптимизированным распределением задач по процессорам,

6) компиляция программ с использованием файла макроопределений символических имен РИО-идентификаторов в формате языка Си и сборка образов операционной системы,

7) составление формализованного описания пользовательского сценария,

8) генерация пользовательского сценария на основе его формализованного описания (полученного на этапе 7) с использованием сценария статической инициализации среды RapidIO (полученного на этапе 3) и файла с определениями символических имён РИО-идентификаторов в формате PSL (полученного на этапе 4).

Этапы 1 и 7 выполняются вручную (с использованием текстового редактора). Этап 2 может быть выполнен вручную, но при интеграции с технологией БПОС возможно его выполнение в автоматическом режиме. Выполнение этапов 3,4,5,8 автоматизировано.

Таким образом, при заранее составленных конфигурационных файлах подготовка задачи к запуску заключается в последовательности вызовов утилиты rio-stat и однократной сборке образов ОС РВ. Все упомянутые вызовы можно свести к однократному вызову утилиты make за счет разработки соответствующего make-файла. При изменениях задачи или состава аппаратуры изменению подлежат только конфигурационные файлы. Если задача полностью описывается формализмом БПОС, то конфигурированию подлежат три файла:

- файл аппаратной конфигурации,
- файл конфигурации задачи,

- файл описания пользовательского сценария.

## 2.2 Формализованное описание аппаратной конфигурации

Аппаратная конфигурация описывается в специфическом текстовом формате (далее - «формат УПЗ-РИО»). Формат обеспечивает описание комплекса в виде совокупности именованных приборов, соединенных связями по протоколу SRIO. Прибор описывается как совокупность сборочных единиц с привязкой к позициям на объединительной плате. В описании прибора указываются тип объединительной платы, типы модуля и установленных на нем мезонинов в каждой позиции объединительной платы. Связи приборов по SRIO описываются посредством указания именованных разъемов соединяемых модулей. Такое описание полностью определяет топологию среды RapidIO, поскольку топологическая структура каждого типа сборочных единиц предопределена.

Формат обеспечивает возможность опционально указывать РИО-идентификаторы оконечных и гибридных устройств в составе модуля в виде числового значения или символического имени. УПЗ-РИО предусматривает возможность выделять в составе комплекса группы приборов и/или модулей, а также указывать опционально включаемые подсистемы.

Представление о формате УПЗ-РИО дает пример, представленный на рисунке 1.

Примечание. В состав прибора могут входить непроцессорные модули и мезонины, не поддерживающие интерфейс RapidIO. Такие модули могут указываться в описании аппаратной конфигурации, но утилита игнорирует их при формировании топологии среды RapidIO. В примере, приведенном на рис.1, помимо модулей с интерфейсом RapidIO указаны модули МП-ВПО, применяемые только при приеме/передаче данных в формате ВСК [6].

Пример описывает комплекс из двух приборов (1-ЦОИ, 3-ЦОИ), собранных на основе объединительной платы ОП-РИО-А. В приборе 1-ЦОИ установлены следующие модули с интерфейсом RapidIO: МР-РИО (в позицияхА02, А12 ), ЦП-РИО-64Г (в позицияхА03, А11, ЦП-РИО-128 с мезонином М-К128 (в позицияхА04, А05, А06, А08, А09, А10). Кабелями SRIO соединены две пары разъемов: А02:Х6 и А12:Х6, А06:Х6 и А08:Х6. В приборе 3-ЦОИ установлены модули МП-ВПО (в позициях А02, А04, А06), а также следующие модули с интерфейсом RapidIO: МР-РИО (в позициях А08 - А11), ЦП-РИО-64Г (в позиции А12). Кабелями SRIO соединены разъемы А06:Х6 и А10:Х9. Между двумя приборами установлено три соединения SRIO. В данном примере процессор модуля ЦП-РИО-64Г, установленного в позиции А03 прибора 1-ЦОИ, назначен «ведущим»: с него будет производиться инициализация RapidIO. Указаны символические имена для РИО-идентификаторов процессоров КОМДИВ128-РИО в приборе 1-ЦОИ, а также для РИО-идентификаторов СБИС 1890ВГ18Я в приборе 3-ЦОИ.

```
*О ОР-RIO-А 1-ЦОИ¶
+ А02: MR-RIO¶
+ А03: CP-RIO-64G ..... !master1¶
+ А04: CP-RIO-128+M2:M-K128 imiter_3 ·· imiter_4 ·· imiter_5¶
+ А05: CP-RIO-128+M2:M-K128¶
+ А06: CP-RIO-128+M2:M-K128 receiver_0 receiver_1 state_0¶
+ А08: CP-RIO-128+M2:M-K128 imiter_0 ·· imiter_1 ·· imiter_2¶
+ А09: CP-RIO-128+M2:M-K128¶
+ А10: CP-RIO-128+M2:M-K128 imiter_6¶
+ А11: CP-RIO-64G¶
+ А12: MR-RIO¶
* {*}-А02:Х6 {*}-А12:Х6¶
* {*}-А06:Х6 {*}-А08:Х6¶
¶
*С ОР-RIO-А 3-ЦОИ¶
+ А02: MP-VPO¶
+ А04: MP-VPO¶
+ А06: MP-VPO¶
+ А08: MR-RIO → → → VG0:VSK_4 :VG3:VSK_3¶
+ А09: MR-RIO → → → VG0:VSK_2 :VG3:VSK_1¶
+ А10: MR-RIO → → → → → VG3:VSK_6¶
+ А11: MR-RIO → → → VG0:VSK_5 ¶
+ А12: CP-RIO-64G¶
* {*}-А06:Х6 {*}-А10:Х9 ·· # к МП-ВПО¶
¶
* {1-ЦОИ}-А02:Х9 {3-ЦОИ}-А09:Х8¶
* {1-ЦОИ}-А09:Х6 {3-ЦОИ}-А10:Х6¶
* {1-ЦОИ}-А12:Х7 {3-ЦОИ}-А08:Х8¶
```

**Рис. 1.** Пример файла аппаратной конфигурации

## 2.3 Процедуры поддержки статической инициализации

### 2.3.1 Автоматизация построения файлов для статической инициализации RapidIO

Группа процедур поддержки статической инициализации позволяет на основе файла аппаратной конфигурации комплекса рассчитать на ИЭВМ данные для статической инициализации среды RapidIO и сгенерировать файл, содержащий последовательность команд программы ПЗУ для инициализации среды RapidIO в соответствии с рассчитанными данными. Под данными для статической инициализации среды RapidIO подразумеваются распределение РИО-идентификаторов оконечных и гибридных устройств и таблицы маршрутизации коммутаторов.

По умолчанию создаются маршруты между каждой парой оконечных/гибридных устройств RapidIO по кратчайшему пути. Поддерживается возможность включить в схему маршрутизации маршруты, специфицированные пользователем (далее - «индивидуальные маршруты»).

Не любая система маршрутов в среде RapidIO корректна. Реализация маршрутов RapidIO посредством таблиц, прописываемых в коммутаторы [1], накладывает следующее ограничение: два маршрута от разных источников к одному приемнику, проходящие через один и тот же порт коммутатора, должны совпадать на участке от коммутатора до приемника. При включении в схему маршрутизации индивидуального маршрута утилита поддерживает корректность схемы

маршрутизации путем корректировки маршрутов, проложенных по умолчанию. Не допускается корректировка ранее включенных индивидуальных маршрутов. В случае возникновений коллизий УПЗ-РИО выдает предупреждение о невозможности включения индивидуального маршрута в схему маршрутизации.

Для включения индивидуальных маршрутов в схему маршрутизации необходимо описать их во входном файле формата «path», специфицирующего РИО-идентификаторы источника и приемника, а также последовательность выходных портов коммутаторов. Построенные утилитой маршруты можно сохранить в файле формата path или экспортировать в формат входных данных «Компилятора высокоуровневого описания системы» (КВОС) [9]. Кроме того, утилита позволяет отобразить маршрут между парой устройств на графическом представлении топологии RapidIO (графическое представление описывается в разделе 2.3.2 настоящей статьи).

Для осуществления статической инициализации RapidIO средствами программы ПЗУ УПЗ-РИО формирует командный файл в формате «BMRW». Файлы такого формата могут быть выполнены командой riobmrw программы ПЗУ[10].

## 2.3.2 Графическое представление топологии RapidIO

Топологию среды RapidIO и маршруты, созданные посредством УПЗ-РИО, можно визуализировать. С этой целью предусмотрена опция для формирования файла в формате «DOT» с данными для графического представления. Графическое представление позволяет пользователю обозреть схему соединений оконечных устройств и коммутаторов топологии среды RapidIO с обозначением приборов, модулей, СБИС, соединений кабелями SRIO, РИО-идентификаторов, загруженности каналов RapidIO, IP-адресов процессоров. Пример графического представления топологии среды RapidIO приведен на рисунке 2.

## 2.3.3 Использование символических имен РИО-идентификаторов

Процедуры поддержки статической инициализации создают два файла:

- файл с макроопределениями символических имен РИО-идентификаторов в формате языка Си, предназначенный для использования в прикладных программах;

- файл с определениями символических имен РИО-идентификаторов в формате описания переменных окружения программы ПЗУ на языке PSL, предназначенный для использования в сценариях программы ПЗУ.

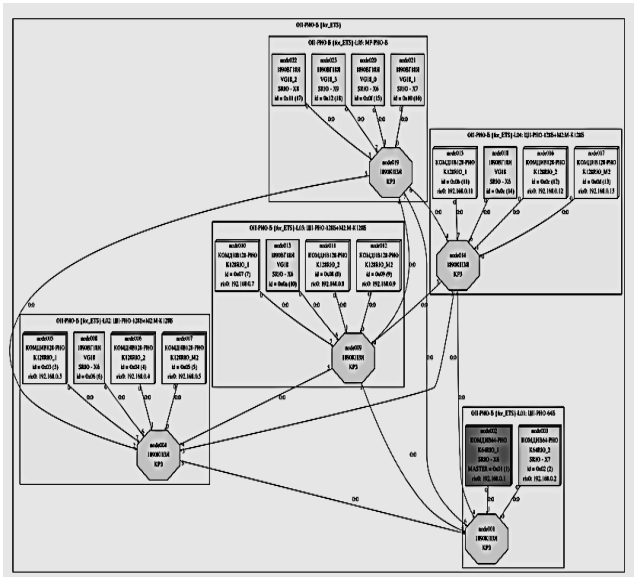


**Рис. 2.** Пример фрагмента графического представления топологии RapidIO в формате «DOT»

В режиме интеграции с БПОС, когда на вход утилиты подается файл конфигурации задачи, утилита автоматически создает массив RIO_MAP и включает его в файл с макроопределениями символических имен РИО-идентификаторов в формате языка Си.

Перечисленные файлы определяют соответствие символических имен числовым значениям РИО-идентификаторов.

Утилита устанавливает это соответствие в результате двух подстановок: 1) соответствие между символическим именем процессора и его физическим расположением, 2) соответствие между числовым значением РИО-идентификатора и физическим расположением процессора.

Соответствие между символическим именем процессора и его физическим расположением (т..е. распределение задач по процессорам) устанавливается одним из следующих способов:

- назначается пользователем и записывается в файле аппаратной конфигурации,

- автоматически составляется утилитой на основе файла конфигурации задачи,

-автоматически устанавливается утилитой в результате оптимизации начального распределения.

Соответствие между числовыми значениями РИО-идентификаторов и физическим расположением процессоров устанавливает статическая инициализация RapidIO. Утилита rio-stat создаёт скрипт формата BMRW для статической инициализации коммуникационной среды RapidIO в соответствии с заданной аппаратной конфигурацией. Таким образом, автоматически поддерживается соответствие РИО-идентификаторов реальному распределению РИО-идентификаторов в приборе.

Утилита создаёт два сценария программы ПЗУ:

- сценарий определений символических имён;

- «пользовательский сценарий».

Вызов первого сценария из других сценариев на языке PSL обеспечивает возможность использования символических имён процессорных элементов вместо числовых значений РИО-идентификаторов. Второй сценарий обеспечивает загрузку и запуск прикладных задач. В результате пользователь избавляется от необходимости обращения к числовым значениям РИО-идентификаторов и может оперировать их символическими именами не только на этапе программирования, но и на всех этапах подготовки задачи к запуску. Единство обозначений РИО-идентификаторов в программах инициализации и в прикладных программах делает привязку функциональности прикладных программ к физическим процессорам инвариантной по отношению к изменениям числовых значений РИО-идентификаторов.

## 3. Процедуры - оптимизации распределения задач по процессорам

В контексте описания процедур оптимизации под «задачей» понимается совокупность процедур приема и обработки данных, выполняемых на процессоре. Одинаковые группы процедур, исполняемые на разных процессорах, считаются разными задачами. При фиксированном распределении РИО-идентификаторов и при фиксированном распределении задач по процессорам символическое имя РИО-идентификатора можно считать символическим именем задачи. Применительно к технологии БПОС, задачей считается совокупность стадий, привязанных к группе процессоров, выполняемых на одном из процессоров группы. В этом случае можно говорить о распределении групп стадий по процессорам. Символические имена РИО-идентификаторов должны быть составлены по следующему шаблону:

<имя группы>_<номер процессора в группе>

где

<имя группы> — название группы, используемое в файле конфигурации задачи БПОС.

Утилита rio-stat предоставляет возможность оптимизировать распределение задач по процессорам с целью уменьшения максимальной нагрузки на физические связи в среде RapidIO. Процедура оптимизации основана на частичном переборе вариантов распределения в соответствии с эвристическим методом. На каждой итерации процедура рассчитывает максимальную нагрузку на физическое соединение на основании информации о маршрутах и о потоках данных в системе. Каждая итерация не ухудшает предыдущий результат. Оптимизированные варианты распределения могут различаться в зависимости от того, какое распределение выбрано в качестве начального.

Начальное распределение задач может быть получено следующими способами:

- явное задание начального распределения пользователем в файле аппаратной конфигурации,
- автоматическое создание начального распределения утилитой в режиме интеграции с БПОС.

Процедура автоматического создания начального распределения может выдавать детерминированный или рандомизированный результат в зависимости от опций командной строки.

Автоматическое распределение задач по процессорам производится с учетом типа процессора. Информация о типе процессора должна быть внесена в описание групп процессоров в файле конфигурации задачи.

Для применения процедур оптимизации необходимо на вход утилиты передать информацию о потоках в системе. При использовании БПОС в описание каждого потока в файле конфигурации задачи необходимо в виде комментария указать количественную характеристику потока. В общем случае потоки данных должны быть описаны в отдельном файле специфического формата «flow».

Оптимизация распределения задач может быть полезна, если начальное распределение не обеспечивает работу ФПО комплекса на заданной частоте. Оптимизация распределения не гарантирует достижения требуемого темпа межпроцессорного обмена, поскольку при расчете максимальной нагрузки учитываются лишь осредненные характеристики потоков. В реальности в некоторые моменты времени нагрузки на физические соединения могут превышать расчетные показатели. Окончательным критерием пригодности того или иного распределения является результат тестирования ФПО на целевой аппаратуре. Если оптимизация распределения задач не привела к желаемому результату, можно попытаться повторить ее при другом начальном распределении (например, использовать опцию рандомизации) или использовать другие подходы (например, изменение схемы маршрутизации за счет построения индивидуальных маршрутов).

## 4. Процедуры генерирования пользовательского сценария

Группа процедур генерирования пользовательского сценария генерирует сценарии на языке PSL для выполнения в среде программы ПЗУ типовых процедур старта прибора, описанного в формате УПЗ-РИО. Функциональность генерируемого сценария и параметры выполняемых им процедур определяются пользователем с помощью специального конфигурационного файла.

Обеспечивается создание сценариев со следующими функциональными возможностями:

- определение текущего режима работы прибора [10],
- запуск тестов по включению [10],

- установка переменных окружения программы ПЗУ на процессорных элементах прибора,
- запуск тестов встроенной системы контроля[10],
- запуск процедуры инициализации коммуникационной среды RapidIO,
-запись файлов из ИЭВМ на модули электронного диска БТМРС4-301 и в РПЗУ процессорных элементов прибора,
- запись файлов, локализованных на модуле электронного диска БТМРС4-301, в РПЗУ процессорных элементов прибора,
- запуск программ из РПЗУ процессорных элементов прибора,
- запись программ из ИЭВМ в ОЗУ процессорных элементов прибора,
- запуск программ из ОЗУ процессорных элементов прибора.

Посредством предлагаемой технологии может быть создан типовой пользовательский сценарий.

# Software start in multiprocessor real-time systems development tool kit

**T.K. Gringauz, A.N.Onin**

**Abstract:** Destination and functionality of development tool kit for software run in RapidIO environment are described. Tool kit provides automation of following instrumental procedure groups: RapidIO: static initialization support, optimization of computing tasks processor distribution, development of embedded script for software loading to target environment. Automation is based on formal descriptions of hardware and data flows.

**Keywords:** RapidIO communication environment, static initialization, optimization, tasks processor distribution, embedded script.

## Литература

1. RapidIO Interconnect Specification (Revision 1.3) Available from: http://www.rapidio.org/specs/current.
2. МИКРОСХЕМА ИНТЕГРАЛЬНАЯ 1890ВМ6Я. Указания по применению, ЮКСУ.431281.102Д4
3. Научно-технический отчёт по ОКР «Разработка базовых узлов для создания субмикронных СБИС для электронных модулей ЭВМ с повышенной устойчивостью к воздействию нейтронов и тепловых режимов эксплуатации бортовой аппаратуры авиационной техники», М., НИИСИ РАН, 2011
4. Научно-технический отчёт по ОКР «Разработка системных контроллеров для высокопроизводительных микропроцессоров, М., НИИСИ РАН, 2011
5. А.Н. Годунов [и др.], Операционная система реального времени Багет 3.0.//Программные продукты и системы. - Тверь, Научно-исследовательский институт «Центрпрограммсистем», 2010, № 4, с. 15 – 19
6. Т.К. Грингауз, А.Н. Онин. Программное обеспечение для приема и передачи данных по высокоскоростному каналу в мультипроцессорных комплексах реального времени.// Труды НИИСИ РАН.- М., НИИСИ РАН, 2014, том 4 №2, с.22-32, ISSN 2225-7349
7. Программное изделие «Библиотека параллельной обработки сигналов для микропроцессоров КОМДИВ64-РИО и КОМДИВ128-РИО» (БПОС), ЮКСУ.91003-01.
8. Программа «Библиотека ВСК», ЮКСУ.90948 01.
9.Программное изделие «Компилятор высокоуровневого описания системы» (КВОС), ЮКСУ.90974-01.
10. Программа ПЗУ. Описание применения, ЮКСУ.90815-01 31 01.

# О некоторых практических вопросах программирования сопроцессора обработки сигналов микропроцессора КОМДИВ128-РИО

**М.С. Аристов, А.С. Осипов[1], А.М. Щегольков**

*1 - кандидат физико-математических наук*

**Аннотация:** В статье рассматривается ряд вопросов, связанных с программированием специализированного сопроцессора обработки сигналов СР2 в составе процессора КОМДИВ128-РИО, разработанного в НИИСИ РАН. Приводятся основные особенности архитектуры КОМДИВ128-РИО и СР2 и возникающие в связи с этим сложности программирования данного сопроцессора. Перечисляются методы повышения производительности разрабатываемых для СР2 программ. Приведены примеры разработки композитных функций СР2, не связанных с алгоритмами быстрого преобразования Фурье и свёртки и используемых для решения конкретных прикладных задач сигнальной обработки.

**Ключевые слова:** встречная оптимизация, микропроцессоры семейства КОМДИВ, К128-РИО, сопроцессор обработки сигналов СР2, БЦОС, композитные функции.

## Введение

В настоящее время одним из перспективных направлений, используемых в разработке микропроцессорной техники, является методика встречной оптимизации (англ. co-design). Одним из вариантов применения данной методики является расширение базовой архитектуры специализированными сопроцессорами, ориентированными на достижение максимальной производительности на заранее определённом классе прикладных задач. В рамках реализации данной методики в НИИСИ РАН разрабатывается семейство высокопроизводительных микропроцессоров КОМДИВ, нашедших своё применение для решения ряда актуальных задач сигнальной обработки, требующих когерентной (согласованной) обработки больших объемов данных. Помимо этого, ряд решений, полученных при разработке данных микропроцессоров (неоднородность архитектуры, организация обмена данными внутри и вне микропроцессора и т. п.), способствует созданию научно-технологического задела для создания отечественных суперкомпьютерных технологий экзафлопсного класса [1].

К микропроцессорам данного семейства следует отнести К64-РИО (1890ВМ6) – суперскалярный микропроцессор 64-разрядной RISC архитектуры MIPS64. Помимо обычного 64-разрядного сопроцессора (СР1) арифметики с плавающей точкой, поддерживающего форматы представления вещественных чисел одинарной и двойной точности, он был дополнен 64-разрядным сопроцессором (СР2) арифметики комплексных чисел, имеющим следующие основные характеристики:

- регистровый файл на 64 64-разрядных регистра с 4 портами записи и 8 портами чтения;
- возможность обмена с памятью 128-разрядными словами;
- возможность обмена данными с регистрами общего назначения и регистрами блока вещественной арифметики вещественных чисел.

Следующий в данном семействе микропроцессор К128-РИО (1890ВМ7) был оснащён 128-разрядным сопроцессором (СР2) обработки сигналов (о нем речь пойдет ниже).

В дальнейшем вычислительный блок сопроцессора арифметики комплексных чисел был дополнен рядом команд, позволяющих ускорить выполнение ряда типичных операций обработки векторов и матриц. В результате такого развития получился 128-разрядный сопроцессор (СР3), который можно охарактеризовать как векторный. Он, как предполагается, будет функционировать в составе разрабатываемых в настоящее время высокопроизводительных микропроцессоров (1890ВМ8) семейства КОМДИВ.

Микропроцессоры данного семейства имеют ряд особенностей архитектуры, которые следует учитывать при разработке эффективных программных приложений, расширяющих круг задач, для которых эти микропроцессоры были изначально предназначены. Часть этих особенностей связана с реализацией методики встречной оптимизации (применительно к К128-РИО некоторые такие особенности будут приведены ниже). Далее в статье будут приведены примеры практической реализации на сопроцессоре СР2 К128-РИО задач, не связанных с алгоритмами быстрого преобразования Фурье (БПФ) и свёртки, для реализации которых данный сопроцессор обработки сигналов предназначен в первую очередь.

# 1. Общая характеристика К128-РИО и сопроцессора СР2

Микропроцессор К128-РИО включает в себя следующие основные функциональные элементы:

- управляющий процессор КОМДИВ64 архитектуры MIPS64, включающий в себя, в частности, сопроцессор вещественной арифметики СР1;
- специализированный сопроцессор СР2;
- контроллер памяти типа DDR2, обеспечивающий доступ к внешней динамической памяти;
- одноканальный контроллер DMA (контроллер прямого доступа к памяти DDR2);
- контроллер шины RapidIO (RIO);
- контроллер последовательного интерфейса SPI.

Дополнительно имеется ряд блоков, необходимых для целостного функционирования системы и отладочных целей: RS-232 контроллер, накристальная статическая память объемом 32 Кбайт, контроллер прерываний и другие элементы (подробнее см. [2]).

Специализированный сопроцессор СР2 ориентирован на выполнение арифметических операций над комплексными числами или векторами длины 2, элементы которых являются значениями одинарной точности с плавающей запятой. Сопроцессор обладает собственной уникальной системой команд и накристальной программной памятью. Этим, в частности, К128 отличается от К64-РИО, который однороден и полностью соответствует спецификациям MIPS64. Поэтому программисту, желающему освоить СР2 К128-РИО, даже при наличии опыта работы с К64-РИО и его сопроцессором арифметики комплексных чисел, вначале следует изучить систему команд СР2 и то, как он взаимодействует с управляющим ядром.

Сопроцессор СР2 имеет SIMD архитектуру: один поток инструкций, множественные потоки данных, что позволяет выполнять от 2 до 40 операций над 32-разрядными вещественными числами за один такт рабочей частоты. Основные структурные элементы СР2:

- 4 вычислительные секции АЛУ;
- локальная память, LMEM (статическое ОЗУ), в каждой вычислительной секции объемом 64 Кбайт (соответственно, общий объём 256 Кбайт);
- 64 64-разрядных регистра FPR в каждой вычислительной секции;
- память инструкций объёмом 64 Кбайт, вмещающая 213 64-разрядных VLIW-инструкций (см. ниже);
- память коэффициентов (Фурье) объёмом 64 Кбайт (8 Кбайт 64-разрядных слов, итого 213 коэффициентов);
- 16 64-разрядных регистров GPR общего назначения;
- 16 64-разрядных интерфейсных регистров IR (блок интерфейсных регистров для обеспечения взаимодействия между СР2 и контроллером DMA);
- устройство генерации адресов AGU, содержащее по 16 троек 13-разрядных регистров: адреса, шага и режима модификации адреса. Поддерживаются режимы линейной адресации, бит-реверсной адресации и адресации по модулю (для работы с циклическими буферами).

Пиковая производительность К128-РИО на вещественных операциях одинарной точности, при работе на частоте 200 МГц, составляет не менее 8 ГФлоп/с (на практике при работе с алгоритмами, основанными на операциях БПФ и свёртки, хорошим результатом является достижение производительности 50-60% от пиковой [3]). Пиковая скорость обмена данными между СР2 и внешней памятью DDR через DMA – 3,2 Гбайт/с. Пиковая скорость внешнего интерфейса типа RapidIO – 1 Гбайт/с (500 Мбайт/с на чтение и 500 Мбайт/с на запись).

За один такт СР2 позволяет выполнять одну VLIW-инструкцию, состоящую из двух команд: одну вычислительную команду, выполняющуюся сразу во всех четырёх вычислительных секциях (но в каждой из секций над данными, находящимися в своем локальном ОЗУ и регистровом файле FPR) и одну команду управляющей секции. К вычислительным командам относятся:

- арифметические операции над комплексными числами;
- арифметические операции над вещественными числами;
- арифметические операции над целыми числами;
- преобразование форматов данных;
- команды сравнения;
- команды для работы с регистрами FPR;
- nop – пустая операция.

В свою очередь, к управляющим командам (операции пересылки и управления) относятся:

- локальные обмены данными;
- команды управления;
- операция nop.

Для выполнения вычислительных и управляющих команд используются два конвейера, имеющие разную длину. Физически сопроцессор СР2 имеет 3 конвейера различных типов:

- Конвейер для выполнения вычислительных команд. Глубина – 9 тактов.
- Конвейер для обмена данными между LMEM и регистрами FPR. Глубина – 4 такта. На данном конвейере выполняются команды обмена данными между регистровым файлом FPR и локальным ОЗУ, а также команды модификации адреса. При этом адрес модифицируется на второй стадии и доступен для последующей команды, выполняемой на данном конвейере по bypass.
- Конвейер для обмена данными между регистрами различных типов и для выполнения операций управления. Глубина конвейера – 3 такта. На данном конвейере выполняются обмены между регистрами AGU, GPR и регистрами управления, а также операции управления.

## 2. Об особенностях программирования СР2

Сопроцессор СР2 К128-РИО (далее СР2) представляет собой систему, обладающую высоким потенциалом производительности, имеющую параллелизм как на уровне команд (VLIW-инструкции), так и на уровне данных (SIMD - архитектура). Однако использование этого потенциала представляет собой непростую задачу. Перечислим основные, на наш взгляд, факторы, определяющие сложность программирования СР2.

Архитектура SIMD подразумевает, что СР2 выполняет одни и те же команды в нескольких вычислительных секциях, но в каждой из секций – над «своими» данными. Соответственно, в программе следует предусмотреть разделение данных для разных вычислительных секций и аккуратную **обработку неполных** (не кратных числу секций) **наборов данных**.

Также следует учитывать, что для выполнения вычислительной и управляющей части VLIW-инструкции используются конвейеры разной длины.

Существенной особенностью СР2 является **отсутствие аппаратного отслеживания зависимостей по данным и конфликтов по ресурсам.**

Например, команда может прочитать значение своего входного регистра R до того, как оно было записано последней из предшествующих команд, модифицирующих R. Программист может сознательно использовать это свойство СР2, чтобы считать предыдущее значение регистра. Это возможно, так как выполнение программы на СР2 строго синхронно. Выполнение команды не может завершиться раньше, чем ожидается, поскольку отсутствуют какие-либо источники прерываний.

Также возможна ситуация возникновения конфликта по записи, когда две команды одновременно записывают значение в один и тот же регистр, при этом результат не определен. Программист обязан учитывать эти аппаратные особенности, при необходимости вставляя в программу команды `nop`.

Вследствие указанных особенностей, при написании программы для СР2 необходимо отслеживать время исполнения команд: для каждой элементарной команды СР2 известно количество тактов, необходимое для ее завершения, по прошествии которого результат команды становится доступным. Для завершения всех текущих операций используется команда `sync`.

Ассемблер для СР2 обеспечивает анализ входной программы на предмет конфликтов по ресурсам и выдаёт соответствующие диагностические сообщения [4].

Отказ от аппаратного контроля подобных ситуаций является следствием реализации методики встречной оптимизации при проектировании СР2 и позволяет существенно его упростить. Можно считать, что такое решение обусловлено тем, что от запускаемых на СР2 программ ожидается максимальная производительность, которая может быть достигнута только при безостановочной работе конвейера.

Также следует отметить, что при обмене данными между памятью СР2 (LMEM) и глобальной памятью (MEM) **аппаратно не контролируется количество считываемых и записываемых данных**. Завершение пересылки данных происходит по окончании программы формирования адресов для MEM, вне зависимости от состояния программы адресов LMEM. Если адреса LMEM не закончились по завершении программы MEM, то по ним не происходит обращений. Если же программа адресов LMEM закончилась раньше MEM, то оставшиеся данные из MEM пересылаются в LMEM по вновь запущенной программе адресов с теми же параметрами, т. е. старые данные стираются. Из LMEM в MEM пересылаются пустые данные с маской, запрещающей запись. Такие ситуации должны отслеживаться программно.

Что касается локальной памяти вычислительной секции, то она имеет объем 64 Кбайт, со следующей организацией: 2 банка по 2048 128-разрядных слов. Минимальной единицей адресации является 64-разрядное слово. Выбор двух независимых банков обусловлен возможностью совмещать вычисления на СР2 и обмен данными с внешней памятью: обращения к одному банку со стороны DMA и к другому со стороны вычислительной секции СР2 могут производиться параллельно. При обращении к одному банку памяти со стороны DMA и со стороны секции может возникнуть конфликт. В этом случае приоритет отдается обращениям со стороны вычислительной секции.

Отметим также такую особенность СР2, как **когерентность данных**. В К128-РИО доступ к MEM со стороны управляющего процессора осуществляется через кэш. Доступ со стороны СР2 происходит через DMA, в обход кэша. Это означает, что если данные в ОЗУ были модифицированы управляющим процессором, то при запуске пересылки данных в LMEM СР2 из MEM посредством DMA могут быть считаны устаревшие данные. Возможна и обратная ситуация: новые данные из LMEM загружены в MEM через DMA, но управляющий процессор считывает устаревшие данные из кэша. В случае, когда управляющий процессор может (в текущий момент времени) считать из ОЗУ корректные данные, эти данные называются CPU-когерентными. Если последняя версия данных может быть считана посредством DMA, то эти данные DMA-когерентны. Всего возможно три различных ситуации:

- данные только CPU-когерентны;
- данные только DMA-когерентны;
- данные и CPU-когерентны, и DMA-когерентны.

При вызове процедуры синхронизации с CPU или DMA фактически происходит сброс кэша, после чего управляющий процессор или DMA гарантированно сможет считать корректные данные. В связи с такой особенностью СР2 его библиотечные процедуры имеют дополнительный параметр – когерентность данных.

Особенности архитектуры СР2 и его система команд (отсутствие ветвлений, ограниченный

аппаратный стек) не позволяют реализовать для него компиляторы с языков высокого уровня. В настоящее время программы для СР2 разрабатываются на языке ассемблера [4].

Перечисленные особенности СР2 свидетельствуют о том, что его оправданное использование, заключающееся в написании ассемблерных программ, эффективно использующих его вычислительный потенциал, представляет собой сложную задачу, требующую больших трудозатрат и высокой квалификации разработчиков.

# 3. Практическое программирование СР2, композитные функции

К настоящему времени написаны десятки прикладных программ для СР2, собранные в несколько пополняемых библиотек. Основная из них – библиотека цифровой обработки сигналов (БЦОС), содержащая ряд базовых функций работы с СР2: инициализация СР2, работа с памятью, копирование, инициализация и преобразование данных, а также ряд основных операций линейной алгебры (сложение и умножение векторов и матриц, скалярное произведение, транспонирование матриц и т. п.) и обработки сигналов (операции БПФ и свертки) [5].

Однако неполнота набора базовых функций, ограничения на условия эффективного быстродействия каждой из них (что, в частности, затрудняет их совместное использование) и требования вычислительной производительности, диктуемые конкретными практическими задачами, заставляет пользователей К128-РИО разрабатывать новые СР2-приложения, называемые композитными функциями. В [3] выделены следующие четыре условных уровня использования СР2:

1. Вызвать готовую библиотечную функцию. Знаний деталей устройства и функционирования К128-РИО не требуется.
2. Составить собственную процедуру, комбинируя вызовы библиотечных функций.
3. В собственной процедуре несколько библиотечных вызовов заменить одним новым. При этом для вычислений на СР2 используются готовые ядра, но последовательность загрузок данных из DDR в локальную память СР2, выгрузок обратно и запусков вычислительных ядер организуется вручную.
4. Написать и использовать новое вычислительное ядро СР2.

Таким образом, композитные функции – это функции, не вошедшие в БЦОС, и их разработка относится к последним двум из приведённых выше уровней использования СР2. Большую часть композитных функций можно было бы заменить последовательностью вызовов готовых библиотечных функций (свести ко второму уровню использования СР2). Такая замена оправдана в случае, если требования к вычислительной производительности не очень высоки. Использование композитной функции вместо последовательности вызовов функций БЦОС позволяет существенно повысить производительность функции, в частности, путём сокращения количества пересылок между памятью СР2 и глобальной памятью (над данными, загруженными в память СР2, выполняется столько вычислений, сколько возможно, и только после этого результаты выгружаются в глобальную память).

В настоящее время в НИИСИ РАН создаются специализированные библиотеки композитных функций, такие как, например, БКФ-x86. Последняя содержит набор композитных функций разной степени общности, ориентированных на выполнение конкретных прикладных задач: от вычисления линейных комбинаций двух комплексных векторов с вещественными коэффициентами или вычисления множественного разложения Холецкого комплексных матриц 15x15, до функций, самостоятельно реализующих значительный фрагмент прикладной задачи, как, например, формирование характеристик направленности антенны в горизонтальной плоскости.

Композитные функции, как и функции БЦОС, имеют ограничения на условия корректной работы, в частности, на условия, обеспечивающие их максимальную производительность. Эти ограничения согласовываются с параметрами соответствующей прикладной задачи. Согласование может происходить по следующей схеме: в приложении определяется размер частотного диапазона (число частот), размеры антенной решетки и другие параметры, по которым формируются входные и выходные массивы композитной функции. Размерности этих массивов должны укладываться в интервалы, определяющие условия корректной работы композитной функции. Тем самым, принцип встречной оптимизации реализуется и при разработки прикладного ПО для СР2.

В настоящее время имеется ряд приемов, призванных повысить производительность разрабатываемых композитных функций [6]. Одним из важных приемов является совмещение вычислений на СР2 (в одной половине его локальной памяти) с пересылками данных через DMA (в другой половине локальной памяти). Организация локальной памяти вычислительных секций СР2 (см. предыдущий раздел) способствует этому совмещению. В [3] (Глава 5) подробно изложено совмещение вычислений с DMA-пересылками на примере реализации алгоритма множественного длинного (от $2^{13}$ элементов) быстрого преобразования Фурье.

Полезным приемом является и параллельное исполнение вычислительных команд и команд пересылки. Как было указано выше, VLIW-инструкция СР2 формируется из двух команд: арифметической и операции пересылки и управления. Эти команды допускают параллельный запуск на разных конвейерах в течение одного такта. Следовательно, в зависимости от готовности аргументов, за один такт рабочей частоты можно выполнить арифметическую операцию и операцию управления/пересылки. Подобное совмещение команд может увеличить производительность до двух раз.

Заслуживает упоминания и такой пример оптимизации, как развертка цикла. Она используется тогда, когда итерации цикла не зависят друг от друга

по данным или от этой зависимости можно избавиться, например, заменив одно накапливаемое значение на несколько значений. Аргументы команд внутри развернутого цикла готовы перед их вызовом, и нет необходимости вставлять команду `nop`. С реализацией этого приема могут быть связаны некоторые проблемы, такие, например, как дефицит регистров или усложнение использования других приемов оптимизации. Но если развертка возможна, то выполнить ее легко, и она может дать большой прирост производительности (до 8 раз).

Конвейеризация циклов [6-7] представляет собой более универсальный метод оптимизации, чем развертка циклов. В отличие от развертки цикла, данный прием может быть применен и в ситуации дефицита свободных регистров. Основан этот метод на учете времени готовности результатов и использовании «старых» данных из регистра, пока вычисляются «новые». У этого метода есть существенный недостаток: вручную его выполнять трудоемко, а полученную программу сложно модифицировать. В настоящее время разрабатываются средства, позволяющие упростить применение процедуры конвейеризации [7]. В результате программист может изначально писать достаточно простые внутренние циклы, которые правильно реализуют вычисления, но могут быть неоптимальными. При применении этих средств конвейеризации исходный цикл трансформируется в конвейеризованный, на котором достигается максимальная или близкая к максимальной эффективность выполнения заданных программистом вычислений.

# 4. Примеры практического программирования СР2

Практическая реализация различных алгоритмов сигнальной обработки часто начинается со стадии приёма и распаковки входных комплексных сигналов. Дело в том, что из аппаратуры предварительной обработки с выхода АЦП комплексные отсчёты подаются в упакованном виде по 16 или 32 разряда (тип `int16` или `int32`), и данная стадия заключается в том, чтобы распаковать каждый комплексный отсчёт путем выделения его действительной и мнимой части и сформировать массив входных отсчётов для дальнейшей обработки.

Для работы с комплексными данными на СР2, в БЦОС предусмотрен тип данных `cfloat`, определяющий комплексное число в виде структуры, содержащей действительную и мнимую части комплексного числа, являющиеся 32-разрядными вещественными числами формата IEEE 754:

```
typedef struct
  { float r; // Действительная часть
             комплексного числа
  float i; // Мнимая часть комплексного
             числа
   } cfloat;
```

Рассмотрим вначале типичный пример последовательной распаковки данных, реализованный по формуле:

$\mathbf{Out}_i = \mathbf{X}_i + j*\mathbf{Y}_i$;

$\mathbf{X}_i = float(\mathbf{In}_i$[в разрядах с 16-го по 31-й]);

$\mathbf{Y}_i = float(\mathbf{In}_i$[в разрядах с 0-го по 15-й]); i=1,...,N;

где **In** - входной массив упакованных по 32 бит N комплексных отсчётов. **Out** - выходной массив длины N распакованных комплексных отсчётов, j – мнимая единица. Величины $\mathbf{X}_i$ и $\mathbf{Y}_i$ представляют собой синфазную (Re) и квадратурную (Im) составляющие i - го отсчёта комплексного сигнала.

В текущей версии библиотеки БЦОС имеется функция `cp2m_i16cvunpack()`, преобразующая вектор данных типа `int16` в вектор комплексных данных с нулевой мнимой частью (все разряды « упакованных данных» переходят в вещественную часть). Очевидно, напрямую данная функция здесь неприменима, а реализация данной задачи в виде последовательности вызовов нескольких функций БЦОС нежелательна, в силу того, что обычно предъявляются высокие требования ко времени распаковки. Это - один из примеров обсуждаемой в предыдущем разделе ситуации, когда требуется написание композитной функции.

Ее написание не представляет из себя сложной задачи, главным образом, в силу наличия в системе команд СР2 команды `unpck16ws2ps`, преобразующей (с учётом знака) четыре 16-разрядных целочисленных полуслова исходного регистра в вещественные 32-разрядные слова в двух результирующих регистрах по следующей схеме:

`unpck16ws2ps` fd, ft, fs;

fs(AB|CD) → fd(*float*(A) | *float*(B)), ft(*float* (C) | *float* (D));

где A,B,C,B обозначают 16-разрядные данные. Соответственно, основная часть ассемблерного кода, реализующая вычислительное ядро СР2, для этой композитной функции имеет вид:

```
do g0 Loop
       lw  f0,  (r1)+1;;sync  ##  Загрузка  в
         ##регистр f0 двух 32-разрядных слов
         ##(упакованных отсчётов);  в  r1  -
         ##адрес входного вектора
       unpck16ws2ps f10,f11,f0;;sync
         ## Реализация распаковки (см. выше)
       sw f10, (r2)+1
Loop:  sw f11, (r2)+1 ## Выгрузка двух
         ##распакованных отсчётов;  в  r2  -
         ##адрес выходного вектора
```

Приведённый пример кода не является оптимизированным, вопросы оптимизации кода подобного вида рассмотрены в [6]. При отладке и практическом применении подобной композитной функции следует учитывать такую особенность архитектуры К128-РИО как порядок байтов от старшего к младшему (англ. big-endian). В то же время, например, в архитектуре x86 порядок байтов обратный (big-endian). Это приводит к тому, что при распаковке данных следует учитывать архитектуру «внешнего» процессора:

```
int i, n = 1, little_endian = 0;
if (*(char *)&n == 1)//little endian check
      little_endian = 1;
      for (i = 0; i < N; i++) {
        if (!little_endian) {
    Out[i].r = (float)(In[i] >> 16);
    Out[i].i = (float)((In[i] << 16) >> 16);
        } else {
```

```
    Out[i].i = (float)(In[i] >> 16);
    Out[i].r = (float)((In[i] << 16) >> 16);
        }
      }
```

На практике входные данные как правило поступают пачками. При этом, данные пачки имеют сложную структуру: каждый из ее элементов (временных отсчётов) обычно характеризуется тремя параметрами (обозначим их за np, nc и nd). Иными словами, пачка данных представляет собой трехмерный массив, структура которого используется при дальнейшей обработке.

Рассмотрим пример композитной функции, реализующей смену порядка осей массива пачки данных. Ее прототип имеет вид:

```
void func_bin2float_3st ( int Np,
int Nd, int Nc,
cfloat In[], cfloat Out[] );
/*
In - входной массив данных вида [Np][Nd][Nc]
Out- выходной массив вида [Nc][Nd][Np]
nc = 0,…,Nc-1; nd = 0,…,Nd-1; np=0,…,Np-1.
*/
Out[nc][nd][np].re = In[np][nd][nc];
Out[nc][nd][np].im = In[np][nd][nc];
```

Наличие опции смены порядка осей является следствием многопроцессорной реализации алгоритмов сигнальной обработки. Использование этой опции зависит от того, каким образом будет распределена информация между процессорами при вводе сигнала.

В случае небольшого размера данных (до 64Кбайт), они помещаются в одной вычислительной секции CP2, и не возникает проблем, связанных с перестановкой элементов внутри памяти CP2 (данные можно считывать и записывать в любой последовательности и в любой размерности). В случае, когда данные не помещаются в одной вычислительной секции (что часто встречается на практике) ситуация существенно усложняется (в частности, потому что DMA осуществляет пересылки последовательных данных, кратных 16 байт), так что для реализации данной композитной функции требуется разработка своей процедуры обмена данными между LMEM и MEM.

Предположим для определенности, что Nc = 4. В этом случае данные из MEM в LMEM загружаются следующим образом: по 32 байта, т. е. по 4 комплексных отсчёта (по одному на каждый nc =0,…,3) с шагом Nd. Итого будет загружено 4 элемента при nd = 0, np = 0, 4 элемента при nd = 0, np = 1, 4 элемента при nd=0, np = 2 и т. д. (см. Рисунок 1).

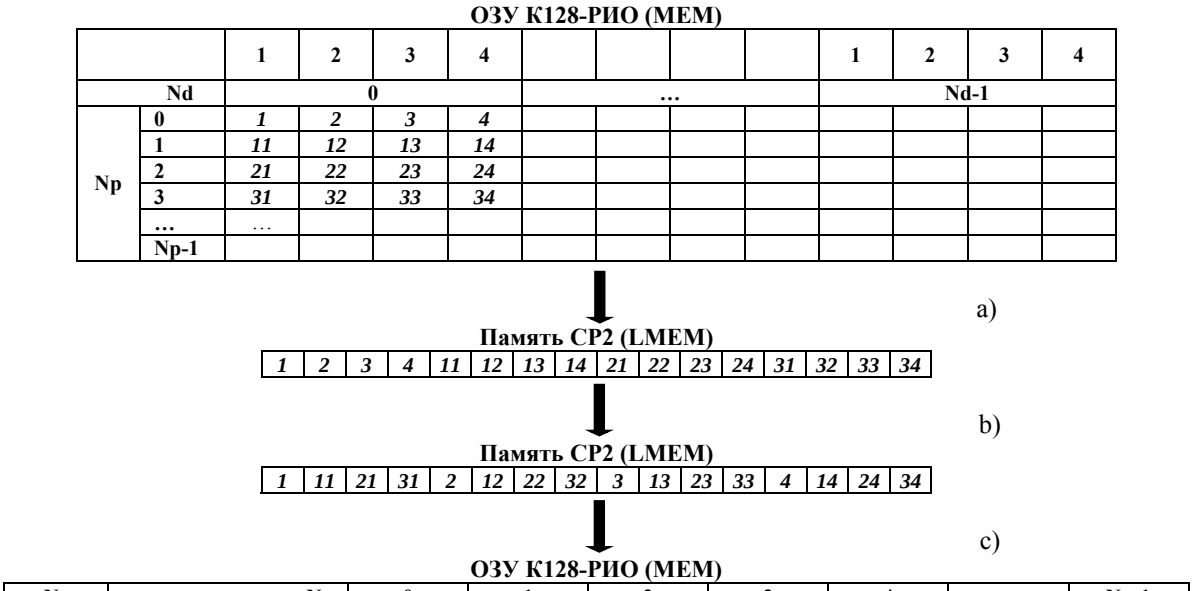
Далее вычислительным ядром CP2 производится считывание, обработка и запись данных. Данные, соответствующие nc=0 записываются в первую четверть памяти CP2, для nc=1 записывается во вторую четверть, для nc=2 и nc=3, соответственно, в третью и четвертую четверть. Приведем фрагмент ассемблерного кода для данного этапа:

```
## в r1 - адрес входного вектора
## в r2-r5 – адреса выходных векторов для
## nc=0,…,3 соответственно; адрес r3 =
## адресу r2 + 4, адрес r4 = адресу r3 + 4,
## адрес r5 = адресу r4 + 4.
##  В комментариях приведены преобразования
## данных из Рис.1 b)
## Цикл по размерности np:
do g0 Loop
      lw f10, (r1)+1 ##  f10 ← 1
      lw f12, (r1)+1 ##  f12 ← 2
      lw f14, (r1)+1 ##  f14 ← 3
      lw f16, (r1)+1 ##  f16 ← 4
      lw f11, (r1)+1 ##  f11 ← 11
      lw f13, (r1)+1 ##  f13 ← 12
      lw f15, (r1)+1 ##  f15 ← 13
      lw f17, (r1)+1 ##  f17 ← 14

      sw f10, (r2)+1 ##  1  → r2
      sw f11, (r2)+1 ##  11 → r2
      sw f12, (r3)+1 ##  2  → r3
      sw f13, (r3)+1 ##  12 → r3
      sw f14, (r4)+1 ##  3  → r4
      sw f15, (r4)+1 ##  13 → r4
      sw f16, (r5)+1 ##  4  → r5
 Loop:  sw f17, (r5)+1 ##  14 → r5, и т. д.
```

Выгрузка данных осуществляется следующим образом. Вначале из CP2 считываются данные, соответствующие nc=0 и последовательно записываются в MEM. Затем считываются данные, соответствующие nc=1 и с шагом = (Np-1)+ Np *(Nd-1)(см. Рисунок 1 часть c)) записываются в MEM, и т. д. Затем данные, соответствующие nc=0 из CP2 с аналогичным шагом записываются в MEM, и т. д. Тем самым, процедура смены порядка осей входного массива будет полностью реализована.

Численные эксперименты с рассмотренными в данном разделе композитными функциями дали следующие результаты. Для функции распаковки данных при длине входного вектора N=1048576 вычисления заняли 0.005281 сек. Для функции смены порядка осей массива при Nh=256, Nd=1024 и Nc = 4, соответственно, 0.011422 сек.




| Nc | Np | | 0 | 1 | 2 | 3 | 4 | … | Np-1 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Nd | 0 | *1* | *11* | *21* | *31* | | | |
| | | 1 | | | | | | | |
| | | 2 | | | | | | | |
| | | … | … | | | | | | |
| | | Nd-1 | | | | | | | |
| 2 | Nd | 0 | *2* | *12* | *22* | *32* | | | |
| | | 1 | | | | | | | |
| | | 2 | | | | | | | |
| | | … | … | | | | | | |
| | | Nd-1 | | | | | | | |
| 3 | Nd | 0 | *3* | *13* | *23* | *33* | | | |
| | | 1 | | | | | | | |
| | | 2 | | | | | | | |
| | | … | … | | | | | | |
| | | Nd-1 | | | | | | | |
| 4 | Nd | 0 | *4* | *14* | *24* | *34* | | | |
| | | 1 | | | | | | | |
| | | 2 | | | | | | | |
| | | … | … | | | | | | |
| | | Nd-1 | | | | | | | |

Рисунок 1 – Этапы реализации алгоритма смены осей массива: a) Загрузка данных из MEM в LMEM, b) Работа вычислительного ядра CP2, c) Выгрузка данных из LMEM в MEM.

## 5. Выводы

В данной статье мы попытались проанализировать накопленный опыт практического применения CP2 с целью расширения круга задач, программируемых с использованием данного сопроцессора.

Естественный путь решения данной задачи – добавление новых функций преобразования данных и математических функций в текущую версию БЦОС, а также пополнение и систематизация содержимого имеющихся на данный момент библиотек композитных функций.

Наряду с пополнением набора готовых функций, представляется весьма целесообразным создание собрания важных для приложений примеров программирования на CP2, реализующих трудные для данной архитектуры ситуации, (таких, например, как упорядочение элементов массива по возрастанию). Вообще на данный момент из содержащихся в доступных источниках примеров программирования CP2 можно упомянуть лишь [3], где содержатся примеры оптимизированных на CP2 операций БПФ и свертки (и отдельные примеры, содержащиеся в [6]-[7].) На наш взгляд, публикация собрания практических примеров программирования CP2 уже назрела.

Наконец, отметим важность создания инструментальных средств оптимизации CP2-кода, типа упоминаемых в [7] средств конвейеризации циклов.

# On some practical issues related to programming of the DSP coprocessor of KOMDIV128-RIO.

**M.S. Aristov, A.S. Osipov, A.M. Shegolkov**

**Abstract**: In the paper, some practical issues related to the DSP coprocessor programming of KOMDIV128-RIO microprocessor, designed by SRISA RAS, are studied. The main features of KOMDIV128-RIO architecture and the programming difficulties arising from these features, are analyzed. Some methods for improving the performance of programs on this coprocessor, are considered. Finally, some examples of design of the DSP composite functions, which are not related to the FFT and convolution algorithms and used for some signal processing practical tasks, are given.

**Keywords:** co-design, microprocessors of the KOMDIV family, KOMDIV-128 RIO, DSP coprocessor CP2, DSPLIB, composite functions.

## Литература

1. В.Б. Бетелин. Отечественные суперкомпьютерные технологии экзафлопсного класса – необходимое условие обеспечения технологической конкурентоспособности России в XXI веке. «Программные продукты и системы», 2013, № 4 (104), с. 4-9.

2. Микросхема интегральная 1890ВМ7Я (КОМДИВ128 - РИО), указания по применению. М.: НИИСИ РАН, 2009.

3. О.Ю. Сударева. Эффективная реализация алгоритмов БПФ и свертки на микропроцессоре Комдив128-РИО. М.: НИИСИ РАН, 2014.

4. Программное изделие «Ассемблер для специализированного сопроцессора (СР2) в составе микропроцессора Комдив128-РИО (АССК128)». Руководство программиста. М.: НИИСИ РАН, 2014.

5. Программа «Библиотека цифровой обработки сигналов (БЦОС)». Справочник по функциям с интерфейсом cp2m программы БЦОС. М.: НИИСИ РАН, 2014.

6. М.С. Аристов. Методы оптимизации программ для процессора КОМДИВ128-РИО. «Труды НИИСИ РАН», 2014, Т. 4, № 2, с. 33-39.

7. Н.И Вьюкова, С.В. Самборский, В.А. Галатенко. Программная конвейеризация циклов для ускорителя плавающей арифметики в составе процессора КОМДИВ128-РИО. «Программные продукты и системы», 2013, № 4 (104), с. 35-43.

# О возможности применения векторного сопроцессора для ускорения операции быстрого преобразования Фурье

**А.А. Бурцев**

*кандидат физико-математических наук*

**Аннотация:** В ФГУ ФНЦ НИИСИ РАН в качестве расширения универсальных микропроцессоров семейства КОМДИВ создан специализированный 128-разрядный сопроцессор, позволяющий ускорять вычисления над векторами комплексных и вещественных чисел одинарной и двойной точности. В статье представлены результаты применения этого сопроцессора, направленные на повышение скорости исполнения операции быстрого преобразования Фурье (БПФ), являющейся одной из основных в задачах цифровой обработки сигналов.

**Ключевые слова:** микропроцессоры семейства КОМДИВ, векторный сопроцессор, цифровая обработка сигналов, операция БПФ.

## Введение

Для поддержки информационной безопасности и технологической конкурентноспособности страны необходимо создавать современные отечественные микропроцессоры, способные обеспечивать в реальном времени высокую производительность научно-технических и инженерных расчётов даже в экстремально жёстких условиях их эксплуатации.

Для решения этой проблемы в ФГУ ФНЦ НИИСИ РАН разрабатывается семейство высокопроизводительных микропроцессоров КОМДИВ [1], в которых в качестве расширения базовой архитектуры предлагаются специализированные сопроцессоры, ориентированные на ускоренное исполнение заданного набора математических функций, наиболее часто употребляемых при решении задач определённой области применения. Такой подход к созданию суперкомпьютеров в ФГУ ФНЦ НИИСИ РАН получил название “встречной оптимизации” [2].

Так микропроцессор К64РИО (1890ВМ6) помимо обычного 64-разрядного сопроцессора (СР1) плавающей вещественной арифметики был дополнен 64-разрядным сопроцессором (СР2) комплексной арифметики, который при дальнейшем развитии был расширен до 128-разрядов и дополнен рядом полезных команд, позволяющих ускорять исполнение ряда типичных операций обработки векторов и матриц. Получившийся в результате 128-разрядный сопроцессор, условно названный векторным (CPV), как предполагается, будет функционировать в составе нового высокопроизводительного микропроцессора (1890ВМ8) семейства КОМДИВ.

На ряде тестовых задач было практически подтверждено [3], что во многих случаях применение CPV позволяет значительно ускорить типичные функции обработки векторов и матриц, характерные для задач линейной алгебры.

В данной статье предпринимается попытка оценить, какой выигрыш (по отношению к СР1) может дать применение этого векторного сопроцессора (CPV) для повышения скорости исполнения операции быстрого преобразования Фурье (БПФ), являющейся ключевой в задачах цифровой обработки сигналов.

## 1. Краткая характеристика векторного сопроцессора

Векторный сопроцессор (CPV) содержит 64 128-битных регистра, в каждом из которых можно хранить:

- 1 комплексное число двойной точности (DC);
- 2 комплексных числа одинарной точности (SC);
- 2 вещественных числа двойной точности (DR);
- 4 вещественных числа одинарной точности (SR).

Для каждого из этих 4-х форматов значений, помещённых в его регистры, CPV обеспечивает соответствующие ему вычислительные команды. Например, каждая из команд умножения с накоплением (см. таблицу 1) выполняет свойственную ей операцию для одной тройки (z,y,x) величин типа DC, либо двух троек типа SC или DR, либо сразу для четырёх троек типа SR.

**Таблица 1.** Вычислительные команды группы умножения

| 1 DC (9)* | 2 SC (7)* | 2 DR (5)* | 4 SR (4)* | cmd z,y,x |
|---|---|---|---|---|
| **cmul.d** | **cmul.s** | **vmul.d** | **vmul.s** | z=y·x |
| **cmadd.d** | **cmadd.s** | **vmadd.d** | **vmadd.s** | z=z+y·x |
| **cmsub.d** | **cmsub.s** | **vmsub.d** | **vmsub.s** | z=z-y·x |
| **cmaddsub.d** | **cmaddsub.s** | **vmaddsub.d** | **vmaddsub.s** | z=z+y·x |
| (t)* - *длительность в тактах (если команда уже в кэше)* | | | | y=z-y·x |

Длительность вычислительных команд зависит от типа обрабатываемых величин (см. в скобках: 4 такта для SR, 5 для DR, 7 для SR и 9 для DC). Но поскольку CPV разрешает на каждом такте начинать исполнение новой команды вычислительного потока, то в итоге можно добиться такой максимальной производительности CPV, при которой n его вычислительных команд смогут исполниться всего за n+8 тактов.

Команды CPV для работы с памятью (см. таблицу 2) позволяют загрузить 128-битное значение целиком в регистр или заполнить его старшую и младшую половину 64-битными значениями по отдельности. Можно одной командой (vldq) загрузить два соседних регистра двумя 128-битными смежными словами из памяти.

Аналогичные команды (vsd, vsdm, vsdq) предусмотрены и для сохранения значений регистров в памяти.

**Таблица 2.** Команды работы с памятью

| | команды загрузки | команды сохранения |
|---|---|---|
| 256 /32 L2(5)* | **vldq**, **vldqx** | **vsdq**, **vsdqx** |
| 64 /8 L1(3)* | **vld**, **vldh**, **vldx**, **vldhx**, **vldlx** | **vsd**,**vsdh**, **vsdx**, **vsdhx** |
| 128 /16 L1(3)* | **vldm**, **vldd**, **vlddh**, **vldmx**, **vlddx**, **vlddhx**, **vlddlx** | **vsdm**, **vsdd**, **vsddh**, **vsdmx**, **vsddx**, **vsddhx** |

** если данные уже в L1(L2)-кэше*

Для указания виртуального адреса в этих командах можно использовать базовую адресацию с относительным смещением (vld) или базовую индексную (vldx). Получаемый адрес должен быть выровнен на границу обрабатываемого слова, т.е. кратен 8, 16 или 32 (см./d).

Архитектурой КОМДИВ предусмотрена возможность в каждом такте взять на исполнение две очередные команды, если эти команды разных потоков. Это позволяет совместить во времени вычислительные команды CPV над одной группой данных с командами CPV для загрузки/сохранения в/из памяти другой группы данных.

Самая продуктивная команда (**cmaddsub**), предусмотренная в CPV над комплексными числами, осуществляет 10 арифметических операций (4 умножения и 6 сложений) над вещественными числами двойной точности или 20 арифметических операций над вещественными числами одинарной точности. Именно такой командой и реализуется базовая операция цифровой обработки сигналов, называемая бабочкой Фурье (БФ).

Таким образом, векторный сопроцессор позволяет ускорить в 10 раз исполнение основной команды (**cmaddsub.d** — бабочки Фурье), многократно используемой для вычисления операции БПФ с комплексными числами двойной точности, и в 20 раз ускорить основную команду (**cmaddsub.s)**, используемую в операции БПФ с комплексными числами одинарной точности, достигая в таком случае потолка своей пиковой производительности (16 ГФлопс на частоте 800 Мгц).

Проанализируем, даёт ли это нам возможность реально добиться на практике такого же ускорения (в 10 и 20 раз соответственно) для операции БПФ?

# 2. Алгоритм БПФ

Пусть задан вектор комплексных чисел $X_N$ длиной N. Его одномерным дискретным преобразованием Фурье (ДПФ) называется комплексный вектор $Y_N$, элементы которого $Y_{m\ (m=0,1,..,N-1)}$ вычисляются по формуле:

$$Y_m = \sum\{X_k \cdot W(m \cdot k,N)\}_{k=0,1,..,N-1}$$

где $W(q,N) = \exp(-i \cdot 2\pi \cdot q/N)$ ( i – мнимая единица).

Вычисление ДПФ непосредственно по этой формуле требует порядка $O(N^2)$ арифметических операций с вещественными числами. Существует, однако, целый класс алгоритмов, объединённых общим названием «быстрое преобразование Фурье» (БПФ), позволяющих вычислить ДПФ для любых значений N всего за $O(N \cdot \log_2 N)$ арифметических операций.

Для выполнения БПФ будем применять вариант известного алгоритма Кули-Тьюки с прореживанием по времени для длины $N=2^P$ (степени двойки). Подробное описание и математическое обоснование такого алгоритма можно найти, например, в [4] или [5, гл.8-12, 31] или в Интернете [6].

В этом алгоритме вектор $Y_N$ получается на месте исходно заданного вектора $X_N$ путём многократного поэтапного выполнения над различными его парами элементов одной и той же базовой операции, так называемой «бабочки Фурье».

Бабочкой Фурье BF(A,B,V) с коэффициентом V над комплексными величинами A и B называется такая единая операция, в результате которой новые значения A′ и B′ для этих величин вычисляются на основе их предыдущих значений по формулам (см. рис. 1):

$A' = A + B \times V$, $B' = A - B \times V$

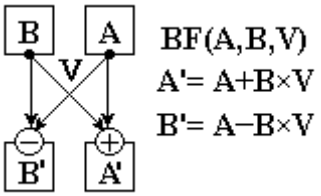



**Рис. 1.** Операция «бабочка Фурье»

Объясним сначала словесно применяемый алгоритм БПФ. Схематично его можно представить состоящим из двух частей (см. рис. 2).

В первой части выполняется перестановка (сортировка) элементов вектора $X_N$ в так называемом бит-реверсном порядке. Для такой бит-реверсной перестановки (БРП) каждому элементу вектора с индексом k, двоичное значение которого задаётся набором битов $k=(b_P,b_{P-1},\ldots,_2,b_1)$ длиной P ($P=\log_2 N$), сопоставляется элемент вектора с индексом m, двоичная запись которого представляется набором тех же битов, но записанных в обратном порядке $m=(b_1,b_2,\ldots,b_{P-1},b_P)$. И все такие пары элементов вектора (для которых $k \neq m$) в процессе БРП меняются местами ($X_k \leftrightarrow X_m$). Например, для вектора X длиной N=16 (P=4) будут переставлены элементы с индексами: $1(0001_2)$ и $8(1000_2)$, $2(0010_2)$ и $4(0100_2)$, $3(0011_2)$ и $12(1100_2)$, $5(0101_2)$ и $10(1010_2)$, $7(0111_2)$ и $14(1110_2)$, $11(1011_2)$ и $13(1101_2)$, а элементы с индексами: $0(0000_2)$, $6(0110_2)$, $9(1001_2)$ и $15(1111_2)$ не изменятся, т.к. у этих индексов битовые коды симметричны и потому совпадают с их бит-реверсными значениями.

Вторая часть алгоритма предполагает цикл из P ($P=\log_2 N$) этапов. На каждом t-ом этапе (t=1,..,P) вектор X длиной N рассматривается разбитым по определённому правилу на N/2 пар элементов.

Пары образуются из элементов, расположенных в массиве на расстоянии s элементов друг от друга (на t-ом этапе $s=2^{t-1}$). Сначала на 1-ом этапе (t=1) пары образуются из соседних элементов (s=1), затем на 2-ом этапе (t=2) пары образуются из элементов, отстоящих друг от друга на 2 элемента (s=2), потом (при t=3) на 4 элемента (s=4), на 8 элементов и т.д., на t-ом этапе расстояние между парами $s=2^{t-1}$, а на последнем этапе (t=P) пары составляются из элементов, стоящих в массиве на расстоянии половины его длины ($s=2^{P-1}=N/2$).

Далее на t-ом этапе над каждой парой вида $(X_m,X_{m+s})$, составленной по описанному выше правилу, выполняется операция бабочки Фурье $BF(X_m,X_{m+s},V_m)$ с коэффициентом $V_m=W(m,h) = \exp(-i \cdot 2\pi \cdot m/h)$, где $h=2^t$.

Для оптимизации вычислений бабочек Фурье целесообразно распределить все пары на группы, собрав в одну k-ую группу (k=0,1,..,G-1) пары элементов вида:

$(X_{k+j\cdot h}, X_{k+j\cdot h+s})$ j=0,1,..,R-1.

Заметим, что для всех пар в одной группе при вычислении бабочки Фурье будет использован одинаковый коэффициент $V_k$, т.к. $V_{k+j\cdot h}=V_k$ для любого j, ибо:

$W(k+j\cdot h,h)= \exp(-i\cdot 2\pi\cdot(k+j\cdot h)/h) = \exp(-i\cdot 2\pi\cdot k)/h)$.

На 1-ом этапе (t=1) будет всего одна группа (G=1), а в ней N/2 пар элементов (R=N/2), и один коэффициент $V_0=W(0,2)$ для всех бабочек. На 2-ом этапе (t=2) будет две группы (G=2), в каждой по N/4 пар элементов (R=N/4), и соответственно два коэффициента: $V_0=W(0,4)$ для всех пар 0-й группы и $V_1=W(1,4)$ для всех пар 1-ой группы.

На каждом следующем этапе число групп уменьшается в 2 раза, а число пар в группах удваивается, так что на t-ом этапе количество групп становится равным $G=2^{t-1}$, а количество пар в каждой группе $R=2^{P-t}$. И на последнем этапе число групп будет $G=2^{P-1}=N/2$, но в каждой группе будет всего одна пара ($R=2^{P-P}=2^0=1$).

Заметим, что используемые коэффициенты можно не вычислять в процессе БПФ, а приготовить заранее в форме массива комплексных констант Cf[0:N], вычислив Cf[q]=W(q,N), полагая, что:

$W(q,N)= \exp(-i\cdot 2\pi\cdot q/N)= \cos(2\pi\cdot q/N) - i\cdot\sin(2\pi\cdot q/N)$.

А поскольку для $u= k\cdot 2^{P-t} = k\cdot N/2^t$ справедливо:

$W(u,N)= \exp(-i\cdot 2\pi\cdot k\ (N/2^t)/N))= \exp(-i\cdot 2\pi\cdot k/2^t)= W(k,2^t)$,

то используя формулу приведения: $W(k,2^t)=W(u,N)$, можно брать требуемые на t-ом этапе коэффициенты $V_k$ из таблицы Cf по индексам $u=k\cdot 2^{P-t}$ соответственно, причём величину $d=2^{P-t}$ не вычислять заново, а формировать постепенно: d:=N/2 на 1-ом этапе и далее d:=d/2 при переходе на следующий этап.

В итоге вторую часть описанного алгоритма БПФ можно представить в следующем виде (на языке Си):

```
s=1; h=2; G=1; R=N/2; d=N/2;
for(t=1, t<=P; t++ ) {
//s=2^(t-1); h=2^t; G=2^(t-1); R=2^(P-t)
  for(k=0,u=0; k<G; k++, u=u+d ) {
    V= Cf[u]; //V=exp(-I*(2*Pi*k/2^t));
    for(j=0,m=k; j<R; j++, m=m+h ) {
      BF(X[m],X[m+s],V); //{*}
    }//for j
  }//for k
  h=h*2; s=s*2; G=G/2; R=R*2; d=d/2;
}//for t
```

Теперь остаётся уточнить, какими действиями (командами, операторами) реализовать в этом алгоритме выделенную (см.**{*}**) операцию бабочки Фурье.

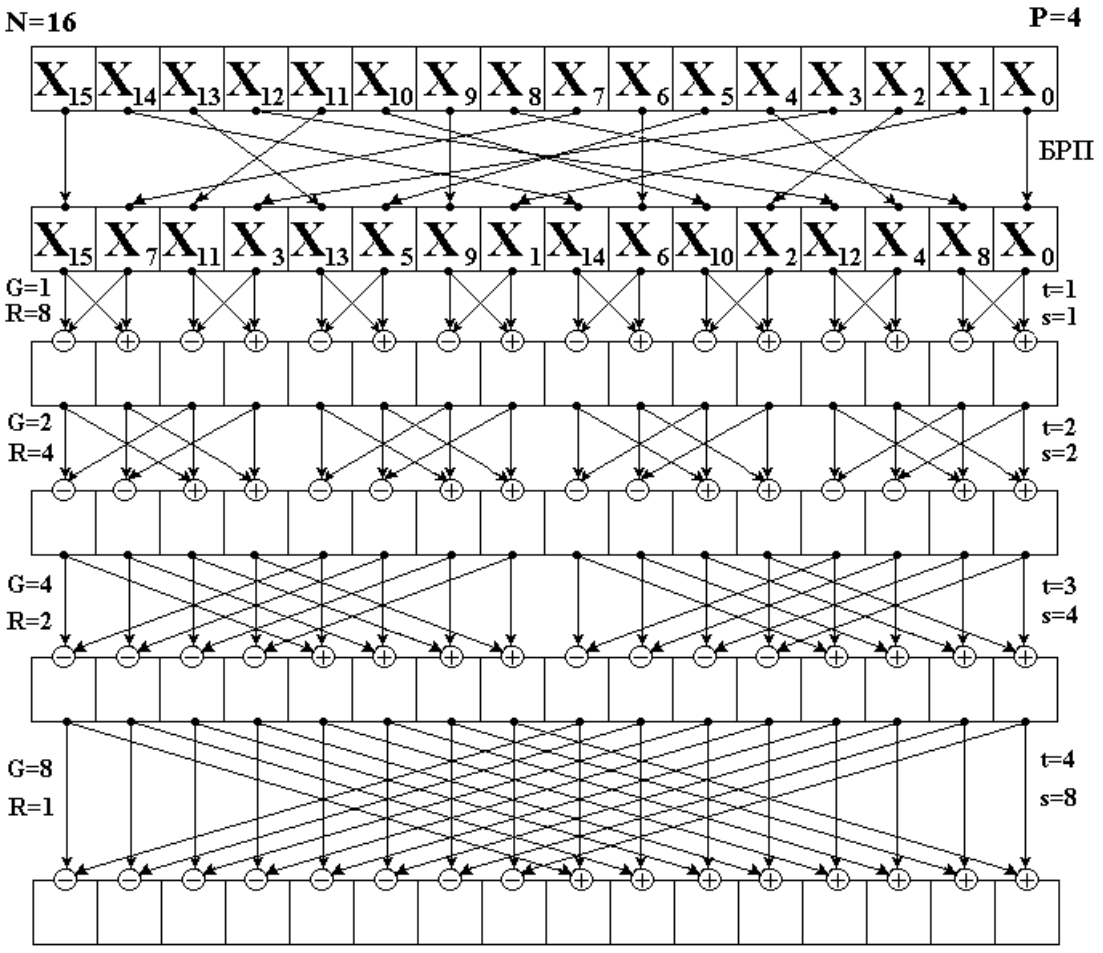



**Рис. 2.** Схема алгоритма БПФ для N=16

# 3. Реализация бабочки Фурье на CP1

Чтобы вычислить бабочку Фурье BF(A,B,V), требуется получить новые значения A′ и B′ для комплексных величин на основе их предыдущих значений:

$A' = A + B \times V;\ B' = A - B \times V$

Если для такой операции не предусмотрена единая команда, то для вычисления бабочки Фурье потребуется три операции над комплексными значениями:

$T = B \times V;\ A = A + T;\ B = A - T;$

и дополнительная память для сохранения вспомогательной комплексной величины T.

Если вообще не предусмотрено арифметических команд (сложения, умножения) над комплексными числами, то выполнение нужных операций умножения и сложения комплексных значений придётся осуществлять серией команд над вещественными числами, представляя каждое комплексное значение парой вещественных. (По сути именно таким способом компилятор с языка Си и реализует работу с комплексными числами, используя обычный сопроцессор плавающей арифметики).

Обозначая каждую комплексную величину парой вещественных: A=(Ar,Ai), B=(Br,Bi), V=(Vr,Vi), представим операцию бабочки Фурье как единое преобразование над парами вещественных чисел:

(Ar′,Ai′) = (Ar+(Br×Vr−Bi×Vi),Ai+(Br×Vi+Bi×Vr)

(Br′,Bi′) = (Ar−(Br×Vr−Bi×Vi),Ai−(Br×Vi+Bi×Vr)

Тогда для реализации операции бабочки Фурье BF(A,B,V) можно применить следующий алгоритм:

```
Ti=(Br×Vi+Bi×Vr);Tr=(Br×Vr-Bi×Vi);
Ai= Ai+Ti; Ar= Ar+Tr;
Bi= Bi-Ti; Br= Br-Tr;
```

Рассмотрим далее, как этот алгоритм может быть реализован для вычисления комплексных значений двойной точности на сопроцессоре CP1 плавающей вещественной арифметики микропроцессора семейства КОМДИВ. В соответствии с этим алгоритмом Си-компилятор сгенерирует ассемблерный (или машинный код) примерно следующего вида:

| вычислительные команды | команды работы с памятью |
|---|---|
| | Ldc1 FVi,8(rV) |
| | Ldc1 FVr,0(rV) |
| | Ldc1 FBi,8(rB) |
| Mul.d FTi,FBi,FVr; | Ldc1 FBr,0(rB) |
| Mul.d FTr,FBi,FVi; | Ldc1 FAi,8(rA) |
| Madd.d FTi,FTi,FBr,FVi; | Ldc1 FAr,0(rA) |
| Msub.d FTr,FTr,FBr,FVr; | |
| Sub.d FBi,FAi,FTi; | |
| Sub.d FBr,FAr,FTr; | |
| Add.d FAi,FAi,FTi; | Sdc1 FBi,8(rB) |
| Add.d FAr,FAr,FTr; | Sdc1 FBr,0(rB) |
| | Sdc1 FAi,8(rA) |
| | Sdc1 FAr,0(rA) |

Назначение вещественных 64-разрядных регистров CP1:
FAr=Ar, FAi=Ai; FBr=Br, Fbi=Bi; FVr=Br, FVi=Bi;
FTr=(Br×Vr−Bi×Vi), FTi=(Br×Vr−Bi×Vi);
А целочисленные регистры rV, rA и rB указывают на расположенные в памяти комплексные значения V, A и B:
rV→(Vr,Vi), rA→(Ar,Ai), rB→(Br,Bi)

В нём 8 вычислительных команд (8M) и 10 команд работы с памятью: 6 команд загрузки (6L) и 4 команды выгрузки (4S). Если и программа, и обрабатываемые данные уже находятся в кэш-памяти, то такой блок из 18 команд выполняется за **28** тактов (установлено экспериментально).

А если этот блок команд будет исполняться многократно в цикле с одинаковыми значениями V, то команды загрузки в регистры FVi и FVr можно поместить перед циклом, и тогда оставшиеся в цикле 16 команд будут исполняться уже за **26** тактов.

Почему же не за 16? Дело в том, что многие команды исполняются более, чем за 1 такт: команды загрузки/выгрузки (Ldc1, Sdc1) – за 2 такта, а вычислительные (Mul.d, Madd.d, Msub.d, Sub.d, Add.d) – от 5 до 8 тактов.

Заметим, что CP1 не позволяет на следующем такте начать исполнять над другими регистрами ту же самую вычислительную команду, которую он только что начал, поэтому дальнейшей существенной оптимизации кода путём совмещения вычислений сразу над несколькими группами данных на CP1 добиться не получается. Но такая оптимизация становится возможной при применении векторного сопроцессора CPV.

# 4. Оптимизация вычислений операций бабочки Фурье на CPV

Каким бы совершенным не был имеющийся аппаратный исполнитель, эффективность реализации требуемой программы во многом определяется совершенством её программного кода. Поэтому анализ возможного выигрыша в производительности, который может обеспечить CPV для вычислений операций бабочек Фурье, будем проводить с учётом известных приёмов оптимизации кода программы для микропроцессоров семейства КОМДИВ.

## 4.1. Одиночное вычисление бабочки

Для вычисления на CPV одиночной операции бабочки Фурье BF(A,B,V) над комплексными величинами двойной точности можно предложить следующий вариант (1) кода:

| ## вариант 1 (18 тактов) | такты | пояснения: |
|---|---|---|
| v**L**dm VV,0(rV) | 3 | VV ← (Vr,Vi) |
| v**L**dm VB,0(rB) | 3 | VB ← (Br,Bi) |
| v**L**dm VA,0(rA) | 3 | VA ← (Ar,Ai) |
| c**M**addsub.d VA,VB,VV | 9 | BF(VA,VB,VV); |
| v**S**dm VB,0(rB) | 3 | VB → (Br,Bi) |
| v**S**dm VA,0(rA) | 3 | VA → (Ar,Ai); |

Назначение 128-разрядных регистров CP2:
VV = V= (Vr,Vi); VB = B = (Br,Bi); VA = A = (Ar,Ai);
rV → V= (Vr,Vi); rB→ B = (Br,Bi); rA→ A = (Ar,Ai);

Но эти 6 команд потребуют аж **18** тактов (5+9+4=18). Так что применение CPV для вычислений бабочек Фурье в таком варианте «по одной за раз» никакой особой выгоды не приносит.

Однако, можно существенно оптимизировать код, чтобы ускорить на CPV вычисления, когда требуется исполнить сразу несколько операций бабочек Фурье.

## 4.2. Групповое вычисление бабочек

Допустим, нужно вычислить группу из n операций бабочек Фурье: BF(Ak,Bk,Vk) k=1,2,..,n над различными не зависимыми друг от друга величинами. (Пусть n<22, чтобы хватило 64-х регистров CPV для размещения в них величин Ak, Bk, Vk для k=1,2,..,n,).

Тогда для вычисления всей этой группы можно предложить следующий вариант (2) кода:

```
## BFG (rA, rB, rV, n)
##  вариант 2 ( 6n+2 тактов на n бабочек при n>8 )
##  rVk → Vk,  rBk → Bk,  rAk → Ak   k=1,2,..,n
 vLdm   VV1,0(rV1); ... vLdm   VVn,0(rVn);
 vLdm   VB1,0(rB1);      vLdm   VA1,0(rA1);
        ...              ...
 vLdm   VBn,0(rBn);      vLdm   VAn,0(rAn);
 cMaddsub.d VA1,VB1,VV1;
  ...
 cMaddsub.d VAn,VBn,VVn;
 vSdm   VB1,0(rB1);      vSdm   VA1,0(rA1)
  ...                     ...
 vSdm   VBn,0(rBn);      vSdm   VAn,0(rAn);
```

Полагая, что команды v**L**dm и v**S**dm могут исполняться так за тактом без задержек (для загрузки и выгрузки разных регистров CPV), и CPV позволяет на каждом такте брать на исполнение очередную команду c**M**addsub.d (над другими регистрами), то весь этот блок из 6·n команд при n>8 можно исполнить без задержек за 6·n+2 тактов (+2 такта потребутся для завершения последней команды v**S**dm). В случае, когда n<9, понадобится 5·n+11 тактов (плюс 9–n тактов задержки перед исполнением первой командой v**S**dm).

В результате такой оптимизации получим скорость вычислений примерно **6** тактов на одну бабочку.

## 4.3. Вычисление бабочек для векторов

Более значительной оптимизации (при чётном n>9) для группового вычисления бабочек Фурье можно добиться, когда элементы пар Aj и Bj для j-ой бабочки являются j-ми элементами векторов A и B, непрерывно лежащих в памяти, а все коэффициенты Vj предварительно уже загружены в регистры CPV.

В таком случае можно использовать одну команду v**L**dq (вместо двух v**L**dm) для загрузки двух элементов вектора в два соседних регистра и одну команду v**S**dq (вместо двух v**S**dm) для выгрузки двух элементов из двух соседних регистров соответственно.

Продемонстрируем вариант (3) кода для n=16:

```
## вариант 3 для n=16 ( 52 такта на 16бабочек )
## BFV16 (rA, rB)
##  rB →B[0:n-1],  rA → A[0:n-1],  p=n/2=8
 vLdq VB0,0(rB);       vLdq VA0,0(rA)
 vLdq VB2,2*16(rB);    vLdq VA2,2*16(rA)
  ...                    ...
 vLdq VB14,14*16(rB); vLdq VA14,14*16(rA)
  cMaddsub.d VA0,VB0,VV0;
  cMaddsub.d VA1,VB1,VV1; ...
  cMaddsub.d VA14,VB14,VV14;
  cMaddsub.d VA15,VB15,VV15;
 vSdq VB0,0(rB);       vSdq VA0,0(rA)
 vSdq VB2,2*16(rB);    vSdq VA2,2*16(rA)
  ...                    ...
 vSdq VB14,14*16(rB); vSdq  VA14,14*16(rA)
```

Такой блок из 48-ми команд (16L+16M+16S) сможет выполниться за 52 такта (+4 такта потребутся для завершения последней команды v**S**dq). В итоге получим скорость вычислений примерно **3.25** такта (52/16=3+4/16= 3.25) на одну бабочку.

## 4.4. Совмещённое вычисление бабочек

Для дальнейшей оптимизации воспользуемся тем, что архитектурой КОМДИВ предусмотрена возможность на каждом такте брать на исполнение сразу две команды разных потоков. Поэтому в одном такте можно совмещать одну вычислительную команду CPV с одной командой загрузки или выгрузки одного или двух регистров. Значит, блок из 48-ми команд, рассмотренных выше в варианте 3, можно исполнить за меньшее число тактов, если каждую команду c**M**addsub.d удастся совместить в одном такте с какой-нибудь командой загрузки v**L**dq или выгрузки v**S**dq для других регистров.

Для этого можно предложить, например, следующий вариант (4) кода:

```
## вариант 4 для n=16 ( 36 тактов на 16 бабочек )
## BFV16 (rA, rB)
##   rB →B[0:n-1],  rA → A[0:n-1],  p=n/2=8
 vLdq VB0,0*16(rB);    vLdq VA0,0*16(rA)
 vLdq VB2,2*16(rB);    vLdq VA2,2*16(rA)
 vLdq VB4,4*16(rB);    vLdq VA4,4*16(rA)
cMaddsub.d VA0,VB0,VV0; vLdq VB6,6*16(rB)
cMaddsub.d VA1,VB1,VV1; vLdq VA6,6*16(rA)
  ...                     ...
cMaddsub.d VA8,VB6,VV8; vLdq VB14,14*16(rB)
cMaddsub.d VA9,VB7,VV9; vLdq VA14,14*16(rA)
cMaddsub.d VA10,VB10,VV10; vSdq VB0,0*16(rB)
cMaddsub.d VA11,VB11,VV11; vSdq VA0,0*16(rA)
cMaddsub.d VA12,VB12,VV12; vSdq VB2,2*16(rB)
cMaddsub.d VA13,VB13,VV13; vSdq VA2,2*16(rA)
cMaddsub.d VA14,VB14,VV14; vSdq VB4,4*16(rB)
cMaddsub.d VA15,VB15,VV15; vSdq VA4,4*16(rA)
 vSdq VB6,6*16(rB);      vSdq VA6,6*16(rA)
  ...                     ...
 vSdq VB14,14*16(rB);    vSdq VA14,14*16(rA)
```

В таком случае 16 бабочек Фурье удастся вычислить за: 6L+10M||10L+6M||6S+10S+4= (32+4)= 36 тактов. А если группу из n бабочек (n=16·p) вычислять циклом по 16 бабочек за один шаг, то на это потребуется: (32+1)·p+4 тактов (+1 на команду перехода с приращением счётчика цикла), что в итоге даст оптимальную скорость вычислений (33·p+4)/(16·p)≈2, т.е. примерно **2** такта на 1 бабочку.

Чтобы добиться ещё большего ускорения, необходимо уменьшать количество команд работы с памятью. Ведь сейчас (см. варианты 3 и 4) на каждую пару команд c**M**addsub.d, в которых задействованы 4 регистра ($VA_i$,$VA_{i+1}$,$VB_i$,$VB_{i+1}$), требуется две команды загрузки v**L**dq и две команды выгрузки v**S**dq для этих регистров. Получается на 2 бабочки такой баланс команд: 2L+2M+2S, который никак не удастся исполнить менее, чем за 4 такта.

В идеальном случае, когда все нужные для вычислений бабочек величины уже хранятся в регистрах и остаются там после исполнения каждой операции, можно достигать предельной скорости вычислений, исполняя одну бабочку за каждый такт. Но для этого

необходимо вначале загрузить из памяти в регистры CPV все величины, участвующие в дальнейших вычислениях бабочек Фурье, затем выполнить вычисления всех бабочек с участием загруженных регистров, и лишь после этого полученные в регистрах значения отправлять в память как новые значения этих величин.

# 5. Реализация БПФ на CPV

Для исполнения операции БПФ над вектором длиной N элементов необходимо вычислить $M=N\cdot\log_2 N/2$ бабочек Фурье и при этом хотя бы по одному разу прочитать из памяти в регистр CPV каждый элемент вектора, а затем записать в него обновлённое значение. Значит, в лучшем случае для вычисления $N\cdot\log_2 N/2$ бабочек потребуется загрузить N элементов вектора (да ещё N/2 коэффициентов) в регистры, а после выгрузить их обратно. Но такой благоприятный вариант возможен только если у CPV хватит регистров для размещения всех N+N/2 величин (для элементов и коэффициентов).

Поскольку количество регистров CPV ограничено (их всего 64), то такой вариант можно применить лишь при выполнении БПФ для векторов малой длины.

## 5.1. БПФ длины 32

Содержать все элементы вектора и все коэффициенты в регистрах CPV на протяжении вычислений всех бабочек Фурье возможно лишь при выполнении БПФ длины не более 32 (если рассматривать в качестве длины только степени двойки $N=2^P$).

Чтобы выполнить БПФ для вектора длиной N=32 (БПФ-32), потребуется вычислить 80 бабочек: по 16 бабочек на каждом из 5 этапов ($P=\log_2 32=5$). Элементы вектора можно прочитать из памяти в 32 регистра лишь один раз перед 1-ым этапом, а далее оставлять их в регистрах, и только при завершении последнего 5-ого этапа отправлять полученные значения в память. Ещё 16 регистров потребуется загрузить коэффициентами, необходимыми для вычисления этих 80 бабочек. Итого будет занято 48 регистров CPV.

Заметим, что первоначальную загрузку значений элементов вектора в регистры можно совместить с их бит-реверсной перестановкой, т.е. в регистр VXk (k=0,1..,31) загружать из памяти такой элемент X[m] вектора, индекс m которого соответствует бит-реверсному двоичному значению индекса k.

В результате для выполнения БПФ-32 потребуется:

1) 8 команд v**L**dq для загрузки 16 коэффициентов;
2) 32 команды v**L**dm для чтения в регистры 32-х элементов вектора в бит-реверсном порядке;
3) 80 команд c**M**addsub.d для вычислений 80 бабочек Фурье (по 16 на каждом этапе);
4) 16 команд v**S**dq для записи из регистров в память новых значений для 32-х элементов вектора.

Используя определения следующих макросов:

```
##    BF(i,j,k) => cmaddsub.d VXi,VXj,VVk;
#define BF(i,j,k) cmaddsub.d VX##i##,VX##j##,VV##k##;
##    LoadX(i,j) => vldm VXi,j*16(rX) ## VXi ← X[j]
#define LoadX(i,j) vldm VX##i##,##j##*16(rX);
##   LoadXX(i) => vldq VXi,i*16(rX) (VXi,VXi+1)←X[i:i+1]
#define LoadXX(i) vldq VX##i##,##i##*16(rX);
## StoreXX(i) => vsdq VXi,i*16(rX) (VXi,VXi+1)➔X[i:i+1]
#define StoreXX(i) vsdq VX##i##,##i##*16(rX);
##   LoadVV(i) => vldq VVi,i*16(rV) (VVi,VVi+1)←V[i:i+1]
#define LoadVV(i) vldq VV##i##,##i##*16(rV);
```

приведём возможный вариант кода, осуществляющий такое БПФ-32:

```
## cp3_VFFT32 (rX,rV)  ## rX → X[0:31], rV → V[0:15]
## load VX00..VX18 || load VV0,VV1:
 LoadVV(0);   LoadX(0,0);   LoadX(1,16);  LoadX(2,8);
 LoadX(3,24);  LoadX(4,4);   LoadX(5,20);  LoadX(6,12);
 LoadX(7,28);  LoadX(8,2);   LoadX(9,18);  LoadX(10,10);
 LoadX(11,26); LoadX(12,6);  LoadX(13,22); LoadX(14,14);
 LoadX(15,30); LoadX(16,1);  LoadX(17,17); LoadX(18,9);
## t=1;(k=0,V0)|| load VX19..VX31 || load VVi, i=8,9,4,5,12,13:
BF(0,1,0);    LoadX(19,25);  BF(2,3,0);    LoadX(20,5);
BF(4,5,0);    LoadX(21,21);  BF(6,7,0);    LoadX(22,13);
BF(8,9,0);    LoadX(23,29);  BF(10,11,0); LoadX(24,3);
BF(12,13,0); LoadX(25,19);  BF(14,15,0); LoadX(26,11);
BF(16,17,0); LoadX(27,27);  BF(18,19,0); LoadX(28,7);
BF(20,21,0); LoadX(29,23);  BF(22,23,0); LoadX(30,15);
BF(24,25,0); LoadX(31,31);  BF(26,27,0); LoadVV(8);
BF(28,29,0); LoadVV(4);     BF(30,31,0); LoadVV(12);
##--- t=2; (k=0,V0), (k=1,V8); || load VVi, i=2,3,6,7,10,11,14,15:
BF(0,2,0);    LoadVV(2);    BF(4,6,0);    LoadVV(6);
BF(1,3,8);    LoadVV(10);   BF(5,7,8);    LoadVV(14);
BF(8,10,0);  BF(12,14,0);   BF(9,11,8);  BF(13,15,8);
BF(16,18,0); BF(20,22,0);   BF(17,19,8); BF(21,23,8);
BF(24,26,0); BF(28,30,0);   BF(25,27,8); BF(29,31,8);
##--- t=3; (k=0,V0), (k=1,V4), (k=2,V8), (k=3,V12);
BF(0,4,0);    BF(8,12,0);   BF(1,5,4);    BF(9,13,4);
  BF(2,6,8);   BF(10,14,8);   BF(3,7,12);   BF(11,15,12);
BF(16,20,0); BF(24,28,0);   BF(17,21,4); BF(25,29,4);
  BF(18,22,8); BF(26,30,8);   BF(19,23,12); BF(27,31,12);
##--- t=4; (для k,Vk·2), (k=0,1,..,7);
BF(0,8,0); BF(1,9,2); BF(16,24,0); BF(2,10,4);
BF(17,25,2); BF(3,11,6); BF(18,26,4); BF(4,12,8);
BF(19,27,6); BF(5,13,10); BF(20,28,8); BF(6,14,12);
 BF(21,29,10); BF(7,15,14); BF(22,30,12); BF(23,31,14);
##--- t=5; (для k,Vk), (k=0,1,8,9,2,3,10,11,4,5);
BF(0,16,0);  BF(1,17,1);  BF(8,24,8);  BF(9,25,9);
BF(2,18,2);  BF(3,19,3);  BF(10,26,10); BF(11,27,11);
BF(4,20,4);  BF(5,21,5);
##--- t=5; (для k,Vk), (k=12,13,6,7,14,15); || VX ➔ X[0:31]
BF(12,28,12); StoreXX(0);    BF(13,29,13); StoreXX(16);
BF(6,22,6);   StoreXX(8);    BF(7,23,7);   StoreXX(24);
BF(14,30,14); StoreXX(2);    BF(15,31,15); StoreXX(18);
## ----- store VX0..VX31 ➔ X[0:31] :
StoreXX(10); StoreXX(26); StoreXX(4); StoreXX(20);
StoreXX(12); StoreXX(28); StoreXX(6); StoreXX(22);
StoreXX(14); StoreXX(30);
```

В нём 136 команд (8L+32L+80M+16S=136) размещены так, чтобы они могли исполниться за 114 тактов (20L+20M||20L+54M+6M||6S+10S=110, плюс 4 такта на завершение последней операции v**S**dq 110+4=114). Итого получается 114/80=**1.425** тактов на 1 бабочку.

## 5.2. БПФ длины $N=2^P>32$

Вторая часть алгоритма БПФ для вектора длиной $N=2^P>32$ , которая начинает выполняться после бит-реверсной перестановки (БРП) элементов вектора, на первых 5-ти этапах по сути совершает над каждой группой из 32-х элементов операцию БПФ длиной 32, но только уже без БРП.

Поэтому в алгоритме БПФ произвольной длины будем применять вариант БПФ-32 (из п.5.1, но без

БРП) для исполнения 5-ти первых этапов (часть 2А). А на последующих этапах (часть 2Б) будем вычислять бабочки Фурье группами (по 16 штук), как было рассмотрено в п.4.3-4.4. В итоге получим алгоритм БПФ, который можно выразить схематично в следующем виде (на языке Си):

```
// часть 1 Бит-Реверсная Перестановка (БРП)
cp3_VFFT_BitReverseExchange(X,BRT,BRN);//{*1}
// часть 2А t=1..5 БПФ-32 (без БРП) над группами по 32
cp3_VWLoad32(Cf32); {*2}
for(k=0;k<N>>5;k++)
 cp3_DoVFFT32(&X[k*32]);//{*3}
// часть 2Б t=6..P
s=32; h=2; G=32; R=N/64; d=N/64;
for(t=6, t<=P; t++ ) {
 for(k=0,u=0; k<G; k=k+16, u=u+16*d ) {
  cp3_VFFT_Load16W(&Cf[u],d); //{*4}
  for(j=0,m=k; j<R; j++, m=m+h ) {
   cp3_VFFT_Do16BF(&X[m],&X[m+s]);//{*5}
  }//for j
 }//for k
 h=h*2; s=s*2; G=G/2; R=R*2; d=d/2;
}//for t
```

Характеристика ассемблерных функций:

**cp3_VFFT_BitReverseExchange** – выполняет бит-реверсную перестановку (БРП) элементов массива

**cp3_VWLoad32** – загружает в регистры VV0,..,VV15 CPV все 16 коэффициентов бабочек Фурье для БПФ-32

**cp3_DoVFFT32** – выполняет БПФ для вектора длиной 32 как описано в п.6.1, но без БРП и без загрузки коэффициентов

**cp3_VFFT_Load16W** – загружает в регистры VV0,..,VV15 CPV очередные 16 коэффициентов бабочек Фурье

**cp3_VFFT_Do16BF** – вычисляет 16 бабочек Фурье BFV16 как описано в п.5.3 (или 5.4)

Приведём вариант первоначальной части кода для **cp3_DoVFFT32**, которой он отличается от кода рассмотренной выше процедуры **cp3_VFFT32** для БПФ-32 в предположении, что все 16 коэффициентов для БПФ-32 уже загружены (перед циклом) в CPV-регистры VV0..VV15 процедурой **cp3_VWLoad32**:

```
##    cp3_DoVFFT32 (rX)      ## rX → X[0:31]
## ---- load X[0:9] ➔ VX0..VX9 (без БРП):
 LoadXX(0); LoadXX(2); LoadXX(4); LoadXX(6); LoadXX(8);
##--- t=1; (k=0,V0) || load X[10:31] => VX10..VX31:
BF(0,1,0);    LoadXX(10); BF(2,3,0);    LoadXX(12);
BF(4,5,0);    LoadXX(14); BF(6,7,0);    LoadXX(16);
BF(8,9,0);    LoadXX(18); BF(10,11,0); LoadXX(20);
BF(12,13,0); LoadXX(22); BF(14,15,0); LoadXX(24);
BF(16,17,0); LoadXX(26); BF(18,19,0); LoadXX(28);
BF(20,21,0); LoadXX(30); BF(22,23,0);
BF(24,25,0); BF(26,27,0); BF(28,29,0); BF(30,31,0);
##--- t=2; (k=0,V0), (k=1,V8);
BF(0,2,0);    BF(4,6,0)       BF(1,3,8);    BF(5,7,8);
BF(8,10,0);  BF(12,14,0);   BF(9,11,8);  BF(13,15,8);
## . . . и далее как в процедуре cp3_VFFT32 . . .
```

В этом варианте кода (16L+80M+16S=112) 112 команд размещены так, чтобы они могли исполняться за (5L+11M||11L+63M+6M||6S+10S=95) 95+4=99 тактов. А при исполнении их в цикле m раз (m=N/32) потратится (95+1)·m+4 тактов, т.е получится примерно (96·m+4)/(80·m)≈96/80= **1.2** тактов на 1 бабочку.

Заметим, что, вообще говоря, выполнить серию БПФ-32 можно быстрее, чем одиночную БПФ-32. Когда предстоят вычисления БПФ-32 для нескольких векторов, то цикл, на одном шаге которого вычисляется БПФ для очередного вектора, можно усовершенствовать.

Во-первых, коэффициенты для бабочек Фурье можно загружать в регистры CPV один раз перед циклом, что и предусмотрено в рассмотренном варианте.

Во-вторых, на каждом шаге цикла вычисления бабочек для текущего вектора можно совмещать с записью в память вычисленных значений для предыдущего вектора и чтением из памяти в регистры значений следующего вектора.

Такое усовершенствование, казалось бы, позволило бы нам добиться скорости вычислений, близкой к идеальной (1 бабочка за такт). Но для его осуществления необходимо выделить регистры CPV для хранения в них значений элементов не одного, а сразу двух векторов. Так что для реализации этой возможности для БПФ длины 32 потребуется 32+32+16=80 регистров, но таким количеством регистров CPV не располагает. Значит, усовершенствовать таким способом рассмотренный вариант кода для БПФ-32 не получится.

Можно, конечно, было бы проанализировать возможный вариант такого усовершенствования для БПФ длиной 16 (БПФ-16). Но БПФ-16 не может быть эффективно реализован на CPV по другой причине: вычисляемые в ходе БПФ-16 32 бабочки Фурье ($16\cdot\log_2 16/2=16\cdot 4/2=32$) не могут уложиться в идеальные 32 такта, т.к. команда cmaddsub.d исполняется аж 9 тактов, а возникающие зависимости между следующими и предыдущими бабочками вынуждено порождают значительные задержки, так что на чистое вычисление 32-х бабочек (без операций чтения/записи) потребуется как минимум 42 такта, что уже уступает по производительности рассмотренному выше оптимальному варианту БПФ-32 (42/32=1.3125 > 1.2).

Таким образом, с помощью векторного сопроцессора удаётся значительно ускорить операцию БПФ для вектора комплексных величин двойной точности. В частности, выполнить БПФ-32 на CPV путём вызова описанной выше функции **cp3_VFFT32** можно почти в **22.5** раза быстрее, чем на CP1 путём вызова Си-функции, алгоритм которой представлен в п.2. На внутренний цикл этой функции, где вычисляется одна бабочка BF(X[m],X[m+s],V) уходит **32** такта (установлено экспериментально). Значит, даже без учёта БРП на вычисление 80 бабочек на CP1 потратится как минимум 32·80 тактов, а на CPV — всего 114 тактов (32·80/114≈22.456).

Для вектора произвольной длины $N=2^P>32$ можно также существенно ускорить (примерно в **16** раз) вторую часть алгоритма БПФ, где производятся многочисленные вычисления бабочек Фурье, т.к. вместо **32** тактов можно затрачивать на выполнение одной бабочки в среднем 1.2 такта (в части 2А) и примерно 2 такта (в части 2Б).

Остаётся выяснить, как векторный сопроцессор способствует ускорению процедуры бит-реверсной перестановки, которую требуется выполнять в первой части алгоритма БПФ.

## 5.3. Оптимизация БРП

Чтобы оптимизировать исполнение бит-реверсной перестановки (БРП) вектора длиной N, можно заранее

подготовить таблицу BRT пар различных индексов ($k_j \neq m_j$), задающих, какие элементы вектора должны будут переставляться. Индексы ($k_j$,$m_j$) в каждой такой паре должны быть друг для друга бит-реверсными перестановками их двоичных кодов: $k_j=(b_P,\ldots,b_2,b_1)$, $m_j=(b_1,b_2,\ldots,b_P)$. Тогда алгоритм БРП будет просто переставлять всевозможные пары $X[k_j] \leftrightarrow X[m_j]$ элементов вектора:

```
for(j=0; j<BRN; j++ )
{ k= BRT[2*j]; m= BRT[2*j+1];
  T= X[m]; X[m]=X[k]; X[k]=T; }//for j
```

Заметим, что для доступа к элементу массива X[k] по индексу (k) потребуется вычислять смещение элемента относительно начала массива путём домножения его индекса на размер элементов массива (k*16). Поэтому для оптимизации БРП целесообразно приготовить в таблице BRT не просто индексы переставляемых элементов, а вычисленные для них смещения, что позволит сократить внутренний цикл БРП на 2 команды. Но для этого и алгоритм БРП (на языке Си) придётся переписать немного иначе:

```
for(j=0; j<BRN; j++ )
{ Xk=(char*)&X+BRT[2*j];
  Xm=(char*)&X+BRT[2*j+1];
  T=*Xm; *Xm=*Xk; *Xk=T; }//for j
```

Для выполнения такой же БРП на CPV можно предложить следующий ассемблерный код:

```
cp3_VFFT_BitReverseExchange: ##
move rX, a0; move rBRT,a1; move rK,a2
BRELoop: ## for(rK=BRN;rK>0;rK=rK-1)
         ## Exchange X[rI]<=>X[rJ] :
 lwu rI,0(rBRT) ## rJ=BRT[2*rK]
 lwu rJ,4(rBRT) ## rJ=BRT[2*rK+1]
    addiu rBRT,rBRT,8
 vldmx VXI,rI(rX) ## VXI:= X[rI]
    addi rK,rK,-1 ## rK:= rK-1
 vldmx VXJ,rJ(rX) ## VXJ:= X[rJ]
 vsdmx VXI,rJ(rX) ## X[rJ]:= VXI;
 bgtz rK,BRELoop
 vsdmx VXJ,rI(rX) ## X[rI]:= VXJ;
 jr ra ; ssnop
```

Но он позволит ускорить процедуру БРП с помощью CPV всего лишь примерно в **1.75** раза (теперь 1 шаг цикла перестановки будет исполняться за 8 тактов вместо 14).

Чтобы уменьшить затраты на БРП можно предложить реализовать (при дальнейшем развитии CPV) специальные режимы так называемой бит-реверсной автоинкрементной адресации, которыми обычно наделяют сопроцессоры, ориентированные на ускоренное исполнение БПФ.

Такой режим адресации, например, был предусмотрен в процессорах обработки сигналов DSP96002 фирмы Motorolla (см.[7], п.3.5, с.235-250), а также осуществлён в сопроцессоре CP2 микропроцессора К128РИО (1890ВМ7) семейства КОМДИВ (см.[4], с.130). Использование такого бит-реверсного режима адресации позволяет выборку очередных элементов вектора из памяти в регистры осуществлять в соответствии с требуемым бит-реверсным порядком, исключая тем самым всякую необходимость в предварительной перестановке элементов вектора.

Поскольку первую часть алгоритма БПФ, в которой выполняется БРП, не удаётся ускорить на CPV в той же степени, что и её вторую часть, в которой производятся вычисления бабочек Фурье, общий выигрыш в ускорении операции БПФ на CPV получится меньше. Насколько меньше? Чтобы оценить итоговый получаемый выигрыш, решим следующую задачу.

Задача Операция P состоит из двух частей $P_1$ и $P_2$: $P=P_1+P_2$, временные доли исполнения которых $d_1$ и $d_2$: $T=T_1+T_2$, $d_1=T_1/T$, $d_2=T_2/T$. Вопрос: если исполнять часть $P_1$ в $k_1$ раз, а часть $P_2$ — в $k_2$ раз быстрее, то во сколько раз (k) ускорится исполнение всей операции ?

Решение Выразим новые времена исполнения для $P_1$, $P_2$, P: $T_1'=T_1/k_1=T\cdot d_1/k_1$, $T_2'=T_2/k_2=T\cdot d_2/k_2$, $T'=T_1'+T_2'$,

$$T'= T\cdot d_1/k_1 + T\cdot d_2/k_2 = T\cdot(d_1\cdot k_2+ d_2\cdot k_1)/(k_1\cdot k_2).$$

Тогда коэффициент k ускорения операции P, равный отношению $k=T/T'$, должен вычисляться по формуле:

$$k= (k_1\cdot k_2) / (d_1\cdot k_2+ d_2\cdot k_1) \qquad \{\ \Phi 1\ \}.$$

А в том случае, когда часть $P_1$ в новом исполнении исчезает совсем ($k_1\to\infty$), вычисление коэффициента k можно упростить:

$T_1'=0$, $T'=T\cdot d_2/k_2$, значит, $k=k_2/d_2$ { Ф2 } .

Используя эти формулы, оценим, какой можно ожидать итоговый выигрыш в ускорении операции БПФ на CPV. При исполнении на CP1 на долю БРП (1-ой части БПФ) затрачивается примерно 5% времени всей операции БПФ (это установлено экспериментально). Значит, для операции БПФ доли её частей будут соответственно: $d_1=0.05$, $d_2=0.95$. Коэффициент ускорения на CPV для 1-ой части $k_1=1.75$, для 2-ой части БПФ-32 — $k_2=22.5$, а для БПФ длиной N>32 — $k_2=16$.

Для операции БПФ длиной N>32 вычисляем итоговый коэффициент ускорения на CPV по формуле Ф1:

$$K_N=(1.75\cdot 16)/(0.05\cdot 16+0.95\cdot 1.75)=$$
$$28/(0.8+1.6625)= 28/2.4625 \approx \mathbf{11.37}.$$

Для БПФ-32 — по формуле Ф2: $K_{32}=22.5/0.95 \approx \mathbf{23.68}$.

# 6. Результаты ускорения на CPV операции БПФ

Коэффициенты возможного ускорения операций БПФ на CPV, выявленные теоретически в ходе проведённого анализа, требовалось проверить на практике. Для этого были составлены специальные программы — тесты, в которых каждую операцию БПФ над вектором заданной длины $N=2^P$ предусматривалось выполнить дважды. Один раз с помощью применения обычного процессора плавающей арифметики CP1 путём вызова функции, составленной на языке Си (по алгоритмам, описанным в п.2 и п.5.3). А другой раз с помощью вызовов одной (как в п.5.1 для БПФ-32) или нескольких ассемблерных функций (как в п.5.2 для БПФ длины N>32), в которых применяются команды векторного сопроцессора.

Тесты прогонялись многократно для векторов различной длины N=32,64,..,4096 (N фиксировалось заранее перед компиляцией программы). Прогоны тестов выполнялись на доступной в настоящее время версии RTL-модели (от 02.09.2015) создаваемого микропроцессора 1890ВМ8 семейства КОМДИВ. В каждом тесте непосредственно перед вызовом самой операции

БПФ её программный код и обрабатываемые данные помещались в кэш-память. Время выполнения операции БПФ в каждом случае оценивалось по числу тактов, затраченных на её исполнение.

## 6.1. Результаты ускорения БПФ для DC

Результаты прогонов тестовых программ и полученные в итоге коэффициенты ускорения операции БПФ для вектора комплексных величин двойной точности (DC – Double Complex) представлены в таблице 3. Её столбец **KV** характеризует коэффициент выигрыша, который показывает, во сколько раз операция БПФ для DC исполняется быстрее при её реализации с применением векторного сопроцессора CPV.

**Таблица 3.** Ускорение на CPV операции БПФ для DC

| N | BFN | %L2(L1) | CP1 | CPV | KV | KE% |
|---|---|---|---|---|---|---|
| 32 | 80 | 0.12 (3.71) | 3385 | 168 | **20.15** | **47.62** |
| 64 | 192 | 0.24 (7.62) | 7675 | 829 | **9.26** | **23.16** |
| 128 | 448 | 0.48 (15.23) | 17161 | 1899 | **9.04** | **23.59** |
| 256 | 1024 | 0.96 (30.86) | 38252 | 4397 | **8.7** | **23.29** |
| 512 | 2304 | 1.93 (61.72) | 84454 | 10013 | **8.43** | **23.01** |
| 1024 | 5120 | 3.88 (124.22) | 190977 | 23716 | **8.05** | **21.59** |
| 2048 | 11264 | 7.76 (248.44) | 466638 | 55620 | **8.39** | **20.25** |
| 4096 | 24576 | 15.58 (498.44) | 1012344 | 125089 | **8.09** | **19.65** |

**BFN** – количество вычисляемых бабочек Фурье
**%L2(L1)** – доля заполненности L2(L1)-кэша (в процентах %)
**CP1** – число тактов, затраченных на операцию БПФ на CP1
**CPV** – число тактов, затраченных на операцию БПФ на CPV
**KV** – коэффициент выигрыша или ускорения, который CPV обеспечивает для БПФ по сравнению с CP1 (= такты **CP1**/ такты **CPV**)
**KE** – коэффициент эффективности использования CPV (в %), равный отношению полученной производительности к пиковой

Полученные экспериментальным путём результаты показывают, что уже сейчас с применением CPV удаётся реально в значительной степени ускорить операцию БПФ для DC (почти в **20** раз операцию БПФ-32 и в **8-9** раз операцию БПФ длины N>32). Но эти коэффициенты ускорения чуть-чуть уступают тем (**23.68** и **11.37**), которые были получены нами в результате теоретического анализа. Чем же можно объяснить эти расхождения?

**Таблица 4.** Ускорение на CPV частей БПФ для DC

| N | | **CP1** (%доля) | **CPV** (%доля) | **KV** |
|---|---|---|---|---|
| 32 | ч1 | 200 (5.85) | 0 (0.0) | |
| | ч2 | 3157 (92.37) | 154 (76.62) | **20.64** |
| 64 | ч1 | 440 (5.74) | 281 (32.41) | **1.57** |
| | ч2 | 7162 (93.45) | 549 (63.32) | **13.05** |
| 128 | ч1 | 860 (5.03) | 509 (25.98) | **1.69** |
| | ч2 | 16165 (94.60) | 1413 (72.13) | **11.44** |
| 256 | ч1 | 1820 (4.78) | 1017 (32.87) | **1.79** |
| | ч2 | 36183 (95.06) | 3384 (62.81) | **10.69** |
| 512 | ч1 | 3621 (4.31) | 1978 (19.6) | **1.83** |
| | ч2 | 80304 (95.61) | 8078 (80.04) | **9.94** |
| 1024 | ч1 | 9273 (4.88) | 5458 (22.84) | **1.7** |
| | ч2 | 180701 (95.08) | 18392 (76.97) | **9.82** |
| 2048 | ч1 | 20422 (4.39) | 15083 (27.10) | **1.36** |
| | ч2 | 444191 (95.59) | 40516 (72.80) | **10.96** |
| 4096 | ч1 | 42006 (4.17) | 31209 (24.94) | **1.35** |
| | ч2 | 966272 (95.83) | 93860 (75.02) | **10.29** |

ч1 (часть 1) – выполнение Бит-Реверсной Перестановки (БРП)
ч2 (часть 2) – вычисление ($N \cdot \log_2 N/2$) бабочек Фурье
для каждой части представлено число тактов, затраченных на неё, и её процентная доля (в скобках) в общем времени исполнения БПФ

Для ответа на этот вопрос, посмотрим, во сколько раз на CPV удаётся ускорять отдельные части алгоритма БПФ и каковы их доли (в процентах) в общем объёме затрат (в тактах) на исполнение всей операции БПФ (см. таблицу 4).

Из этой таблицы видно, что вычисления бабочек Фурье реально удаётся ускорить на CPV лишь в **10-13** раз (вместо 16) для БПФ длины N>32 и в **20.6** раза (вместо 22.5) для БПФ-32, а операцию БПР — в 1.6-1.8 раза, т.е. почти, как и предполагалось (1.75). (Резкое ухудшение показателей БРП для N= 2048, 4096, вероятно, можно объяснить серьёзной пробуксовкой кэш-памяти 1-го уровня).

При более детальном анализе (см. таблицу 5) оказывается, что в части 2А операции БПФ удаётся достигать скорости вычислений 1.9-1.6 тактов на 1 бабочку и приблизиться к расчётной (1.2), а в части 2Б этого добиться не получается: скорость вычислений бабочек держится в районе 7.7 - 5.2 (вместо 2).

**Таблица 5.** Скорость вычисления бабочек для БПФ на CPV

| N | часть | число вычисляемых бабочек | количество тактов | **тактов на 1 бабочку** |
|---|---|---|---|---|
| 32 | ч2А | 80 | 154 | **1.93** |
| | ч2Б | | | |
| 64 | ч2А | 160 | 302 | **1.89** |
| | ч2Б | 32 | 247 | **7.72** |
| 128 | ч2А | 320 | 558 | **1.74** |
| | ч2Б | 128 | 855 | **6.68** |
| 256 | ч2А | 640 | 1078 | **1.68** |
| | ч2Б | 384 | 2306 | **6.00** |
| 512 | ч2А | 1280 | 2106 | **1.64** |
| | ч2Б | 1024 | 5972 | **5.83** |
| 1024 | ч2А | 2560 | 4157 | **1.62** |
| | ч2Б | 2560 | 14235 | **5.56** |
| 2048 | ч2А | 5120 | 8256 | **1.61** |
| | ч2Б | 6144 | 32260 | **5.25** |
| 4096 | ч2А | 10240 | 16448 | **1.60** |
| | ч2Б | 14336 | 77412 | **5.40** |

Прогнав тесты в специальном режиме, можем узнать, сколько реально тактов затрачивается сейчас на исполнение ключевых процедур при прогоне тестов на RTL-модели имеющейся версии (см. таблицу 6).

**Таблица 6.** Время исполнения основных процедур БПФ

| часть | процедура | расчётное | реальное |
|---|---|---|---|
| БПФ32 | cp3_VFFT32 | 110 | 139 |
| БПФ ч2А | cp3_DoVFFT32 | 95 | 122 |
| БПФ ч2Б | cp3_VFFT_Do16BF | 32 | 60 |

Сравнивая эти реальные показатели с теми, которые в ходе проведённого нами теоретического анализа были рассчитаны на основе описанных в п.1 возможностей CPV, обнаруживаем, что они немного уступают расчётным (особенно для процедуры cp3_VFFT_Do16BF части 2Б БПФ).

Причиной тому, как выясняется, являются ограничения, которые используемая в настоящее время RTL-модель накладывает на групповое исполнение команд (vldq, vsdq), обращающихся с кэш-памятью 2-го уровня. Оказывается, что после исполнения подряд 4-х таких команд следующие такие же команды смогут начать своё исполнение только после некоторой весьма ощутимой задержки.

Таким образом, различия между теоретическими оценками и полученными на практике коэффициентами ускорения операции БПФ (для DC) вызваны выявленными ограничениями в исполнении команд доступа к кэш-памяти 2-го уровня

### 6.2. Результаты ускорения БПФ для SC

До сих пор анализировалась возможность ускорения на CPV процедуры БПФ и операции бабочки Фурье для комплексных величин двойной точности (DC). При обработке комплексных величин одинарной точности (SC – Single Complex) одна команда cmaddsub.s исполняет сразу две бабочки Фурье (одну для величин из левых половин, а другую для величин из правых половин регистров). Даёт ли это возможность ускорить БПФ для SC в **2** раза более, чем для DC?

**Таблица 7.** Ускорение на CPV операции БПФ для SC

| N | BFN | %L2(L1) | CP1 | CPV | KV | KE% |
|---|---|---|---|---|---|---|
| 32 | 80 | 0.07(2.15) | 3296 | 152 | **21.68** | **26.32** |
| 64 | 192 | 0.14 (4.49) | 7524 | 715 | **10.52** | **13.43** |
| 128 | 448 | 0.28 (8.98) | 16771 | 1546 | **10.85** | **14.49** |
| 256 | 1024 | 0.57 (18.36) | 37289 | 3408 | **10.94** | **15.02** |
| 512 | 2304 | 1.15 (36.72) | 82183 | 7378 | **11.14** | **15.61** |
| 1024 | 5120 | 2.32 (74.22) | 180176 | 16215 | **11.11** | **15.79** |
| 2048 | 11264 | 4.64 (148.44) | 400610 | 38417 | **10.43** | **14.66** |
| 4096 | 24576 | 9.33 (298.44) | 954779 | 90118 | **10.59** | **13.64** |

Коэффициенты ускорения операции БПФ для SC, полученные экспериментальным путём, демонстрируются в таблице 7. Как видно из этой таблицы, выигрыш в ускорении оказывается не столь уж большим (Он всё же выше, чем для аналогичных операций БПФ для DC, но не в 2 раза, как спрашивалось в поставленном выше вопросе).

Это объясняется тем, что для использования команды cmaddsub.s приходится тратить дополнительные команды для упаковки в один регистр двух величин, прочитанных из разных элементов вектора (при их выборке из памяти в бит-реверсном порядке) или из разных регистров (после 1-го этапа вычислений бабочек Фурье).

## Заключение

Проведённый анализ и полученный практический опыт разработки ряда БПФ-процедур для имеющегося варианта векторного сопроцессора, реализованного в составе микропроцессора 1890ВМ8 семейства КОМДИВ, позволяют сделать вывод о возможности его применения для ускорения операций быстрого преобразования Фурье над векторами комплексных чисел, непрерывно расположенных в памяти.

Данный векторный сопроцессор CPV реально обеспечивает ускорение (по отношению к CP1) в **8-9** раз для операций БПФ с непрерывными векторами длиной $N=2^P>32$ и в **20** раз для БПФ с векторами длиной 32 (если они уже в кэш-памяти).

# About possibility of use of the vector coprocessor for acceleration of operation of fast Fourier transformation

**A.A.Burtsev**

**Abstract**. In Scientific Research Institute for System Analysis of the Russian Academy of Sciences (SRISA RAS) as expansion of universal microprocessors of the KOMDIV family the specialized 128-digit coprocessor allowing to accelerate calculations over vectors of complex and real numbers of unary and double accuracy has been created. The article presents results of use of this coprocessor directed on increase of the execution speed of the operation of the fast Fourier transformation (FFT) which is one of the main in tasks of digital signal processing.

**Keywords:** microprocessors of the KOMDIV family, vector coprocessor, digital signal processing, operation FFT.

## Литература

1. В.Б. Бетелин. Отечественные суперкомпьютерные технологии экзафлопсного класса – необходимое условие обеспечения технологической конкурентноспособности России в XXI веке. // Программные продукты и системы. 2013. №4 (104). С.4-9.

2. Методы встречной оптимизации в задачах обработки сигналов. / Сб. под ред. академика В.Б. Бетелина М.: Изд-во НИИСИ РАН, 2005. 130 с.

3. А.А. Бурцев. О возможности оптимизации некоторых функций библиотеки линейной алгебры с помощью векторного сопроцессора // Труды НИИСИ РАН, т.4 №2. М.: Изд-во НИИСИ РАН, 2014. с.5-15.

4. О.Ю. Сударева. Эффективная реализация алгоритмов быстрого преобразования Фурье и свёртки на микропроцессоре КОМДИВ128-РИО. // под ред. академика В.Б. Бетелина М.: НИИСИ РАН, 2014. 266 с.

5. Стивен Смит. Цифровая обработка сигналов. Практическое руководство для инженеров и научных работников / Стивен Смит; пер. с англ. А.Ю. Линовича, С.В. Витязева, И.С. Гусинского. — 720 с.

6. Быстрое преобразование Фурье. URL: http://psi-logic.shadanakar.org/fft/fft.htm (дата обращения: 02.09.2015).

7. В.В. Корнеев, А.В. Киселёв. Современные микропроцессоры. М.: НОЛИДЖ, 2000. 320с.

# Поддержка интерфейса MPI в ядре ОС Linux для многопроцессорных модулей на базе высокоскоростных каналов RapidIO

**А.С. Кулешов**

**Аннотация.** В статье приводятся результаты работы типовых тестов NAS Parallel Benchmarks и синтетических тестов NetPIPE и OSU Micro-Benchmarks на вычислительном комплексе из нескольких микросхем 1890ВМ6Я с использованием протокола MPI поверх высокоскоростных каналов RapidIO. Приведены показатели скорости передачи данных между двумя узлами и задержки в системе из 8 узлов, а также сравнение результатов в зависимости от частоты работы процессора.

**Ключевые слова.** 1890ВМ6Я, MPI, RapidIO, NAS Parallel Benchmarks, OSU Micro-Benchmarks.

## 1. Введение

В НИИСИ РАН несколько лет назад была разработана микросхема 1890ВМ6Я – система на кристалле, состоящая из 64-х разрядного ядра с архитектурой MIPS, системного контроллера, поддерживающего механизм DMA, традиционные контроллеры прерываний, PCI и Ethernet (100 Мбит), а также контроллер высокоскоростного интерфейса стандарта RapidIO. Также была разработана микросхема 8-портового коммутатора RapidIO [1], [2]. Отечественные многопроцессорные комплексы с использованием этих микросхем нашли своё применение для решения специализированных прикладных задач под управлением различных операционных систем.

В работе [3] приведены результаты автора по расширению области применения такого рода комплексов путём установки на них рабочего окружения общего типа — ядра ОС общего назначения Linux и стандартного протокола MPI v1.1 [4]. Для этого в реализацию стандарта MPI библиотеку MPICH [5] и ядро ОС Linux была добавлена поддержка контроллера сообщений RapidIO микросхемы 1890ВМ6Я с возможностью передачи некоторого класса данных без копирования на процессоре (*zero-copy*). Установка на отечественных многопроцессорных комплексах рабочего окружения общего типа упрощает разработку прикладного ПО, но остаются открытыми вопросы об эффективности специализированного ПО, работающего в таком окружении и возможности эффективного исполнения в таком окружении типовых параллельных вычислительных алгоритмов.

В попытке ответа на эти вопросы автор провёл измерение производительности ряда параллельных вычислительных алгоритмов на вычислительном комплексе из 8 микросхем 1890ВМ6Я, соединённых параллельными каналами RapidIO пиковой пропускной способностью 500 + 500 МБ/с через 8-портовый коммутатор 1890КП3Я. Для специальных приложений, создающихся с нуля, пределы эффективности задаются, в основном, двумя характеристиками: скоростью передачи и задержками. Эти две характеристики измерялись на двух специализированных синтетических тестах, результаты которых описаны в разделе 2.

Эффективность на типовых параллельных вычислительных алгоритмах измерялась на тестах NAS Parallel Benchmarks класса A [6]. Результаты измерений показывают, что приведённая в [3] реализация протокола MPI для RapidIO достаточно эффективна, что позволяет использовать отечественные многопроцессорные комплексы на базе микросхем 1890ВМ6Я и 1890КП3Я под управлением ОС Linux. При этом большинство параллельных вычислительных алгоритмов NAS Parallel Benchmarks класса A на комплексах описанной архитектуры демонстрируют хорошую масштабируемость.

## 2. Измерение пропускной способности и задержек коммуникационной системы на синтетических тестах

Для измерения двух описанных показателей отечественного 8-процессорного комплекса на базе микросхемы 1890ВМ6Я были использованы специализированные синтетические тесты: NetPIPE [7] и OSU Micro-Benchmarks [8].

### 2.1. Тест NetPIPE

Тест NetPIPE измеряет пропускную способность канала и задержки по следующему упрощённому алгоритму, рассматривая только тривиальный случай однонаправленной передачи данных. В тесте участвуют только два узла. Один узел является отправителем, другой — получателем. Отправитель засекает время и затем циклически (достаточно большое количество раз, исключающее влияние посторонних программно-аппаратных факторов) совершает единичную отправку данных определённого размера (от 1 байта до 8 МБ) с

последующим ожиданием завершения каждой отправки. При выходе из цикла снова засекается время. Такой эксперимент проводится несколько раз, и на основе разностей временных отсчётов в каждом эксперименте выводятся характеристики скорости передачи данных и задержек. Скорость вычисляется по формуле *размер / задержка*, где *задержка* — это минимальная разность времени среди всех экспериментов для данного размера. NetPIPE способен производить такого рода замеры для различных коммуникационных интерфейсов: tcp (стек Ethernet TCP/IP), mpi, mx (интерфейс Myrinet eXpress), ib (InfiniBand) и другие.

В нашем случае сначала использовался тест NetPIPE для измерения производительности интерфейса mx, реализованного автором в [3]. Ниже на графиках приведены результаты измерений для двух микросхем 1890ВМ6Я, связанных по параллельному каналу RapidIO. Пунктирной линией изображены результаты при частоте ядра 200 МГц, сплошной — 270 МГц.

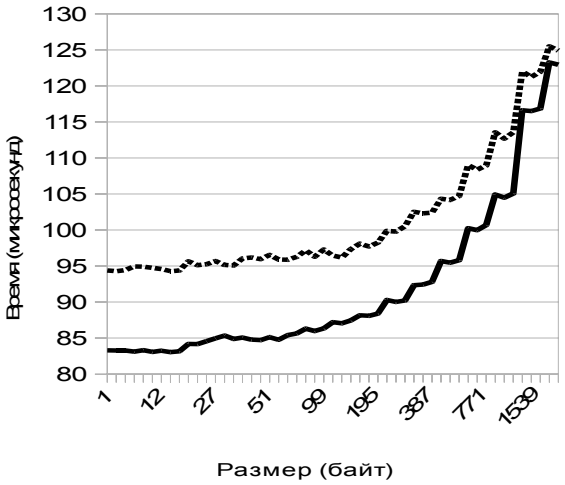



**Рис. 1.** Задержки для размера данных до 2КБ.

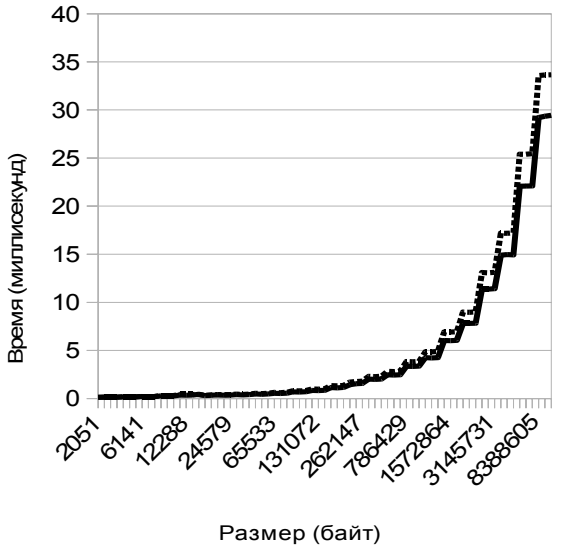



**Рис. 2.** Задержки для данных больше 2КБ.

Как видно из первого графика на рисунке 1, частота процессора существенно влияет на «задержку»: для малых размеров передаваемых данных увеличение частоты на 70 МГц даёт уменьшение задержки на 11 микросекунд — 94 микросекунд (мкс) против 83 мкс. С ростом размера передаваемых данных разрыв сокращается и для данных размером 8 МБ составляет меньше 5 мкс.

На рисунке 3 изображён график с соответствующими показателями скорости. Для данных достаточно большого размера скорость передачи на частоте ядра 270 МГц на 10% выше скорости на частоте 200 МГц: для данных размером 8 МБ имеем скорость 2190 Мбит/сек против 1906 Мбит/сек.

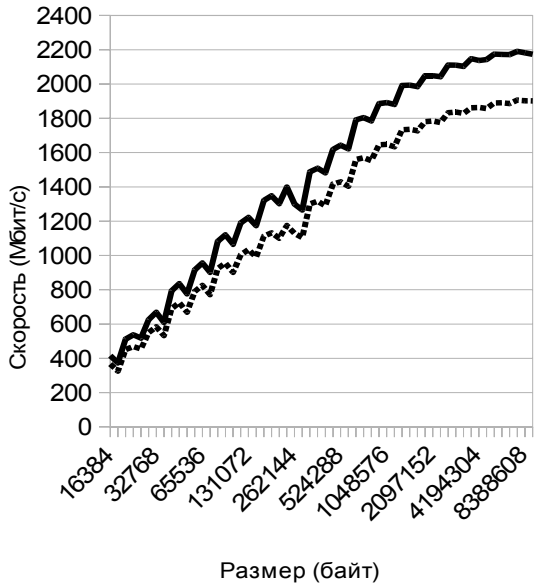



**Рис. 3.** Скорость передачи данных между двумя узлами.

Далее были проведены измерения производительности для интерфейса MPI. В этом случае NetPIPE показал минимальную задержку в 156 мкс на 270 МГц и 172 мкс на 200 МГц. Максимальная скорость передачи составила 2160 Мбит/с и 1890 Мбит/с соответственно. С помощью получившихся задержек можно оценить накладные расходы использования интерфейса MPI. Задержка составила около 70 мкс — столько времени процессор потратил на работу библиотеки MPICH; пропускная способность, тем не менее, практически не изменилась.

## 2.2. Тест OSU Micro-Benchmarks

В отличие от теста NetPIPE, тест OSU Micro-Benchmarks работает по другому алгоритму. Он пытается достичь максимальной эффективности работы всего стека передачи данных и для этого выставляет целый ряд запросов либо только на передачу, либо на приём данных, и только после выставления всех запросов ожидает их завершения.

Пакет тестов OSU Micro-Benchmarks содержит в себе тесты производительности,

написанные с использованием интерфейса MPI разных классов: pt2pt, collective, one-sided. Класс pt2pt содержит тесты, измеряющие производительность между двумя узлами MPI. Эти тесты измеряют пропускную способность передачи сообщений между двумя узлами в одну сторону, в обе одновременно, и измеряют задержки. Класс collective – измерение производительности MPI операций Alltoall, Reduce, Barrier, Scatter, Gather, Broadcast на множестве узлов. Класс one-sided – измерение односторонних коммуникаций чтения/записи памяти удалённого узла, в который входят тесты, измеряющие пропускную способность в одну сторону (скорость чтения или записи памяти удалённого узла), в обе стороны (запись с обоих узлов в удалённые памяти) и задержки доступа в память удалённого узла.

Операция Barrier используется для синхронизации группы узлов. Выполнение программы на узлах останавливается на точке выполнения операции Barrier, пока все остальные узлы в группе не дойдут до этой операции.

Операция Alltoall используется для распределения данных между узлами, где каждый узел рассылает всем узлам свои данные и принимает ото всех других узлов соответствующие им данные. Например, такая операция необходима для транспонирования матрицы, распределённой по узлам вычислительной системы (рис. 4, здесь и далее серым прямоугольником обозначен узел, белые квадраты — порция данных на узле).

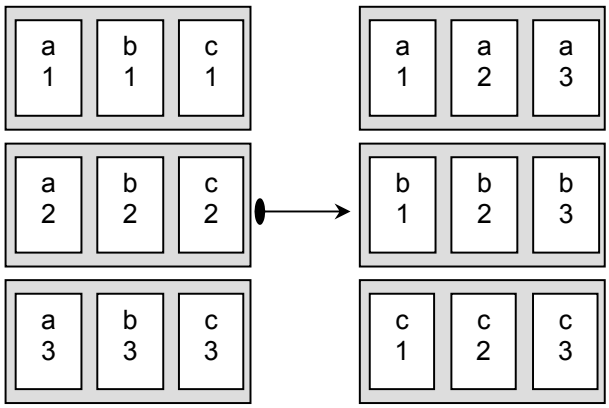



**Рис. 4.** Операция Alltoall.

Операция Reduce выполняет сбор данных с группы указанных узлов и совершает над ними указанную операцию, например, сложить, вычесть, умножить, произвести логическое И, ИЛИ, или воспользоваться некой частной определённой функцией. Результат выполнения операции над группой данных возвращается либо определённому узлу, либо всем (операция Allreduce).

Операция Scatter распределяет массив данных от одного узла по остальным узлам в группе (рис. 5).

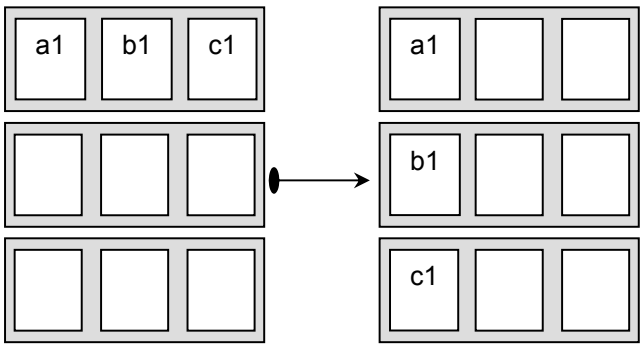



**Рис. 5.** Результат операции Scatter.

Операция Gather делает операцию, обратную Scatter — сбор данных со всех узлов на один узел (рис. 6).

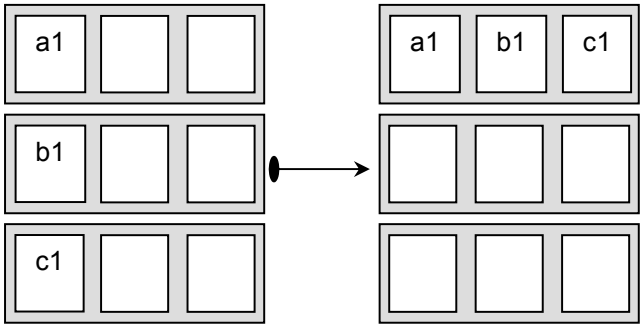



**Рис. 6.** Результат операции Gather.

Операция Broadcast рассылает данные одного узла по остальным узлам в группе (рис. 7).

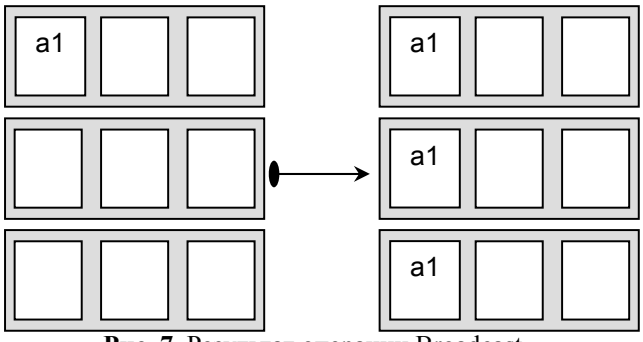



**Рис. 7.** Результат операции Broadcast.

Тест osu_bw из класса pt2pt, измеряет скорость однонаправленной передачи данных. На узле-отправителе выставляется ряд запросов на отправку данных, затем происходит ожидание их завершения. На узле-получателе выставляется ряд запросов на приём данных и затем также происходит ожидание их завершения. Время замеряется на узле-отправителе первый раз до выставления запросов, и второй раз после завершения ожидания всех операций. Для получения результатов, разность полученных временных отсчётов делится на размер передаваемых данных. На рисунке 8 приведён результат работы этого теста для частоты ядра 270 МГц (сплошная линия) и 200 МГц (пунктирная).

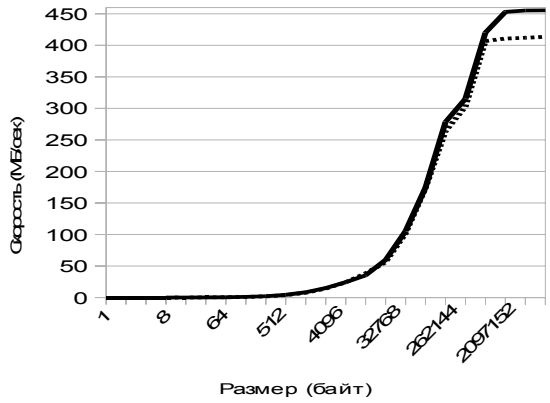



**Рис. 8.** Сравнение показателей скорости теста osu_bw для частот процессора 200 МГц и 270 МГц.

Начиная с размера 1 МБ, показатели скоростей вышли на свои максимумы и составили 413 МБ/с при частоте 200 МГц и 455 МБ/с при 270 МГц — это 82,6% и 91% от пиковой односторонней пропускной способности канала RapidIO соответственно.

Тест osu_bibw из класса pt2pt, измеряет пропускную способность канала в обоих направлениях. На узле-отправителе выставляется ряд запросов на приём и ряд запросов на отправку, а затем происходит ожидание сначала всех запросов на отправку, потом на приём. На узле-получателе делается тоже самое. Время засекается на узле-отправителе сначала до выставления всех запросов, второй раз — после завершения ожидания всех запросов (на приём и отправку). Результатом является скорость передачи, вычисленная делением размера на разность полученных временных отсчётов. На рисунке 9 показано сравнение результатов этого теста с показателями из теста osu_bw только для частоты ядра 200 МГц, где сплошной линией изображён результат теста osu_bibw, а пунктирной — osu_bw.

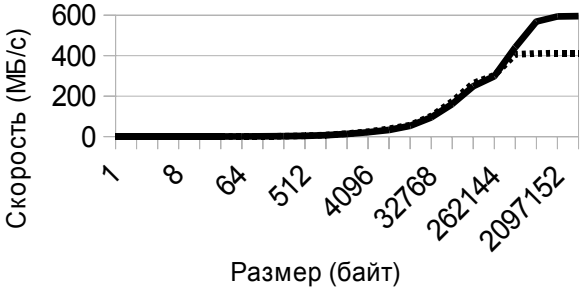



**Рис. 9.** Сравнение результатов тестов osu_bw (пунктирной линией) и osu_bibw (сплошной линией).

Скорость двунаправленной передачи данных вышла на свой максимум 595 МБ/с, начиная с размера 2 МБ.

Несмотря на реализацию поддержки в стеке драйверов RapidIO только стандарта MPI v1.1, операции удалённого доступа к памяти в библиотеке MPICH (из стандарта MPI v2) реализованы через эмуляцию передачи обычного рода сообщений. Из-за такой эмуляции скорость передачи данных в односторонних операциях (класс one-sided) оказалась посредственной. Максимальная скорость передачи составила 37 МБ/сек, а показатели задержки начинались с 450 микросекунд для 1 байта данных и увеличивались с ростом объёма данных до нескольких десятков миллисекунд для размера 4МБ.

Тест osu_barrier принадлежит классу collective и проводит измерение времени выполнения операции Barrier. Время засекается до вызова операции и после её завершения на каждом узле. Затем со всех узлов это время собирается (Reduce с операцией сложения) и делится на количество узлов. Тест запускался на комплексе из 8-ми узлов на частоте процессоров 270 МГц.

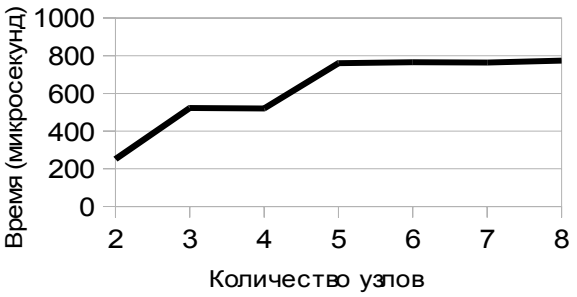



**Рис. 10.** Результат теста osu_barrier для разного количества узлов.

По результатам теста osu_barrier, приведённым на рисунке 10, время синхронизации 8-ми узлов составило немногим меньше 800 микросекунд. С увеличением числа узлов от 2 до 5 время синхронизации росло с 251 мкс до 760 мкс. Затем, рост количества узлов не внёс значительного ухудшения в показатели задержки.

Все остальные тесты из класса collective показали схожие друг с другом результаты: для двух узлов минимальная задержка составила 250 мкс, и, увеличиваясь с ростом объёма данных, достигла при передаче больше 1 МБ данных несколько десятков миллисекунд.

## 3. Тесты NAS Parallel Benchmarks

Показатели синтетических тестов NetPIPE и OSU Micro-Benchmarks позволяют дать только приближенную оценку реальной картине и отвечают на вопрос: что можно ожидать от системы в идеальных условиях. На приближенных к реальным вычислительных задачах результаты могут оказаться другими. Тесты NAS Parallel Benchmarks (NPB) представляют собой описание вычислительных алгоритмов для распределённых систем, которые направлены на снятие показателей производительности узлов в условиях, приближенных к реальным. Результаты тестов представляются в виде единицы измерения «Миллион Операций в секунду» (МОп/с). В распределённый пакет тестов NPB входят тесты CG, MG, IS, FT, LU. Эти тесты могут быть запущены на комплексах из 8-ми узлов. Каждый тест реализует свой вычислительный метод: CG – метод сопряжённых градиентов, MG – многосеточный метод, IS – сортировка целых чисел, FT –

преобразование Фурье, LU – LU-разложение матрицы. Для каждого теста возможно задание размера данных и числа итераций — в NPB обозначается как «класс» (S, W, A, B, C, D, E, F). Ниже приведены результаты запусков этих тестов класса A на вычислительном комплексе из 8-ми узлов, работающих на частоте ядра 270 МГц, где пунктирной линией показан график идеальной масштабируемости (по формуле *МОп/с на одном узле * количество узлов*), а сплошной — фактической.

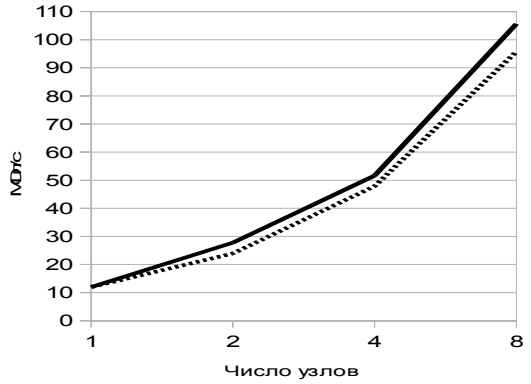



**Рис. 11.** Результаты теста CG.

На рисунке 11 тест CG фактически превзошёл показатели идеальной масштабируемости, что означает, что эталонный тест CG класса A не раскрыл потенциала одного процессора.

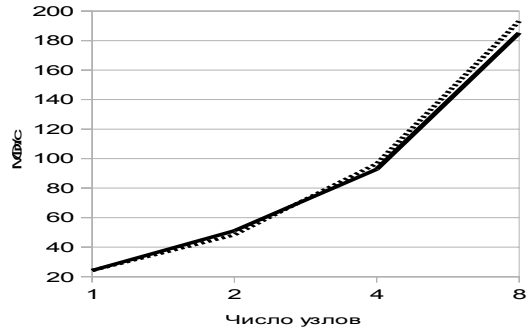



**Рис. 12.** Результаты теста MG.

На рисунке 12 представлен результат теста MG. Начиная с 4-х узлов, фактический показатель начал отставать от идеального с разрывом в 10 МОп/с.

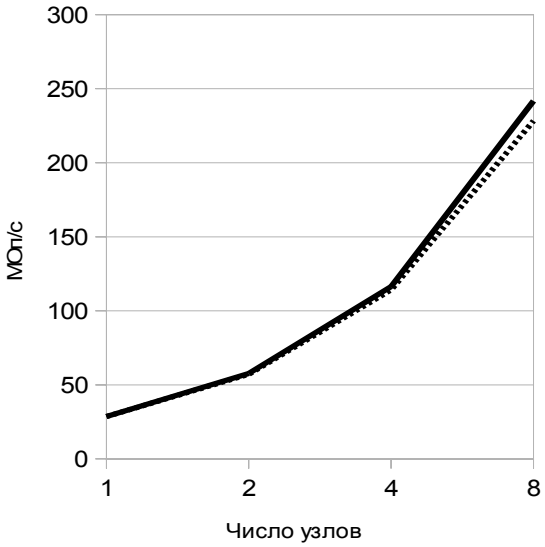



**Рис. 13.** Результаты теста LU.

Тест LU на рисунке 13, так же как и CG, в итоге превзошёл показатель идеальной кривой.

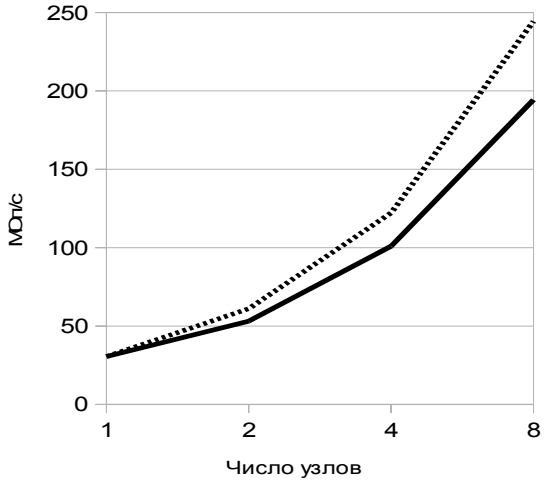



**Рис. 14.** Результаты теста FT.

На рисунке 14 приведён результат работы теста FT. Как видно из графика, фактический результат достиг отставания от идеального на 50 МОп/с, то есть примерно на 20%.

## 4. Заключение

Поддержка стандарта MPI для высокоскоростных каналов RapidIO позволяет использовать высокопроизводительные вычислительные многопроцессорные комплексы, построенные на отечественной элементной базе, для решения широкого класса математических задач с помощью хорошо апробированных параллельных вычислительных алгоритмов. Протестированы и верифицированы разного рода MPI задачи, а все запускаемые тесты из пакетов OSU и NPB показали хороший коэффициент масштабируемости, за исключением теста NPB FT, на котором масштабируемость всего лишь удовлетворительна.

Для улучшения полученных показателей, необходимо проведение дополнительных исследований и продолжение развития текущих наработок в стиле *co-design* [9], где разработка программного стека и аппаратных компонент ведётся соответствующими разработчиками совместно и нацелена на решение проблем с учётом технологических возможностей обеих сторон.

# MPI support in Linux kernel for multiproprocessor systems with high-speed RapidIO interconnect

**A.S.Kuleshov**

**Abstract.** The paper presents results of standard tests NAS Parallel Benchmarks and synthetic tests NetPIPE and OSU Micro-Benchmarks on computing system consisted of several 1890VM6IA chips using MPI protocol with high-speed RapidIO channels support. Also the paper presents results of bandwidth for two nodes and latency for 8 nodes, and result comparison depending on CPU frequency.

**Keywords.** 1890VM6IA, MPI, RapidIO, NAS Parallel Benchmarks, OSU Micro-Benchmarks.

## Литература

1. С.Г.Бобков, А.А.Ерёмин, Н.В.Кондратьева, О.В.Сердин. Разработка высоконадежных многопроцессорных модулей на базе высокоскоростных каналов RapidIO // Программные продукты и системы. 2013. № 4. с. 49–55.
2. RapidIO interconnect specification, RapidIO Trade Association. http://www.rapidio.org, 2002
3. А.С.Кулешов. Поддержка протокола MPI в ядре ОС Linux для многопроцессорных вычислительных комплексов на базе высокоскоростных каналов RapidIO // Программные продукты и системы. 2015. № 4.
4. Message Passing Interface Forum. MPI: A Message Passing Interface Standard, *http://www.mpi-forum.org*
5. William Gropp, et al. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel computing* 22.6 (1996): с. 789-828.
6. D. Bailey, J. Barton, T. Lasinski, and H. Simon (Eds.). The NAS Parallel Benchmarks. NAS Technical Report RNR-91-002, NASA Ames Research Center, Moffett Field, CA, 1991
7. Quinn O.Snell, Armin R. Mikler, and John L. Gustafson. Netpipe: A network protocol independent performance evaluator. IASTED International Conference on Intelligent Information Management and Systems. Vol. 6. 1996.
8. Micro-Benchmarks OSU. OSU Network-Based Computing Laboratory. URL: http://mvapich.cse.ohio-state.edu/benchmarks.
9. G. Shainer, T. Wilde, P. Lui, T. Liu, M. Kagan, M. Dubman, Y. Shahar, R. Graham, P. Shamis, and S. Poole. The co-design architecture for exascale systems, a novel approach for scalable designs. Computer Science-Research and Development, May 2012