НИЦ «КУРЧАТОВСКИЙ ИНСТИТУТ» НИИСИ

Том 15 N 3
ISSN 3033-6422

Труды НИИСИ

SRISA PROCEEDINGS

Москва, 2025

Федеральное государственное автономное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований
Национального исследовательского центра «Курчатовский институт»
(НИЦ «Курчатовский институт» — НИИСИ)

ТРУДЫ НИИСИ

SRISA PROCEEDINGS

ТОМ 15 № 3

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2025

Учредитель и издатель
Федеральное государственное автономное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт» (НИЦ «Курчатовский институт» — НИИСИ)

«Труды НИИСИ» — это рецензируемый научный журнал, в котором публикуются научные статьи по следующим специальностям и отраслям наук:

- 1.2.1. «Искусственный интеллект и машинное обучение» (физико-математические науки);
- 2.3.1. «Системный анализ, управление и обработка информации, статистика» (физико-математические и технические науки);
- 2.3.2. «Вычислительные системы и их элементы» (технические науки);
- 2.3.3. «Автоматизация и управление технологическими процессами и производствами» (технические науки);
- 2.3.5. «Математическое и программное обеспечение вычислительных систем , комплексов и компьютерных сетей» (физико-математические и технические науки).

Направления исследований, по которым журнал публикует оригинальные статьи определены, но не ограничены следующим перечнем: системный анализ; математика; математическое и компьютерное моделирование; задачи автоматизации и управления; обработка сигналов; компьютерное зрение и обработка изображений; распознавание образов; статистика; технологии искусственного интеллекта; информационные технологии; информационная безопасность; вычислительные системы и их элементы; микро- и наноэлектроника; высокопроизводительные вычисления; вопросы численного анализа; нейроморфные и мягкие вычисления; оптико-нейронные технологии; история науки, техники и персоналий. Журнал предназначен для научных сотрудников, инженеров и аспирантов, работающих в указанных направлениях исследований.

Миссия журнала — развитие перечисленных научных направлений в России и за рубежом, включая широкое освещение результатов исследований и обеспечение высококвалифицированных кадров печатными площадями, обеспечение высокого качества исследований путем развития механизма профессионального и общественного обсуждения научных результатов и воспитания молодого поколения ученых-исследователей.

Политика журнала ориентирована на пропаганду передовых научно-технических идей и решений в рамках развития важнейших наукоемких технологий и участия в реализации приоритетов научно-технологического развития Российской Федерации. До 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН» (ISSN 2225-7349). В настоящее время сетевому рецензируемому научному журналу «Труды НИИСИ» присвоен ISSN 3033-6422.

Founder and Publisher

Scientific Research Institute for System Analysis of the National Research Center "Kurchatov Institute" (NRC "Kurchatov Institute" - SRISA)

SRISA Proceedings is a peer-reviewed journal that covers the following key areas of research:

- Artificial intelligence and machine learning
- System analysis, control, information processing, statistics
- Computing systems and their components
- Automation and control in manufacturing
- Mathematical and software support for computing systems, complexes and computer networks .

We publish original articles on topics including, but not limited to: system analysis; mathematics; computer simulation; automation and control; signal processing; computer vision and image processing; pattern recognition; statistics; artificial intelligence; information technologies; cybersecurity; computing

systems and their components; micro- and nanoelectronics; high-performance computing; numerical analysis; neuromorphic and soft computing; optoneural technologies; and the history of science, technology, and researchers. Our readers include researchers, engineers, and doctoral students.

Our mission is to advance these research areas in Russia and worldwide by publishing significant results and offering leading professionals a platform to share their work. We are committed to maintaining high research standards through professional and public review while fostering the next generation of researchers.

The journal's policy is to promote advanced research and innovative solutions, foster the development of high-tech fields, and contribute to key national priorities in science and technology. Until 2025, the journal was published in print as *Trudy NIISI RAN* (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349. Currently, *SRISA Proceedings* online peer-reviewed journal has been assigned ISSN 3033-6422.

Sergei I. Aryashev, PhD (Engineering), Deputy Director for Microelectronics and Computer Systems, NRC "Kurchatov Institute" - SRISA, Moscow
Alexander N. Godunov, PhD (Phys&Math), Senior Researcher, Head of the Department of System Programming, NRC "Kurchatov Institute" - SRISA, Moscow
Andrei I. Griuntal, PhD (Phys&Math), Head of the Mathematics Department, NRC "Kurchatov Institute" - SRISA, Moscow
Yakov M. Karandashev, PhD (Phys&Math), Leading Researcher, Center for Optic-Neural Technologies, NRC "Kurchatov Institute" - SRISA, Moscow
Anatoly G. Kushnirenko, PhD (Phys&Math), Assoc. Prof., Head of IT for Education Department, NRC "Kurchatov Institute" - SRISA, Moscow
Alexander N. Muranov, PhD (Engineering), Assoc. Prof., Head of the Manufacturing Automation and Control Laboratory, NRC "Kurchatov Institute" - SRISA, Moscow
Konstantin A. Petrov, PhD (Engineering), Senior Researcher, High Performance Microprocessor Architectures Department, NRC "Kurchatov Institute" - SRISA, Moscow
Ilya V. Semenov, PhD (Phys&Math), Leading Researcher, Department of Computational Mathematics, NRC "Kurchatov Institute" - SRISA, Moscow
Andrey I. Tsimbalov, PhD (Engineering), Deputy Director for Microtechnology, NRC "Kurchatov Institute" - SRISA, Moscow

Тематика номера:
Вычислительные системы, их элементы и программное и математическое обеспечение

The topic of the issue:
Computing Systems, Hardware Components and Software

СОДЕРЖАНИЕ

CONTENT

Формирование матрицы проводимостей с помощью кусочно-постоянных сигналов

Г. А. Бесхлебнова[1], В. Б. Котов[2]

[1] НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, beskhlebnova@niisi.ras.ru;

[2] НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, v1111111k1111@gmail.com

Аннотация. Для формирования матрицы проводимостей массива переменных резисторов необходима процедура произвольного изменения проводимостей резисторов массива при использовании ограниченного числа управляющих сигналов — напряжений на проводниках структуры типа кроссбар. Поскольку число проводников значительно меньше числа резисторов, такая процедура должна быть многошаговой. На каждом шаге происходит изменение проводимостей целевых резисторов, число которых не больше числа управляющих сигналов. При этом неизбежно меняются проводимости и некоторых нецелевых резисторов. Соответствующие изменения необходимо компенсировать. В работе рассмотрена процедура записи с использованием в качестве управляющих сигналов высокочастотных кусочно-постоянных сигналов. На основе анализа с использованием модели простого резисторного элемента показана возможность формирования произвольной (в известных пределах) матрицы проводимостей. На каждом шаге формируется (изменяется) строка или столбец матрицы. Обсуждаются условия, обеспечивающие выполнимость и удобство такой процедуры.

Ключевые слова: переменный резистор, кусочно-постоянный сигнал, резисторная матрица, матрица проводимостей

1. Введение

Большие резисторные матрицы могут стать основными блоками нейроморфных систем, поскольку они позволяют реализовать вектор-матричное умножение – наиболее трудоёмкую операцию нейровычислений. Элементы матрицы-множителя определяются проводимостями соответствующих резисторов [1]. Чтобы резисторный массив обеспечивал умножение на разные матрицы, резисторы должны быть переменными с возможностью управления проводимостями резисторов. Наибольшую популярность получили переменные резисторы, изменяющие своё сопротивление под действием протекающего тока («мемристоры») [2-4] благодаря своей простоте, компактности и энергоэффективности.

Для больших массивов подобных элементов формирование матрицы проводимостей – нетривиальная задача [1,5]. Дело в том, что организовать индивидуальный доступ к каждому резистору нереально из-за огромного числа элементов. В регулярных структурах типа кроссбара каждый резистор соединяет два проводника, к каждому из которых присоединено много резисторов. Число управляющих сигналов – напряжений на проводниках – много меньше числа резисторов. Поэтому необходимо пошаговое формирование матрицы проводимостей, причём на каждом шаге обычно формируется строка или столбец матрицы проводимостей. Использование для записи постоянных (точнее, однополярных) сигналов-напряжений встречает значительные трудности – при воздействии на целевые резисторы структуры изменяются проводимости и многих нецелевых резисторов. Знак этих нежелательных изменений разный для разных резисторов, из-за чего эти изменения трудно компенсировать. Часто эти трудности пытаются обойти, предполагая пороговый характер механизма изменения проводимости [2,3,6]. Однако на практике подобный трюк не работает – природа не всегда следует желаниям учёных.

Можно добиться легко компенсируемых изменений в нецелевых резисторах, если для записи информации использовать переменный (высокочастотный) сигнал [7]. В работах [8,9] были рассмотрены методы записи информации в массив переменных резисторов типа кроссбар с помощью высокочастотных гармонических сигналов. Показано, что для случая однонаправленных переменных резисторов использование знакопеременных гармонических напряжений в качестве управляющих сигналов позволяет записывать произвольную (в определённых пределах) матрицу проводимостей.

Помимо гармонических управляющих сигналов можно использовать другие периодические высокочастотные сигналы. С практической точки зрения такие сигналы должны быть достаточно простыми для формирования. Среди таких сигналов выделяются кусочно-постоянные сигналы,

которые могут иметь всего несколько значений. Применение именно таких сигналов для формирования матрицы проводимостей рассмотрено в настоящей статье. Предполагаем, что изменение состояния переменного резистора в результате воздействия одного периода управляющего напряжения невелико, для записи информации (изменения проводимостей резисторов) требуется много периодов сигнала. Такое предположение позволяет называть сигнал высокочастотным. Среди значений сигналов должны быть как положительные, так и отрицательные, в противном случае имеем однополярный сигнал, близкий по своему действию к постоянному сигналу. При анализе предполагаем, что переменный резистор описываются моделью простого резисторного элемента [10].

2. Уравнения записи

Проводимость G и сопротивление $R=1/G$ простого резисторного элемента выражается через единственную переменную состояния x: $G=G(x)$. Считаем, что переменная состояния меняется от 0 до 1. Состояние $x=0$ – основное состояние – это состояние с максимальным сопротивлением, а состояние $x=1$ соответствует минимальному сопротивлению резистора. Изменение переменной состояния описывается уравнением [10]

$$\frac{dx}{dt} = F(x,u), \qquad (1)$$

где u – напряжение на резисторе. Заметим, что в общем случае правая часть уравнения (1) зависит как от напряжения u, так и от тока резистора I. Однако закон Ома (справедливость которого подразумевает использование термина резистор)

$$u = R(x)I \qquad (2)$$

позволяет выразить I через u и x, что даёт уравнение (1).

Запишем функцию F () в виде

$$F(x,u) = F_0(x) + F^+(x,u) + F^-(x,u), \qquad (3)$$

где слагаемое $F_0(x) = F(x,u=0)$ описывает спонтанную (в отсутствие сигнала) релаксацию состояния резистора к основному состоянию, слагаемое $F^+(x,u) = \theta(u)(F(x,u) - F_0(x))$ описывает тенденцию увеличения переменной состояния x при положительном напряжении, а слагаемое $F^-(x,u) = \theta(-u)(F(x,u) - F_0(x))$ описывает тенденцию уменьшения переменной x (ускоренную релаксацию) при отрицательном напряжении. (Здесь $\theta(x)$ – функция Хевисайда).

Если характер зависимости величины $F^+(x,u)$ ($F^-(x,u)$) от x слабо зависит от u, можно провести факторизацию:

$$F^+(x,u) = F_x^+(x)F_u^+(u), F^-(x,u) = \\ = F_x^-(x)F_u^-(u), \qquad (4)$$

в результате получаем следующее представление функции F:

$$F(x,u) = F_0(x) + F_x^+(x)F_u^+(u) + \\ + F_x^-(x)F_u^-(u) \qquad (5)$$

Функции одной переменной $F_0(x), F_x^\pm(x), F_u^\pm(u)$ наряду с функцией $G(x)$ назовём характеристическими функциями резистора.

Характеристические функции $F_x^\pm(x), F_u^\pm(u)$ не задаются однозначно формулами (4). Будем считать, что функции $F_x^\pm(x)$ нормированы на единицу, то есть равны 1 в характерных точках. (в одной из граничных точек). Характеристические функции $F_0(x), F_x^\pm(x)$ определяют скорости записи/стирания в зависимости от текущего состояния резистора. Естественно принять, что они непрерывные на отрезке [0,1], положительные в интервале (0,1), и выполняются соотношения

$$F_0(0) = 0, F_x^+(0) = 1, F_x^+(1) = \\ = 0, F_x^-(0) = 0, F_x^-(1) = 1 \qquad (6)$$

Характерный пример таких функций:

$$F_0(x) = fx^{\gamma_0}(1-x)^{\gamma_1}, F_x^+(x) = \\ = (1-x)^{\beta_+}, F_x^-(x) = x^{\beta_-}, \qquad (7)$$

причём показатели γ_0, β_+, β_- положительные, а показатель γ_1 неотрицательный, f – положительный коэффициент. Заметим, что при $\gamma_1>0$ имеем $F_0(1)=0$, а при $\gamma_1=0$ $F_0(1)=1$.

Характеристические функции $F_u^+(u), F_u^-(u)$ задают зависимости скоростей изменения состояния x от напряжения на резисторе при положительных и отрицательных напряжениях соответственно. Наиболее типичный вид функций $F_u^\pm(u)$ – степенной на соответствующих полуосях:

$$F_u^+(u) = A_+\theta(u)u^{\alpha_+}, F_u^-(u) = \\ = -A_-\theta(-u)(-u)^{\alpha_-} \qquad (8)$$

с положительными коэффициентами $A_\pm$ и показателями $\alpha_\pm$.

Пусть на переменный резистор подается периодическое кусочно-постоянное напряжение $u(t)$. Обозначим u_k , $k=1,\ldots,K$ – значения напряжения, а T_k – времена действия соответствующих значений в течение одного периода T. Изменение состояния резистора за один период управляющего сигнала (напряжения) согласно уравнению (1) с учётом формулы (3) можно записать в виде

$$x(t+T) - x(t) = F_0(x)T + \sum_{k,u_k>0} F^+(x,u_k)T_k + \\ + \sum_{k,u_k<0} F^-(x,u_k)T_k. \qquad (9)$$

Нас интересуют медленные (усреднённые по периоду сигнала) изменения состояния резистора (малыми колебаниями переменной состояния внутри периода пренебрегаем из-за их малости при действии высокочастотного сигнала). Для таких изменений получаем

уравнение

$$\frac{dx}{dt} = \frac{x(t+T)-x(t)}{T} = F_0(x) + \\ + \sum_{k,u_k>0} F^+(x,u_k)\tau_k + \sum_{k,u_k<0} F^-(x,u_k)\tau_k \quad (10)$$

где $\tau_k = T_k/T$ – временная доля ненулевого значения напряжения u_k.

Уравнение (10) получено без использования предположения о факторизации (4). Напомним, что при использовании гармонического сигнала [7] факторизация необходима для получения рабочего уравнения. В рассматриваемом случае при условии факторизации уравнение записи (10) можно записать в виде

$$\frac{dx}{dt} = F_0(x) + F_x^+(x)M^+ + F_x^-(x)M^-, \quad (11)$$

где $M^+ = \sum_{k,u_k>0} F_u^+(u_k)\tau_k, M^- = \\ = \sum_{k,u_k<0} F_u^-(u_k)\tau_k$ – величины, характеризующие интенсивность воздействия положительных и отрицательных напряжений соответственно.

Второе слагаемое в правой части уравнения (10) или (11) положительное, а первое и второе слагаемые отрицательные. Поэтому правая часть уравнений (10), (11) может быть как положительной, так и отрицательной. С учётом свойств (6) характеристических функций получаем, что при $x=0$ правая часть уравнения (11) положительная, а при $x=1$ – отрицательная. Это означает, что уравнение стационарной точки уравнения (11)

$$P(x) = 0, \quad (12)$$

где $P(x)$ – правая часть уравнения (11) (или (10) в более общем случае), обязательно имеет решение внутри интервала (0,1). При реалистических (не слишком экзотических) характеристических функциях это решение единственное. Обозначим его x_{st}. При $x<x_{st}$ согласно (11) имеем $dx/dt>0$, а при $x>x_{st}$ имеем $dx/dt<0$. Уравнение (11) описывает монотонное приближение переменной состояния x к стационарной точке x_{st}. Скорость движения равна $P(x)$. Эту скорость характеризует эффективность записи с помощью кусочно-постоянного высокочастотного сигнала.

Уравнение стационарной точки (12) для уравнения (11) можно записать в виде

$$F_x^+(x)M^+ = -F_0(x) - F_x^-(x)M^-. \quad (13)$$

Обычно запись информации осуществляется со скоростями, большими по сравнению со скоростью спонтанной релаксации. При этом можно пренебречь членом $F_0(x)$ в уравнении записи и, следовательно, в уравнении стационарной точки. В результате получим уравнение стационарной точки вида

$$\frac{F_x^-(x)}{F_x^+(x)} = \frac{M^+}{-M^-}. \quad (14)$$

При принятых разумных предположениях левая часть уравнения (14) неограниченно увеличивается от 0 при изменении переменной x от 0 до 1. Правая часть — положительное число для заданного сигнала $u(t)$. Значение отношения $\mu = M^+/(-M^-) = M^+/|M^-|$ определяет положение стационарной точки. При $\mu\to 0$ $x_{st}\to 0$, а при $\mu\to\infty$ $x_{st}\to 1$. Обычно зависимость $x_{st}(\mu)$ — монотонно возрастающая.

Для характеристических функций (7) уравнение (14) принимает вид

$$\frac{x^{\beta_-}}{(1-x)^{\beta_+}} = \mu. \quad (15)$$

Это уравнение имеет единственное решение в интервале (0,1). Получить явное выражение для x_{st} можно только при некоторых соотношениях показателей степеней. Так, при $\beta_+ = \beta_- = \beta$ получаем

$$x_{st} = \frac{\mu^{1/\beta}}{1+\mu^{1/\beta}}. \quad (16)$$

Зависимости $x_{st}(\mu)$ при разных значениях β_+, β_- имеют сходный характер. Это видно из того, что при $\mu<<1$ из формулы (15) следует $x_{st} \approx \mu^{1/\beta_-}$, а при $\mu>>1$ имеем $x_{st} \approx 1 - \mu^{-1/\beta_+}$.

Чтобы оценить влияние спонтанной релаксации, подставим выражения (7) в уравнение (13). Получающееся уравнение

$$\frac{x^{\beta_-}}{(1-x)^{\beta_+}} = \mu - \frac{f}{-M^-}x^{\gamma_0}(1-x)^{\gamma_1-\beta_+} \quad (17)$$

отличается от уравнения (15) наличием дополнительного отрицательного члена в правой части, пропорционального отношению $f/|M^-|$. Влияние этого члена сводится к эффективному уменьшению величины μ, то есть к смещению стационарной точки влево (в сторону граничной точки $x=0$). В частности, при $\gamma_0 = \beta_+ = \beta_- = \beta, \gamma_1 = 0$ получаем простое выражение

$$x_{st} = \frac{\hat{\mu}^{1/\beta}}{1+\hat{\mu}^{1/\beta}}, \text{ где } \hat{\mu} = \frac{M^+}{-M^-+f} = \frac{\mu}{1-f/M^-}, \quad (18)$$

аналогичное выражению (16), но с перенормированной величиной μ.

На практике запись информации должна производиться достаточно быстро, так что в процессе записи спонтанная релаксация состояния резистора не играет роли. В уравнениях (10), (11) можно отбросить первый член в правой части. Изменение состояния резистора определяется величинами M^+, M^- (или аналогичными величинами при отсутствии факторизации в уравнении (10)). Отношение этих величин определяет стационарное состояние резистора при действии данного управляющего сигнала.

Как влияют параметры сигнала на величины M^+, M^-, то есть на процесс записи? К параметрам сигнала относятся используемые ненулевые уровни напряжения u_k, $k=1,\ldots,K$ и относительные длительности действия этих напряжений τ_k, $k=1,\ldots,K$. Правые части уравнений (10), (11) линейно зависят от параметров τ_k, а вот зависимость от уровней

напряжения u_k может быть нелинейной. При этом обычно имеются широкие возможности регулировки скорости записи и положения стационарной точки. Рассмотрим эти возможности на примере простейшего сигнала с двумя ненулевыми уровнями напряжения. Именно такие сигналы есть смысл применять для формирования матрицы проводимостей кроссбара.

3. Простейший управляющий сигнал

Итак, пусть на переменный резистор подается периодическое кусочно-постоянное напряжение с двумя ненулевыми уровнями, один из которых положительный, а другой отрицательный. Для наглядности обозначим их u_+ и u_-. Соответствующие относительные времена действия обозначим τ_+, τ_-. Уравнение (10) здесь принимает вид

$$\frac{dx}{dt} = F_0(x) + F^+(x,u_+)\tau_+ + F^-(x,u_-)\tau_- . \quad (19)$$

При фиксированных значениях u_+, u_- нет необходимости предполагать факторизуемость функций $F^+()$, $F^-()$. Достаточно знать только их зависимость от x при заданном значении второго аргумента. Однако, если мы хотим использовать сигналы с разными значениями амплитуды, надо знать зависимость $F^+()$, $F^-()$ от второго аргумента. Предполагая справедливость разложений (4), приходим к уравнению (11), в котором

$$M^+ = F_u^+(u_+)\tau_+, M^- = F_u^-(u_-)\tau_-, \quad (20)$$

Величины M^+, M^- пропорциональны значениям τ_+, τ_- соответственно, а их зависимости от u_+, u_- задаются характеристическими функциями. С помощью простейшего сигнала можно определить характер этих зависимостей. Для этого надо зафиксировать три из четырёх параметров сигнала, а один параметр — u_+ или u_- — изменять. Измерение скорости записи при разных значениях варьируемого параметра позволит определить тип соответствующей характеристической функции.

Одновременное пропорциональное изменение уровней u_+, u_- (при условии неизменности их отношения) позволяет проверить одинаковость изменения величин M^+, M^- при изменении амплитуды сигнала. Если степенные характеристические функции (8) имеют одинаковые показатели степени: $\alpha_+=\alpha_-$, отношение μ не зависит от амплитуды сигнала, и положение стационарной точки практически не зависит от амплитуды. Для линейных характеристических функций $F_u^+(u), F_u^-(u)$ ($\alpha_+=\alpha_-=1$) каждая из величин M^+, M^- зависит линейно только от своей комбинации $u_+\tau_+$, $u_-\tau_-$. Здесь число параметров сигнала, влияющих на процесс записи, сокращается до двух.

Если мы хотим ещё больше упростить управляющий сигнал, надо наложить дополнительные ограничения. Для сигнала с нулевым средним имеем условие

$$u_+\tau_+ + u_-\tau_- = 0 \quad (21)$$

При наличии такой связи можно говорить о сигнале с амплитудой u_+ (или $-u_-$), форма которого задаётся параметрами τ_+, τ_-. Расположение интервалов постоянства внутри периода сигнала не имеет значения. В случае линейных характеристических функций $F_u^+(u), F_u^-(u)$ величины M^+, M^- зависят от единственной комбинации параметров $u_+\tau_+$. Это не так, если функции $F_u^+(u), F_u^-(u)$ нелинейные на соответствующих полуосях. Такое различие можно использовать для проверки наличия нелинейности характеристических функций.

Дополнительные ограничения, приводящие к упрощению сигнала, выражаются равенствами

$$u_- = -u_+ \quad (22)$$

$$\tau_+ = \tau_- \quad (23)$$

Заметим, что для сигнала с нулевым средним одно из равенств (22), (23) влечёт другое. В следующем разделе именно сигналы, удовлетворяющие условиям (22), (23), используются для формирования матрицы проводимостей кроссбара. Для такого сигнала имеется всего два независимых параметра, определяющих величины M^+, M^-и, следовательно, скорость записи (то есть скорость изменения проводимости резистора). Согласно (19), (20) в данном случае

$$P(x,u_0,\tau) = F_0(x) + F^+(x,u_0)\tau + F^-(x,-u_0)\tau, \quad (24)$$

$$M^+ = F_u^+(u_0)\tau, M^- = F_u^-(-u_0)\tau, \quad (25)$$

$$P(x,u_0,\tau) = F_0(x) + F_x^+(x)F_u^+(u_0)\tau + F_x^-(x)F_u^-(-u_0)\tau, \quad (26)$$

где $\tau=\tau_+=\tau_-$ – степень заполнения, а $u_0=u_+=-u_-$ – амплитуда сигнала. Напомним, что $P(x, u_0,\tau)$ – правая часть уравнения записи (с учётом зависимости от амплитуды сигнала и его степени заполнения). Формула (26) записана для случая факторизации функций $F^+(x,u)$, $F^-(x,u)$, а формула (24) – для общего случая. Как видим, принципиальной разницы между этими случаями нет. Для определённости используем формулу (26), причём в правой части равенства можно опустить первое слагаемое, описывающее спонтанную релаксацию.

4. Процедура формирования матрицы проводимостей

Резистор кроссбара R_j^i соединяет i-ый

горизонтальный проводник с j-ым вертикальным проводником. На проводники подаются потенциалы от источников напряжения. Напряжение на резисторе R_j^i равно

$$u_j^i = V^i - V_j, \quad (27)$$

где V^i, V_j – потенциалы i-го горизонтального и j-го вертикального проводников.

Пусть на проводники поданы потенциалы:

$$V^{i\neq k}(t) = 0,$$
$$V^k(t) = V_0\sigma(t/T), \quad (28)$$
$$V_j(t) = V_0\sigma(t/T + \delta_j),$$

где V_0 – амплитуда сигналов, δ_j – фазовые сдвиги ($0\leq\delta_j<1$), $\sigma(y)$ – периодическая кусочно-постоянная функция с периодом 1, для которой

$$\sigma(y) = 1, 0 < y < 1/2,$$
$$\sigma(y) = -1, 1/2 < y < 1 \quad (29)$$

(значения функции в точках разрыва не играют роли). Функция $\sigma(y)$ – кусочно-постоянный аналог синуса, её даже можно определить с помощью функции sin():

$$\sigma(y) = sign(\sin(2\pi y)) \quad (30)$$

(sign() – функция знака).

Распределение потенциалов (29) означает, что на k-ый горизонтальный проводник подан стандартный сигнал, остальные горизонтальные проводники заземлены. На вертикальные проводники поданы сигналы, получающиеся из стандартного сигнала с помощью фазового (временного) сдвига. Сдвиг фазы свой для каждого вертикального проводника. Резисторы, присоединённые к k-му горизонтальному проводнику, то есть резисторы k-ой строки, оказываются в выделенном положении. Будем называть эти резисторы целевыми.

Распределение напряжений на резисторах кроссбара согласно (27), (28) имеет вид

$$u_j^{i\neq k}(t) = -V_0\sigma(t/T + \delta_j),$$
$$u_j^k(t) = V_0(\sigma(t/T) - \sigma(t/T + \delta_j)). \quad (31)$$

На нецелевые резисторы действуют простейшие сигналы, удовлетворяющие условиям (22), (23). Существенные параметры этих сигналов $\tau=1/2$, $u_0=V_0$ одинаковые для всех нецелевых резисторов.

Напряжения на целевых резисторах – также простейшие сигналы, удовлетворяющие условиям (22), (23). Для них $u_0=2V_0$, а вот параметр τ зависит от фазового сдвига: для резистора R_j^k этот параметр равен

$$\tau_j = min(\delta_j, 1 - \delta_j). \quad (32)$$

При изменении δ_j от 0 до 1/2 параметр τ_j линейно растёт от 0 до 1/2, а при изменении δ_j от 1/2 до 1 параметр τ_j линейно падает от 1/2 до 0. Для получения полного диапазона изменения параметра τ можно ограничиться половиной диапазона изменения фазового сдвига. Считая, что $0\leq\delta_j\leq1/2$, получаем согласно (32)

$$\tau_j = \delta_j. \quad (33)$$

Запишем формулы для изменения состояний резисторов под действием напряжений (31), предполагая, что резисторы кроссбара одинаковые, выбрано базовое состояние резисторов x_b, информация записывается в виде малых отклонений от базового состояния x-x_b. С помощью формулы (26) получаем для изменений состояний резисторов

$$\Delta x_j^{i\neq k} = P(x_b, V_0, 1/2)t_r =$$
$$= (F_x^+(x_b)F_u^+(V_0) + F_x^-(x_b)F_u^-(-V_0))\frac{t_r}{2},$$
$$\Delta x_j^k = P(x_b, 2V_0, \delta_j)t_r = \quad (34)$$
$$= (F_x^+(x_b)F_u^+(2V_0) + F_x^-(x_b)F_u^-(-2V_0))t_r\delta_j$$

(t_r – время записи). Нецелевые резисторы испытывают одинаковое изменение состояния, не зависящее от фазовых сдвигов. В отличие от них, целевые резисторы изменяют своё состояние на величину, пропорциональную соответствующему сдвигу фаз, при условии, что коэффициент пропорциональности не равен нулю.

Чтобы избавиться от изменения состояний нецелевых резисторов, можно выбрать базовое состояние совпадающим со стационарным состоянием под действием стандартного сигнала: $x_b=x_{st}(u_0=V_0)$. При таком выборе $\Delta x\approx0$ для нецелевых резисторов. Важно, чтобы стационарное состояние под действием напряжения на целевых резисторах заметно отличалось от базового состояния: $x_b\neq x_{st}(u_0=2V_0)$. В противном случае изменения состояний целевых резисторов будут незначительны. Как мы видели в разделе 3, такой неблагоприятный случай имеет место при условии, что характеристические функции $F_u^+(u), F_u^-(u)$ изменяются на своих полуосях сходным образом – показатели α_+, α_- равны друг другу для случая степенных характеристических функциях. Здесь $x_{st}(u_0=V_0)=x_{st}(u_0=2V_0)$ (при пренебрежении спонтанной релаксацией), и избавиться от смещения состояний нецелевых резисторов с помощью выбора базового состояния не получится.

При произвольном выборе базового состояния необходимо компенсировать изменение состояний нецелевых резисторов. Это можно сделать, подав постоянное напряжение между всеми горизонтальными и всеми вертикальными проводниками:

$$V^i(t) = V^0,$$
$$V_j(t) = 0. \quad (35)$$

Напряжение V^0 и время действия этого напряжения t_{r2} (время записи второго этапа) должны удовлетворять условию

$$\begin{gathered}\left(F_x^+(x_b)F_u^+(V^0)+F_x^-(x_b)F_u^-(V^0)\right)t_{r2}= \\ =P(x_b,V_0,1/2)t_r= \\ =\left(F_x^+(x_b)F_u^+(V_0)+F_x^-(x_b)F_u^-(-V_0)\right)\frac{t_r}{2}.\end{gathered} \quad (36)$$

Полярность постоянного напряжения зависит от выбора базового состояния. При $x_b<x_{st}(u_0=V_0)$ напряжение V^0 должно быть отрицательным, а при $x_b>x_{st}(u_0=V_0)$ – положительным.

После двух этапов записи имеем изменения состояний резисторов

$$\begin{gathered}\Delta x_j^{i\neq k}=0, \\ \Delta x_j^k=\left(F_x^+(x_b)F_u^+(2V_0)+\right. \\ \left.F_x^-(x_b)F_u^-(-2V_0)\right)t_r\delta_j- \\ -\left(F_x^+(x_b)F_u^+(V_0)+F_x^-(x_b)F_u^-(-V_0)\right)\frac{t_r}{2}.\end{gathered} \quad (37)$$

В итоге состояния нецелевых резисторов остались прежними, а изменения состояний целевых резисторов линейно зависят от соответствующих фазовый смещений. Полный диапазон изменения состояния целевого резистора, соответствующий изменению фазового сдвига от 0 до 1/2, зависит от взаимного расположения состояний x_b, $x_{st}(u_0=V_0)$, $x_{st}(u_0=2V_0)$. В типичном случае $x_b<x_{st}(u_0=V_0)<x_{st}(u_0=2V_0)$ нулевое значение изменения состояния для целевого резистора находится внутри доступного диапазона Δx. При этом с учётом произвольности выбора времени записи t_r можно записать любое (небольшое) значение Δx.

Таким образом, можно осуществить произвольные (в разумных пределах) изменения состояний резисторов выбранной строки кроссбара. Выполняя описанную процедуру для разных строк резисторной матрицы, можно записать заданную информацию в резисторную матрицу – кроссбар. Заметим, что вместо построчной записи можно говорить о постолбцовой записи. Вообще, названия «строки» и «столбцы», как и «горизонтальные» и «вертикальные проводники», условные – поворот кроссбара на 90 градусов превращает горизонтальные проводники в вертикальные, а строки в столбцы.

Можно было бы попытаться изменить на одном этапе не одну, а несколько строк матрицы проводимостей, подав простейший сигнал не на один горизонтальный проводник (как в (28)), а на несколько. Такие сигналы могут иметь фазовые сдвиги друг относительно друга. Однако в этом случае нельзя обеспечить произвольные изменения состояний целевых резисторов из-за недостаточного количества управляющих параметров. Здесь достижимы только изменения специального вида.

5. Заключительные замечания

Итак, использование простейших высокочастотных кусочно-постоянных сигналов позволяет формировать произвольные матрицы проводимостей. При записи матрицы «с нуля» процесс записи многошаговый – на каждом шаге записывается одна строка или один столбец матрицы. Если же уже имеется некоторая матрица, и требуется только небольшая её корректировка, можно производить изменения в отдельных строках или столбцах резисторной матрицы.

Удобство и сама возможность использования высокочастотных сигналов для формирования матрицы зависит от вида характеристических функций резистора. В частности, наиболее благоприятные условия возникают, если функция $F_I^+(I)$ растёт быстрее функции $-F_I^-(-I)$ при увеличении тока I [7]. В этом случае доступен в принципе весь диапазон состояний резистора, и легко получить как положительные, так и отрицательные значения Δx. В противном случае доступный диапазон состояния уже, меньше возможностей для выбора базового состояния, время записи увеличивается, поскольку нельзя использовать большие амплитуды тока для получения положительных смещений Δx. Для функций $F_0(x), F_x^+(x), F_x^-(x)$ мы приняли выполнение условий (6). Невыполнение этих условий может сделать невозможной запись с помощью высокочастотных сигналов. Впрочем, условия (6) кажутся вполне естественными и даже имеют экспериментальные подтверждения [6,10].

Процедура записи и условия её реализуемости похожи при использовании кусочно-постоянных и гармонических сигналов [8]. Естественно, возникает вопрос о том, какой сигнал удобнее с практической точки зрения. С одной стороны, гармонические сигналы привычны для радиотехников и совместимы с элементами цепей переменного тока, прежде всего, с конденсаторами, что позволяет использовать большой накопленный опыт в области линейных радиотехнических цепей (например, использовать различные диапазоны частот для разделения записи и считывания информации). С другой стороны, использование кусочно-постоянных сигналов позволяет осуществлять более тонкое и удобное управление процессом записи. В отличие от гармонического сигнала, кусочно-постоянный сигнал не обязан иметь нулевое среднее. В частности, постоянный сигнал хорошо совместим с кусочно-постоянным сигналом, поскольку является его частным случаем. При использовании кусочно-постоянного сигнала

результат записи линейно зависит от фазового сдвига (см. (37)), в отличие от нелинейной зависимости для случая гармонических сигналов [8]. Это позволяет быстрее и точнее найти нужные фазовые сдвиги. Да и само формирование кусочно-постоянных сигналов при современном развитии цифровой электроники нельзя назвать сложной задачей.

Кроме того, кусочно-постоянные сигналы удобны с методической точки зрения. При фиксированных значениях сигнала достаточно знать скорость изменения состояния резистора только при этих значениях напряжения, для получения простых оценок не обязательно предполагать факторизуемость правой части уравнения записи. Не возникает проблем с изменением формы сигнала, которое может быть использовано для выявления нелинейных свойств характеристических функций. При использовании гармонического сигнала его форма фиксирована. Можно, конечно, использовать сумму гармонических сигналов [11], однако это более сложный путь. Заметим, что любой сигнал можно приближенно представить кусочно-постоянным сигналом. Но практической пользы от этого нет, поскольку при большом числе уровней сигнала теряется его главное преимущество – простота. Для формирования матрицы проводимостей кроссбара использовался простейший кусочно-постоянный сигнал — с двумя ненулевыми значениями и равными промежутками действия значений. Именно простота такого сигнала даёт надежду на практическую полезность рассмотренной методики записи информации в резисторную матрицу.

Работа выполнена в рамках государственно го задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0001 "Создание и реализация доверенных систем искусственного интеллекта, основанных на новых математических и алгоритмических методах, моделях быстрых вычислений, реализуемых на отечественных вычислительных системах" (1023032100070-3-1.2.1).

Crossbar Array Programming Using Piecewise-Constant Signals

G. A. Beskhlebnova, V. B. Kotov

Abstract. To program a crossbar array, we need to adjust the resistor conductance using a limited number of control signals, which are voltages applied to the crossbar lines. Since the number of lines is significantly smaller than the number of resistors, this is a multi-step procedure. At each step, the conductances of the selected resistors are adjusted. The number of such resistors is no greater than the number of control signals. This inevitably changes the conductivity of some half-selected resistors, too. These unwanted changes must be compensated for. We examined a crossbar programming procedure using high-frequency piecewise-constant control signals. Our analysis involved a simple resistive element model. We demonstrated that an arbitrary (within known limits) conductance matrix can be programmed. At each step, a row or column of the crossbar array is generated or adjusted. We discussed the feasibility and convenience of such a procedure.

Keywords: variable resistor, piecewise constant signal, resistor array, conductivity matrix.

Литература

1. Kotov V.B., Beskhlebnova G.A. Specifics of Crossbar Resistor Arrays. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research VI (NEUROINFORMATICS 2022). Studies in Computational Intelligence. Vol. 1064. Cham: Springer. 2023. PP. 292–304. https://doi.org/10.1007/978-3-031-19032-2_31.
2. Adamatzky A., Chua L. Memristor Networks. Springer International Publishing (2014).
3. Advances in Memristors, Memristive Devices and Systems. / Edited by S. Vaidyanathan and C. Volos. Springer International Publishing AG (2017).
4. Kim S. Ju, Kim S., Jang H.W. Competing memristors for brain-inspired computing. iScience 24, 101889, January 22, 2021.
5. Kotov V.B., Beskhlebnova G.A. Generation of the Conductivity Matrix. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research V (NEUROINFORMATICS 2021). Studies in Computational Intelligence. Vol. 1008. Cham: Springer. 2022. PP. 276-284.

6. Surazhevsky I.A. at all. Noise-assisted persistence and recovery of memory state in amemristive spiking neuromorphic network.Chaos, Solitons and Fractals. 146 (2021). 110890.
7. Beskhlebnova G.A., Kotov V.B. The Variable Resistor Under a High-Frequency Signal. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research VII (NEUROINFORMATICS 2023). Studies in Computational Intelligence. Vol. 1120. Springer Nature Switzerland AG. 2023. PP. 257–266. https://doi.org/10.1007/978-3-031-44865-2_28.
8. Kotov V.B., Beskhlebnova G.A. Use of High-Frequency Signals to Generate a Conductivity Matrix. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research VIII (NEUROINFORMATICS 2024). Studies in Computational Intelligence. Vol. 1179. Springer Nature Switzerland AG. 2025. PP. 265-272.
9. Котов В.Б., Бесхлебнова Г.А. Поточечная запись информации в резисторную матрицу. // Труды НИИСИ РАН. Т. 14. №4. С. 33-40.
10. Kotov V.B., Yudkin F.A. Modeling and Characterization of Resistor Elements for Neuromorphic Systems. Optical Memory and Neural Networks (Information Optics). 2019, v.28, No.4, P. 271-282.
11. V.B. Kotov, Z. B. Sokhova. Two-frequency recording of information into a resistor array. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research IX (NEUROINFORMATICS 2025). Studies in Computational Intelligence. Vol. 1241. Springer Nature Switzerland AG. 2026. PP. 463-476.

Crossbar Array Programming Using Piecewise-Constant Signals

G. A. Beskhlebnova[1], V. B. Kotov[2]

[1]Scientific Research Institute for System Analysis of the National Research Centre Kurchatov Institute, Moscow, Russia, beskhlebnova@niisi.ras.ru;

[2]Scientific Research Institute for System Analysis of the National Research Centre Kurchatov Institute, Moscow, Russia, v1111111k1111@gmail.com

Abstract. To program a crossbar array, we need to adjust the resistor conductance using a limited number of control signals, which are voltages applied to the crossbar lines. Since the number of lines is significantly smaller than the number of resistors, this is a multi-step procedure. At each step, the conductances of the selected resistors are adjusted. The number of such resistors is no greater than the number of control signals. This inevitably changes the conductivity of some half-selected resistors, too. These unwanted changes must be compensated for. We examined a crossbar programming procedure using high-frequency piecewise-constant control signals. Our analysis involved a simple resistive element model. We demonstrated that an arbitrary (within known limits) conductance matrix can be programmed. At each step, a row or column of the crossbar array is generated or adjusted. We discussed the feasibility and convenience of such a procedure.

Keywords: variable resistor, piecewise-constant signal, crossbar array, conductance matrix

1. Introduction

Large crossbar arrays can become the basic building blocks of neuromorphic systems. They perform vector–matrix multiplication, the most computationally intensive operation in neural computing. The values of the multiplier matrix elements are represented by the conductances of the crossbar array resistors [1]. To use he same crossbar array for multiplication by different matrices, the resistors must be variable and controllable. Memristors are variable resistors whose resistance changes based on current [2-4]. They are commonly used due to their simplicity, compact size, and low energy consumption.

Programming large crossbar arrays is a challenging problem [1, 5]. It is impossible to control each resistor individually since their number is huge. In crossbars, two conductors connect to not one but multiple resistors. In crossbars, each conductor connects to multiple resistors, not just one. Therefore, a step-by-step programming procedure is required. With each step, a row or column of the crossbar is programmed. Using constant (more precisely, unipolar) voltage signals for crossbar programming leads to significant difficulties: the conductances of both selected and many half-selected resistors are affected. The direction of these undesirable changes varies for each half-selected resistor, making it difficult to compensate for them. These difficulties are often circumvented by assuming a threshold behavior of the conductance [2, 3, 6]. However, this does not work in real life. Nature does not always meet the expectations of scientists.

To achieve easily compensable changes in half-selected resistors, we can apply a modulated high-frequency signal to program the crossbar [7]. The use of high-frequency harmonic signals for crossbar programming is discussed in [8, 9]. It is shown that for unidirectional variable resistors, applying alternating harmonic voltages as control signals allows for programming virtually any (within certain limits) crossbar.

Other periodic high-frequency signals can also be used. Such signals should be sufficiently simple to generate to be practical. Piecewise-constant signals stand out as they can have only a few values. We studied the application of such signals to crossbar programming. We assumed that applying a single period of the control voltage produces an insignificant change in the variable resistor's conductance; multiple periods are required to program the crossbar (i.e., change the resistor conductance). It means that a high-frequency signal is needed. The signal values must be both positive and negative. Otherwise, it is a unipolar signal similar to a constant signal in its effect. We also assumed that the variable resistor is represented by a simple resistor element model [10].

2. Crossbar Programming Equations

The conductivity G and resistance R=1/G of a simple resistor element are expressed by only one state variable x: $G=G(x)$. We assumed that the state variable ranges from 0 to 1. The high-resistance

state ($x = 0$) corresponds to the maximum resistance, and the low-resistance state ($x = 1$) corresponds to the minimum resistance. The change in the state variable is described by equation [10]

$$\frac{dx}{dt} = F(x,u), \quad (1)$$

where u is the voltage applied to the resistor. Note that, in general, the right-hand side of equation (1) depends on both the resistor voltage u and current I. Using Ohm's law (it can be applied to resistors by definition)

$$u = R(x)I \quad (2)$$

we can express I in terms of u and x as equation (1).

Let us express the function F () as

$$F(x,u) = F_0(x) + F^+(x,u) + F^-(x,u) \quad (3)$$

Where the term $F_0(x) = F(x, u = 0)$ describes the spontaneous change of the resistor toward its high-resistance state in the absence of any control signal; the term $F^+(x,u) = \theta(u)(F(x,u) - F_0(x))$ represents the tendency of the state variable x to increase under a positive voltage; the term $F^-(x,u) = \theta(-u)(F(x,u) - F_0(x))$ represents the tendency of x to decrease (accelerated relaxation) under a negative voltage. (Here $\theta(x)$ is the Heaviside function).

If $F^+(x,u)$ ($F^-(x,u)$) vs. x weakly depends on u, we can use factorization:

$$F^+(x,u) = F_x^+(x)F_u^+(u), F^-(x,u) = \\ = F_x^-(x)F_u^-(u), \quad (4)$$

As a result, we obtain the following function F:

$$F(x,u) = F_0(x) + F_x^+(x)F_u^+(u) + \\ + F_x^-(x)F_u^-(u) \quad (5)$$

The functions of a single variable $F_0(x), F_x^{\pm}(x), F_u^{\pm}(u)$ and the function G(x) are called characteristic functions of a resistor.

The characteristic functions $F_x^{\pm}(x), F_u^{\pm}(u)$ are not unambiguously defined by equations (4). Let us assume that the functions $F_x^{\pm}(x)$ are normalized to unity, i.e., they are equal to 1 at characteristic points (at one of the boundary points).

. The characteristic functions $F_0(x), F_x^{\pm}(x)$ define how quickly a resistor's state changes during programming and erasing, depending on its current state. It is natural to assume that they are continuous on [0,1], positive on (0,1), and satisfy the following relations:

$$F_0(0) = 0, F_x^+(0) = 1, F_x^+(1) = \\ = 0, F_x^-(0) = 0, F_x^-(1) = 1 \quad (6)$$

A typical example of such functions:

$$F_0(x) = fx^{\gamma_0}(1-x)^{\gamma_1}, F_x^+(x) = \\ = (1-x)^{\beta_+}, F_x^-(x) = x^{\beta_-}, \quad (7)$$

where γ_0, β_+, β are positive; γ_1 is non-negative; f is a positive coefficient. Note that for $\gamma_1>0$ we have $F_0(1) = 0$, and for $\gamma_1=0$, $F_0(1) = 1$.

The characteristic functions $F_u^+(u), F_u^-(u)$ define the relationships between the rate of change of the state variable x and the positive or negative voltage applied to the resistor, respectively. The most typical type of functions $F_u^{\pm}(u)$ is an exponent function defined on the respective semi-axes:

$$F_u^+(u) = A_+\theta(u)u^{\alpha_+}, F_u^-(u) = \\ = -A_-\theta(-u)(-u)^{\alpha_-} \quad (8)$$

with positive coefficients $A_{\pm}$ and values $\alpha_{\pm}$.

Let a periodic piecewise constant voltage $u(t)$ be applied to the variable resistor. Let us denote u_k, k=1,…, K as the voltages, and T_k as the duration of the voltage application during one period T. The change in the resistor's state during one period of the control signal (voltage) according to equation (1) and considering equation (3) can be expressed as

$$x(t+T) - x(t) = F_0(x)T + \\ + \sum_{k,u_k>0} F^+(x,u_k)T_k + \sum_{k,u_k<0} F^-(x,u_k)T_k \quad (9)$$

We are interested in slow (averaged over the signal period) changes in the resistor's state (we neglect small fluctuations of the state variable within the period due to their insignificance when a high-frequency signal is applied). For such changes, the equation is

$$\frac{dx}{dt} = \frac{x(t+T)-x(t)}{T} = F_0(x) + \\ + \sum_{k,u_k>0} F^+(x,u_k)\tau_k + \sum_{k,u_k<0} F^-(x,u_k)\tau_k \quad (10)$$

where $\tau_k=T_k/T$ is the fraction of time when u_k is nonzero.

Equation (10) was obtained without assuming the factorizability of equation (4). Let us recall that for harmonic signals [7], factorization is required to obtain a correct equation. If we apply factorization to this case, equation (10) can be expressed as

$$\frac{dx}{dt} = F_0(x) + F_x^+(x)M^+ + F_x^-(x)M^-, \quad (11)$$

where $M^+ = \sum_{k,u_k>0} F_u^+(u_k)\tau_k, M^- = \\ = \sum_{k,u_k<0} F_u^-(u_k)\tau_k$ are the values that represent the responses to positive and negative voltages, respectively.

The second term on the right-hand side of equation (10) or (11) is positive, while the first and third terms are negative. Therefore, the right-hand side of equations (10) and (11) can be either positive or negative. Considering the properties of the characteristic functions expressed in (6), we obtain that when $x = 0$, the right-hand side of equation (11) is positive, and when $x = 1$, it is negative. This means that the equilibrium point equation (11) is

$$P(x) = 0, \quad (12)$$

where $P(x)$ is the right-hand side of equation (11) (or (10) in a more general case), always has a solution within the (0,1) range. For real-life (and not too exotic) characteristic functions, this is the only solution. Let us denote it as x_{st}. According to (11), for $x<x_{st}$, $dx/dt>0$, and for $x>x_{st}$, $dx/dt<0$. Equation (11) describes the monotonic approach of the state variable x to the equilibrium point x_{st}. The approximation rate is P(x). The rate characterizes the efficiency of crossbar programming using a piecewise-constant high-frequency signal.

The equilibrium point equation (12) used in

equation (11) can be expressed as

$$F_x^+(x)M^+ = -F_0(x) - F_x^-(x)M^-. \quad (13)$$

Usually, the crossbar programming rate is higher than the rate of spontaneous relaxation. Therefore, the term $F_0(x)$ in the programming equation and equilibrium point equation can be neglected. The resulting equilibrium point equation is

$$\frac{F_x^-(x)}{F_x^+(x)} = \frac{M^+}{-M^-}. \quad (14)$$

Under reasonable assumptions, the left side of equation (14) increases indefinitely starting from 0 as the variable x changes from 0 to 1. The right side is a positive number for a given signal $u(t)$. The value of $\mu = M^+/(-M^-) = M^+/|M-|$ defines the position of the equilibrium point. For $\mu \to 0$ $x_{st} \to 0$, and for $\mu \to \infty$ $x_{st} \to 1$. Usually, $x_{st}(\mu)$ is monotonically increasing.

For characteristic functions (7), equation (14) takes the form

$$\frac{x^{\beta_-}}{(1-x)^{\beta_+}} = \mu. \quad (15)$$

This equation has a unique solution in the (0, 1) range. An explicit expression for x_{st} can only be obtained for certain exponents. For example, for $\beta_+ = \beta_- = \beta$ we obtain

$$x_{st} = \frac{\mu^{1/\beta}}{1+\mu^{1/\beta}}. \quad (16)$$

We also get more cumbersome explicit expressions for x_{st} for $\beta_+ = 2\beta_-$ and $2\beta_+ = \beta_-$. The $x_{st}(\mu)$ relationships for different values of β_+, β_- are similar. This comes from the fact that when $\mu \ll 1$ (refer to equation (15)), it follows that $x_{st} \approx \mu^{1/\beta_-}$, and when $\mu \gg 1$, we have $x_{st} \approx 1 - \mu^{-1/\beta_+}$.

To assess the effect of spontaneous relaxation, we substituted expressions (7) into equation (13). The resulting equation is

$$\frac{x^{\beta_-}}{(1-x)^{\beta_+}} = \mu - \frac{f}{-M^-} x^{\gamma_0}(1-x)^{\gamma_1-\beta_+} \quad (17)$$

It differs from equation (15) by an additional negative term on the right-hand side, proportional to $f/|M^-|$. The effect of this term is an effective reduction of μ, i.e., a shift of the equilibrium point to the left (towards the boundary point x=0). In particular, for $\gamma_0 = \beta_+ = \beta_- = \beta, \gamma_1 = 0$, we obtain a simple expression

$$x_{st} = \frac{\hat{\mu}^{1/\beta}}{1+\hat{\mu}^{1/\beta}}, where\ \hat{\mu} = \frac{M^+}{-M^-+f} = \frac{\mu}{1-f/M^-}, \quad (18)$$

similar to expression (16), but with a renormalized μ.

In real-life applications, the crossbar programming rate should be sufficiently high to avoid the effects of spontaneous relaxation of the resistor state during the process. In equations (10) and (11), the first term on the right-hand side can be discarded. The change in the resistor's state is defined by M^+, M^- (or similar values if equation (10) is not factorized). The ratio of these values defines the steady state of the resistor when the control signal is applied.

How do signal parameters affect M^+, M^-, i.e., the programming? The signal parameters are the non-zero voltage levels u_k, k=1,…,K and the relative durations of these voltages τ_k, k=1,…,K. The right-hand sides of equations (10) and (11) vary linearly with τ_k, but their dependence on the voltage levels u_k may be nonlinear. There are usually extensive options for adjusting the programming rate and the position of the equilibrium point. Let us consider these options for the simplest signal with two non-zero voltage levels. It is feasible to use such signals for crossbar programming.

3. The Simplest Control Signal

Let a periodic piecewise constant voltage with two non-zero levels, positive and negative, be applied to the variable resistor. We denote them as u_+ and u_-. Let us denote the respective relative periods of voltage application as τ_+, τ_-. Then equation (10) takes the form

$$\frac{dx}{dt} = F_0(x) + F^+(x, u_+)\tau_+ + F^-(x, u_-)\tau_- \quad (19)$$

For constant u_+, u_-, there is no need to assume the factorizability of the functions $F^+()$, $F^-()$. It is sufficient to know only their dependence on x for a given value of the second argument. However, if we want to use signals with different amplitudes, we also need to know the dependence of $F^+()$, $F^-()$ on the second argument. Assuming the validity of decompositions (4), we arrive at equation (11), where

$$M^+ = F_u^+(u_+)\tau_+, M^- = F_u^-(u_-)\tau_-, \quad (20)$$

The values M^+, M^- are proportional to τ_+, τ_- respectively, and their dependencies on u_+, u_- are represented by characteristic functions. These dependencies can be obtained by analyzing the simplest signal. For this, we need to fix three of the four signal parameters and change only one: u_+ or u_-. By measuring the programming rate at different values of the variable parameter, we can determine the type of the characteristic function.

By changing u_+, u_- voltages simultaneously (provided that their ratio is constant), we can check whether M^+, M^- vary uniformly as the signal amplitude changes. If the exponential characteristic functions (8) have identical exponents: $\alpha_+ = \alpha_-$, μ does not depend on the signal amplitude, and the position of the equilibrium point is virtually independent of the amplitude. For linear characteristic functions $F_u^+(u), F_u^-(u)$ ($\alpha_+ = \alpha_- = 1$), each of the M^+, M^- values depend linearly only on its combination of $u_+\tau_+$, $u_-\tau_-$. In this case, the number of signal parameters affecting the programming process is reduced to two.

If we want to simplify the control signal even further, additional constraints should be introduced.

For a signal with a zero mean value, the condition is

$$u_+\tau_+ + u_-\tau_- = 0. \quad (21)$$

If such a relationship exists, we have a signal with amplitude u_+ (or $-u_-$) with its shape defined by τ_+, τ_-. The positions of the constant intervals within the signal period are irrelevant. If the characteristic functions $F_u^+(u), F_u^-(u)$ are linear, M^+, M^- depend on a single combination of the $u_+\tau_+$ parameters. This is not so if the $F_u^+(u), F_u^-(u)$ functions are nonlinear on the respective half-axes. We can use this difference to determine whether the characteristic functions are nonlinear.

The additional constraints that simplify the signal are as follows.

$$u_- = -u_+ \quad (22)$$

$$\tau_+ = \tau_- \quad (23)$$

Note that for a signal with a zero mean value, one of the equalities (22) and (23) implies the other. In the next section, the signals that satisfy conditions (22) and (23) are used to program the crossbar. For such a signal, there are only two independent parameters that determine M^+, M^- and, consequently, the crossbar programming rate (the rate of change in the resistor's conductivity). According to (19), (20), in this case

$$P(x, u_0, \tau) = F_0(x) + F^+(x, u_0)\tau + \\ + F^-(x, -u_0)\tau, \quad (24)$$

$$M^+ = F_u^+(u_0)\tau, M^- = F_u^-(-u_0)\tau, \quad (25)$$

$$P(x, u_0, \tau) = F_0(x) + F_x^+(x)F_u^+(u_0)\tau + \\ + F_x^-(x)F_u^-(-u_0)\tau, \quad (26)$$

where $\tau=\tau_+=\tau_-$ is the duty ratio, and $u_0=u_+=-u_-$ is the signal amplitude. Since $P(x, u_0,\tau)$ is the right-hand side of the programming process equation (considering its dependence on the signal amplitude and its duty ratio). Equation (26) applies to the case when functions $F^+(x,u)$, $F^-(x,u)$ are factorized, and equation (24) applies to the general case. There is no fundamental difference between these cases. For certainty, we will use equation (26), and on the right-hand side of the equality, we can discard the first term that represents spontaneous relaxation.

4. Crossbar Array Programming Procedure

The crossbar resistor R_j^i connects the i^{th} word line to the j^{th} bit line. Voltage sources apply potentials to the lines. The voltage applied to the resistor R_j^i is

$$u_j^i = V^i - V_j, \quad (27)$$

where V^i, V_j are the potentials of the i^{th} word line to the j^{th} bit line.

Suppose the following potentials are applied to the lines:

$$V^{i\neq k}(t) = 0, \\ V^k(t) = V_0\sigma(t/T), \\ V_j(t) = V_0\sigma(t/T + \delta_j), \quad (28)$$

where k is the index of the selected row, V_0 is the amplitude of the signals; δ_j is the phase shift ($0\leq\delta_j<1$), $\sigma(y)$ is a periodic piecewise constant function with period 1, for which

$$\sigma(y) = 1, 0 < y < 1/2, \\ \sigma(y) = -1, 1/2 < y < 1 \quad (29)$$

(The values of the function at the points of discontinuity are irrelevant). Function $\sigma(y)$ is a piecewise constant equivalent of the sine function. It can even be defined using the sin() function:

$$\sigma(y) = sign(\sin(2\pi y)) \quad (30)$$

where sign() is the sign function.

The distribution of potential represented by equation (29) means that a standard signal is applied to the k^{th} word line, while the other word lines are grounded. Signals obtained from a standard signal by a phase shift (in the time domain) apply to the bit lines. The phase shift is specific to each bit line. Resistors connected to the k^{th} word line are selected.

The voltage distribution across the crossbar resistors according to (27), (28) is as follows.

$$u_j^{i\neq k}(t) = -V_0\sigma(t/T + \delta_j), \\ u_j^k(t) = V_0(\sigma(t/T) - \sigma(t/T + \delta_j)) \quad (31)$$

The half-selected resistors are affected by the simplest signals that satisfy conditions (22) and (23). The essential parameters of these signals $\tau=1/2$, $u_0=V_0$ are identical for all half-selected resistors.

The voltages applied to the selected resistors are also the simplest signals satisfying conditions (22) and (23). For them, $u_0=2V_0$, but the parameter τ depends on the phase shift: for resistor R_j^k it is equal to

$$\tau_j = min(\delta_j, 1 - \delta_j) \quad (32)$$

When δ_j changes from 0 to 1/2, the parameter τ_j increases linearly from 0 to 1/2, and when δ_j changes from 1/2 to 1, the parameter τ_j decreases linearly from 1/2 to 0. To obtain the full range of the parameter τ variation, it is sufficient to consider half of the phase shift range. Assuming that $0\leq\delta_j\leq1/2$, according to (32) we obtain

$$\tau_j = \delta_j \quad (33)$$

Let us express equations to represent the resistor state changes under voltages (31), assuming that the crossbar resistors are identical, the baseline state of the resistors is x_b, and small deviations from the baseline state x-x_b are used for crossbar programming. Using equation (26), we get

$$\Delta x_j^{i\neq k} = P(x_b, V_0, 1/2)t_r = \\ = (F_x^+(x_b)F_u^+(V_0) + F_x^-(x_b)F_u^-(-V_0))\frac{t_r}{2}, \\ \Delta x_j^k = P\big(x_b, 2V_0, \delta_j\big)t_r = \\ = (F_x^+(x_b)F_u^+(2V_0) + F_x^-(x_b)F_u^-(-2V_0))t_r\delta_j \quad (34)$$

(where t_r is the programming period). Half-selected resistors experience an identical change in their states, independent of the phase shifts. Unlike that, selected resistors change their states by an amount proportional to the phase shift, provided that the proportionality coefficient is not zero.

To avoid changes to the states of half-selected resistors, we can select a baseline state identical to the steady state under a standard signal (first equation (32)): $x_b=x_{st}(u_0=V_0)$. In this case, for half-selected resistors$\Delta x\approx 0$. The steady state under voltage applied to the selected resistors should differ significantly from the baseline state: $x_b\neq x_{st}(u_0=2V_0)$. Otherwise, changes in the states of the selected resistors would be insignificant. As we showed in Section 3, such an unfavorable case occurs when the characteristic functions $F_u^+(u), F_u^-(u)$ change similarly along their semi-axes. The values α_+, α_- are equal for exponential characteristic functions. Here, $x_{st}(u_0=V_0)=x_{st}(u_0=2V_0)$ (spontaneous relaxation is discarded), so it is not possible to avoid affecting half-selected resistors by selecting an appropriate baseline state.

When the baseline state is arbitrary, the changes to the states of half-selected resistors must be compensated for. For this, we can apply a constant voltage between all word and bit lines:

$$V^i(t) = V^0, \\ V_j(t) = 0. \quad (35)$$

The voltage V^0 and the period of its application t_{r2} (second programming step) must satisfy the condition:

$$\big(F_x^+(x_b)F_u^+(V^0) + F_x^-(x_b)F_u^-(V^0)\big)t_{r2} = \\ = P(x_b, V_0, 1/2)t_r = \\ = (F_x^+(x_b)F_u^+(V_0) + F_x^-(x_b)F_u^-(-V_0))\frac{t_r}{2}. \quad (36)$$

The polarity of the constant voltage depends on the selected baseline state. When $x_b<x_{st}(u_0=V_0)$, the voltage V^0 must be negative, and when $x_b>x_{st}(u_0=V_0)$, it must be positive.

After two programming steps, we have the following changes in the states of the resistor

$$\Delta x_j^{i\neq k} = 0, \\ \Delta x_j^k = (F_x^+(x_b)F_u^+(2V_0) + \\ + F_x^-(x_b)F_u^-(-2V_0))t_r\delta_j - \\ -(F_x^+(x_b)F_u^+(V_0) + F_x^-(x_b)F_u^-(-V_0))\frac{t_r}{2}. \quad (37)$$

As a result, the states of the half-selected resistors remain unchanged, while the changes in the states of the selected resistors linearly depend on the phase shifts. The full range of changes in the state of the selected resistor for a phase shift change from 0 to 1/2, depends on the relative positions of the states x_b, $x_{st}(u_0=V_0)$, $x_{st}(u_0=2V_0)$. In a typical case, $x_b<x_{st}(u_0=V_0)<x_{st}(u_0=2V_0)$, the zero change of the state for the selected resistor is within the available range Δx. Since the programming time t_r is arbitrary, we can use any (small) value Δx.

In this way, arbitrary (within reasonable limits) changes to the states of the resistors in the selected crossbar line can be made. By repeating this procedure for each word line, we can program the crossbar. Both row-wise and column-wise programming processes are possible. Actually, the names like “rows” and “columns” or “horizontal” and “vertical” are only conventional. Rotate the crossbar by 90 degrees to turn rows into columns, and columns into rows.

We can try to change multiple crossbar lines by applying the simplest signal to them (as described in (28)). Such signals may have phase shifts relative to each other. However, in this case, we cannot arbitrarily change the states of the selected resistors because the number of control parameters is insufficient. Only some special cases can be achieved.

5. Conclusion

Arbitrary conductance matrices can be generated by applying simple high-frequency piecewise-constant signals. Programming a pristine crossbar is a multi-step process: at each step, a single row or column is programmed. After initial programming, the crossbar requires only minor adjustments to individual rows or columns.

The convenience and the very possibility of using high-frequency signals for crossbar programming depend on the characteristic functions of the resistors. In particular, the most favorable conditions are when the function $F_I^+(I)$ grows faster than the function $-F_I^-(-I)$ as the current I increases [7]. In this case, the entire range of resistor states is available in principle, and it is easy to obtain both positive and negative Δx values. Otherwise, the available range is smaller, there are fewer baseline state options, and the programming time increases because large currents cannot be used to obtain positive Δx offsets. For the $F_0(x), F_x^+(x), F_x^-(x)$ functions, we assumed that conditions (6) are satisfied. If not, high-frequency signals may not be used. However, conditions (6) seem quite natural and were even tested experimentally [6,10].

The programming procedure and its prerequisites are similar for piecewise-constant and harmonic signals [8]. The question arises: which signal is more convenient in real-life applications? On the one hand, harmonic signals are familiar to radio engineers and compatible with AC circuitry

(primarily capacitors), so we can build on the extensive experience in linear RF circuits (for example, we can use various frequency ranges to separate programming and reading). On the other hand, the use of piecewise-constant signals enables more precise and convenient control over the programming process. Unlike a harmonic signal, a piecewise constant signal does not necessarily have a zero mean. In particular, a constant signal is well compatible with a piecewise constant signal, since it is a special case of the latter. For piecewise-constant signals, the result of programming depends linearly on the phase shift (Eq. (37)), in contrast to the nonlinear dependence for harmonic signals [8]. With this, the phase shifts can be found faster and more accurately. Moreover, for advanced digital electronics, the generation of piecewise-constant signals is not too complex.

Piecewise-constant signals are also convenient for analysis and simulation. For given signal values, it is sufficient to know the rate of change of the resistor state only at these voltages; to obtain simple estimates, the factorizability of the right-hand side of the programming equation is not required. There are no problems with modifying the signal waveform, which can be used to identify nonlinear properties of the characteristic functions. The waveform of a harmonic signal is fixed. We can use a sum of harmonic signals [11], but this is a more complicated approach. Note that any signal can be approximated by a piecewise-constant signal. However, this is of no practical use, since with a large number of signal levels, simplicity as its main advantage is lost. For crossbar programming, the simplest piecewise-constant signal consists of two nonzero values applied for equal time intervals. The simplicity of this signal suggests promising practical applications for the proposed crossbar programming method.

This study is a part of the FNEF-2024-0001 Deployment of Trusted AI Systems based on New Mathematical and Algorithmic Approaches and Fast Computing Models Compatible with Domestic Computer Hardware (1023032100070-3-1.2.1) government contract granted to Scientific Research Institute for System Analysis of the National Research Centre Kurchatov Institute, Russian.

References

1. Kotov V.B., Beskhlebnova G.A. Specifics of Crossbar Resistor Arrays. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research VI (NEUROINFORMATICS 2022). Studies in Computational Intelligence. Vol. 1064. Cham: Springer. 2023. PP. 292–304. https://doi.org/10.1007/978-3-031-19032-2_31.
2. Adamatzky A., Chua L. Memristor Networks. Springer International Publishing (2014).
3. Advances in Memristors, Memristive Devices and Systems. / Edited by S. Vaidyanathan and C. Volos. Springer International Publishing AG (2017).
4. Kim S. Ju, Kim S., Jang H.W. Competing memristors for brain-inspired computing. iScience 24, 101889, January 22, 2021.
5. Kotov V.B., Beskhlebnova G.A. Generation of the Conductivity Matrix. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research V (NEUROINFORMATICS 2021). Studies in Computational Intelligence. Vol. 1008. Cham: Springer. 2022. PP. 276-284.
6. Surazhevsky I.A. at all. Noise-assisted persistence and recovery of memory state in a memristive spiking neuromorphic network.Chaos, Solitons and Fractals. 146 (2021). 110890.
7. Beskhlebnova G.A., Kotov V.B. The Variable Resistor Under a High-Frequency Signal. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research VII (NEUROINFORMATICS 2023). Studies in Computational Intelligence. Vol. 1120. Springer Nature Switzerland AG. 2023. PP. 257–266. https://doi.org/10.1007/978-3-031-44865-2_28.
8. Kotov V.B., Beskhlebnova G.A. Use of High-Frequency Signals to Generate a Conductivity Matrix. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research VIII (NEUROINFORMATICS 2024). Studies in Computational Intelligence. Vol. 1179. Springer Nature Switzerland AG. 2025. PP. 265-272.
9. Kotov V.B., Beskhlebnova G.A. Local Point Recording of Information into a Crossbar Resistor Array // SRISA Proceedings. Vol. 14. No. 4 Pp. 33–40 (in Russ.)
10. Kotov V.B., Yudkin F.A. Modeling and Characterization of Resistor Elements for Neuromorphic Systems. Optical Memory and Neural Networks (Information Optics). 2019, v.28, No.4, P. 271-282.
11. V.B. Kotov, Z. B. Sokhova. Two-frequency recording of information into a resistor array. // B. Kryzhanovsky et al. (Eds.). Advances in Neural Computation, Machine Learning, and Cognitive Research IX (NEUROINFORMATICS 2025). Studies in Computational Intelligence. Vol. 1241. Springer Nature Switzerland AG. 2026. PP. 463-476.

Современные достижения и тенденции в области разработки микросхем на основе чиплетов

А. В. Андреев[1], Г. И. Зебрев[2], К. А. Петров[3]

[1]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, alandreev@cs.niisi.ras.ru;

[2]Национальный исследовательский ядерный университет «МИФИ», Москва, Россия, gizebrev@mephi.ru;

[3]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, petrovk@cs.niisi.ras.ru

Аннотация. В обзоре сформулирован ряд проблем, которые решает технология чиплетов при переходе от «систем на кристалле» (СнК) к «системам в корпусе» (СвК). Рассмотрены достижения в области технологии чиплетов и приведены конкретные примеры СвК на ее основе. Рассмотрены ключевые преимущества и границы применимости технологии чиплетов.

Ключевые слова: чиплет, система в корпусе, система на кристалле, сверхбольшая интегральная схема, сложно-функциональные блоки, программируемая логическая интегральная схема, интерпозер, Through-Silicon Via, Redistribution Layer, микроба

1. Введение

Развитие интегральной электроники на протяжении более 60-ти лет заключалось в увеличении степени интеграции, что позволило уменьшать размеры и потребление интегральных схем, повышая их производительность и снижая стоимость. Однако из-за уменьшения технологических норм и физических ограничений технологий фотолитографии значительно возрастает сложность и стоимость проектирования, а также снижается выход годных кристаллов [1]. В качестве ответа на эти вызовы в 10-х годах в мире развивается технология использования вместо одного большого монолитного кристалла отдельных кристаллов, называемых чиплетами, объединенных между собой в систему в корпусе [2, 3]. Каждый чиплет при этом может выполнять свою функцию (ядра процессора, графический сопроцессор или память), и изготавливаться по разным технологическим нормам – такие системы называются гетерогенными. Чиплеты также имеют применение и в гомогенных системах, когда функционал отдельных чиплетов одинаков (например, многоядерный центральный процессор разные ядра которого находятся на отдельных чиплетах) [4, 5].

Целью данного исследования является анализ подходов и методик разработки подложек для корпусов микросхем на основе чиплетов с точки зрения решения технологических проблем современной микроэлектроники.

Статья организована следующим образом: во втором разделе обосновывается актуальность проблем, решаемых с использованием технологии чиплетов; в третьем рассматриваются достижения в области технологии чиплетов и приведены конкретные примеры СвК на ее основе, в четвертом - границы применимости технологии чиплетов.

2. Актуальные проблемы, решаемые с помощью технологии чиплетов

Стоимость и длительность разработки и изготовления сверхбольшой интегральной схемы (СБИС) зависит от комплекса технических и технологических факторов, таких как уменьшение проектных норм, увеличение площади кристалла и сложности составных сложно-функциональных (СФ) блоков, что ведет к росту затрат на разработки и производство, а также к снижению выхода годных изделий.

С ростом сложности задач, стоящих перед индустрией разработки вычислительных систем необходимо разрабатывать и изготавливать все более сложные СБИС. Такие СБИС необходимо производить по передовым проектным нормам для обеспечения растущих требований по параметрам энергопотребления, размерам и набору, и типу функциональных ядер и интерфейсов. Снижение проектных норм изготовления СБИС резко увеличивает стоимость разработки и изготовления отдельной системы-на-кристалле (СнК), что показано на рисунке 1.

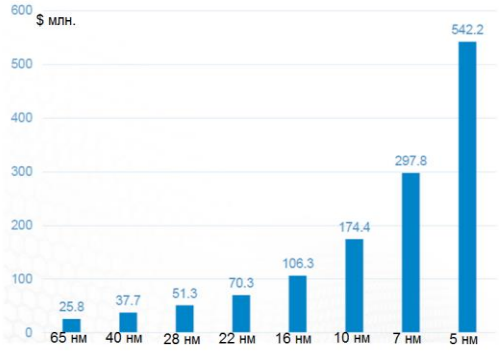

Рис. 1. Стоимость разработки микросхем с уменьшением проектных норм [6]

Увеличение площади кристалла приводит к снижению выхода годных, а значит – к росту стоимости конечных изделий. На рисунке 2 показано влияние перехода к чиплетному дизайну, по сравнению с монолитной СБИС [7].

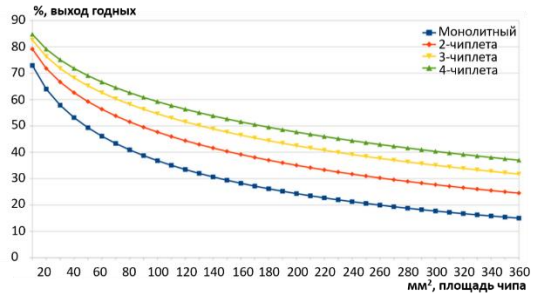

Рис. 2. Зависимость выхода годных от площади изготавливаемого кристалла

Таким образом, монолитный кристалл площадью 200 мм² будет иметь выход только 24%. Однако микросхема на основе четырех чиплетов, каждый из которых имеет площадь 50 мм², будет иметь выход 49%, что более чем вдвое превышает выход монолитного кристалла.

Использование большего количества СФ-блоков при проектировании повышает вероятность возникновения ошибок в СБИС. Их исправление требует дорогостоящего перепроизводства (перезапуска) всего кристалла. Это приводит к необходимости применения дорогостоящих средств отладки и верификации на всех этапах проектирования. Помимо стоимости, большое количество перезапусков увеличивает длительность цикла разработки новых СБИС и вычислительных систем на их основе.

Эти проблемы особенно актуальны при разработке российскими дизайн-центрами в условиях ограниченной серийности изделий для отечественного микроэлектронного рынка.

Технология чиплетов позволяет применить следующие решения при создании СБИС:

1. Разбиение СБИС большой площади на отдельные кристаллы – чиплеты. Изготовление кристаллов меньшей площади позволяет достигнуть большего выхода годных (рисунок 2), а значит, существенно снизить стоимость микросхемы, изготавливаемой как «система-в-корпусе» по сравнению с микросхемой, изготавливаемой как «система-на-кристалле» (при одинаковом функционале и сравнимой общей площади кристаллов). При малом размере отдельных кристаллов также возможно изготовление на одной полупроводниковой пластине нескольких различных разработанных кристаллов, что дополнительно снижает стоимость изготовления комплекта фотошаблонов при мелкосерийном производстве.
2. Использование различных проектных норм для изготовления чиплетов. Разбиение на отдельные чиплеты позволяет использовать различные проектные нормы при изготовлении различных частей «системы-в-корпусе». В результате возможно изготовление СБИС, оптимальной по функциональным характеристикам и стоимости, содержащей, например, центральный вычислительный чиплет, произведенный по предельной технологической норме для обеспечения максимальной производительности и ответные чиплеты, произведенные по более грубым проектной нормам для снижения их стоимости.
3. Изготовление СФ-блоков в виде отдельных чиплетов. Изготовление СФ-блока собственной разработки в качестве отдельного чиплета позволяет снизить стоимость тестового запуска и отладки при условии его совместимости с универсальным центральном чиплетом. Это позволяет снизить вероятность ошибок при запуске СФ-блока в составе полноценного изделия и сократить стоимость разработки изделия.

Таким образом, применение технологий чиплетов создает предпосылки для значительного снижения стоимости, сроков проектирования и изготовления СБИС для вычислительных систем. Теоретические выгоды чиплетного подхода, обоснованные в разделе 2, нашли свое практическое подтверждение и развитие в ряде успешных коммерческих и исследовательских изделий. Рассмотрим способы применения чиплетов в современных процессорах и вычислительных системах, начиная с исторически значимых и заканчивая передовыми решениями, демонстрирующими эволюцию и многообразие новых методов и технологий в микросхемах на основе чиплетов.

3. Примеры применения чиплетов в процессорах

3.1. Первый успешный пример использования чиплетов

Первая реализация чиплетной технологии в её современном виде была осуществлена компанией Xilinx в 2011 году, когда она перевела свои передовые программируемые логические интегральные схемы (ПЛИС) на чиплетную архитектуру. Этот шаг не только решил конкретную проблему (низкий выход годных), но и открыл новый вектор развития всей отрасли. Основная трудность заключалась в том, что при производстве нового поколения ПЛИС в виде системы на кристалле (SoC) с использованием 28-нм технологического процесса на фабрике TSMC выход годных чипов оказался крайне низким из-за значительных размеров кристалла. Оптимизация дизайна SoC, заключающаяся в разделении ПЛИС на четыре меньших кристалла — чиплета, позволила повысить выход годных кристаллов, что сделало новое поколение ПЛИС экономически эффективным.

После производства чиплеты корпусировались с применением новой технологии CoWoS (Chip-on-Wafer-on-Substrate), также разработанной TSMC в 2011 году. Эта технология представляет собой 2.5D - интеграцию, обеспечивающую как вертикальное, так и преимущественно горизонтальное соединение между четырьмя чиплетами. Для связи чиплетов (28 nm FPGA Die Slice) между собой и с подложкой (Package Substrate) используется интерпозер с TSV и RDL (рисунок 3).

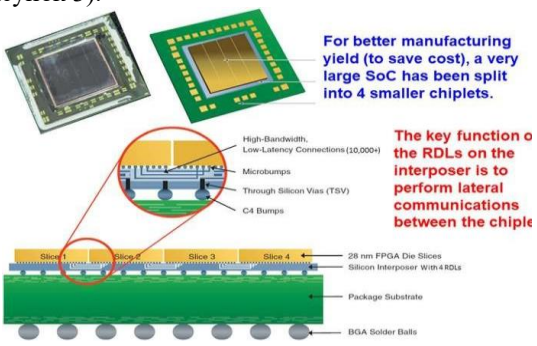

Рис. 3. Разделение крупного кристалла FPGA на 4 чиплета от Xilinx/TSMC [7]

Список технологий, примененных в этой микросхеме:

- интерпозер – дополнительная кремниевая подложка между коммутационной платой микросхемы и кристаллом с переходными отверстиями типа TSV и слоями разводки RDL;
- TSV (Through-Silicon Via) – сквозные металлизированные отверстия в кремнии;
- RDL (redistribution layer) – слой металлизации в интерпозере, который обеспечивает соединения между чиплетами и корпусом микросхемы.

Минимальный шаг четырёх слоёв перераспределения (RDL) на интерпозере составляет 0,4 мкм, что видно на рисунке 4.

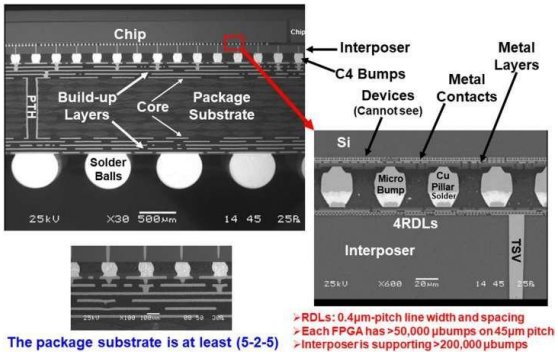

Рис. 4. 2,5D интеграция ИС от Xilinx/TSMC [7]

В 2013 году Xilinx и TSMC совместно объявили о запуске серийного производства семейства чипов Virtex-7 HT с использованием 28-нм технологического процесса. Это событие стало значимым моментом в индустрии как первое серийное производство с дизайном чиплетов. С тех пор было представлено множество успешных микросхем с использованием дизайна чиплетов от ведущих мировых компаний. В последующих разделах данного обзора будут подробно рассмотрены некоторые из них.

3.2. Технология серверного процессора EPYC

В середине 2019 года AMD представила второе поколение процессоров EPYC (Extreme-Performance Yield Computing) серии 7002 под кодовым названием Rome, удвоив количество ядер до шестидесяти четырех. Второе поколение EPYC представляет собой класс серверных процессоров, устанавливающих более высокие стандарты для центров обработки данных. В серверном продукте Rome используется большая структура слоев подложки 9–2–9 (рисунок 5). Один из слоев с разводкой сигналов показан на рисунке 6 вместе с физическим расположением вычислительных кристаллов – CCD (CPU Complex Die), кристалла ввода-вывода – IOD (IO Die), а также основных внешних интерфейсов DRAM (Dynamic Random Access Memory) и SerDes (Serialization/Deserialization).

В системе на кристалле (SoC) AMD EPYC реализуется чиплетная структура, в которой передовые и дорогостоящие технологии, такие как 7-нм процесс на базе TSMC, применяются

исключительно для CPU-ядер, в то время как I/O и интерфейсы памяти продолжают производиться по более старой технологии (14-нм, GlobalFoundries).

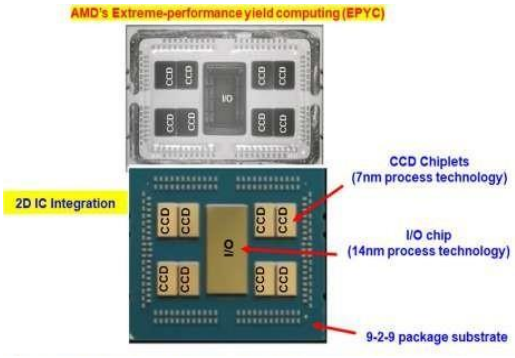

Рис. 5. Чиплетный дизайн процессора EPYC от AMD [8]

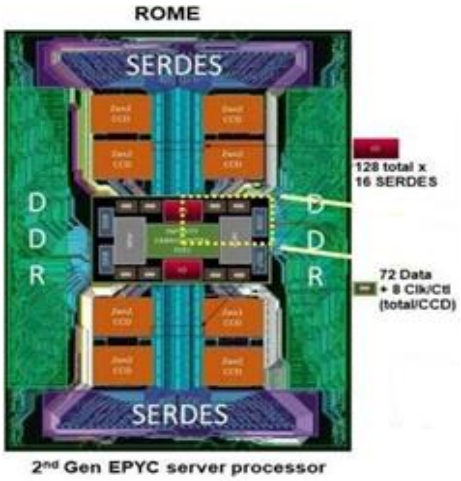

Рис. 6. Топология подложки процессора EPYC от AMD [9]

Необходимость сохранить неизменными размер корпуса и расположение выводов EPYC обуславливает важность тесного взаимодействие между разработчиками кристаллов и корпуса. Это особенно актуально с учетом увеличения числа чиплетов с четырех в первом поколении EPYC до девяти во втором. Такой стратегический подход позволяет снижать затраты и значительно повышать выход качественных чиплетов за счет применения более компактных чиплетов. Однако, по мере роста числа ядер и сложности вычислительных задач, традиционная двумерная компоновка чиплетов стала наталкиваться на физические ограничения, связанные прежде всего с растущими задержками доступа к кэш-памяти L3 и ограниченной пропускной способностью межкристальных соединений. Потребовались новые, еще более радикальные подходы к интеграции.

3.3. Технология 3D V-Cache

На конференциях IEEE/ISSC 2022 [10] и IEEE/ECTC 2022 [11] компания AMD представила свой дизайн чиплетов 3D V-Cache, которая схематично изображена на рисунке 7.

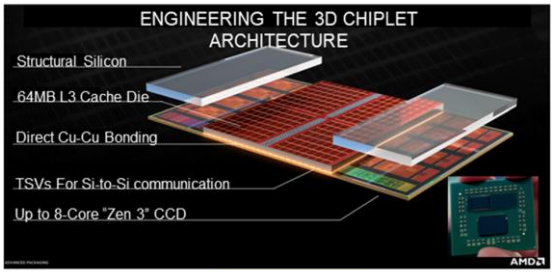

Рис. 7. 3D V-Cache от AMD [11]

Нижний кристалл (81 мм²) — это вычислительный процессор «Zen 3», изготовленный по 7-нм технологическому процессу TSMC. Верхний кристалл (41 мм²) — это расширенный кэш L3 (SRAM), изготовленный по тому же процессу. Нижний кристалл с TSV расположен лицевой стороной вниз с использованием C4-бампов. Верхний кристалл также расположен лицевой стороной вниз и соединен с нижним кристаллом методом прямого соединения чипов Cu-Cu (Hybrid Bonding) при помощи микробампов.

На рисунке 8 также можно заметить использование структурных кремниевых кристаллов по бокам над «Zen 3» CCD (Core Complex Die) для обеспечения баланса и плоскостности сборки, что приводит к лучшему отводу тепла от нижнего вычислительного кристалла.

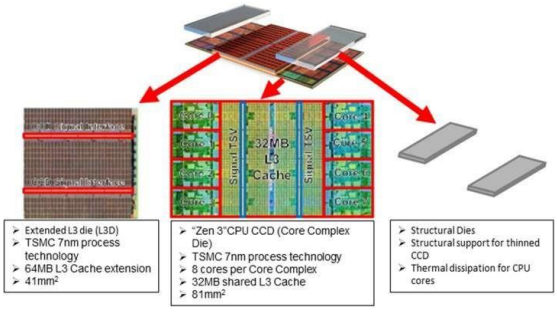

Рис. 8. Расположение кристаллов в 3D V-Cache от AMD [12]

На рисунке 9 показан процесс соединения, выполненный по технологии SoIC (system on integrated chips) от TSMC. Минимальный шаг микробампов при использовании метода прямого соединения чипов Cu-Cu составляет 9 мкм.

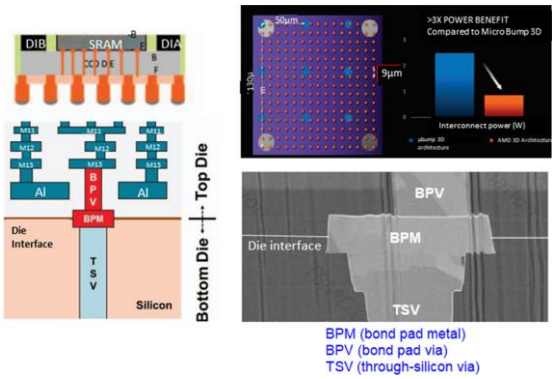

Рис. 9. Гибридное соединение SRAM (с лицевой стороны) и процессорного чипа (с обратной стороны) по технологии SoIC от TSMC для 3D V-Cache AMD [12]

Эта технология устанавливает новый стандарт плотности вертикальных соединений для специализированных пар кристаллов. Однако, масштабирование такого прямого соединения на большее количество разнородных кристаллов в единой 3D-структуре представляет значительную сложность.

3.4. Технология Foveros

Для решения задачи интеграции множества разнородных компонентов (CPU, память, FPGA, ускорители) в сложные гетерогенные 3D-системы Intel предложила альтернативную архитектурную платформу Foveros, анонсированную в декабре 2018 года. Ее основу составляет активный TSV-интерпозер (рисунок 10), который подобен кристаллу и соединен методом «face-to-face» микробампами с шагом 50 мк с чиплетами или системой на кристалле (SoC).

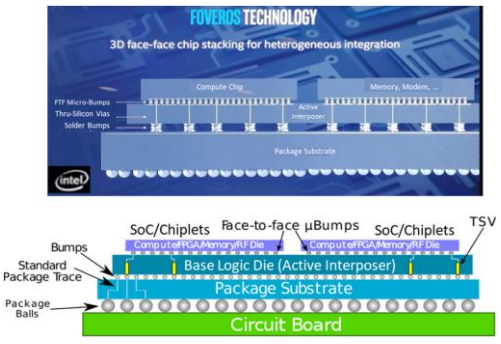

Рис. 10. 3D интеграция ИС Foveros от Intel [13]

Для ODI Type 1 (рисунок 11(а)): два активных TSV-интерпозера (кристаллов) расположены между большим Compute/FPGA/Memory кристаллом и подложкой. Сверху и снизу интерпозеров расположены микробампы, которые соединяют их с верхним кристаллом и подложкой корпуса.

Для ODI Type 2 (рисунок 11(б)): активный TSV-интерпозер (мост, кристалл) расположен между двумя двумя чиплетами Compute/FPGA/Memory и подложкой. Таким образом, этот интерпозер выполняет роль подложки для интерконнекта между двумя верхними кристаллами.

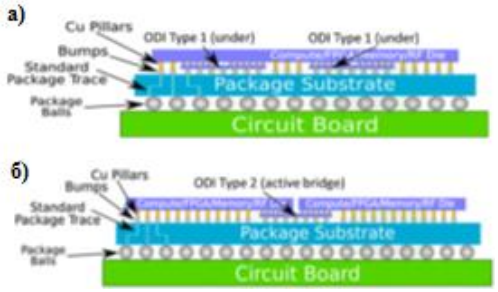

Рис. 11. ODI [14]: (а) - тип 1; (б) - тип 2

Intel также анонсировала интерфейс Management Data Input/Output (MDIO) для межкристального взаимодействия, который призван заменить текущую шину Advanced Interface Bus (AIB). Все эти гетерогенные интеграции направлены на достижение экстремально высокой производительности, а активные TSV-интерпозеры помогают достичь еще более высоких показателей [15-17].

3.5. Особенности мобильного процессора Lakefield

Первым и знаковым практическим воплощением технологии Foveros на массовом рынке стал мобильный процессор Intel Lakefield, поставки которого начались в июле 2020 года. Этот процессор для ноутбуков наглядно демонстрирует, как архитектурные принципы Foveros применяются для создания компактной и энергоэффективной SoC. Система на кристалле (SoC) разделена на функциональные блоки (CPU, GPU, LPDDR4 и т.д.), а CPU дополнительно поделен на один крупный и четыре малых, как показано на рисунке 12. Чиплеты в нижней части сборки соединены микробампами методом «face-to-face». Нижний кристалл является активным TSV-интерпозером. Соединение между интерпозером и подложкой корпуса выполняется с помощью бампов C4, а между подложкой и печатной платой – BGA-шариками (Ball Grid Array).

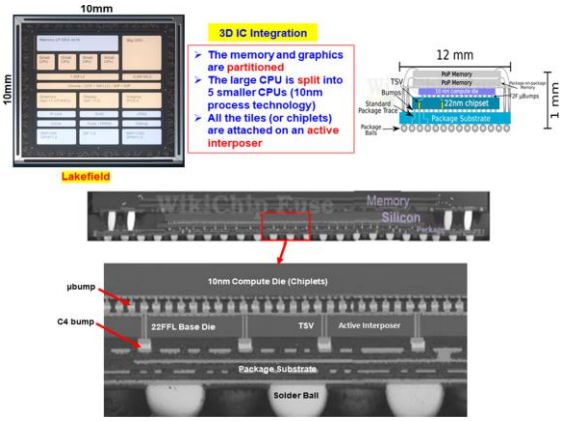

Рис. 12. Структура и схематичное изображение Lakefield от Intel [18]

Конечный формат корпуса представляет собой структуру PoP (корпус на корпусе) размером 12 мм × 12 мм × 1 мм (рисунок 12). Дизайн чиплетов и гетерогенная интеграция реализованы в нижнем корпусе, тогда как верхний корпус содержит память, соединённую по технологии проволочного монтажа (wire bonding). Изготовление верхнего кристалла выполнено по 10-нм технологическому процессу Intel, а нижнего – интерпозера – по 22-нм. Стоит отметить, что это первое в истории высокообъёмное производство процессоров для мобильных устройств (например, ноутбуков), реализованное с применением трёхмерной интеграции чиплетов.

Значение Lakefield как первого массового процессора с 3D-интеграцией и практической демонстрации технологии Foveros трудно переоценить. Его реализация с использованием микробампов с шагом 50 мкм обозначила потенциал для дальнейшего увеличения плотности межкристальных соединений и снижения задержек – ключевых параметров для будущих высокопроизводительных и компактных решений на базе Foveros.

3.6. Технология Foveros-Direct

Уже через месяц после начала поставок Lakefield, в августе 2020 года на Intel Architecture Day, компания анонсировала ключевое направление эволюции Foveros – внедрение гибридного бондинга (Cu-Cu hybrid bonding) для прямого соединения кристаллов. На конференции IEEE Hot Chip Conference (август 2021 года) эта технология уже была представлена под названием FOVEROS-Direct [16], где Intel продемонстрировала, что при использовании гибридного соединения (bumpless hybrid bonding) шаг соединений может быть уменьшен до 10 мкм, в отличие от 50 мкм, как в случае с процессором Lakefield, что показано на рисунке 13.

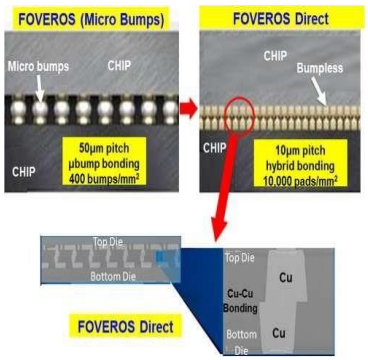

Рис. 13. Гибридное соединение Foveros-Direct [17]

3.7. Технология графического процессора Ponte Vecchio

Развитие технологий гибридной сборки, анонсированных в Foveros-Direct, нашло свое наиболее масштабное и комплексное воплощение в графическом процессоре Ponte Vecchio, или, как его называют, «космический корабль среди GPU» [19, 20]. Этот процессор стал крупнейшим и наиболее сложным проектом с использованием чиплетов на сегодняшний день, что показано на рисунках 14-16. Процессор Ponte Vecchio был выпущен в 2023 году, однако уже в мае 2024 года компания приняла решение о прекращении его производства. Intel намерилась сосредоточиться на более конкурентоспособных и актуальных на данный момент ускорителях, ориентированных для задач искусственного интеллекта.

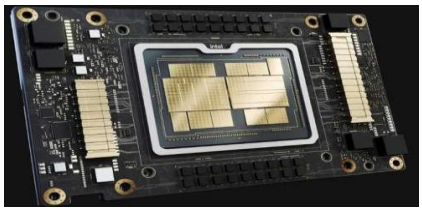

Рис. 14. Графический процессор Ponte Vecchio компании Intel [20]

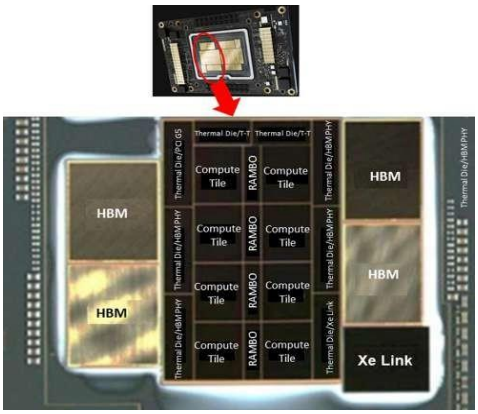

Рис. 15. Расположение чиплетов выделенной зоны процессора Ponte Vecchio [21]

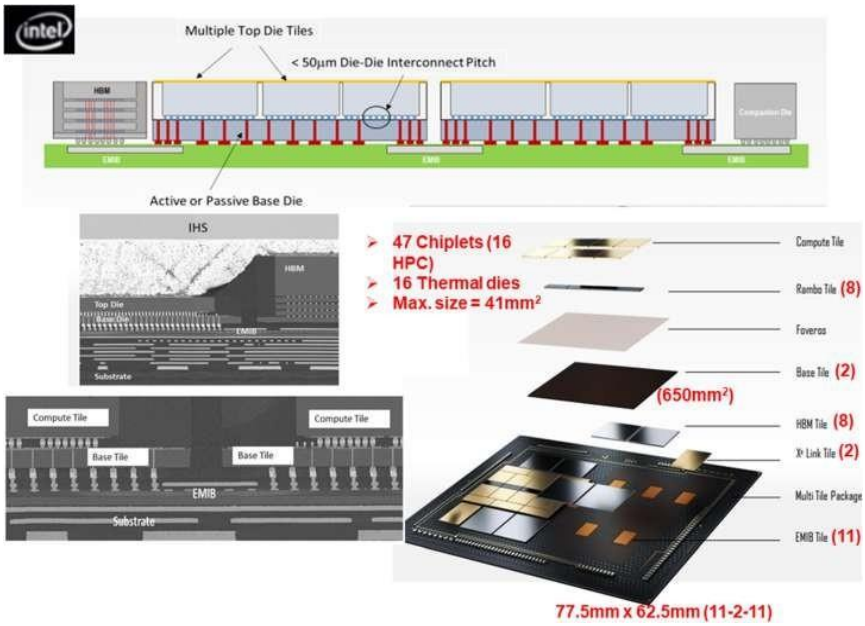

Рис.16. Расположение чиплетов выделенной зоны процессора Ponte Vecchio [21]

В графическом процессоре Ponte Vecchio используется несколько ключевых технологий, которые обеспечивают работу 47 различных функциональных кристаллов, основанных на разных технологических процессах и архитектурах. Также дополнительно используется 16 термокристаллов, которые обеспечивают более равномерное распределение тепла от активных чипов. В составе процессора присутствуют два нижний (базовых) кристалла (642 мм^2 каждый), 16 вычислительных чиплетов, восемь кристаллов кеш-памяти RAMBO, столько же стеков памяти HBM и рядом расположены два кристалла шины Xe Link. За связь между всеми чиплетами в графическом процессоре отвечает шина EMIB (11 кристаллов). EMIB представляет собой ключевую технологию межкристального соединения, обеспечивающую высокоскоростную коммуникацию между отдельными чиплетами. Детальное изображение EMIB показано на рис. 16. В процессоре в основном используются чиплеты с собственным 7-нм технологическим процессом от Intel, но компания также использовала некоторые вычислительные кристаллы на сторонних фабриках (например, TSMC с их 5-нм процессом). Ключевые параметры процессора отображены в таблице 1.

Таблица 1. Ключевые характеристики GPU Ponte Vecchio.

Кол-во чиплетов	47 функциональных+16 термокристаллов
Кол-во слоев подложки	11-2-11 (24)
Габариты подложки	77,5 x 62,5 мм (4844 мм^2)
Кол-во выводов подложки	4468
Общая площадь чипов (с термокристаллами / без них)	3100 мм^2/2330 мм^2
Общее кол-во транзисторов в СвК	Более 100 млрд
Шаг микробампов, соединяющих кристаллы	36 мкм
Максимальная площадь верхнего кристалла	41 мм²
Площадь нижнего кристалла	650 мм²
Потребляемая мощность СвК	600 Вт

4.Заключение

В статье были рассмотрены современные достижения и актуальные тенденции в области разработок микросхем на основе чиплетов от крупных коммерческих компаний с 2011 до 2024 год.

Технология чиплетов дает возможность снизить затраты при проектировании (вместо нового монолитного чипа можно использовать готовые чиплеты), увеличить выход годных (маленькие размеры чиплета снижают вероятность дефекта), увеличить гибкость при производстве (разные чиплеты могут быть изготовлены по разным технологическим процессам, оптимизированным под конкретные задачи).

Границами применимости в разработке микросхем на основе чиплетов являются:

1. Повышенная технологичность изготовления подложек для корпусов микросхем.
2. Увеличение площади подложки из-за разделения монолитного кристалла на чиплеты.
3. Повышается сложность разработки корпуса из-за высоких требований применения новых интерфейсов и обеспечения их электрических характеристик.
4. Традиционные методики разработки корпуса микросхемы менее подходят для работы с чиплетами, что требует модернизации САПР.

Ключевые преимущества использования чиплетов по сравнению с монолитными системами на кристалле (SoC):

1. Увеличение выхода годных кристаллов в процессе производства из-за меньшего размера чипа.
2. Размещение меньших кристаллов на подложке ведет к равномерному и более эффективному распределению тепла в корпусе микросхемы.
3. Применение чиплетов позволяют обновлять или модифицировать многокристальную сборку, так как можно легко заменить или улучшить отдельный чиплет без необходимости полного перепроектирования всей микросхемы.

Чиплетные сборки становятся ключевым подходом в разработке высокопроизводительных микросхем, особенно с учётом растущих затрат на проектирование и производство при переходе к более тонким техпроцессам. Эта технология позволяет значительно улучшить выход годных кристаллов, снизить стоимость и обеспечить гибкость при разработке сложных систем в корпусе.

Публикация выполнена в рамках государственного задания НИЦ "Курчатовский институт" - НИИСИ по теме № FNEF-2024-0003.

Modern Achievements and Trends in the Development of Chiplet-Based Integrated Circuits

A. V. Andreev, G. I. Zebrev, K. A. Petrov

Abstract. This review identifies a number of challenges that chiplet technology addresses in the transition from systems-on-chip (SoC) to systems-in-package (SIP). Advances in chiplet technology are examined, and specific examples of SIP-based systems are given. The key advantages and limits of applicability of the chiplet technology are considered.

Keywords: chiplet, system-in-package, system-on-chip, very-large-scale integration, complex functional blocks, Field-Programmable Gate Array, interposer, Through-Silicon Via, Redistribution Layer, microbump

Литература

1. Shalf, J. (2020). The future of computing beyond moore's law. Phil. Trans. R. Soc. A., 378(2166), 20190061. https://doi.org/10.1098/rsta.2019.0061

2. Бобков С.Г. Технология чиплетов - перспективное направление развития российской микроэлектроники // Электронная техника. Серия 3: Микроэлектроника. 2022. № 1 (185). С. 42-51.

3. Liu, Y., Li, X. & Yin, S. Review of chiplet-based design: system architecture and interconnection. Sci. China Inf. Sci. 67, 200401 (2024). https://doi.org/10.1007/s11432-023-3926-8.

4. J. Lan, V. P. Nambiar, R. Sabapathy, M. D. Rotaru and A. T. Do, "Chiplet-based Architecture Design for Multi-Core Neuromorphic Processor," 2021 IEEE 23rd Electronics Packaging Technology Conference (EPTC), Singapore, Singapore, 2021, pp. 410-412.

5. S. Naffziger et al., "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families: Industrial Product," 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 2021, pp. 57-70.

6. Neuffer, J., 2019, "An Industry Perspective on Government Action: The DARPA Electronics Resurgence Initiative," Semiconductor Industry Association, Washington.

7. Netronome, 2020 "It's Time for Disaggregated Silicon!," Netronome, CranberryTwp., PA, accessed Oct. 2022, https://www.netronome.com/blog/its-timedisaggregated-silicon/

8. Naffziger, S., Lepak, K., Paraschour, M., and Subramony, M., 2020, "AMD Chiplet Architecture for High-Performance Server and Desktop Products," IEEE/ISSCC Proceedings, Virtual conference, Feb. 16–20, pp. 44–45.

9. Naffziger, S., 2020, "Chiplet Meets the Real World: Benefits and Limits of Chiplet Designs," Symposia on VLSI Technology and Circuits, Virtual conference, June 14–19, pp. 1–39.

10. Wuu, J., Agarwal, R., Ciraula, M.,Dietz, C., Johnson, B., Johnson, D., Schreiber, R., et al., 2022, "3D V-CacheTM: The Implementation of a Hybrid-Bonded 64MB Stacked Cache for a 7 nm _86-64 CPU," Proceedings of IEEE/ISSCC, San Francisco, CA, Feb. 20–26, pp. 1–36.

11. Agarwal, R., Cheng, P., Shah, P.,Wilkerson, B., Swaminathan, R., Wuu, J., and Mandalapu, C., 2022, "3D Packaging for Heterogeneous Integration," IEEE/ ECTC Proceedings, San Diego, CA, May 31–June 3, pp. 1103–1107.

12. Swaminathan, R., 2022, "The Next Frontier: Enabling Moore's Law Using Heterogeneous Integration," Chip Scale Rev., 26(3), pp. 11–22.

13. Prasad, C., Chugh, S., Greve, H., Ho, I., Kabir, E., Lin, C., Maksud, M., et al., 2020, "Silicon Reliability Characterization of Intel's Foveros 3D Integration Technology for Logic-on-Logic Die Stacking," Proceedings of IEEE Interna tional Reliability Physics Symposium, Grapevine, TX, Apr. 28–May 3, pp. 1–5.

14. Wade, M., Anderson, E., Ardalan, S., Bhargava, P., Buchbinder, S., L. Davenport, M., Fini, J., et al., 2020, "TeraPHY: A Chiplet Technology for LowPower, High-Bandwidth in-Package Optical I/O," IEEE Comput. Soc., 40(2), pp. 63–71.

15. Ingerly, D., Amin, S., Aryasomayajula, L., Balankutty, A., Borst, D., Chandra, A., Cheemalapati, K., et al., 2019, "Foveros: 3D Integration and the Use of Face to-Face Chip Stacking for Logic Devices,"

IEEE/IEDM Proceedings, San Francisco, CA, Dec. 7–11, pp. 19.6.1–19.6.4.

16. Gomes, W., Khushu, S., Ingerly, D., Stover, P., Chowdhury, N., O'Mahony, F., Balankutty, A., et al., 2020, "Lakefield and Mobility Computer: A 3D Stacked 10nm and 2FFL Hybrid Processor System in 12x12 mm2, 1mm Package-on Package," IEEE/ISSCC Proceedings, Hiroshima, Japan, Nov. 9–11, pp. 40–41.

17. Mahajan, R., and Sane, S., 2021, "Advanced Packaging Technologies for Heterogeneous Integration," Proceedings of IEEE Hot Chip Conference, Palo Alto, CA, Aug. 22–24, pp. 1–44.

18. Intel, 2020, Intel Architecture Day, Intel, Santa Clara, CA.

19. Gomes, W., Koker, A., Stover, P., Ingerly, D., Siers, S., Venkataraman, S., Pelto, C., Shah, T., et al., 2022, "Ponte Vecchio: AMulti-Tile 3D Stacked Processor for Exascale Computing," Proceedings of IEEE/ISSCC, San Francisco, CA, Feb. 20–26, pp. 42–44.

20. Gelsinger, P., 2021, Engineering the future, Intel Unleashed Webcast, Intel, Santa Clara, CA.

21. Sheikh, F., Nagisetty, R., Karnik, T., and Kehlet, D., 2021, "2.5D and 3D Heterogeneous Integration – EmergingApplications," IEEE Solid-State Circuits Mag., 13(4), pp. 77–87.

Обзор трансформируемой архитектуры системы на кристалле MONARCH

П. С. Остапенков[1], А. М. Чупрунов[2]

[1]ФГБОУ ВО «НИУ «МЭИ», Москва, Россия, OstapenkovPS@mpei.ru;

[2]ФГБОУ ВО «НИУ «МЭИ», Москва, Россия, ChuprunovAM@mpei.ru;

Аннотация. В статье приводится обзор архитектуры системы на кристалле MONARCH, которая была представлена в 2007 году исследователями и аспирантами Университета Южной Калифорнии с участием инженеров компании Raython по гранту Агентства перспективных исследований министерства обороны США, и которая до сих пор остаётся актуальной. Описываются основные характеристики, компоненты архитектуры и режимы работы полиморфного процессора. Приводятся примеры применения MONARCH в модульных многопроцессорных измерительно-управляющих системах компании Raython.

Ключевые слова: MONARCH, система на кристалле, полиморфный процессор, высокопроизводительные вычисления, цифровая обработка сигналов, энергоэффективность, модульность, автономность, бортовые измерительно-управляющие системы

1. Проект MONARCH

Проект MONARCH [1] финансировался агентством Министерства обороны США по перспективным научным исследованиям (англ. DARPA) с начала 2000-х г. в рамках программы «Полиморфная компьютерная архитектура», - грант DARPA BAA 00-59 от 2000 года в сфере исследований: автоматизированные системы радиоэлектронной борьбы (РЭБ), включая исследования в области «Искусственный интеллект для радиоэлектронной борьбы» (англ. AI-EW).

2. Разработка

По гранту DARPA в США на I и II этапах в программе «Полиморфная компьютерная архитектура» принимали участие Институт информационных наук Университета Южной Калифорнии (англ. USC-ISI) совместно с группой передовых концепций и технологий Raytheon Space and Airborne Systems. В рамках реализации программы была разработана трансформируемая архитектура MONARCH (англ. Morphable Networked Micro-Architecture).

2.1. Университет и промышленность

В число разработчиков от университета USC-ISI под руководством директора отдела перспективных систем Джона Гранацки (англ. John Granacki) входили две команды исследователей возглавляемые Джеффом ЛаКоссом (англ. Jeff LaCoss) и Джеффом Дрейпером (англ. Jeff Draper). Команда Джеффа Дрейпера из двух инженеров и четырёх аспирантов внёсла значительный вклад в разработку микросхемы MONARCH. Исследователи разработали и реализовали: RISC-процессоры (англ. Reduced Instruction Set Computer), сеть на кристалле с маршрутизаторами пакетов (англ. packet buffers - PBufs), блоки с плавающей запятой в арифметических кластерах FPCA (Field Programmable Computer Array), интерфейс Flash (с первоначальной загрузкой системы на кристалле) и встроенные контроллеры DRAM (англ. Dynamic Random-Access Memory) .

Помимо команд университета USC-ISI, в разработке участвовали исследователи и представители из других институтов и предприятий США: Exogi Inc., Технологический институт Джорджии (англ. Georgia Tech), причём один из ведущих инженеров James Kulp совмещал работу в Mercury Computer Systems и в подразделении IBM Global Engineering Solutions.

2.2. Результаты

По гранту с DARPA на III этапе необходимо было реализовать прототип микросхемы. В ходе предварительных испытаний прототип микросхемы «система на кристалле» MONARCH обеспечил:

- производительность на операциях с 32-битными числами с плавающей запятой 64 Гфлопс/с,
- на операциях обращения к внешней памяти более 60 Гбайт/с,
- пропускную способность межсистемных обменов вне микросхемы более 43 Гбайт/с.

Со стороны исполнителя Университета USC-ISI руководитель работ Джон Гранацки отметил: «То, что мы создали, — это, по сути, суперкомпьютер на кристалле, и не просто суперкомпьютер, а гибкий суперкомпьютер,

который перестраивается в оптимальный суперкомпьютер для каждой конкретной части многоэтапной задачи», а Джефф Дрейпер добавил: «На момент внедрения чип MONARCH был самой большой стандартной ASIC, которую IBM когда-либо создавала по 90-нанометровой технологии» [1].

По словам Джона Гранацки, полиморфные возможности и сверхэффективность MONARCH позволяют разрабатывать системы для Министерства обороны США, которые должны быть очень компактными, высокопроизводительными, малопотребляющими, а в некоторых случаях (особенно для систем, используемых в космосе) устойчивыми к радиации.

Со стороны заказчика компании Raython комментарии руководителей были превосходными. Как объяснил Ник Урос (Nick Uros), вице-президент группы перспективных концепций и технологий компании Raytheon Space and Airborne Systems: «Микроархитектура MONARCH уникальна своей способностью перестраиваться для оптимизации обработки данных на лету. MONARCH обеспечивает исключительную вычислительную мощность и гибкую пропускную способность для передачи данных, а также высочайшую энергоэффективность и полную программируемость», главный исследователь проекта в Raytheon Майкл Вейи (Michael Vahey) добавил про энергоэффективность: «MONARCH превзошёл четырёхъядерный процессор Intel Xeon в 10 раз» [1].

В дальнейшем данную разработку использовали для создания высокопроизводительных компьютерных систем бортового назначения, поскольку «система на кристалле» обладает наилучшими характеристиками по стоимости вычислений на Ватт подводимой электроэнергии.

3. Технология изготовления

Микросхема MONARCH была изготовлена компанией IBM по технологии CMOS (англ. Complementary Metal–Oxide–Semiconductor) по техпроцессу 90 нм с 8-ю медными слоями, на которых можно разместить до $72 \cdot 10^6$ логических элементов [2]. Размер микросхемы 18,76 x 18,76 мм.

4. Энергоэффективность и пиковая производительность

При работе на частоте 333 МГц (первые экземпляры) пиковая производительность составила 64 млрд. операций с 32-разрядными числами с плавающей запятой в секунду или 64 Гфлопс/с при номинальном потреблении 8-50 Вт. Энергоэффективность в расчёте на 1 Вт составляла от 3 до 6 Гфлопс/Вт [1].

5. Реконфигурируемый массив вычислительных узлов

Базовый вычислительный узел «системы на кристалле» MONARCH состоит из четырёх взаимосвязанных компонентов:

- многопоточного RISC-процессора,
- встроенной динамической памяти (англ. EDRAM, Embedded DRAM),
- буфера пакетов кольцевой сети (анг. PBuf Ring),
- интерфейса FPCA-узла (англ. Array Node Bus Interface, ANBI).

Объединённая структура базовых вычислительных узлов показана на рис. 1.

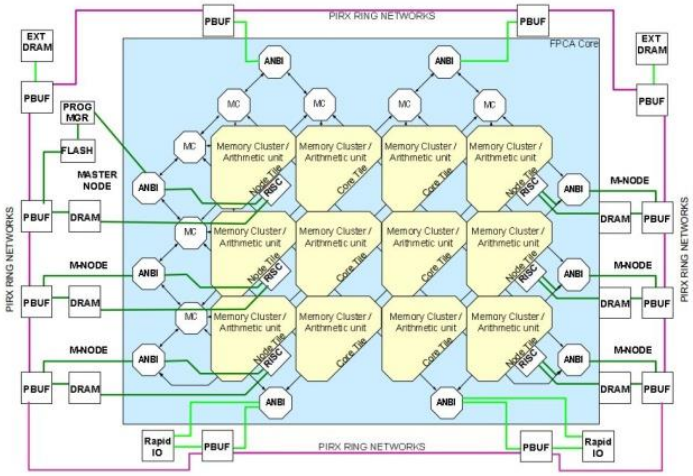

Рис. 1. Архитектура системы на кристалле MONARCH [3]

На рисунке 1 обозначены:

PBuf — это встроенный сетевой маршрутизатор пакетов, циркулирующих по двунаправленному кольцу между вычислительными узлами; ANBI-интерфейс соединяет основную часть FPCA с PBuf, интерфейс предоставляет вычислительным потокам порт для доступа к памяти других вычислительных узлов, находящихся в кольце (англ. PIRX RING NETWORK). Через PBuf осуществляется дистанционный прямой доступ (англ. RDMA - Remote Direct Memory Access) к внешней памяти (англ. ExtDRAM – External DRAM) и к бортовому интерфейсу RapidIO, при этом поддерживается согласованность кэшей (англ. Cashe Coherence).

6. Трансформируемые кольцевые сети

В основе архитектуры MONARCH лежит двунаправленная кольцевая сеть из 12 узлов PBuf, обладающая способностью к трансформации, как внутри микросхем MONARCH, так и к объединению этих микросхем в единую многопроцессорную измерительно-управляющую систему. Основной принцип функционирования двунаправленной кольцевой сети - сохранение работоспособности при выходе из строя линий связи или узлов сети с трансформацией из кольца в цепочку или цепочки узлов. Такое трансформирование колец с узлами различного назначения было использовано в стандартах IEEE-802.17 на региональные информационные сети и в нестандартных фирменных суперкомпьютерных решениях. Американские Cray XK7, IBM Blue Gene/Q имеют топологии 3D-тор и 5D-тор соответственно, и были созданы после воплощения MONARCH.

Разработанная в 2013 году российская сеть АНГАРА от НИЦЭВТ, подобная американским решениям, имела топологию 4D-тор, и воплощена в СБИС ЕС8430 [4]. Изготовленная по 68-нм техпроцессу СБИС ЕС8430 имеет размеры 13 х 10,5 мм, и, потребляя 36 Вт, обеспечивает пропускную способность 8 Гбайт/с. При этом пропускная способность Cray XK7 составляет 9.6 Гбайт/с, а IBM примерно 20 Гбайт/с [4]. Это почти двукратное превосходство IBM Blue Gene/Q, может быть объяснено участием инженеров от IBM в более ранней разработке MONARCH.

Кольцевая сеть, реализованная в MONARCH, не имеет аналогов среди систем на кристалле и является прототипом для более современных решений.

7. Переключаемые модели вычислений

В системе на кристалле MONARCH поддерживается переключение режимов вычислений во время работы, что делает её уникальной в своём роде. При этом вычислительный FPCA-массив (англ. Field Programmable Computer Array) «системы на кристалле» MONARCH может выполнять загруженные программы в одном из трёх возможных режимов: MIMD, Stream, SIMD [5].

1. В режиме MIMD (англ. Multiple Instruction stream, Multiple Data stream) шесть 32-битных RISC-процессоров симметрично обрабатывают данные (англ. Symmetric Multiprocessing, SMP), находящиеся во встроенной DRAM-памяти, используя её как общую память. При этом обеспечивается дистанционный прямой доступ в память (англ. RDMA - Remote Direct Memory Access), подключенную к двум интерфейсам PBuf (см. на рис.1).

2. В режиме Stream 96 арифметическо-логических устройств (англ. ALU, Arithmetic Logic Unit) соединяются матрицей переключателей (англ. Crossbar) для выполнения операций умножения и сложения 32-битовых целых чисел и чисел с плавающей запятой в однопоточном или многопоточном исполнении.

3. В режиме SIMD (англ. Single Instruction, Multiple Data) семь встроенных векторных процессоров, выполняют SIMD-инструкции. При этом используется AltiVec-набор SIMD-инструкций, принадлежащий альянсу AIM (компаниям Apple Computer, IBM и Motorola), который особенно эффективен в задачах цифровой обработки сигналов, таких как цифровая фильтрация, алгоритмах быстрого преобразования Фурье, свёрточного кодирования и декодирования, обработки изображений, видеопотоков, шифрования и дешифрования и пр.

8. Интерфейсы MONARCH

Микросхема MONARCH содержит 16 портов-HSIO (англ. High-speed I/O) с двунаправленными каналами связи между узлами полиморфной вычислительной архитектуры (FPCA):

- 12 портов для внутримодульной связи,
- 4 для межмодульной связи с общей пропускной способностью 42,67 Гбайт/с.

Кроме того, для подключения к измерительно-управляющему оборудованию

бортовых систем имеются два порта интерфейса 4x Serial RapidIO с суммарной пропускной способностью 1.25 Гбайт/с [6].

9. MONARCH в бортовых измерительно-управляющих системах

На пленарном заседании 19 сентября 2006 года на конференции по высокопроизводительным встроенным вычислениям (англ. High Performance Embedded Computing, HPEC-2006) разработчиками от Университета Южной Калифорнии и компании Raython был представлен доклад «MONARCH: Полиморфный вычислительный процессор первого поколения» [6], в котором описана информационная измерительно-управляющая система, состоящая из четырёх микросхем MONARCH, которая решает задачи цифровой обработки сигналов на борту БПЛА (беспилотный летательный аппарат) типа Global Hawk.

На следующей конференции HPEC-2008 через два года инженерами компании Raytheon был представлен стендовый доклад [7] с описанием передвижной компьютерной томографической системы, построенной на восьми модулях с MONARCH, см. рис. 2.

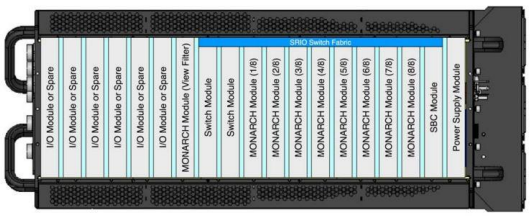

Рис. 2. Модульная система с MONARCH [7]

Система была построена в модульном стандарте VME/VPX (англ. VMEbus Switched Serial), который был создан специально для рынка авиационно-космических и оборонных систем. Система имела воздушное охлаждение, 432 МБ встроенной DRAM-памяти, 72 ГБ внешней DDR2-DRAM-памяти, обеспечивая суперкомпьютерную производительность 2.3 Тфлоп/с (англ. TFLOPS/s) при потребляемой мощности от 720 Вт до 1.26 кВт [7].

Быстрое развитие беспилотных авиационных систем (БАС) предъявляет новые требования к бортовым измерительно-управляющим комплексам. Современные БАС должны быть оснащены современными нейросетевыми функциями, которые позволяют принимать большинство решений без участия операторов, обеспечивая требуемую автономность своей миссии в воздухе.

Следует отметить, что по теме создания трансформируемых вычислительных архитектур с 2008 года в мире не было выпущено значимых публикаций, а широкие разработки, подобные проекту MONARCH велись внутри фирм. Так, например, фирма nVidia разработала своё закрытое решение NVLink [8], которое объединяет графические процессоры GPU (англ. Graphics Processing Unit) в единую вычислительную систему, которая работает в основном режиме SIMD. Переключение во время работы в другие режим, например, в MIMD, как в системе MONARCH, не предусмотрено. Для использования GPU в режиме MIMD графические процессоры подключаются по отдельным интерфейсам к центральным процессорам CPU (англ. Central Processing Unit) вычислительной системы. Для решения задач стандартного подключения графических процессоров к GPU, консорциум ведущих мировых фирм AMD, Broadcom, Cisco, Google, Hewlett Packard Enterprise, Intel и Microsoft в апреле 2025 года выпустил спецификацию открытого интерфейса UELink [9]. Разрабатываемая спецификация UELink определяет правила объединения GPU с CPU различных производителей в единую вычислительную систему для обучения больших нейронных сетей посредством оборудования Ethernet. Однако и это стандартное решение избыточно для бортовых систем, на создание которых нацелена архитектура MONARCH.

Со времени появления архитектуры MONARCH актуальность трансформируемых микропроцессорных архитектур и «систем на кристалле», позволяющих подстраиваться и решать задачи автономного управления объектами при минимальном энергопотреблении, только возрастает.

Применения нейросетей и развитие систем искусственного интеллекта приводит к необходимости структурного объединения элементов вычислительной, измерительной техники, размещённых на одном кристалле и имеющих возможность, как для трансформации под выполняемые вычислительные задачи, так и межмодульного высокоскоростного объединения. Прототипом современных бортовых многопроцессорных измерительно-управляющих систем стала архитектура MONARCH.

Overview of the Transformable Architecture of the System-On-Chip MONARCH

P. S. Ostapenkov, A. M. Chuprunov

Abstract. The paper provides an overview of the MONARCH system-on-chip architecture. The architecture was presented in 2007 by researchers and graduate students of the University of Southern California with the participation of Raython engineers under a grant provided by the US Department of Defense Advanced Research Agency. The main characteristics, architecture components, and operating modes from the polymorphic processor are described. Examples of MONARCH applications in modular multiprocessor measurement and control systems from Raython are given.

Keywords: MONARCH, system on a chip, polymorphic processor, high-performance computing, digital signal processing, energy efficiency, modularity, autonomy, on-board measurement and control systems

Литература

1. John Granacki MONARCH: Next Generation SoC (Supercomputer on a Chip) / John Granacki [Электронный ресурс] // Researchgate : [сайт]. — URL: https://www.researchgate.net/publication/235116338_MONARCH_Next_Generation_SoC_Supercomputer_on_a_Chip (дата обращения: 28.11.2025).

2. IBM introduces next-gen Asic for 90nm / [Электронный ресурс] // : [сайт]. — URL: https://www.electronicsweekly.com/news/archived/resources-archived/ibm-introduces-next-gen-asic-for-90nm-2002-06/ (дата обращения: 28.11.2025).

3. PBuf: An On-Chip Packet Transfer Engine for MONARCH // 49th IEEE International Midwest Symposium on Circuits and Systems URL: https://ieeexplore.ieee.org/document/4267408 (дата обращения: 01.12.2025).

4. На пути к созданию отечественного суперкомпьютера субэкзафлопсной производительности СБИС ЕС8430 // МСКФ-2013, 23 октября 2013 г. URL: https://www.ospcon.ru/files/media/Simonov.pdf (дата обращения: 28.11.2025).

5. Mike Vahey (Raytheon), John Granacki (USC-ISI) and others // MONARCH: A First Generation Polymorphic Computing Processor / Mike Vahey [Электронный ресурс] // HPEC-06 : [сайт]. — URL: https://archive.ll.mit.edu/HPEC/agendas/proc06/Day1/05_Vahey_Pres.pdf (дата обращения: 28.11.2025).

6. William, G. W. Reconfigurable Architecture Targets Military Sensor Systems / G. W. William. — Текст : электронный // ElectronicDesign : [сайт]. — URL: https://www.electronicdesign.com/technologies/industrial/boards/article/21766384/reconfigurable-architecture-targets-military-sensor-systems (дата обращения: 28.11.2025).

7. Using MONARCH for High Performance Processing A Case Study for CT Reconstruction. — Текст : электронный // HPEC 2008 : [сайт]. — URL: https://archive.ll.mit.edu/HPEC/agendas/proc08/Day1/10-Day1-PosterDemoA-Prager-abstract.pdf (дата обращения: 28.11.2025).

8. NVIDIA NVLink High-Speed GPU Interconnect // NVIDIA URL: https://www.nvidia.com/en-eu/products/workstations/nvlink-bridges/ (дата обращения: 28.11.2025).

9. Ultra Accelerator Link Consortium, Inc. Specification UALink_200 Rev 1.0 // UALink URL: https://ualinkconsortium.org/wp-content/uploads/2025/12/UALink200_Specification_v1.0_Evaluation_Copy-v2.pdf (дата обращения: 28.11.2025).

Аппаратные основы адаптивной безопасности: технологии изоляции и их интеграция в современные системы защиты

С. И. Земков[1], Н. А. Гревцев[2], П. А. Чибисов[3]

[1]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, zemkov@cs.niisi.ras.ru;

[2]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, ngrevcev@cs.niisi.ras.ru;

[3]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, chibisov@cs.niisi.ras.ru

Аннотация. Статья исследует аппаратные основы адаптивной безопасности — подхода к созданию систем, архитектурно устойчивых к атакам. Рассматриваются ключевые технологии изоляции (TEE, SEV, TPM) и их интеграция в адаптивные системы безопасности. Делается вывод о необходимости комплексного подхода, сочетающего аппаратные гарантии с динамическим программным анализом и оркестрацией.

Ключевые слова: адаптивная безопасность, аппаратная изоляция, доверенные среды выполнения

1. Введение

Экспоненциальный рост сложности и масштаба киберугроз заставляет постоянно пересматривать устоявшиеся подходы к защите информации. Классические методы кибербезопасности, основанные на статичных периметрах и сигнатурном анализе, оказываются недостаточно эффективными для отражения современных целевых атак. В качестве ответа сформировалась адаптивная модель безопасности, основанная на принципе «предполагай нарушение». Её суть — не только в прогнозировании и предотвращении угроз, но и в постоянной готовности к инциденту: она обеспечивает непрерывный мониторинг, анализ поведения и контекстно-зависимое автоматизированное реагирование для сдерживания атаки и минимизации ущерба даже внутри условно скомпрометированной среды.

В то же время развивается иной, архитектурно-ориентированный вектор, который ставит во главу угла не реагирование на нарушение, а его принципиальную невозможность за счёт проектирования систем, устойчивых к компрометации. Этот подход, известный как кибериммунитет, часто опирается на принципы изоляции компонентов (схожие с архитектурой MILS — Multiple Independent Levels of Security and Safety [1]), минимальных привилегий и верифицируемой безопасности. Таким образом, если адаптивная безопасность делает систему устойчивой к последствиям взлома, то кибериммунитет стремится сделать её архитектурно невзламываемой.

В основе таких систем лежат аппаратные механизмы доверия и изоляции, однако защита от киберугроз может дополняться программными средствами, реализующими принципы изоляции и минимизации доверенной вычислительной базы, например, кибериммунной операционной системой KasperskyOS и изолированными средами выполнения AWS Nitro Enclaves. В статье проводится обзор технологий Trusted Execution Environments (TEE), шифрования памяти AMD SEV и систем измерения целостности на базе TPM. Анализируется их роль в создании доверенных сред выполнения и обеспечении достоверной телеметрии, критически важной для оркестрации механизмов безопасности.

2. Основные аппаратные технологии изоляции применимые для адаптивной безопасности

Для построения систем по принципам адаптивной безопасности требуется использование аппаратных технологий изоляции — механизмов создания защищенных сред выполнения. В главе рассматриваются три ключевых подхода: TEE, архитектура Haven и шифрование памяти AMD.

2.1 Trusted Execution Environment

Trusted Execution Environment — это аппаратно изолированная зона в процессоре, обеспечивающая конфиденциальность и целостность кода и данных даже при компрометации ОС. TEE использует изоляцию

(например, Intel SGX) и криптографическую удаленную аттестацию для проверки своей корректности [2].

В работе [3] рассматривается многообразие подходов к реализации TEE в архитектуре RISC-V, от программных решений (например, Keystone), использующих стандартные механизмы защиты, до архитектур, требующих глубоких аппаратных модификаций (Sanctum, HECTOR-V). Это разнообразие объясняется стремлением к компромиссу между силой изоляции, производительностью и масштабируемостью. Легковесные проекты могут быть уязвимы к сложным микроархитектурным атакам таким как Spectre [4], тогда как более защищенные системы теряют в универсальности и производительности из-за требований к нестандартному оборудованию.

2.2 Архитектура Haven

Архитектура Haven [5] расширяет TEE, помещая в анклав SGX не только приложение, но и минимальную ОС (LibOS). Это позволяет запускать legacy-приложения без изменений, полностью изолируя их от скомпрометированной основной ОС.

Для адаптивных систем Haven демонстрирует важный принцип: создание доверенных контейнеров для сложных рабочих нагрузок. Однако, высокое потребление аппаратных ресурсов и зависимость от микроархитектуры ограничивают прямое применение этой технологии, так что она ценна только в качестве исследовательской модели.

2.3 Шифрование памяти AMD

В основе технологии AMD SME/SEV, представленной группой исследователей из AMD [6], лежит шифрование данных в оперативной памяти на уровне процессора. SEV предоставляет каждой виртуальной машине уникальный ключ шифрования, который генерируется и хранится внутри защищённого аппаратного модуля процессора и недоступен гипервизору. Данные виртуальной машины автоматически шифруются и расшифровываются контроллером памяти при обращении к ОЗУ, что обеспечивает конфиденциальность даже при компрометации гипервизора. При этом базовая версия SEV не обеспечивает защиту целостности данных, которая реализуется в расширениях SEV-ES и SEV-SNP.

В контексте адаптивной безопасности этот подход создает дополнительный барьер, повышая стоимость атак на конфиденциальность. Однако данная технология защиты пассивна и уязвима к атакам на архитектурном уровне, поэтому может быть использована как часть более широкой системы мониторинга и ответа.

3. Архитектурные подходы к адаптивной безопасности

В главе рассматриваются подходы к построению адаптивных систем безопасности на архитектурном уровне. Анализируется роль аппаратных механизмов (таких как TPM) в обеспечении доверенной телеметрии для непрерывного контроля целостности, а также обсуждаются принципиальные ограничения изолированных сред, требующих дополнительных методов поведенческого анализа для обнаружения скрытых угроз.

3.1 Адаптивные системы измерения целостности

Адаптивные системы целостности на основе TCG, предложенные Sailer et al. [7], используют аппаратный модуль TPM для создания криптографически верифицируемой цепочки измерений всех критических компонентов системы с момента ее загрузки. Это обеспечивает удаленную аттестацию и проверку соответствия эталону.

В контексте адаптивной безопасности эти непрерывные измерения становятся ключевым источником доверенной телеметрии. Система анализирует их в реальном времени, обнаруживая несанкционированные изменения и автономно запуская ответные действия, такие как изоляция узла.

3.2 Проблемы безопасности анклавов

Исследование Schwarz et al. [8] показало, что защищенные анклавы (например, Intel SGX) могут быть использованы для создания скрытного вредоносного ПО. Такое ПО становится невидимым для традиционных средств защиты, использующих анализ памяти или сигнатуры. Это доказывает, что аппаратная изоляция не самодостаточна. Для противодействия необходимо внешнее наблюдение: адаптивные системы, анализирующие аномалии в поведении (сеть, вызовы API), могут обнаружить злоупотребление доверенной средой.

4. Интеграция аппаратной изоляции в адаптивные системы безопасности

4.1 Принципы интеграции аппаратной изоляции

Анализ рассмотренных работ позволяет выделить ключевые принципы интеграции аппаратных механизмов изоляции в адаптивные

системы безопасности:

• Многоуровневая изоляция: комбинирование различных технологий (SGX, SEV, TPM) для создания защитных барьеров на разных уровнях системы.

• Динамическая аттестация: непрерывная верификация целостности изолированных компонентов.

• Адаптивное управление доступом: автоматическое изменение прав доступа на основе оценки рисков в реальном времени.

• Оркестрация безопасности: координация работы различных механизмов защиты через единую систему управления.

4.2 Развитие адаптивной безопасности как кибериммунитета

Кибериммунитет — это архитектурная парадигма, проектирующая системы, изначально устойчивые к атакам. Признавая невозможность абсолютной защиты, подход фокусирует усилия на критических компонентах, компрометация которых недопустима. Это становится возможным благодаря использованию встроенных механизмов строгого разделения и контролируемого взаимодействия всех подсистем.

Искусственные иммунные системы (AIS) применяют биологический принцип распознавания «свой/чужой» (self/nonself) для обнаружения аномалий в IT-средах. Как отмечено в обзоре [9], архитектура таких систем основывается на кодировании данных (антигенов/антител), алгоритмах генерации детекторов (например, негативной селекции) и эволюционном механизме для адаптации к новым угрозам. Последующие исследования [10] развивают эту тему, предлагая конкретные архитектурные решения для IDS, вдохновленные иммунной системой, включая использование не только теории «свой/чужой», но и теории опасности (Danger Theory).

В работе [11] показано, как принципы кибериммунитета реализуются в микросервисных архитектурах. Их естественная изоляция компонентов сама по себе увеличивает безопасность ситемы, однако для повышения устойчивости к киберугрозам необходимы централизованные защищенные механизмы. Такие механизмы должны контролировать все взаимодействия между сервисами, предотвращая распространение атаки при компрометации одного из них.

5. Заключение

Проведенный анализ позволяет сделать вывод о том, что эффективная адаптивная безопасность не может быть достигнута исключительно за счет программных решений и надстроек над существующей инфраструктурой. Современный уровень угроз требует пересмотра маршрута построения вычислительных систем. Ключевым направлением становится проектирование безопасности на аппаратном уровне, начиная с этапа разработки микросхем и процессоров.

Необходимо закладывать в архитектуру аппаратных компонентов специализированные механизмы изоляции, доверия и измерения целостности, такие как защищенные среды выполнения, криптографическое шифрование памяти на уровне процессора и аппаратные модули доверия. Внедрение этих механизмов позволяет создать доверенную и верифицируемую платформу, которая будет способна предотвращать проникновение и противостоять угрозам в уже скомпрометированной системе. Такой подход позволяет гарантировать доверенность критически важных компонентов адаптивной безопасности — агентов мониторинга, систем анализа и механизмов автоматизированного реагирования. Это требует тесного сотрудничества между разработчиками микропроцессорной техники, производителями аппаратных платформ и архитекторами систем программной безопасности.

Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0003.

Hardware foundations of adaptive security: isolation technologies and their integration into modern protection systems

S. I. Zemkov, N. A. Grevtsev, P. A. Chibisov

Abstract. The article explores the hardware foundations of adaptive security—an approach to creating systems that are architecturally resilient to attacks. It examines key isolation technologies (TEE, SEV, TPM) and their integration into adaptive security systems. The conclusion emphasizes the necessity of a comprehensive approach that combines hardware guarantees with dynamic software analysis and orchestration.

Keywords: adaptive security, hardware isolation, trusted execution environments

Литература

1. Alves-Foss J. et al. The MILS architecture for high-assurance embedded systems // International journal of embedded systems. – 2006. – Vol. 2. – No. 3-4. – P. 239-247.
2. Hoekstra M., Lal R., Pappachan P., Phegade V., Del Cuvillo J. Using innovative instructions to create trustworthy software solutions // Proc. of the Network and Distributed System Security Symposium (NDSS 2013). 2013.
3. Boubakri, M.; Zouari, B. A Survey of RISC-V Secure Enclaves and Trusted Execution Environments. Electronics 2025, 14, 4171.
4. P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg,M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, “Spectre attacks: Exploiting speculative execution,” Communications of the ACM, vol. 63, pp. 93–101, 2020.
5. Baumann A., Peinado M., Hunt G. Shielding applications from an untrusted cloud with Haven // ACM Transactions on Computer Systems (TOCS). 2015. Vol. 33. No. 3. Pp. 1–26.
6. Kaplan D., Powell J., Woller T. AMD Memory Encryption. White paper, 2016.
7. Sailer R., Zhang X., Jaeger T., Van Doorn L. Design and implementation of a TCG-based integrity measurement architecture // Proc. of the 13th USENIX Security Symposium. 2004. Pp. 223–238.
8. Schwarz M., Li S., Weiser S., Gruss D. Practical enclave malware with Intel SGx. arXiv preprint arXiv:2002.05649, 2020.
9. Smith J., Johnson M. A survey of artificial immune system based intrusion detection // Journal of Network and Computer Applications. 2015. Vol. 52. Pp. 1–20.
10. Wilson P., Taylor S. Artificial immune systems in local and network cybersecurity: An overview of intrusion detection strategies // Proc. of the International Conference on Cybersecurity. 2021. Pp. 45–62.
11. Соболев С. П. Кибериммунный подход к разработке. Иллюстрация применения на базе микросервисной архитектуры // Вестник СПбГУ. Серия 10. Прикладная математика. Информатика. Процессы управления. 2024. № 1.

Подходы к переводу и компиляции в многоязыковой системе

В. А. Ковыршина[1], А. Г. Леонов[2], М. В. Райко[3]

[1] НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, potchtovy_jashik@mail.ru

[2] НИЦ «Курчатовский институт» - НИИСИ, МПГУ, МГУ, Москва, Россия, dr.l@vip.niisi.ru

[3] НИЦ «Курчатовский институт» - НИИСИ, МПГУ, Москва, Россия, rayko@niisi.ru

Аннотация. Цифровая трансформация образования актуализирует задачу снижения порога входа в программирование для самой юной аудитории. В качестве решения предлагается использование блочных сред, таких как «ПиктоМир-К», которые позволяют концентрироваться на алгоритмической составляющей, минуя сложности профессиональных инструментов. В статье детально описывается ядро многоязыковой учебной среды программирования, построенное на разделении универсального синтаксического дерева SyntaxTree), хранящего семантику программы, и визуального дерева (VisualTree), ответственного за отрисовку представлений программы на различных языках программирования. Такой подход реализует функции многоязыковости, позволяя мгновенно переключать представление кода между различными синтаксисами (КуМир, Python, C++). Кроме того, синтаксическое дерево используется для компиляции программы в набор инструкций для виртуальной стековой машины. Показано, что предложенная архитектура является гибкой и расширяемой, открывая возможности для поддержки новых языков и трансляции в различные исполняемые форматы.

Ключевые слова: компиляторы, стековая машина, синтаксическое дерево

1. Введение

Цифровая трансформация общества обуславливает растущий интерес к преподаванию азов программирования детям самых ранних возрастов. Ключевой задачей становится снижение порога входа в деятельности, которые предлагаются детям для ознакомления с основными понятиями программирования. Среди таких деятельностей наиболее важной и трудной является самостоятельное составление программ в какой-то среде программирования. Традиционные среды, как правило, требуют от учащихся непривычных или логически непростых действий: клавиатурного ввода текста; чтения и ввода служебных слов в латинской транскрипции и, что труднее всего, поиска и исправления синтаксических ошибок в программе. Требуются специальные подходы, для того, чтобы не погасить интерес учащихся из-за необходимости выполнять подобные действия.

Одним из успешно работающих подходов служат блочные среды программирования, применяемые во вводных курсах программирования. В таких средах используется традиционное текстовое изображение кода программы, однако ввод кода проводится не клавиатурно, а пиктограммно, путем манипуляций блоками, представляющими синтаксически правильные конструкции используемого языка программирования.

Примером отечественной блочной среды программирования является среда ПиктоМир-К [1], которая предоставляет учащемуся предоставляет ограниченный, но педагогически достаточный набор школьного алгоритмического языка. Главное преимущество среды ПиктоМир-К — обеспечение корректности синтаксиса на всех этапах составления программы, что делает ненужными этапы лексического и синтаксического анализа.

Традиционный подход, при котором начинающие сразу работают со сложными учебными или даже промышленными языками программирования (школьный алгоритмический язык в среде КуМир, C++, Python), вынуждает новичка при решении учебных задач одновременно преодолевать и трудности придумывания алгоритма и трудности освоения интерфейса среды разработки и синтаксиса языка программирования. Инструменты блочного программирования, будучи специализированным педагогическим решением, демонстрируют большую эффективность на начальном этапе, уменьшая затраты на освоение интерфейса среды программирования и синтаксиса используемого языка. Это позволяет учащемуся

концентрироваться на алгоритмических аспектах решаемых учебных задач.

Особенности блочной организации процесса составления программы позволяют разработчикам среды программирования ПиктоМир-К избежать использования тяжеловесных библиотек для компиляции и выполнения программ и создать собственные упрощенные алгоритмы компиляции и интерпретации кода программы, ориентированные на упрощенный синтаксис и использование всего лишь двух базовых типов данных.

2. Описание языка ЦОС ПиктоМир-К

Программирование в ПиктоМир-К начинается с визуального конструирования: ученик собирает код из пиктограмм, которые система автоматически преобразует в текст, объединенный в логические блоки. В качестве основы используется подмножество школьного алгоритмического языка, реализованного в отечественной учебной среде программирования КуМир. Функционал выбранного подмножества языка сознательно ограничен. Для лучшего представления структуры кода задействованы синтаксические диаграммы Вирта — метод визуализации, разработанный для языка Паскаль.

Центральная концепция — "исполнитель". Доступно 9 различных исполнителей (Вертун, Черепаха, Робот и др.), каждый со своим уникальным набором команд, представленных в виде пиктограмм. Все действия происходят в виртуальной "обстановке", где робот, соответствующий выбранному исполнителю, выполняет составленную учащимся программу.

Программа структурно состоит из директивы подключения исполнителя и основного алгоритма без имени. Для создания функций (без параметров или с параметрами) по умолчанию предусмотрены пять стандартных имен (первые буквы алфавита). Отдельные команды исполнителей (например, у Черепахи) также используют параметры. Таким образом, ученик, манипулируя командами, может управлять роботом, рисуя и перемещая его по полю.

Основу языка Пикто-К составляют следующие элементы:

Управляющие конструкции: несколько видов циклов и условные операторы.

Работа с данными: объявление переменных, присваивание значений и математические операции.

Система типов: язык строго типизирован и включает только два типа — целый и логический.

Условные операторы представлены формами «если-то» и «если-то-иначе». Условия в них задаются с помощью логических выражений, переменных или констант «да» и «нет».

Грамматика алгоритмического языка Пикто-К подробно изложена в статье «Пиктограммный язык программирования «Пикто» авторов Бесшапошников Н.О., Леонов А.Г. [2]

3. Многоязыковость и архитектура визуализации

Представление программы в системе ПиктоМир-К основано на параллельном существовании двух синхронизированных структур данных:

Универсальное синтаксическое дерево (*SyntaxTree*): содержит чистую семантику программы, независимую от способа её отображения.

Универсальное визуальное дерево (*VisualTree*): отвечает за отрисовку программы, используя модули визуального представления (рендереры).

Синтаксическое дерево *SyntaxTree* состоит из узлов класса *Node*, а визуальное дерево *VisualTree* состоит из узлов класса *NodeLayer* или дочерних от него классов. Узлы этих двух деревьев находятся во взаимно однозначном соответствии: каждому узлу *Node* в *SyntaxTree* соответствует узел *NodeLayer* в *VisualTree* и наоборот.

По завершении сеанса работы пользователя с какой-либо учебной программой, в архиве пользователя сохраняется только *SyntaxTree*, и по нему при новом открытии данной учебной программы в среде ПиктоМир-К заново строится *VisualTree*. После этого пара деревьев (*SyntaxTree*, *VisualTree*), соответствующих данной программе, в ходе сеанса редактирования учебной программы пользователем изменяются одновременно и взаимосогласованно.

Пользователь работает с программой посредством взаимодействия с *VisualTree*. Визуальное дерево, будучи *UI*-объектом отлавливает и обрабатывает действия пользователя (перетаскивания, клики), в частности, добавление/удаление/изменение узлов, а также смену формы графического представления программы. Создание, изменение или удаление визуального (*NodeLayer*) и синтаксического (*Node*) узлов происходит одновременно и согласованно, за счёт этого и обеспечивается полное соответствие синтаксического и визуального деревьев.

В то время как класс *Node* хранит всю

логическую и смысловую информацию (тип узла *type*, значение *value*, идентификатор *id* и связи с родителем *parent* и дочерними узлами *children*[]), он не содержит никакой информации о том, как соответствующий узел будет изображен графически. Его графическая реализация *NodeLayer* содержит шаблон для визуального представления *Node* и знает, как «перевести» этот шаблон на конкретный язык программирования. Данный шаблон состоит из компонент-рендереров, которые неизменны и не зависят от синтаксиса, они перерисовываются при его смене. То, в каком синтаксисе мы работаем, учитывается непосредственно при рендеринге *NodeLayer*.

Такой подход позволяет реализовать в системе ПиктоМир-К возможность программирования на разных языках без перестройки самой программы. Для одной и той же пары (*SyntaxTree*, *VisualTree*) достаточно выбрать модуль визуального представления, и все узлы *NodeLayer* будут автоматически нарисованы с новыми терминалами. При переключении языка представления программы каждая вершина *VisualTree* перерисовывается независимо и автоматически. Опишем подробнее, как это происходит.

3.1. Принцип работы механизма визуализации

Класс *NodeLayer* (и, соответственно, всякий дочерний от него класс) хранит следующую информацию:

parent (**NodeLayer*) – родительский узел в *VisualTree*,

node (**Node*) – соответствующий узел в *SyntaxTree,*

syntax (*String*) – текущий модуль визуального представления (например, "kumir", "python", "cpp"), то есть синтаксис, в котором представлена программа.

Рассмотрим, по какому принципу строится и изменяется *NodeLayer*, на примере языков Python и Кумир, реализованных в ПиктоМир-К. *NodeLayer*'ы всех типов имеют универсальные конструкции, «перевод» на конкретный язык программирования происходит посредством замены терминалов его конструкции. Ниже приведена таблица 1, описывающая эти замены.

Таблица 1. Описание замены терминалов

Тип Node	**Конструкция NodeLayer**	**Python**		**КуМир**	
		компонента	**терминал**	**компонента**	**терминал**
Prog	использовать <robot name> eol	использовать eol	*from* *import **	использовать eol	*использовать* -
MainAlg	алг нач <body> кон	алг нач кон	- - -	алг нач кон	*алг* *нц* *кц*
Func	алг <identifier> [<args>] eol нач <body> кон	алг eol нач кон	*def* *:* - -	алг eol нач кон	*алг* - *нц* *кц*
Body	<Empty>	<tab> <Empty>		■ <Empty>	
Statement	<identifier> := <expression> ;	:= ;	= -	:= ;	:= ;

IfStatement	если <condition> eol то <body> [ElseBlock] все	если eol то все	*if* : - -	если eol то все	*если* - *то* *всё*
ElseBlock	[иначе eol <body>]	иначе eol	*else* :	иначе eol	*иначе* -
Loop (nTimes)	нц <expression> раз <body> кц	нц раз кц	for _ in range( ) : -	нц раз кц	*нц* *раз* *кц*
Loop (for)	нц для <identifier> “от” <expression> до <expression> eol <body> кц	нц для от до eol кц	for in range(,) : -	нц для от до eol кц	*нц для* *от* *до* *eol* *кц*
Loop (while)	нц пока <condition> eol <body> кц	нц пока eol кц	*while* : -	нц пока eol кц	*нц пока* - *кц*
Type	тип	тип	-	тип	цел | лог
Identifier	“a1, a2, …”, “л1, л2, ..»	“a1, a2, …”, “л1, л2, ..»		“a1, a2, …”, “л1, л2, ..»	
Number	0 - 999	0 - 999		0 - 999	
Bool	value	value	True | False	value	да | нет
Expression	“add” | “sub” | “mult” | “divide”	+ | - | * | /		+ | - | * | /	
LogicExpression	“llr”, “rll”, “equal”, ”not-equal”, ‘llr-equal’, ‘rll-equal’	< | > | == | != | <= | >=		< | > | == | != | <= | >=	
Alloc	тип <identifier>	тип	-	тип	цел | лог
Empty	-	-		-	

Условные обозначения в таблице 1:

- `eol` — конец строки (учитывается в форматировании)
- `<tab>` — табуляция
- [ ] — необязательные элементы
- <Empty> – не отображаемая

графически компонента, позволяющая вставлять в указанное место новые узлы.

Аналогичным образом происходит перевод языка Кумир с русского на английский: терминалы на русском заменяются в соответствии с таблицей терминалами на английском.

3.2. Преимущества архитектуры с двумя параллельно поддерживаемыми согласованными деревьями

Данный механизм имеет четыре ключевых преимущества:

Универсальность синтаксического дерева: SyntaxTree служит единым источником истины для компилятора, отладчика и визуализатора, обеспечивая согласованность всех компонентов системы.

Удобство технической реализации всей системы: для компиляции программы достаточно SyntaxTree, а для визуализации – VisualTree. Разделение этих двух задач делает процессы компиляции и визуализации независимыми, что ускоряет работу системы, также упрощает архитектуру узлов Node и NodeLayer.

Модульность визуализации: Система визуализации построена как набор независимых модулей-рендереров. Добавление поддержки нового языка программирования (например, C, JavaScript) сводится к созданию нового модуля, который "знает" соответствующие терминалы для универсальных шаблонов NodeLayer.

Производительность: При смене языка представления не требуется перестраивать структуру деревьев — достаточно обновить терминалы в существующих узлах NodeLayer, что обеспечивает мгновенное переключение между языками.

Такой подход не только реализует многоязыковость, но и создает фундамент для будущих расширений, таких как поддержка новых парадигм программирования или специализированных предметно-ориентированных языков (DSL).

В таблице 2 приведен пример того, что должны были бы начать делать разработчики среды ПиктоМир-К, если бы потребовалось добавить представление кода программы в синтаксисе языка C++:

Таблица 2. Перевод для добавления языка C++

Тип Node	**Конструкция NodeLayer**	**C++**	
		компонента	**терминал**
Prog	использовать <robot name> eol	использовать eol	#include “ >.h”

MainAlg	алг
	нач
	<body>
	кон
		алг
		нач
		кон
			void main()
			{
			}

Func	алг <identifier> [<args>] eol
	нач
	<body>
	кон
		алг
		eol
		нач
		кон
			void
			-
			{
			}

Body	<Empty>	<tab> <Empty>	
Statement	<identifier> := <expression> ;	:= ;	= ;

IfStatement	если <condition> eol
	то
	<body>
	[ElseBlock]
	все
		если
		eol
		то
		все
			if (
			)
			{
			}

ElseBlock (частный случай Body)	[иначе eol <body>]	иначе eol	} *else* {

Loop (nTimes)	нц <expression> раз
	<body>
	кц
		нц
		раз
		кц
			for(j = 0; j <=
			; j ++) {
			}

Loop (for)	нц для <identifier> “от” <expression>
	до <expression> eol
	<body>
	кц
		нц для
		от
		до
		eol
		кц
			for(
			=
			; <identifier> <=
			; <identifier> ++) {
			}

Loop (while)	нц пока <condition> eol
	<body>
	кц
		нц пока
		eol
		кц
			while (
			){
			}

Type	тип	тип	int | bool
Identifier	“a1, a2, …”, “л1, л2, ..»	“a1, a2, …”, “л1, л2, ..»	
Number	0 - 999	0 - 999	
Bool	value	value	true | false
Expression	“add” | “sub” | “mult” | “divide”	+ | - | * | /	
Condition	Команды вопросы конкретного робота	Команда вопрос	
LogicExpression	“llr”, “rll”, “equal”, ”not-equal”, ‘llr-equal’, ‘rll-equal’	< | > | == | != | <= | >=	
Action	Методы конкретного робота или название вызывамой функции	Метод или функция	
Alloc	тип <identifier> eol	тип eol	int | bool ;

Empty	-	-

Как видно, нам бы потребовалось добавить компоненту `eol` в конец AllocNode (NodeLayer для Node типа Alloc), однако для этого достаточно было бы доопределить замены добавленной компоненты `eol` на пустой терминал - в Python и в Кумире. Таким образом, функционал ПиктоМир-К может быть легко адаптирован в случае добавления нового языка, даже если синтаксические конструкции каких-то его компонент не укладываются в имеющийся шаблон.

3.3. Компиляция программы

На этапе компиляции осуществляется обход синтаксического дерева и генерации соответствующих инструкций. В результате обхода дерева появляется набор инструкций для виртуальной стековой машины. Эта виртуальная машина выполняет определенные действия в зависимости от типа инструкции и ее данных. Для выполнения любой программы на языке Пикто-К. требуется всего лишь 14 инструкций, подробно описанных в таблице 3.

Таблица 3. Описание инструкций для выполнения программы на языке Пикто-К

№	Тип	Описание	Аргументы
0	Execute	Выполняет метод робота либо подпрограмму. При выполнении подпрограммы с вершины стека достаются параметры, в стек заносится текущая позиция в списке инструкций, а в регистр вызовов функций заносится текущая позиция в стеке для определения области локальных данных этого метода.	isNative– является ли вызовом подпрограммы methodID – уникальный идентификатор метода робота paramCount– количество параметров, которые нужно взять из регистра.
1	Check Condition	Выполняет проверку условия для робота и кладет результат на вершину стека.	robot – ссылка на исполнителя condition – уникальный идентификатор условия для проверки роботом
2	Start loop	Является индикатором начала цикла, задает начальное значение итератору цикла. Достает с вершины стека текущее значение итератора и конечное, если текущее значение> конечного, то переходит на конец блока цикла. В противном случае кладет на вершину стека начальное и конечное значение итератора.	blockEndLabel – ссылка на конец блока цикла repType – тип цикла
3	End loop	Достает с вершины стека текущее и конечное значения итератора. Если итератор <= конечному значению, то выполняет переход в начало цикла, увеличивает итератор на единицу и кладет в стек текущее значение итератора и конечное.	jumpLabel – ссылка для перехода для выполнение следующей итерации цикла robot – ссылка на исполнителя
4	Return	Выполняет выход из подпрограммы, если есть выходные параметры, то достает их из вершины стека и кладет в регистр параметров, далее получает из стека вызовов функций позицию, начиная с которой в нем расположены локальные данные этой функции. Удаляет все локальные данные и последнее значение из регистра вызовов. Если регистр вызовов пуст – программа	paramQuantity – количество параметров, которые нужно взять из регистра

		завершилась.	
5	Jump_if	Переход по ссылке, если условие на вершине стека верно.	jumpLabel – ссылка для перехода
6	Jump_nif	Переход по ссылке, если условие на вершине стека неверно.	jumpLabel – ссылка для перехода
7	Jump	Безусловный переход по ссылке.	jumpLabel – ссылка для перехода
8	Alloc	Выделяет место для переменной в памяти, если вызов осуществляется вне функции или кладет переменную в стек, если в регистре вызовов функции уже есть вызов.	varType – тип переменной varName – имя переменной value – значение переменной
9	Fetch	Кладет на вершину стека значение переменной, ищет значение этой переменной в стеке в области видимости этой функции.	varName – имя переменной
10	Push	Кладет в стек значение	value - значение
11	Store	Сохраняет значение с вершины стека или из регистра параметров в переменной.	varName – имя переменной fromReg – взять значение из регистра параметров или из стека
12	Pop	Удалить последнее значение из стека	-
13	Calc	Выполняет операцию над последними двумя значениями с вершины стека.	operation - операция

Таким образом, программа, описываемая синтаксическим деревом, переводится в набор ассемблер-подобных инструкций для виртуальной исполняющей машины. Виртуальная исполняющая машина состоит из стека, регистров и памяти. Стек хранит локальные данные функций и операций. Регистр вызовов функций сохраняет в себе позицию в стеке, начиная с которой расположены локальные данные вызванной функции. Регистр параметров сохраняет входные и выходные параметры функций. Память – это ассоциативный массив глобальных переменных в смысле работы [3].

4. Универсальность компилятора и целевые платформы

Универсальное синтаксическое дерево служит не только для мультиязыковой визуализации, но и как идеальное промежуточное представление (Intermediate Representation, IR) для компиляции. Набор инструкций для стековой машины, генерируемый из дерева, является лишь одной из возможных целевых платформ.

Алгоритмы обхода дерева и генерации кода являются универсальным. Это открывает возможности для трансляции программ ПиктоМир-К в другие формы исполнения:

Трансляция в «нативный код»: при наличии соответствующего бэкенда, обходчик синтаксического дерева может генерировать не инструкции для виртуальной машины, а, например, исходный код на C или JavaScript, который затем можно выполнить с максимальной производительностью.

Исполнение на микроконтроллерах: Тот же самое дерево можно использовать для генерации машинного кода или упрощенного байт-кода для образовательных робототехнических платформ (например, на базе Arduino). Это позволит программам, написанным детьми в ПиктоМире, управлять реальными физическими устройствами.

Интеграция с другими образовательными средами: Синтаксическое дерево можно сериализовать в универсальный формат (например, JSON или XML) и импортировать в другие системы, что делает ПиктоМир-К открытой и интегрируемой платформой.

5. Заключение

Описанный метод компиляции и выполнения программы реализован на языке JavaScript [4] в обучающей среде программирования ПиктоМир-К. Данная система позволяет выполнять программы, созданные в среде ПиктоМир-К для целей обучения азам алгоритмики.

Предложенная архитектура с универсальным синтаксическим деревом в качестве ядра и независимыми модулями визуализации и компиляции демонстрирует высокую степень гибкости и расширяемости. Она не только решает задачу обучения основам алгоритмизации, но и создает прочный фундамент для будущего развития системы. За счет описанных принципов обеспечивается:

Низкий порог для добавления поддержки новых языков программирования.

Легкость модификации существующих языковых представлений.

Потенциал для трансляции в различные исполняемые форматы и платформы.

Это делает ПиктоМир-К не просто еще одной средой блочного программирования, а универсальным инструментом для построения моста между визуальным, блочным и текстовым программированием.

Работа выполнена в рамках темы государственного задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0001, этап 2025 года (1023032100070-3-1.2.1).

Approaches to Translation and Compilation in a Multilingual System

V. A. Kovyrshina, A. G. Leonov., M. V. Rayko

Abstract. Digital transformation in education brings to the forefront the task of lowering the entry barrier into programming for the youngest audience. As a solution, the use of block-based environments, such as "PiktoMir-K," is proposed. These environments allow users to focus on the algorithmic component, bypassing the complexities of professional tools. The article details the system's core, which is built on the separation of a universal SyntaxTree, storing the program's semantics, and a VisualTree, responsible for its rendering. This approach implements a multilingualism function, allowing for instant switching of the code representation between different syntaxes (KuMir, Python, C++). Furthermore, the syntax tree is used to compile the program into a set of instructions for a virtual stack machine. It is shown that the proposed architecture is flexible and extensible, opening up possibilities for supporting new languages and translation into various executable formats.

Keywords: compilers, stack machine, syntax tree

Литература

1. Стартовая страница проекта «ПиктоМир - К» на сайте НИЦ «Курчатовский институт» - НИИСИ . URL: https://www.niisi.ru/piktomir/ (дата обращения 01.10.2025)
2. Бесшапошников Н.О., Леонов А.Г. Пиктограммный язык программирования «Пикто» // Вестник кибернетики. 2017. № 4 (28). С. 173–180
3. Райко М.В. Построение компилятора-интерпретатора для гибридной текстово-пиктограммной цифровой образовательной среды ПиктоМир-К / М.В. Райко, Д.Б. Аглямутдинова, А.Г. Леонов // Труды НИИСИ РАН. — 2020. — Т. 10, № 5-6. — С. 148–160
4. ISO/IEC 22275:2018. Information technology — Programming languages, their environments, and system software interfaces — ECMAScript Specification Suite.

Опыт организации курсов алгоритмики для дошкольников и младшеклассников с помощью отечественной предметно-цифровой образовательной среды «ПиктоМир»

А. И. Аханкина[1], А. Г. Кушниренко[2], А. Г. Леонов[3], М. В. Райко[4], У. М. Солопова[5]

[1] НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, ahankina@niisi.msk.ru

[2] НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, agk_@mail.ru

[3] НИЦ «Курчатовский институт» - НИИСИ, МПГУ, МГУ, Москва, Россия, dr.l@vip.niisi.ru

[4] НИЦ «Курчатовский институт» - НИИСИ, МПГУ, Москва, Россия, rayko@niisi.ru

[5] НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, solopova@niisi.ru

Аннотация. В статье представлен многолетний опыт внедрения и развития предметно-цифровой среды ПитоМир в системе дошкольного и начального школьного образования РФ. Особое внимание уделено методическим аспектам обучения основам алгоритмики и программирования, которое основано на опыте педагогов-практиков. Рассмотрены два варианта годового курса алгоритмики для дошкольников возраста 5-7 лет, и продолжающий их курс алгоритмики для младшеклассников. Свободно распространяемое программное и программно-методическое обеспечение курсов размещено и доступно для скачивания на сайте НИЦ «Курчатовский институт»-НИИСИ. В статье подробно обсуждаются структурные компоненты и методология годового курса «Алгоритмика 1-36», рассчитанного на 36 получасовых занятий и направленного на практическое освоение детьми основных понятий программирования и развитие алгоритмического мышления в предметно-цифровой среде «ПиктоМир».

Ключевые слова: алгоритмическое мышление, программирование, дошкольники, предметно-цифровая образовательная среда, ПиктоМир, пиктограммы, алгоритмика, бестекстовая среда программирования

1. Введение

В 2009 году сотрудники отдела учебной информатики Научного Центра НИИСИ РАН в инициативном порядке начали работу над бестекстовой цифровой образовательной средой «ПиктоМир». По инициативе академика В.Б. Бетелина в 2010 года работа получила высокий статус и стала выполняться в рамках госзадания по заказу Российской Академии Наук. Среда «ПиктоМир» создавалась с целью предоставления дошкольникам и младшеклассникам инструментов для систематического и непрерывного изучения основ программирования, формирования алгоритмического мышления и развития интереса к информационным технологиям [1].

В современном мире стремительной и глобальной цифровизации инновационный, свободно распространяемый отечественный программный инструмент «ПиктоМир» позволяет детям дошкольного и младшего школьного возраста освоить, в игровой форме, основные принципы построения программ управления как экранными виртуальными роботами, так и реальными роботами игрушками. В результате, доступный детям программный инструмент, сочетающий в себе игровую и образовательную составляющую, способствует всестороннему развитию детей, формируя, в частности, высокий уровень алгоритмического мышления - необходимого навыка для решения сложных, как повседневных, так и аналитических задач, а также для самостоятельного структурирования повседневных действий.

Одним из главных преимуществ среды «ПиктоМир» стала возможность обучения детей без использования синтаксически сложного текстового языка программирования. Вместо этого дети начинают погружение в алгоритмику, используя пиктокарточки и пиктокубики с изображенными на них пиктограммами команд, для описания маршрута движения реального

материального Робота, по игровому полю, составленному из ковриков с цифрами. Робот перемещается по игровому полю, выполняя команды по алгоритму, составленному ребенком [2]. «ПиктоМир» позволяет воспитателю организовать не только индивидуальную работу одного ребенка, но и командную, кооперативную работу детей, что развивает социально–коммуникативные способности детей в соответствии с ФГОС дошкольного образования. Ознакомившись с базовыми принципами алгоритмики путем проведения действий в материальном мире, дети переходят в цифровую среду и начинают использовать пиктограммный язык, основанный на графических, пиктограммных, представлений команд вместо представлений текстовых. Такая простота, доступность и геймификация образовательного процесса позволяет начинать ознакомление детей с программированием в возрасте, когда они ещё не умеют читать и писать. Многолетняя практика работы с тысячами детей дошкольного возраста показала, что даже в раннем возрасте дети способны на практике освоить основные понятия структурного программирования.

Очевидно, что раннее обучение детей азам программирования с каждым годом становится все более актуальной задачей, необходимость решения которой вызвана национальной стратегией развития в России информационного общества. Буквально с самого рождения дети сегодня погружены в информационное цифровое пространство и раннее освоение цифровой грамотности из особого конкурентного преимущества, что мы наблюдали еще лет 10 назад, становится просто базовым навыком для успешной адаптации в современном обществе. Благодаря доступности, привлекательности и элементам геймификации, а также соответствию всем нормативно-правовым правилам, «ПиктоМир» нашел широкое применение в системе дополнительного образования России и активно используется в дошкольных учреждениях.

2. Первое массовое внедрение предметно-цифровой среды «ПиктоМир» в практику дошкольных учреждений

Начиная с 2014 года НИИСИ РАН и департамент образования г. Сургута начали разработку и частичное внедрение дополнительной образовательной программы для подготовительных групп г. Сургута. В ходе развития и совершенствования образовательной программы, уже в 2018-2019 годах началось массовое внедрение годовой программы курса «Алгоритмика для дошколят» во все подготовительные группы всех детских садов города Сургута: данная программа «пропускает через себя» более 6000 тысяч детей в год.

В основе курса лежит бестекстовая методика, в которой дети дошкольного возраста работают с пиктограммами команд, составляют алгоритмы, выполняемые разными типами виртуальных роботов, составляют подпрограммы с однобуквенными именами, алгоритмы с повторителями и используют базовые управляющие конструкции [3].

На начальном этапе, важной составляющей программы является допланшетный период обучения вне цифровой среды, в ходе которого дети действую в реальном мире: составляют игровые поля для роботов из сочленяемых пластиковых ковриков, учатся с помощью звукового пульта управлять движением робота-игрушки по игровым полям, учатся составлять из материальных объектов — карточек и кубиков с пиктограммами –программы перемещения робота-игрушки по игровым полям.

Внедренный в Сургуте годовой курс «Алгоритмика для дошколят» был рассчитан на тридцать пять занятий: тридцать основных, которые подробно описаны авторами курса и предоставляются педагогам в методическом пособии, а также пять дополнительных/резервных занятий, которые планируются и составляются самим педагогом.

Тридцать основных занятий, представленных в курсе, можно поделить на пять этапов [4]:

Первый этап предусматривает 5 занятий, посвященных общей теме «Алгоритмы управления роботами. Основные понятия программирования». Дети начинают развивать представления о базовых понятиях программирования и работают с реальным Роботом «Ползуном», составляют простые линейные программы;

Второй этап, состоящий из пяти занятий, посвящён теме «Линейные алгоритмы управления роботами». На этом этапе дети начинают осваивать работу на планшетах в цифровой среде программирования «ПиктоМир». Дети знакомятся с несколькими экранными виртуальными роботами (Вертун, Двигун, Тягун, Ползун), с их системой команд, разбираются со сходствами и различиями наборов команд этих роботов. Помимо этого, дети начинают осваивать кооперативное программирование и получают навыки работы в команде.

Третий этап сфокусирован на теме

«Алгоритмы управления роботами. Повторители» и направлен на освоение детьми алгоритмов с числовыми повторителями и накоплению практического опыта составления и отладки алгоритмов с числовыми повторителями на планшетах.

Четвертый этап предусматривает 10 занятий и посвящен теме «Алгоритмы управления Роботами. Повторители и подпрограммы». Данный этап объединяет работу с подпрограммами и повторителями, в программах появляется блочная структура, вложенность, алгоритмы становятся сложнее.

Пятый, завершающий, этап знакомит с командами-вопросами роботов (команды с обратной связью) и организуемыми с их помощью ветвлениями и циклами с неизвестным числом повторений, и счетными операциями. На этом этапе дети осваивают конструкции «если» и «цикл пока», а также работают с исполнителем «Волшебный Кувшин», который используется в программах в качестве «счётчика». Этот пятый, завершающий этап закладывает основу для развития навыков программирования в начальной школе.

В рамках реализованной массово в г. Сургуте годовой программы курса «Алгоритмика для дошколят», в каждой группе, в течение учебного года, два раза, осенью и весной проводился мониторинг результатов учебного процесса. Этот мониторинг неизменно показывал, что программа достаточно успешна. Например, осенью 2019/2020 учебного года мониторинг учебного процесса в одном из муниципальных детских садов города Сургута показал, что успешно справились с предложенными заданиями 100% из 50 учащихся, 8 из которых были детьми с ОВЗ [5]. Этот опыт говорит о том, что в типовом российском дошкольном образовательном учреждении стандартного типа, все дошкольники без каких-либо сложностей способны осваивать азы программирования в игровой форме.

3. Опыт интеграции методики педагогами

Опыт внедрения программы «Алгоритмика для дошколят» в Сургуте, вскрыл несколько вопросов, подлежащих разрешению. В ходе внедрения среди родителей дошкольников несколько раз проводился опрос: «Нужно ли обучать детей программированию в дошкольном возрасте?» «Так ли необходимо использовать гаджеты в образовательном процессе детского сада?». По результатам опроса выяснилось, что бóльшая часть родителей (63%) считали, что рано внедрять гаджеты в дошкольную образовательную деятельность. А 55% опрошенных родителей считали, что рано обучать дошкольников программированию [6].

Под впечатлением результатов этих опросов, в 2019 году группа инициативных педагогов из детских садов «Чудо-Град» и «Семицветик» (СП ГБОУ СОШ «ОЦ «Южный город»», посёлок Придорожный, Волжский район, Самарская область) совместно с сотрудниками Отдела учебной информатики ФГУ ФНЦ НИИСИ РАН поставили перед собой задачу — опровергнуть стереотип о вреде использования гаджетов в дошкольном возрасте. Была сделан попытка на практике показать, как можно эффективно обучать дошкольников основам программирования с помощью цифровой образовательной среды «ПиктоМир». Результатом работы экспертной группы стало развертывание в РАН сетевой инновационной площадки «ПиктоМир» и апробация двух образовательных программы на базе обобщения опыта годовой программы «Алгоритмика для дошколят».

Были разработаны – и сегодня активно используются дошкольными учреждениями, следующие две программы.

Дополнительная общеобразовательная программа «Алгоритмика «Старт» 4+» для детей 4–8 лет. Эта программа рассчитана на три года обучения и создана для погружения дошкольников в программирование уже начиная с четырехлетнего возраста;

Парциальная образовательная программа «По алгоритмическим дорожкам» для детей 5–8 лет. Данная программа составлена на два года обучения, рассчитана на дошкольников начиная с пяти лет и построена на использовании сюжетов и персонажей классических художественных произведений для детей

В структуре каждой из двух программ пять последовательных этапов обучения программированию:

1) вызов интереса;
2) контроль имеющихся усвоенных знаний и введение нового;
3) беспланшетные игры и переход в цифровую среду;
4) физические упражнения;
5) рефлексия.

Все занятия и в рамках первой внедренной годовой программы «Алгоритмика для дошколят» и в рамках двух созданных позже трехлетней и двухлетней программ эксплуатируют некий единый сюжет – космическую легенду о роботах «ПиктоМира» и их предназначении. Уже на ранних стадиях обучения, до перехода к работе в цифровой среде

в так называемый «допланшетный период», дети успешно справляются с построением линейных алгоритмов. Наряду с воображаемыми космическими обстановками, в начальном обучении используются реальный предметы и сюжеты действий в реальном мире: сочленяемые коврики с цифрами, мягкие фигурки для представления экранных роботов, реальный робот Ползун с пультовым управлением, маршрутные стрелки, карточки и кубики с пиктограммами команд. При этом, даже после введения в учебный процесс цифровой среды, бóльшую часть времени занятия по-прежнему, занимает бескомпьютерная образовательная часть занятия.

Такая образовательная деятельность в реальном, материальном мире формирует заинтересованность ребенка программированием, способна мотивировать ребёнка к последующему погружению в данную предметную область, способна избавить ребенка от ненужных стрессов при переходе в школу и усложнении осваиваемого материала в сторону меньшей конкретности.

4. Расширение полученного опыта в дошкольных учреждениях. Профессиональное сообщество «ПиктоМир»: профессиональное взаимодействие

Любое образовательное учреждение РФ, обладающее комплектом из 10, 12 или 15 планшетов или любых других компьютеров сегодня имеет возможность интегрировать «ПиктоМир» в свой образовательный процесс в группах размером 10, 12 или 15 детей соответственно, выбрав один из двух возможных способов.

4.1. Самостоятельная интеграция

Поскольку предметно-цифровая образовательной среда «ПиктоМир» свободно распространяема, доступна для всех популярных операционных систем и может быть беспрепятственно установлена практически на любом оборудовании, то любой заинтересованный педагог может самостоятельно внедрить «ПиктоМир» в образовательный процесс. Необходимые для этого программное обеспечение, программно-методическое обеспечение и чисто методическое обеспечение могут быть свободно скачаны с сайта НИЦ «Курчатовский институт»-НИИСИ и использованы для осуществления образовательной деятельности в любых формах, в том числе и в коммерческих целях.

На сегодняшний день образовательное учреждение, обладающее комплектом планшетов или других компьютеров в нужном количестве, может самостоятельно развернуть следующие курсы алгоритмики.

Годовой курс для дошкольников «Алгоримика 1-30».

Для проведения этого курса с сайта НИИСИ могут быть скачаны:

а) Собственно ЦОС «ПиктоМир» и комплект из 30 проводимых в «ПиктоМире» компьютерных игр, по одной игре для каждого из 30 занятий. Это комплект носит название Мир «Алгоритмика 1-30».

б) Подробная поминутная (поэпизодная) методичка по проведению 30 получасовых занятий с номерами 1-30.

Годовой курс для дошкольников «Алгоримика 1-36».

Для проведения этого курса с сайта НИИСИ могут быть быть скачаны:

а) ПиктоМир в комплекте с Миром «Алгоритмика 1-36».

б) Подробная поминутная (поэпизодная) методичка по проведению 36 получасовых занятий с номерами 1-36. Объем методички более 200 страниц.

Годовые курсы «Алгоримика 1-30» и «Алгоритмика 1-36» очень похожи. Отличия в том, что курс «Алгоримика 1-30» ориентирован на проведение всех активностей детей в чисто цифровой среде и не требует закупки дополнительного оборудования. В то время, как более эффективный методически курс «Алгоритмика 36» рассчитан на организацию активностей детей в предметно-цифровой среде, что требует закупки дополнительного оборудования.

Годовой курс «Алгоритмика 31-60».

Это курс для второго года обучения алгоритмике. Он продолжает курс «Алгоритмика 1-30» и так же как курс первого года может быть проведен без закупки дополнительного оборудования. Курс может быть использован для детей, которые прошли курс «Алгоритмика 1-30» в подготовительной группе и продолжают изучение алгоритмики в первом классе. Кроме того, пара курсов «Алгоритмика 1-30» и «Алгоритмика 31-60» могут быть использованы в школьной системе образования, как двухлетний курс, например, в 1-2 классах или в 3-4 классах при условии проведения одного занятия в неделю, или как годовой курс при условии проведения двух занятий в неделю.

Для проведения курса «Алгоритмика 31-60»

с сайта НИИСИ могут быть скачаны:

а) ЦОС «ПиктоМир» вместе с Миром «Алгоритмика 31-60».

б) Подробная поминутная (поэпизодная) методичка по проведению 30 получасовых занятий с номерами 31-60.

Для того, чтобы облегчить дошкольным образовательным учреждениям внедрение курсов «Алгоритмика 1-30» и «Алгоритмика 1-36» в НИИСИ разработан и размещен в открытом доступе шаблон документа под названием

Дополнительная общеобразовательная программа - общеразвивающая программа «Алгоритмика 1-30» технической направленности

Этот шаблон построен в соответствии с требованиями органов образования РФ. Аналогичный шаблон разработан и для курса *«Алгоритмика 1-36».*

4.2. Присоединение к сетевой инновационной площадке «ПиктоМир»

Эта площадка была организована в НИИСИ РАН в 2020 году. Сегодня образовательное учреждение может присоединиться и получить официальный статус инновационной площадки НИЦ «Курчатовский институт» – НИИСИ. Такое присоединение позволит педагогам организации получать регулярную методическую поддержку от разработчиков проекта «ПиктоМир», включая консультационную помощь и доступ к большому количеству методических материалов, накопленных на площадке и доступных ее участникам. Кроме того, присоединение к площадке позволит образовательной организации участвовать в региональных и федеральных мероприятиях и публикациях, регулярно организуемых на площадке.

На конец октября 2025 года в число участников сетевой инновационной площадки «ПиктоМир» входит 630 российских дошкольных образовательных учреждений (детских садов) и 210 школ. Анализ 50 наиболее подробных отчетов дошкольных учреждений - участников инновационной площадки за 2024/2025 учебный год, показывает, что в этих ДОУ было проведено более 5000 занятий, из них более половины, а именно, около 2900, проводилось с использованием робототехнического набора «ПиктоМир», то есть проводилось не в чисто цифровой, а в предметно-цифровой среде. Количество детей, занимающихся на данных площадках: около 2300, из них около 400 детей с ОВЗ. По выраженным в отчетах субъективным мнениям педагогов, успешно осваивают программу 85% детей, причем около 70% детей демонстрируют высокий уровень освоения материала.

В число основных достижений в развитии детей благодаря участию в программе, которые отмечают в своих отчетах педагоги, в первую очередь входят формирование алгоритмического мышления, освоение основ программирования, развитие компьютерной грамотности, повышение познавательного интереса к современным технологиям.

Профессиональное сообщество площадки «ПиктоМир» состоит из практикующих педагогов, которые систематически, из недели в неделю и из года в год используют и развивают предметно-цифровую среду «ПиктоМир. Это сообщество представляет собой активно развивающуюся экосистему, где педагоги участвуют не только во внедрении и распространении образовательной среды «ПиктоМир», но и сами совершенствуют современные образовательные технологии. Так, например, участники сетевой инновационной площадки «ПиктоМир» из 11 часовых поясов России, согласно предоставленным отчетам за 2024/2025 учебный год, разработали около 40 методических продуктов для работы с детьми, внедрили около 30 методических разработок по работе с родителями и представили свой инновационный опыт на 60+ мероприятиях разного уровня.

Собранный на площадке «ПиктоМир» в ходе многолетней практики опыт педагогов впечатляющ. Научные и методические результаты деятельности педагогов распространяются по России путем участия в профессиональных мероприятиях различного уровня. Педагоги регулярно выступают на межрегиональных, всероссийских и международных конференциях. Они активно представляют свой опыт работы с предметно-цифровой средой, делятся успешными методическими разработками по внедрению современных технологий в практику. Помимо этого, научная работа педагогов находит свое отражение в публикациях в научных и образовательных изданиях. Все результаты их практических разработок способствуют развитию в России современного образовательного пространства.

В 2024 году вышла книга «Сборник трудов педагогов-участников сетевой инновационной площадки ПиктоМир» под редакцией А.Г. Кушниренко и М.В. Райко [7]. Сборник представляет собой профессиональное издание, основанное на итогах работы педагогов-практиков в рамках сетевой инновационной образовательной деятельности. В сборнике собраны методические разработки и научные

статьи педагогов, принимающих участие в работе инновационной площадки. Эти разработки отражают практический опыт реализации предметно-цифровой среды «ПиктоМир» в дошкольных учреждениях и начальных классах.

Сборник фокусируется на описании эффективных методик работы с детьми и методическими рекомендациями по использованию образовательной среды «ПиктоМир». В материалах сборника подробно анализируется работа в каждой возрастной категории, выявляя ее особенности.

По материалам сборника мы можем выделить ключевые направления исследований:

а) формирование социального опыта детей дошкольного возраста через предметно-цифровую среду «ПиктоМир»;

б) адаптацию цифровой образовательной среды «ПиктоМир» для детей с нарушением зрения;

в) интеграцию в детскую игру алгоритмических задач;

г) сотрудничество с семьями для создания единого образовательного пространства.

Все материалы сборника адресованы педагогам дошкольных учреждений, учителям начальных школ, а также методистам и специалистам, которые работают в сфере образования. Электронная версия сборника доступна на сайте НИЦ «Курчатовский институт» - НИИСИ.

5. Методическое обеспечение проекта «ПиктоМир» достигло высокой степени зрелости

Методическое обеспечение предметно-цифровой среды «ПиктоМир» покрывает все потребности педагога и обеспечивает комплексный подход к обучению. Оно включает в себя систему материалов, направленных на эффективное освоение основ программирования, а также на развитие алгоритмического мышления. Все методические материалы строго выдерживают принцип доступности для разных возрастных категорий, начиная с дошкольного возраста и заканчивая младшими школьниками. Методическое обеспечение позволяет организовать постепенное вхождение ребенка в мир программирования, от легких линейных программ, к более сложным циклическим программам и программам с подпрограммами.

Перечислим содержимое учебно-методического комплекса более подробно. Комплекс включает в себя:

1) цифровую образовательную среду «ПиктоМир», которая свободно распространяется и доступна на сайте НИЦ «Курчатовский институт»-НИИСИ https://www.niisi.ru/piktomir.htm;
2) учебное пособие «Навигатор к учебно-методическому комплекту «Алгоритмика для дошкольников и учащихся начальных классов с использованием робототехнического образовательного набора и цифровой образовательной среды «ПиктоМир» [8] (первая версия этого методического пособия была издана в г. Самара ООО «Инженерная сила» в 2021 году;
3) парциальная образовательная программа по развитию алгоритмического мышления дошкольников "По алгоритмическим дорожкам" (для детей 5-8 лет) М.В. Богомолова, О.К. Пересыпкина, М.В. Райко [9];
4) упомянутые выше методические указания по проведению цикла занятий «Алгоритмика 1-30» в подготовительных группах дошкольных образовательных учреждений с использованием свободно распространяемой учебной среды «ПиктоМир» авторства А.Г. Кушниренко, А.Г. Леонов, М.В. Райко;
5) методические указания по проведению цикла занятий «Алгоритмика 31-60» для учащихся начальной школы с использованием свободно распространяемой учебной среды «ПиктоМир» авторства А.Г. Кушниренко, А.Г. Леонов, М.В. Райко;
6) доступные участникам сетевой инновационной площадки «ПиктоМир» конспекты и видеозаписи занятий курсов, проходящих апробацию на данной площадки;
7) доступные участникам сетевой инновационной площадки «ПиктоМир» конспекты и видеозаписи занятий третьего и четвертого года обучения алгоритмике в начальной школе (проводятся в гибридной среде программирования ПиктоМир-К) [10].

Как уже говорилось, методическое сопровождение курсов «Алгоритмика 1-30», Алгоритмика 1-36» и «Алгоритмика 31-60» включает подробный, буквально поминутный разбор всех занятий этих курс. В комплекте методического сопровождения также входят готовые раздаточные материалы для печати, которые можно распечатать и активно использовать в ходе практических занятий. Все

эти материалы четко структурированы и логически продуманы на основе многолетнего опыта.

6. Высокая степень отработанности дополнительной образовательной программы для дошкольников «Алгоритмика 1-36»

На сегодняшний день, отделом учебной информатики НИЦ «Курчатовский институт» - НИИСИ, завершена разработка годовой дополнительной образовательной программы технической направленности «Алгоритмика 1-36». Данная программа представляет собой комплексный образовательный курс, который рассчитан на детей возраста 5-7 лет. Основной фокус программы ориентирован на развитие алгоритмического мышления через работу в предметно-цифровой среде «ПиктоМир».

Объём программы составляет 36 получасовых занятий и включает в себя [11]:

- инвариантную часть из 13 занятий, знакомящих учащихся с базовыми принципами алгоритмического подхода и программирования;
- вариативную часть из 23 занятий, направленных на более глубокую проработку базового материала.

Программа реализуется в форме групповых занятий - 10-15 человек.

Занятия проводятся в деятельностно-игровой форме и в начальный период – мы называем его допланшетным – проходят без индивидуального использования компьютеров детьми. По завершении допланшетного периода начинается постепенное погружение в цифровую образовательную среду «ПиктоМир» путем индивидуального или кооперативного составления учебных программ на планшете или компьютере.

В ходе первых занятий учащиеся, часто в виде командной работы, знакомятся с роботами: Ползун, Вертун, Двигун, Тягун и на практике учатся создавать линейные алгоритмы.

В середине курса начинается более глубокое освоение методов и приемов составления алгоритмов: использование повторителей, подпрограмм, циклов и команд с обратной связью, а также использование конструкции ветвления в форме «если-то».

Задания-игры в данной образовательной программе распределены таким образом, чтобы обеспечить последовательное формирование у детей алгоритмического мышления через составление требуемых последовательностей команд, отладку полученного алгоритма через пошаговую проверку правильности хода выполнения программы в цифровой среде. Успешный опыт планирования будущих действий, приобретенный в ходе освоения азов программирования, вольно или невольно оказывается для ребенка некоторым образцом при планированиях любых иных действий. Помимо конкретного опыта программирования и планирования будущих действий, в ходе прохождения курса у детей развивается логическое и пространственное мышление, цифровая грамотность, память и внимание, коммуникативные способности и навыки взаимопомощи.

7. Заключение

Обсуждаемые в настоящей статье программы курсов алгоритмики соответствует государственным стандартам и федеральным образовательным программам дошкольного и начального школьного образования, учитывают психофизические особенности дошкольников и младшеклассников.

Реализация программы опирается на создание сложной системы взаимодействий между участниками образовательного процесса. Педагог, создавая особую образовательную среду, является главным связующим звеном, который может раскрыть весь потенциал учащихся. Такой подход к организации образовательного процесса позволяет:

- развивать как технические, так и коммуникационные компетенции;
- развивать опыт конструктивного взаимодействия;

Особое значение, для детей дошкольного и младшего школьного возраста, имеет то, что все грани развития детей реализуются в игровой форме. Это позволяет делать процесс обучения естественным и привлекательным для детей, и при этом соответствующим всем нормативно-правовым документам, направленных на организацию образовательной среды в дошкольном учреждении и начальной школе. Представленные в настоящей статье курсы алгоритмики носят системный, комплексный характер, обогащают все сферы жизни ребенка, развивают познавательные, речевые и социально-коммуникативные способности ребенка, помогают формированию у ребенка целостную картину современного мира и создают прочную основу для успешной социализации и дальнейшего освоения естественно-научной тематики.

Работа выполнена в рамках государственного

задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2.1 этап 2025 года)Experience in Organizing an Additional Educational Program for Preschoolers and Firstgraders in the PiktoMir Subject-Digital Educational Environment

Experience in Organizing an Additional Educational Program for Preschoolers and Firstgraders in the PiktoMir Subject-Digital Educational Environment

A. I. Akhankina, A. G. Kushnirenko, A. G. Leonov, M. V. Rayko, U. M. Solopova

Abstract. This article presents long-term experience in implementing and developing the PiktoMir subject-digital environment in the preschool and primary school education system of the Russian Federation. Particular attention is paid to the methodological aspects of teaching the fundamentals of algorithms and programming, based on the experience of practicing teachers. Two versions of a year-long algorithms course for preschoolers aged 5-7 years and a continuing algorithms course for elementary school students are considered. Freely distributed software and software-methodological support for the courses are posted and available for download on the website of the National Research Center "Kurchatov Institute" - NIISI. The article discusses in detail the structural components and methodology of the yearlong course "Algorithmics 1-36," designed for 36 half-hour lessons and aimed at practical mastery of basic programming concepts and the development of algorithmic thinking in the PiktoMir subject-based digital environment.

Keywords: algorithmic thinking, programming, preschoolers, subject-based digital educational environment, PiktoMir, pictograms, algorithms, text-free system

Литература

1. Н.О. Бесшапошников, А. Г. Кушниренко, А.Г Леонов, М.В. Райко, О.В. Собакинских. Цифровая образовательная среда «Пиктомир»: Опыт разработки и массового внедрения годового годового курса программирования для дошкольников. Информатика и образование. 2020. № 10(319), 28-40.

2. И. Н. Грибанова, А. Г. Кушниренко, А. Г. Леонов, М. В. Райко. Отечественные программные средства и методики для годового курса «Алгоритмика для дошколят». Ученые записки НТГСПИ. Серия: Педагогика и психология. 2025. № 2, 21-27.

3. Методические указания по проведению цикла занятий «Алгоритмика» в подготовительных группах дошкольных образовательных учреждений с использованием свободно распространяемой учебной среды ПиктоМир. URL: https://www.niisi.ru/piktomir/m.pdf (дата обращения 20.10.2025).

4. А. Г. Кушниренко, А. Г. Леонов, И. Н. Грибанова, М. В. Райко. Годовой цикл занятий «Алгоритмика для дошкольников» в подготовительных группах дошкольных образовательных учреждений. Вестник кибернетики. 2018. № 2(30), 138-144.

5. А. Г. Леонов, М.В. Райко, О.В. Собакинских, Н.В. Собянина. Результаты освоения годовой программы "Алгоритмика для дошколят" подготовительными группами муниципального ДОУ. Труды НИИСИ РАН (2020). Том 10, номер 5-6, стр. 195-199, URL: https://trudy.niisi.ru/2020_T10_N5.pdf (дата обращения 20.10.2025).

6. И. Алькина, Е. Жукова, О. Пересыпкина и др. Алгоритмика для дошкольников. «Обруч». 2023. №2, 12-13.

7. А. Г. Кушниренко, М. В. Райко. Сборник трудов педагогов – участников сетевой инновационной площадки «ПиктоМир» по итогам работы в 2024 году: сборник научных статей. М., Наука, 2024.
URL: https://www.niisi.ru/sbornik2024.pdf (дата обращения 20.10.2025).

8. А. Г. Леонов, М.В. Райко и др. Навигатор к учебно-методическому комплекту «Алгоритмика

для дошкольников и учащихся начальных классов с использованием робототехнического образовательного набора и цифровой образовательной среды ПиктоМир»: методическое пособие / М.В. Райко [и др.]. — Самара: ГАУ ДПО СО "Самарский областной институт повышения квалификации и переподготовки работников образования", 2023. — 69 с. — ISBN 978-5-98229-456-2.

9. М.В. Богомолва, О.В. Пересыпкина, М.В. Райко М. Парциальная образовательная программа по развитию алгоритмического мышления дошкольников «По алгоритмическим дорожкам» (для детей 5-8 лет)// «Инженерная сила» — 2024 — 119 с. — ISBN 978-5-907707-50-4.

10. А. Г. Леонов, М.В. Райко. Знакомство с ЦОС Пиктомир-К: от знаков к тексту // STEAM-образование: от дошкольника до выпускника вуза: материалы Всероссийской научно-практической конференции / под ред. Е.В. Малеевой, Ю.В. Скоробогатовой. — Нижний Тагил; Екатеринбург: Изд-во НТГСПА, 2023. — С. 22–27.

11. Программа «Алгоритмика 1-36». URL: https://www.niisi.ru/piktomir.htm (дата обращения 20.10.2025).

Статическая инфраструктура для сборки кросс-компилятора

А. А. Асонов[1], С. В. Самборский[2]

[1]НИЦ «Курчатовский институт» - НИИСИ, Москва, РФ, asonow@niisi.ras.ru;

[2] НИЦ «Курчатовский институт» - НИИСИ, Москва, РФ, sambor@niisi.msk.ru

Аннотация. В работе рассматривается проблема эксплуатации кросс-компилятора на базе GCC и бинарных утилит для отечественных микропроцессоров в условиях разнообразия аппаратных платформ и дистрибутивов ОС Linux. Показано, что использование разделяемых библиотек (прежде всего glibc) и зависящих от конкретного дистрибутива сборок приводит к множеству несовместимостей и усложняет сопровождение. Предложена статическая «самодостаточная» инфраструктура для сборки кросс-компилятора, включающая компилятор, бинарные утилиты и набор библиотек, собранные с использованием mus - альтернативной реализации стандартной библиотеки языка С. Описана многошаговая процедура раскрутки (bootstrapping), позволяющая получить конечный комплект инструментов, не использующий разделяемые библиотеки и минимально зависящий от параметров конкретной системы (кроме разрядности процессора и интерфейса ядра Linux). Обсуждаются ограничения, связанные с отказом от разделяемых библиотек (поддержка LTO, санитайзеров), и возможные направления расширения инфраструктуры за счет включения дополнительных утилит для сборки, тестирования и отладки.

Ключевые слова: кросс-компилятор, статическая линковка, инфраструктура сборки, GCC, бинарные утилиты, musl libc, glibc, отечественные микропроцессоры, Linux-дистрибутивы

1. Введение

При портировании и доработке для отечественных микропроцессоров и ОС реального времени кросс-компилятора на основе GCC [1] и бинарных утилит [2, 3] возникла проблема с тем, как обеспечить эксплуатацию этих утилит на разнообразной вычислительной технике.

Даже если ограничиться только операционной системой Linux на 32-битной архитектуре Intel (80386 и выше) и 64-битной архитектуре AMD (X86-64), компьютеры могут отличаться разрядностью процессора, установленными дистрибутивами ОС Linux, версиями этих дистрибутивов и ранее установленным программным обеспечением.

Можно точно задать конфигурацию компьютера для эксплуатации компилятора и бинарных утилит, точно указав версию дистрибутива, и потребовать, чтобы использовалась только эта конфигурация. Такой подход оказался не очень удобен, так как выбор дистрибутива ОС Linux и его версии часто определяется необходимым дополнительным программным обеспечением, опытом и предпочтениями пользователя или историческими причинами.

Использование для эксплуатации компилятора и бинарных утилит отдельного компьютера (пусть даже виртуального) крайне неудобно. Технологии контейнеризации, подобные docker, могут быть незнакомы пользователям, а также требуют достаточно свежего ядра системы Linux.

Собирать кросс-компилятор и бинарные утилиты по отдельности для использования на разных версиях всевозможных дистрибутивов ОС Linux – плохое решение, так как потребует много времени и соответствующих компьютеров (хотя бы виртуальных). А главное: на разработчиков возлагается ответственность за работоспособность каждого варианта сборки и, следовательно, требуется отдельное тестирование.

Более реалистичный подход – собирать ограниченное число вариантов поставок, для конкретных конфигураций, на которых работоспособность гарантируется, но при этом стремиться к тому, чтобы собранные кросс-компилятор и бинарные утилиты запускались и нормально работали на разнообразных (по возможности) компьютерах с операционной системой Linux.

2. Причины несовместимости

Причины, по которой собранные на одном компьютере программы не будут работать на другом компьютере, можно разделить на две основные группы: аппаратные несовместимости и программные различия.

Очевидная аппаратная несовместимость

возникает при попытке запустить 64-битные программы (скомпилированные для x86-64) на 32-битном процессоре. Впрочем, сейчас 32-битные процессоры остались в различном встроенном оборудовании, где не требуется запускать компилятор и бинарные утилиты. Настольных компьютеров и, тем более, серверов с 32-битными процессорами осталось очень мало.

Другая аппаратная проблема возникает с тем, что при сборке некоторых библиотек, на этапе конфигурирования изучается процессор компьютера, на котором идет сборка, для того чтобы использовать возможности данного процессора (обычно векторное расширение) с целью повышения производительности изготавливаемой библиотеки. В результате собранная с использованием этой библиотеки программа не будет работать на компьютере с процессором, не поддерживающим эту возможность. Именно это может случиться с библиотекой GMP, реализующей арифметику повышенной точности, необходимую для сборки компилятора GCC, подробнее можно прочитать в [4].

Для борьбы с такой несовместимостью нужно запретить настройку на конкретный процессор при сборке программного обеспечения. Во всех случаях следует указывать при конфигурации базовую архитектуры без всяких расширений и дополнений. Здесь следует иметь в виду, что вызванная этим потеря в производительности относится только к скорости компиляции, а не к производительности программ для отечественных микропроцессоров, скомпилированных этим компилятором.

Несовместимости, вызванные программными различиями, возможны разные, но для компилятора и бинарных утилит основная причина, препятствующая их установке на компьютеры с разной конфигурацией – использование разделяемых библиотек. В случае простого отсутствия необходимой разделяемой библиотеки ее можно дополнительно установить, если есть свободный доступ в интернет. Хуже ситуация, когда для нового программного обеспечения требуется версия разделяемой библиотеки более старая, чем уже установленная на компьютер. Одновременное использование нескольких версий одной библиотеки возможно, но часто требует ручной настройки.

Еще хуже, если на компьютер установлены разделяемые библиотеки, специфичные для конкретного дистрибутива ОС Linux. Тогда при сборке программного обеспечения на этом компьютере могут быть созданы исполняемые файлы, которые в принципе нельзя будет запустить на компьютерах с другими дистрибутивами.

Для того чтобы избежать этих проблем, желательно собирать программное обеспечение так, чтобы минимизировать зависимости от прочего программного обеспечения. Подобные «самодостаточные» программные пакеты должны содержать в себе все необходимые разделяемые библиотеки или не использовать их совсем.

Вариант без разделяемых библиотек выглядит предпочтительнее, поскольку выигрыш от собственных разделяемых библиотек – небольшая экономия дискового пространства и оперативной памяти, что не так актуально для современных компьютеров. При этом будет проигрыш в усложнении процесса запуска программы и потенциальных проблемах с тем, что динамический загрузчик не найдет нужную библиотеку.

Среди разных библиотек следует особенно выделить библиотеку libc, которая содержит runtime-поддержку языка C (ввод/вывод, динамическая память, работа со строками и еще очень много всего). Кроме этого, libc обеспечивает интерфейс с операционной системой, поэтому практически невозможно представить полезную программу, запускаемую в ОС Linux, которая бы не использовала прямо или косвенно функции из libc.

В большинстве дистрибутивов ОС Linux библиотека libc реализована в пакете glibc [5, 6] (и связанных с ним). Эта реализация содержит весьма продвинутую функциональность, но ценой большой сложности. При этом есть статичная (неразделяемая) версия libc (пакет glibc-static в дистрибутиве Fedora), но с неразделяемой версией libc много проблем, что признают авторы дистрибутива:

«*The glibc-static package contains the C library static libraries for -static linking. You don't need these, unless you link statically, which is highly discouraged*».

Компилятору и бинарным утилитам требуется не очень много функциональности libc, поэтому собрать их с неразделяемой версией libc из glibc-static удается, но с проблемами, которые к тому же зависят от версии библиотеки.

3. Альтернативы glibc

Существует много реализаций runtime-поддержки языка C кроме glibc. Некоторые альтернативные реализации libc ориентированы

на встроенные системы, ограничены по возможностям, зато крайне нетребовательны к ресурсам. Другие реализации полностью поддерживают стандартные возможности и могут быть полноценной заменой glibc, и действительно заменяют glibc в некоторых дистрибутивах ОС Linux.

Одной из таких полноценных альтернатив является пакет musl [7]. Данная библиотека доступна с 2011 года и продолжает развиваться, а в качестве одного из основных преимуществ заявлена качественная поддержка статической линковки, т. е. использования неразделяемой библиотеки (в документации musl охарактеризована как «static-linking-friendly»). Поэтому musl представляется подходящей заменой glibc для сборки самодостаточного программного обеспечения.

4. Что требуется

Поставлена цель: сделать так, чтобы можно удобно собирать кросс-компилятор и бинарные утилиты для отечественных микропроцессоров, не используя никаких разделяемых библиотек, получая тем самым «самодосточный» пакет без зависимостей. Самый удобный способ, чтобы не было путаницы с тем, какая именно библиотека использована: изготовить специальный комплект из компилятора (не кросс-компилятора!), бинарных утилит и всех необходимых библиотек. При этом, компилятор и утилиты должны быть настроены на использование по умолчанию «комплектных» библиотек. Тем самым исключаются недоразумения с ошибочно подключенным glibc.

Таким образом, задача сборки кросс-компилятора и кросс-утилит сводится к сборке (не «кросс») компилятора, утилит и библиотек. При этом, для того чтобы ими было удобно пользоваться на разных компьютерах, эту «инфраструктуру» также желательно собрать с неразделяемыми библиотеками.

5. Как сделано

Для того чтобы собрать самодостаточный комплект из компилятора, утилит и библиотек, была использована раскрутка («bootstrapping»), состоящая из последовательной сборки компиляторов, утилит и библиотек таким образом, чтобы последний вариант удовлетворял нашим требованиям.

Последовательные шаги:

1) Прежде всего копируются и распаковываются дистрибутивы GCC (компилятора), бинарных утилит, библиотеки musl и трех библиотек, необходимых для сборки GCC: GMP, mpfr и mpc.

2) Дальше компилируются только неразделяемые варианты библиотек GMP, mpfr и mpc и устанавливаются в первый временный каталог.

3) Затем собираются и устанавливаются бинарные утилиты во второй временный каталог. Далее собирается компилятор с использованием библиотек из первого временного каталога, при этом для изготовления своих библиотек и стартовых файлов (libgcc.a, crt1.o и т. п.) он использует бинарные утилиты из второго временного каталога, туда же он и устанавливается.

4) Далее еще один раз собираются бинарные утилиты уже при помощи бинарных утилит и компилятора из второго временного каталога и устанавливаются в третий временный каталог.

Потом изготавливается неразделяемая версия библиотек из пакета musl также при помощи бинарных утилит и компилятора из второго временного каталога, и устанавливается в третий временный каталог.

Еще раз собирается компилятор, уже при помощи компилятора из второго временного каталога, и устанавливается в третий временный каталог. При этом он собирается с библиотекой libc.a, установленной в третий временный каталог, т. е. уже не glibc, а musl. Поэтому новый компилятор конфигурируется так, что он собирается статически (без разделяемых библиотек).

5) Последний раз собираются бинарные утилиты и устанавливаются в окончательный каталог (где планируется эксплуатация инфраструктуры). Собираются они при помощи компилятора, бинарных утилит и библиотек (musl!) из третьего временного каталога.

С использованием компилятора и бинарных утилит из третьего временного каталога снова собираются библиотеки из пакетов musl, GMP, mpfr и mpc и устанавливаются в окончательный каталог.

Наконец собирается компилятор при помощи компилятора и бинарных утилит из третьего временного каталога, но библиотек из окончательного каталога. Компилятор устанавливается в окончательный каталог.

Все вышеописанное удобно оформить в виде единого рекурсивного make-файла, в котором используется параллельная сборка на всех доступных ядрах процессора. Если есть много оперативной памяти, то можно значительно ускорить процедуру, разместив все кроме окончательного каталога в оперативной памяти (используя файловую систему tmpfs). Осталось не забыть в конце удалить все временные каталоги (особенно, если они занимают место в

оперативной памяти).

6. Что получилось, не получилось и что еще можно сделать

Сборка всего установленного в окончательный каталог (и компилятора, и бинарных утилит) производится только с неразделяемыми библиотеками.

По-умолчанию сборка происходит для 32-битной (i686) или 64-битной (x86-64) архитектуры в зависимости от архитектуры операционной системы. Но можно также собирать на 64-битном компьютере инфраструктуру для сборки 32-битного программного обеспечения.

Возможно, часть шагов можно исключить, но так как вся описанная последовательность действий осуществляется только один раз, то нет необходимости ее сокращать. Более того, можно сделать еще один шаг, пересобрав все еще один раз с включенной оптимизацией (оптимизировать раньше – пустая трата времени). Также при последней сборке компилятора можно заказать дополнительные языки программирования, кроме C и C++, необходимых для пересборки компилятора.

Окончательный вариант инфраструктуры можно пополнить дополнительными утилитами. Цель – сделать так, что при установке на «чистую» машину было все необходимое не только для простой пересборки кросс-компилятора и бинарных утилит, но и для тестирования, отладки, модификации. Например, можно дополнительно собирать texinfo, isl, bison, flex, expect, tcl, dwarfdump, elfutils, gdb, diff, patch, automake, autoconf, libtool. Из всего этого наиболее актуален texinfo, так как если он отсутствует, то его необходимо в первую очередь пересобрать или установить собранный.

Сборка без разделяемых библиотек плохо совместима с некоторыми возможностями компилятора, например, lto («link time optimization»), а также с санитайзерами (верификация времени выполнения). Впрочем, технология lto не особенно популярна при изготовлении встроенного программного обеспечения.

С санитайзерами ситуация сложнее: во-первых, большая часть санитайзеров не может быть включена при кросс-компиляции, так как отсутствует поддержка на целевой платформе. Но сам кросс-компилятор (и кросс-утилиты) могли бы быть собраны с включенным тем или иным санитайзером. Кроме того, санитайзер неопределенного поведения, UBSan («Undefined Behavior Sanitizer») не требует никакой особенной поддержки. Тем самым желательно его включать при конфигурировании компилятора (как кросс-компилятора, так и компилятора для сборки кросс-компилятора).

7. Заключение

Можно собрать комплект из библиотек, бинарных утилит и компилятора для сборки кросс-компилятора и кросс-утилит без использования разделяемых библиотек. Причем сам этот комплект также не будет использовать разделяемых библиотек.

Тем самым исключаются все зависимости, кроме зависимости от аппаратуры (разрядность процессора) и интерфейса ядра ОС Linux.

Собственно, разделяемые библиотеки, и в первую очередь glibc, должны устранять зависимость от ядра, так как программы не делают системные вызовы сами, а вместо этого обращаются к библиотеке. Но похоже, что «лекарство хуже болезни»: если интерфейс ядра изменяется редко и при этом сохраняется поддержка старых программ, то glibc меняется существенно чаще.

«Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» - НИИСИ» по теме FNEF-2024-0001».

Static Infrastructure for Building a Cross Compiler

A. A. Asonov, S. V. Samborskiy

Abstract. The paper addresses the problem of deploying a GCC- and binutils-based cross-compiler toolchain for domestic microprocessors in the context of a wide variety of hardware platforms and Linux distributions. It is shown that the use of shared libraries (primarily glibc) and distribution-specific builds leads to numerous incompatibilities and complicates maintenance. A static, self-contained infrastructure for building a cross-compiler is proposed, comprising the compiler, binary utilities, and a set of libraries built using musl – an alternative implementation of the C standard library. A multi-stage bootstrapping procedure is described that makes it possible to obtain a final toolchain which does not utilize shared libraries and only minimally dependent on the parameters of a specific system (apart from CPU word size and the Linux kernel interface). The limitations associated with abandoning shared libraries (LTO support, sanitizers) are discussed, as well as possible directions for extending the infrastructure by adding auxiliary tools for building, testing, and debugging.

Keywords: cross-compiler, static linking, build infrastructure, GCC, binutils, musl libc, glibc, domestic microprocessors, Linux distributions

Литература

1. GCC, the GNU Compiler Collection. https://gcc.gnu.org/ (дата обращения 08.11.2025).
2. GNU Binutils. https://www.gnu.org/software/binutils/ (дата обращения 08.11.2025).
3. gdb and binutils. https://sourceware.org/git/binutils-gdb.git (дата обращения 08.11.2025).
4. В. А. Галатенко, Г. Л. Левченкова, С. В. Самборский. Особенности сборки кросс-компилятора GCC и бинарных утилит. «Труды НИИСИ РАН», Т. 12 (2022), № 4, 43-49.
5. The GNU C Library. https://www.gnu.org/software/libc/ (дата обращения 08.11.2025).
6. The GNU C Library (glibc). https://sourceware.org/glibc/ (дата обращения 08.11.2025).
7. musl libc. https://musl.libc.org/ (дата обращения 08.11.2025).

Разработка веб-ресурса для индивидуального подбора косметических средств для лица

Е. С. Галайтата[1], С. Г. Еловой[2]

[1]БУ ВО «Сургутский государственный университет», Сургут, Россия, galajtata_es@edu.surgu.ru;

[2]БУ ВО «Сургутский государственный университет», Сургут, Россия, elovoj_sg@surgu.ru

Аннотация. В научно-исследовательской работе представлен проект разработки веб-ресурса «ClearSkin». Разрабатываемая информационная система позволит пользователям получать персонализированные рекомендации по уходу за кожей на основе ответов в опросе, что сделает выбор косметики более осознанным и эффективным. На начальном этапе научно-исследовательской работы была изучена предметная область. Проведен обзор аналогов разрабатываемой информационной системы, выявлены отличия, а также достоинства и недостатки каждой из систем. Исходя из этого, сделан вывод, что рассматриваемые ресурсы имеют ограниченный функционал, с невозможностью полностью охватить решение поставленной задачи, поэтому разработка собственного веб-ресурса позволит наиболее полно отвечать существующим потребностям пользователей.

Ключевые слова: веб-сайт, персонализация, опрос, рекомендации, интерфейс пользователя, прототипирование, информационная система

1. Введение

Кожа человека – это уникальный орган, который выполняет множество жизненно важных функций. Прежде всего, она служит защитным барьером, оберегая организм от негативного воздействия окружающей среды - ультрафиолетового излучения, перепадов температур и загрязнений. Кроме того, кожа участвует в процессах терморегуляции, дыхания и обмена веществ. Ее состояние напрямую влияет на наше самочувствие и внешний вид.

Особое значение имеет кожа лица – самая заметная и в то же время наиболее уязвимая часть нашего тела. В отличие от других участков, она практически постоянно находится под воздействием внешних факторов и поэтому требует особого внимания и ухода. Современные исследования в области дерматологии показывают, что правильный уход за кожей лица способен не только поддерживать ее здоровье, но и предотвращать преждевременное старение, сохраняя молодость и свежесть на долгие годы.

Актуальность данной работы обусловлена необходимостью создания веб-ресурса, который автоматически подбирает персонализированную систему ухода за кожей лица на основе ответов пользователя на вопросы о состоянии его кожи. Такой подход поможет избежать ошибок при выборе косметики и сделать уход более эффективным.

Разрабатываемая система имеет два ключевых сценария применения. Первичный и основной сценарий предполагает ее самостоятельное использование конечными потребителями для получения первичных, обоснованных рекомендаций по подбору косметики в удобном онлайн-формате. Второй, профессиональный сценарий, рассматривает «ClearSkin» как инструмент поддержки работы косметологов. Специалист может использовать систему для предварительного сбора анамнеза и формирования базовых рекомендаций перед очной консультацией, что позволяет оптимизировать время приема, повысить его объективность и наглядность для клиента.

Такая двойственная направленность позволяет охватить как широкую аудиторию, заинтересованную в самостоятельном уходе, так и профессиональное сообщество, нуждающееся в цифровых помощниках для стандартизации и улучшения клиентского опыта.

2. Сравнение аналогов

Выбрано четыре оценочных критерия для сравнительного анализа конкурирующих решений [2]:

1. Разнообразие брендов – Этот критерий определяет широту представленности производителей косметической продукции в формируемых подборках уходовых средств. Данный параметр напрямую влияет на вариативность потребительского выбора и

потенциал для оптимального подбора индивидуальной косметики.

2. Описание этапов подобранного ухода – Критерий характеризует степень проработанности и доступности разъяснений относительно рекомендуемых процедур для различных типов кожи.

3. Дополнительная информация о типах кожи - Этот критерий включает в себя наличие образовательного контента о различных типах кожи.

4. Удобство использования - Этот критерий оценивает, насколько интуитивно понятен и удобен интерфейс сайта. Важно, чтобы пользователи могли легко находить нужную информацию, быстро проходить опрос на определение типа кожи и получать рекомендации по уходу.

Сравним 3 аналога веб-сайта по определению типа кожи по выбранным критериям в таблице 1. Таблица приведена в Приложении 1. Сделан вывод, что рассматриваемые аналоги имеют ограниченный функционал, с невозможностью полностью охватить решение поставленной задачи. Поэтому разработка собственного веб-сайта позволит наиболее полно отвечать существующим потребностям пользователей.

3. Проектирование

В ходе анализа функциональных требований была разработана UML-диаграмма вариантов использования, которая отображает основные сценарии взаимодействия пользователя с системой (см. Приложение 2, Рис. 1) [4, 5].

Для визуализации структурной организации системы и взаимосвязей между её компонентами была создана диаграмма компонентов (см. Приложение 3, Рис. 2) [3].

Спроектированная структура базы данных (см. Приложение 4, Рис. 3) включает четыре взаимосвязанные таблицы. Центральное место в структуре занимает таблица «Продукт», которая взаимодействует с тремя справочными таблицами: «Бренд», «Этап» и «Тип_кожи». В таблице «Бренд» систематизирована информация о производителях косметической продукции, «Этап» содержит классификацию ступеней уходового ритуала, а «Тип_кожи» — категории дерматологических характеристик. Ключевая таблица «Продукт» аккумулирует сведения о наименовании косметического средства, его визуальном представлении, рекомендациях по применению, а также содержит ссылки на связанные сущности через внешние ключи, обеспечивая корреляцию с соответствующим брендом, стадией ухода и типом кожи.

На этапе прототипирования интерфейса с помощью платформы Figma был разработан кликабельный прототип веб-сайта для персонализированного подбора поэтапного ухода за кожей лица. Результаты, включающие макеты главной страницы, страницы опроса и раздела с информационными материалами, представлены в Приложении 5 (Рис. 4–6) [8, 9].

4. Описание методологии

Алгоритм определения типа кожи реализует вероятностный подход к классификации на основе анализа ответов пользователя в специализированном опросе. Каждому возможному типу кожи присваивается буквенный идентификатор из набора A, B, C, D. На этапе подготовки системы каждый вариант ответа в анкете получает вероятностный вес в процентах и набор кодов релевантных типов кожи, которым соответствует данный ответ.

В процессе анкетирования пользователь последовательно отвечает на вопросы системы. После завершения опроса для каждого типа кожи вычисляется суммарный вероятностный вес на основе выбранных пользователем ответов. Итоговый тип кожи определяется как тот, который набрал наибольшую суммарную вероятность среди всех рассматриваемых категорий. В зависимости от полученного результата, система выводит пользователю персонализированные рекомендации в виде трёх готовых наборов уходовой косметики. Каждый набор представляет собой готовую последовательность шагов ухода (например, очищение, тонизирование и т.п.), где для каждого шага подобран конкретный продукт из базы данных системы.

Логика формирования конечных рекомендаций построена следующим образом. Пользователю демонстрируются три альтернативных варианта полного ухода за кожей лица. Каждый вариант является целостной линейкой продуктов от одного бренда и включает подробное описание последовательности применения всех средств (очищение, тонизирование и т.д.). Ключевыми критериями отбора конкретных продуктов в набор являются их соответствие определённому этапу ухода и выявленному типу кожи пользователя.

В текущей версии система не учитывает ценовое позиционирование брендов и сравнительную стоимость итоговых наборов. Данное ограничение является направлением для дальнейшего развития, в рамках которого планируется ввести фильтрацию и

ранжирование рекомендаций по ценовому сегменту для адаптации под индивидуальный бюджет пользователя.

5. Верификация результатов

Разработанный веб-сайт показал высокую эффективность, подтвержденную в ходе тестирования с точностью определения типа кожи 93%, в тестировании участвовали 15 пользователей. Рекомендации системы были составлены и проверены при участии трех экспертов-врачей-косметологов. Это доказывает, что веб-сайт «ClearSkin» является работоспособным и эффективным инструментом для персонализированного подбора косметических средств.

6. Заключение

В результате выполнения данной научно-исследовательской работы была успешно разработана концепция веб-сайта «ClearSkin» для персонализированного подбора косметического ухода. Актуальность проекта подтверждена анализом существующих аналогов, которые обладают ограниченным функционалом и не способны в полной мере удовлетворить потребности пользователей в индивидуальном подходе к выбору косметики.

В рамках исследования была подробно изучена предметная область, сформулированы цели и задачи системы, определены функциональные требования к веб-ресурсу. Проведенное сравнение аналогов позволило выявить ключевые преимущества разрабатываемого решения, включая разнообразие брендов, подробное описание этапов ухода и удобный интерфейс.

На этапе проектирования были созданы модели бизнес-процессов в нотации BPMN, разработаны UML-диаграммы вариантов использования и компонентов, а также спроектирован кликабельный прототип пользовательского интерфейса с помощью инструмента Figma. Это позволило визуализировать взаимодействие пользователя с системой и проработать архитектуру веб-сайта.

Приложение 1 – Сравнительный анализ

Таблица 1. Сравнительный анализ

Аналог/ Критерий	geltek.ru [6]	mesopharm.ru [1]	art-fact-products.com [7]	ClearSkin (моя ИС)
Разнообразие брендов	-	-	-	+
Описание этапов подобранного ухода	-	-	+	+
Дополнительная информация о типах кожи	+	+	+	+
Удобство использования	+	+	-	+

Приложение 2 – Диаграмма вариантов использования

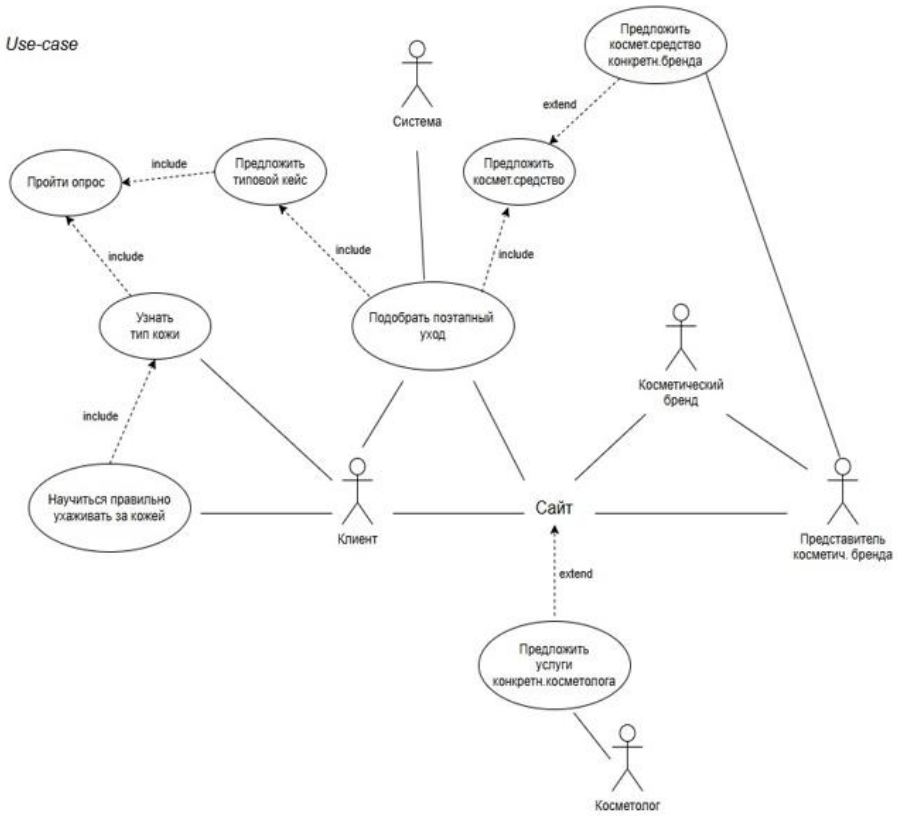

Рис. 1. Диаграмма вариантов использования

Приложение 3 – Диаграмма компонентов

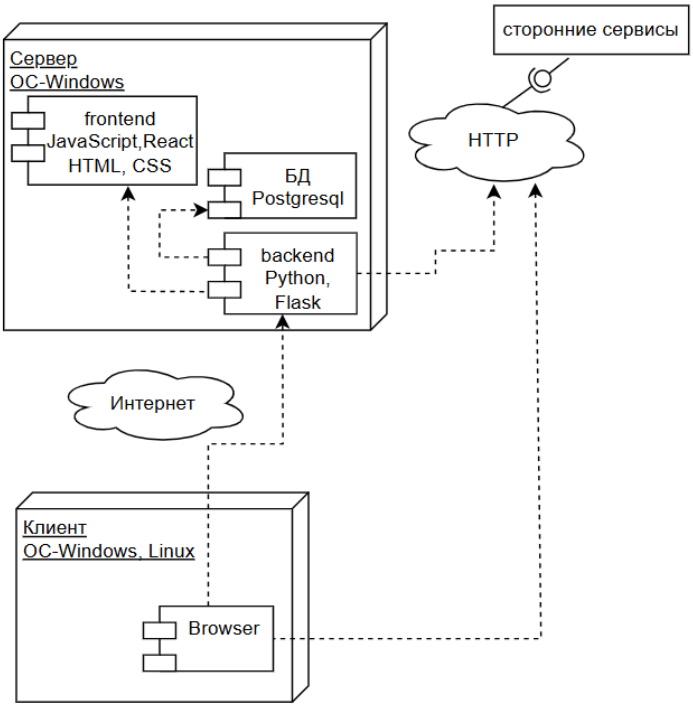

Рис. 2. Диаграмма компонентов

Приложение 4 – Схема базы данных

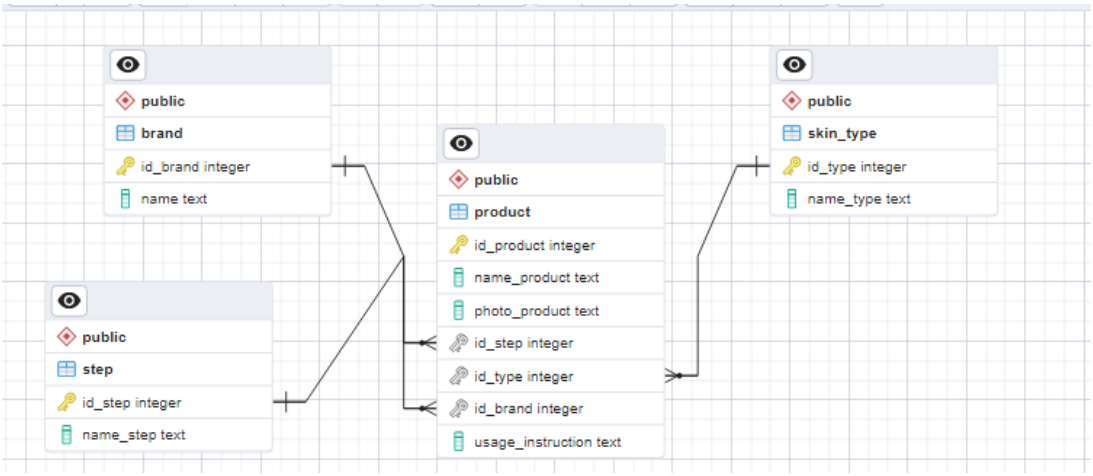

Рис. 3. Схема базы данных

Приложение 5 – Проектирование пользовательского интерфейса

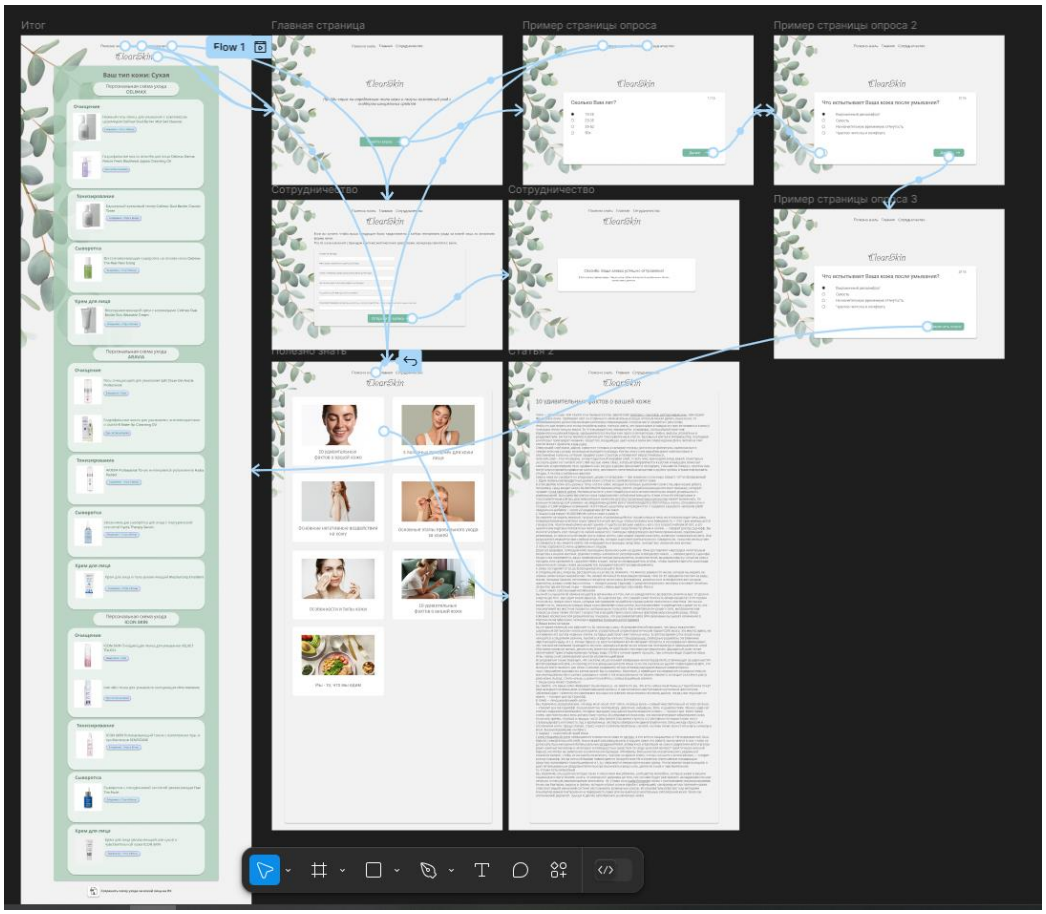

Рис. 4. Прототип интерфейса веб-сайта «ClearSkin»

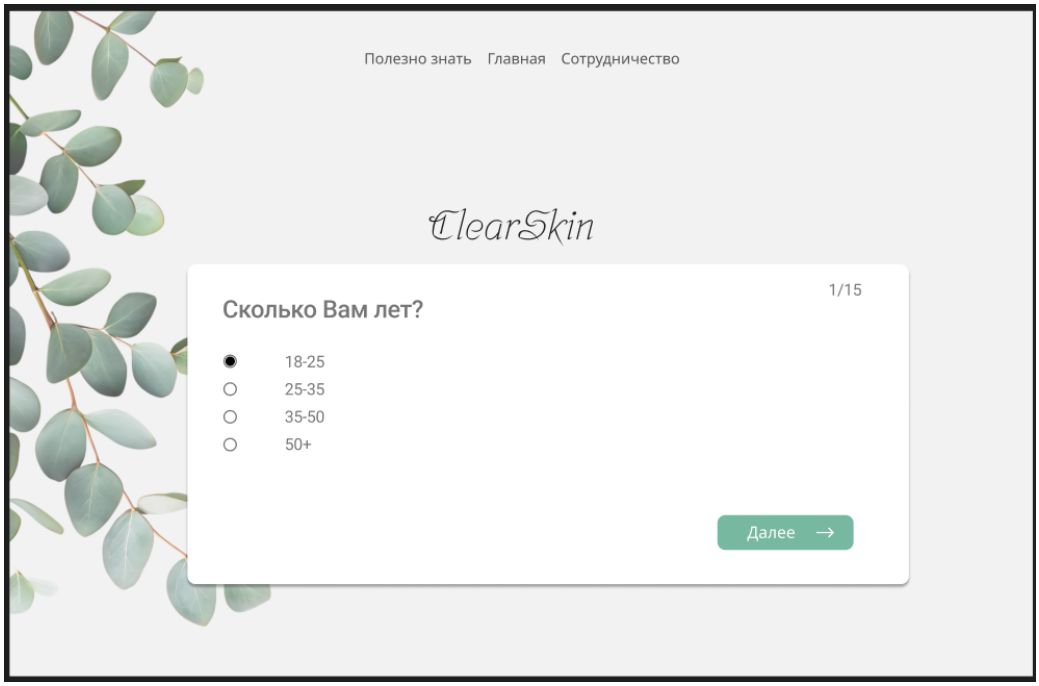

Рис. 5. Пример страницы опроса

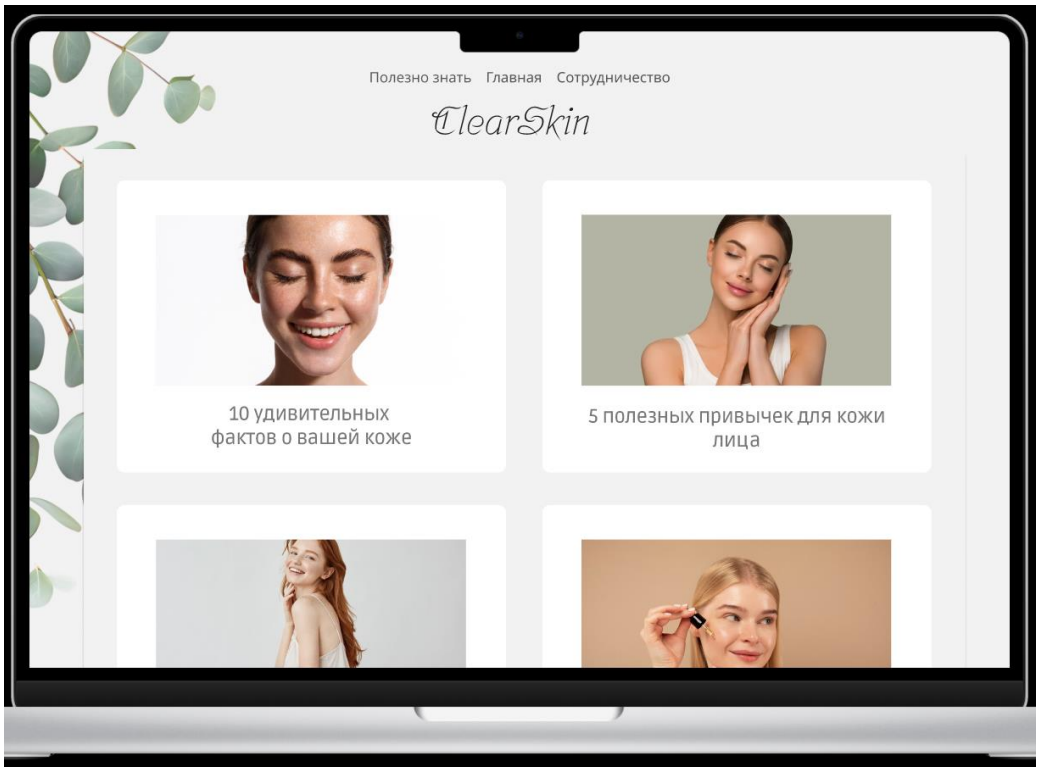

Рис. 6. Раздел веб-сайта «Полезно знать»

Development of a Web Resource for Individualized Selection of Facial Cosmetics

E. S. Galaitata, S. G. Elovoj

Abstract. This research paper presents the development project of the "ClearSkin" web resource. The proposed information system will enable users to receive personalized skincare recommendations based on their survey responses, thus making cosmetic selection more informed and effective.The initial stage of the research involved studying the subject domain. A review of analogues to the developed information system was conducted, identifying their distinctive features, advantages, and shortcomings. Based on this analysis, it was concluded that the considered resources possess limited functionality and cannot fully address the assigned task. Therefore, the development of a proprietary web resource will allow for the most comprehensive response to existing user needs.

Keywords: website, personalization, survey, recommendations, user interface, prototyping, information system

Литература

1. Анкета на определение типа кожи и подбор уходовой косметики от Mesopharm URL: https://mesopharm.ru/anketa-na-opredelenie-tipa-kozhi-i-podbor-ukhodovoy-kosmetiki-ot-mesopharm/ (дата обращения: 01.10.2025).

2. Галайтата, Е. С. Веб-ресурс персонализированного подбора косметического ухода / Е. С.Галайтата, С. Г. Еловой // XXVII Всероссийская студенческая научно-практическая конференция Нижневартовского государственного университета : Материалы конференции, Нижневартовск, 09–10 апреля 2025 года. – Нижневартовск: Нижневартовский государственный университет, 2025. – С. 287-292. – EDN ZDAJGO.

3. Дакет, Д. HTML и CSS: Разработка и дизайн веб-сайтов URL: пер. с англ. / Д. Дакет. – Москва, 2019 - .480 с.

4.Диаграмма вариантов использования (UseCase diagram) URL: https://flexberry.github.io/ru/fd_use-case-diagram.html (дата обращения: 05.11.2025).

5. Использование диаграммы вариантов использования UML при проектировании программного обеспечения URL: https://habr.com/ru/articles/566218/ (дата обращения: 05.11.2025).

6. Косметика для вашего типа кожи URL: https://geltek.ru/quiz/ (дата обращения: 05.11.2025).

7. Получите свой персонализированный подбор средств по уходу за кожей лица URL: https://www.art-fact-products.com/online-test/ (дата обращения: 05.11.2025).

8. Проектирование пользовательского опыта (UX/UI Design) URL: https://hsmi.msu.ru/curriculums/stp/program/proektirovanie-polzovatelskogo-opyta-uxui-design (дата обращения: 29.09.2025).

9. Сычев, А. В. Теория и практика разработки современных клиентских веб-приложений [Текст]: курс лекций / А.В. Сычев. – Москва: Интуит НОУ, 2016. — 483 с.

Наименование: сетевой рецензируемый научный журнал «Труды НИИСИ»
ISSN: 3033-6422
Сведения о переименовании: до 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН», ISSN 2225-7349
Журнал основан: 2011 г.
Периодичность: 4 раза в год
Учредитель и издатель: НИЦ «Курчатовский институт» — НИИСИ
Главный редактор: Бетелин Владимир Борисович, д. ф.-м. н., профессор, академик РАН
Адрес учредителя и издателя: 117218, Москва, Нахимовский проспект, д.36, к.1
Адрес редакции: 117218, Москва, Нахимовский проспект, д.36, к.1
Контакты: Тел.: +7 (925) 924-83-46; muranov@niisi.msk.ru
Подписка: Электронная версия журнала находится в свободном доступе на сайте журнала, а также в базах данных открытого доступа

Title: *SRISA Proceedings* online peer-reviewed journal
ISSN: 3033-6422
Former title: until 2025, the journal was published in print under the name *Trudy NIISI RAN* (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349
Published since: 2011
Publication frequency: Quarterly
Founder and publisher: NRC "Kurchatov Institute" - SRISA
Chief Editor: Vladimir B. Betelin, Doctor of Science (Phys&Math), Prof., member of the Russian Academy of Sciences,
Address of the founder and publisher: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1
Address of the editorial office: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1
Contacts: Tel.: +7 (925) 924-83-46; muranov@niisi.msk.ru
Subscription: The electronic version of the journal is freely available on the journal's website, as well as in open access databases

Подписано в печать 28.11.2025 г.
Формат 60x90/8
Печать цифровая. Условных печатных листов – 9,0