УДК 004.023 DOI 10.18522/2311-3103-2025-6-33-46

А.А. Диченко, И.И. Левин, Д.А. Сорокин

МЕТОД ГЕНЕРАЦИИ ТОПОЛОГИЧЕСКИХ ОГРАНИЧЕНИЙ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР ДЛЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Для реконфигурируемых вычислительных систем на базе ПЛИС эффективными прикладными программами являются параллельно-конвейерные программы, обеспечивающие реальную производительность более 50% от пиковой. Статья посвящена решению проблемы сокращения времени их разработки. Вычислительные структуры таких программ используют большой объём вычислительного ресурса ПЛИС, функционирующих на высокой тактовой частоте. Однако одновременная максимизация объёма задействованного ресурса ПЛИС и тактовой частоты находится в некотором противоречии, поскольку при большом заполнении снижается вариативность размещения функциональных узлов вычислительных структур и коммутационная матрица ПЛИС при трассировке информационных каналов между ними не обеспечивает требуемых характеристик по времени распространения сигналов. Более того в современных САПР алгоритмы размещения и трассировки учитывают только архитектурные и геометрические особенности ПЛИС. Поэтому при использовании большого числа специализированных примитивов, вариативность размещения которых крайне мала, достижение высоких тактовых частот в автоматическом режиме синтеза практически невозможно. Для решения этой проблемы также необходимо учитывать информационные зависимости между функциональными узлами вычислительных структур, но характер информационных зависимостей решаемых задач различных предметных областей может существенно отличаться. Поэтому разработчики вынуждены каждый раз вручную размещать на ПЛИС функциональные узлы путём создания скриптовых инструкций топологических ограничений. Время формирования топологических ограничений для ПЛИС прежних поколений было приемлемым, поскольку они содержали, как правило, до нескольких сотен специализированных примитивов. Однако в современных ПЛИС их количество достигает нескольких тысяч и даже десятков тысяч штук, что приводит к значительному увеличению времени разработки эффективных прикладных программ. Предлагаемый метод позволяет автоматизировать процесс разработки топологических ограничений вычислительных структур. Исследования были проведены при разработке прикладных программ решения ряда задач на основе алгоритмов БПФ, AES и LU-разложения для реконфигурируемого компьютера «Tertius-2». В результате значительного сокращения временных затрат, обусловленных числом итераций оптимизации вычислительных структур, общее время синтеза было сокращено до трех раз.

Реконфигурируемая вычислительная система; ПЛИС; топологические ограничения вычислительных структур; синтез параллельно-конвейерных программ.

A.A. Dichenko, I.I. Levin, D.A. Sorokin

METHOD FOR GENERATING TOPOLOGICAL CONSTRAINTS OF COMPUTATIONAL STRUCTURES FOR RECONFIGURABLE COMPUTING SYSTEMS

For reconfigurable computing systems based on FPGAs, efficient application programs are parallel-pipeline programs that achieve real performance exceeding 50% of the peak. This article addresses the problem of reducing the development time of such programs. The computational structures of these programs utilize a large volume of FPGA resources operating at high clock frequencies. However, simultaneously maximizing both the amount of FPGA resources used and the clock frequency presents a certain contradiction: as resource utilization increases, the placement flexibility of the functional units of the computational structures decreases, and the FPGA switching matrix fails to provide the required signal propagation characteristics when routing information channels between them. Moreover, in modern CAD tools, placement and routing algorithms consider only the architectural and geometric features of the FPGA. Therefore, when a large number of specialized primitives with very limited placement flexibility are used, achieving high clock frequencies in automatic synthesis mode becomes virtually impossible. To address this problem, it is also necessary to consider the information dependencies between the functional units of the computational structures, but the nature of these dependencies in tasks from different subject areas can vary significantly. As a result, developers are often forced to manually place the functional units

of the computational structures on the FPGA by creating script-based instructions for topological constraints. In earlier generations of FPGAs, the time required to generate topological constraints was acceptable, as they typically contained only a few hundred specialized primitives. However, in modern FPGAs, the number of such primitives reaches several thousand or even tens of thousands, significantly increasing the development time of efficient application programs. The proposed method makes it possible to automate the process of developing topological constraints for computational structures. The research was carried out during the development of application programs for solving a range of problems based on FFT, AES, and LU decomposition algorithms for the reconfigurable computer "Tertius-2." As a result of significantly reducing the time required for optimization iterations of computational structures, the total synthesis time was reduced by up to three times.

Reconfigurable computing system; FPGA; topological constraints of computational structures; synthesis of parallel-pipeline programs.

Введение. В настоящее время реконфигурируемые вычислительные системы (РВС) [1] на базе программируемых логических интегральных схем (ПЛИС) успешно применяются для решения задач математической физики, криптографии, машинного обучения, цифровой обработки сигналов и многих других. Благодаря адаптации вычислительной структуры к информационному графу решаемой задачи [2], РВС демонстрируют высокую реальную производительность (более 50% от пиковой). В то же время реальная производительность традиционных высокопроизводительных систем на базе центральных и графических процессоров обычно не превышает 10-15% [3]. Такая эффективность РВС, как правило, обеспечивается за счёт работы на высоких по меркам ПЛИС тактовых частотах (выше 300 МГц) при использовании не менее 60% вычислительного ресурса (число конфигурируемых логических блоков CLB, блоков памяти BRAM или URAM, арифметических блоков DSP).

Однако время разработки прикладных программ для современных РВС, обеспечивающих высокую реальную производительность, даже высококвалифицированными специалистами составляет, как правило, от нескольких месяцев до года (иногда и больше). Это обусловлено не только применением низкоуровневых языков описания цифровых схем, таких как VHDL или Verilog, но и необходимостью синтеза вычислительной структуры параллельно-конвейерной программы, оптимальной по композиции двух параметров: объёму занимаемого вычислительного ресурса (заполнению) и тактовой частоте. При этом стандартные средства САПР [4, 5] автоматически не обеспечивают требуемых тактовых частот в вычислительных структурах с высоким заполнением ПЛИС.

В работе [6] на примере реализации задачи нагрузочного тестирования потребляемой мощности РВС описан типовой процесс, применяемый при разработке высокопроизводительных параллельно-конвейерных программ для РВС. Показано, что при автоматическом синтезе в САПР Vivado вычислительной структуры, заполняющей ПЛИС по CLB примерно на 70%, по DSP – на 100%, реконфигурируемый компьютер (РК) «Tertius» будет функционировать на частоте 200 МГц. Если же ужесточить требования ко времени распространения сигнала в синтезируемой вычислительной структуре, то с помощью подбора оптимальной стратегии размещения и трассировки и наложения топологических ограничений на размещение примитивов DSP, возможно обеспечить функционирование РК «Tertius» на частоте 400 МГц, и, соответственно, поднять его производительность в два раза без дополнительных аппаратных затрат. Однако для максимизации производительности за счёт увеличения частоты до 500 МГц (необходимое условие для качественной оценки потребляемой мощности РВС задачей нагрузочного тестирования) кроме перечисленного выше, потребовалось существенно модернизировать вычислительную структуру, включая переработку функциональных узлов. Это привело к значительному увеличению времени разработки прикладной программы, что обусловлено современными методологиями программирования для ПЛИС [9–11], предполагающими только ручную оптимизацию размещения вычислительной структуры с помощью топологических ограничений.

Как показывают исследования, время разработки топологических ограничений может быть сопоставимо со временем разработки самой вычислительной структуры, а иногда может и превышать его, поскольку для оптимального размещения её функциональ-

ных узлов на поле ПЛИС необходимо учитывать не только свойства примитивов, архитектурные и геометрические особенности ПЛИС, но и функциональные зависимости между узлами, а также зависимости от управления потоками данных [7, 8].

Кроме того, большое значение играет количество используемых примитивов типа BRAM, URAM или DSP в ПЛИС, которое может составлять от нескольких сотен до нескольких тысяч штук. При этом с каждым новым поколением степень интеграции ПЛИС неуклонно растёт (рис. 1), поэтому при переходе на новую элементную базу время разработки имеет тенденцию к пропорциональному увеличению. В связи с этим, *разработка методов генерации топологических ограничений вычислительных структур, позволяющих сократить время синтеза параллельно-конвейерных программ для РВС, является особо актуальной научной задачей*.

Рис. 1. Рост степени интеграции на примере ПЛИС AMD – Xilinx

Проблематика достижения высокой реальной производительности при высоком заполнении ПЛИС. Реальную производительность РВС P_R можно оценить по следующей формуле:

$$P_R = N \cdot \nu \cdot S, \tag{1}$$

где N – число функциональных устройств, ν – тактовая частота, S – скважность подачи данных.

Ключевым параметром, определяющим реальную производительность РВС при высоком заполнении ПЛИС, является тактовая частота работы вычислительной структуры. Скважность подачи данных также влияет на производительность, но для параллельно-конвейерной обработки она, как правило, равна единице (другие варианты скважности в данной статье не рассматриваются).

Предельно достижимая величина частоты в РВС зависит от физических параметров ПЛИС. Одним из таких параметров является класс скорости (Speed Grade) [12] используемой ПЛИС, обусловленный технологией изготовления. Однако, наибольшее влияние на величину тактовой частоты оказывает схема размещения вычислительной структуры в ПЛИС, которая во многом определяется архитектурными особенностями микросхемы. Как показано на рис. 2, современные ПЛИС имеют различное расположение встроенных интерфейсов и контактов ввода/вывода, могут содержать встроенные процессоры, а также зоны сопряжения кристаллов для составных ПЛИС. При переходе через такие специализированные области или при их обходе трассы между примитивами значительно удлиняются, что приводит к увеличению времени распространения сигнала.

Например, области физических выводов могут располагаться в средней части кристалла, что приводит к снижению тактовых частот при пересечении данных областей каналами, соединяющими функциональные узлы (рис. 3). Также распространённым случаем является пересечение каналом встроенных интерфейсов, таких как Ethernet, PCI Express, JTAG и тому подобных [7]. Основная проблема в данных случаях заключается в большой длине информационного канала, пересекающего специализированную область ПЛИС. Его длина вносит существенную задержку во время распространения сигнала, особенно если

содержит несколько уровней вложенной логики. Как показано в работе [8], данная проблема решается путём размещения функциональных узлов в непосредственной близости от границ пересекаемой специализированной области и добавлением блоков с нулевой логической нагрузкой типа «триггер-триггер». Также важными являются геометрические особенности кристалла. Например, в системах на кристалле часто встречается L-образная форма поля ПЛИС (семейства Versal, Arria 10 SX на рис. 2). При таком исполнении топология размещения усложняется, так как повышается вероятность возникновения более длинных трасс между функциональными узлами вычислительной структуры.

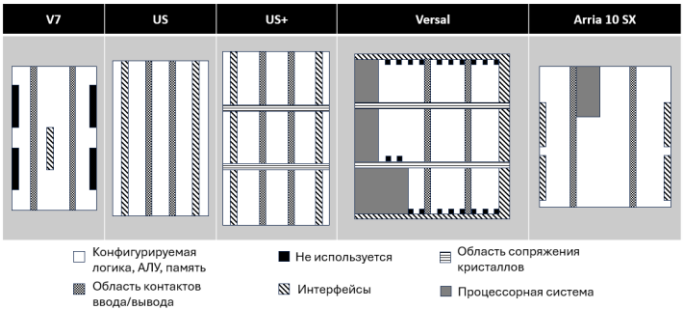

Рис. 2. Архитектура современных ПЛИС

Рис. 3. Пример пересечения связью области физических выводов ПЛИС

На величину тактовой частоты также влияет мощность коммутационного ресурса используемой ПЛИС (количество доступных физических каналов). На рис. 4 приведена фотография фрагмента коммутационной подсистемы ПЛИС XCVU37P.

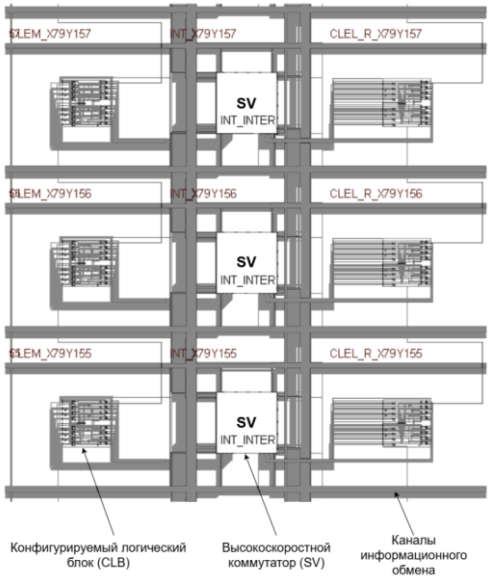

Рис. 4. Фрагмент коммутационной подсистемы ПЛИС XCVU37P

Коммутационный ресурс ПЛИС представляет собой совокупность горизонтальных и вертикальных каналов информационного обмена, объединённых коммутаторами (SV). Количество каналов между SV ограничено, поэтому если функциональные узлы используют большое количество каналов и находятся в разных топологически разнесённых друг от друга конфигурируемых вычислительных блоках (CLB), то при высоком заполнении ресурсов, информационные каналы прокладываются неоптимальным маршрутом, что зачастую приводит к увеличению времени распространения сигнала между примитивами и, соответственно, падению тактовой частоты. Данная проблема особенно усиливается при расположении CLB в разных регионах синхронизации тактового сигнала.

Следует отметить, что современным ПЛИС свойственна неравномерность распределения вычислительного ресурса. Как правило, для синтеза вычислительных структур сложных задач необходимо задействовать не только логический ресурс (CLB), но и специализированный ресурс DSP, BRAM и URAM. Проведённые исследования показывают: при использовании в ПЛИС более 60% DSP и BRAM (URAM) без предварительного размещения таких примитивов САПР в автоматическом режиме обеспечивает реальную рабочую тактовую частоту ниже достижимой в два и более раз. Это обусловлено тем, что специализированного вычислительно ресурса в сотни раз меньше, чем логического, и распределён такой ресурс по кристаллу неравномерно (рис. 5).

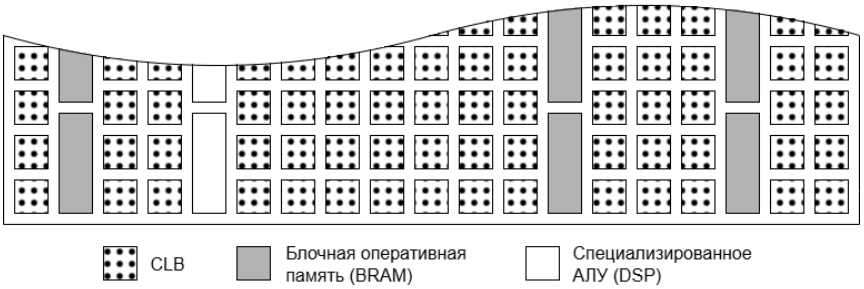

Рис. 5. Пример неравномерного распределения специализированного ресурса в ПЛИС XCVU095

При высоком заполнении ПЛИС и высоких требованиях к тактовой частоте вариативность размещения специализированных примитивов DSP, BRAM и URAM в автоматическом режиме САПР становится крайне низкой. Поэтому методология производителей САПР предполагает ручное размещение примитивов вычислительной структуры с помощью инструментов локализации в определённых областях ПЛИС [9, 11]. Стоит отметить, что процесс оптимизации топологического размещения вычислительной структуры достаточно длительный и трудоёмкий, поскольку необходимо учитывать не только местоположение входных и выходных интерфейсов вычислительной структуры, местоположение встроенных периферийных устройств, схему распределения вычислительного ресурса на ПЛИС, но и функциональные зависимости между фрагментами вычислительной структуры.

При высоком заполнении и тактовой частоте размещение вычислительных структур разработчиком вручную было возможным для поколений ПЛИС Xilinx Series-6 [14, 15] и более ранних (Series-5, Series-4, Stratix III и т.п.), поскольку объем их специализированного вычислительного ресурса не превышал нескольких сотен примитивов. Однако для более современных поколений ПЛИС (UltraScale, UltraScale+, Arria 10, Agilex и т.п.) количество DSP, BRAM, URAM исчисляется тысячами и десятками тысяч штук [16–18] (табл. 1).

В таких микросхемах ручное формирование схем размещения вычислительных структур требует существенно больше времени, поскольку для каждого используемого специализированного примитива формируются топологические ограничения в виде скриптовых инструкций САПР. Например, если для ПЛИС поколения Series-6 разработчику при-

ходилось «расставлять» от 600 до 4000 примитивов, то для ПЛИС поколения UltraScale+ этот объём составляет от 2000 до 15000. Кроме того, время формирования топологических ограничений может ещё увеличиться, потому что современные САПР не имеют инструментов автоматической проверки. Так, в процессе верификации схемы размещения в САПР необходимо выполнить размещение (Place Design) примитивов синтезируемой структуры на поле ПЛИС и проанализировать отчёты САПР на корректность применения топологических ограничений. В случае наличия ошибок файл топологических ограничений исправляется, и процесс верификации схемы размещения запускается заново. Стоит отметить, что для больших ПЛИС описание топологических ограничений при высоком заполнении может содержать десятки тысяч строк кода, что значительно увеличивает количество ошибок, связанных с человеческим фактором (опечатки, пропуски и т.п.).

Таблица 1

Объем специализированного вычислительного ресурса для различных поколений ПЛИС

Тип примитивов	Поколение ПЛИС				
	Virtex-6	**Virtex-7**	**Virtex UltraScale**	**Virtex UltraScale+**	**Versal Premium**
DSP, шт.	288-2016	1120-3600	600-2880	1320-12288	1140-14352
BRAM, шт.	312-2128	1500-3760	1260-2520	720-2016	535-6808
URAM, шт.	—	—	—	320-960	345-2549

После размещения выполняется этап трассировки (Route Design) связей между примитивами синтезируемой структуры, а затем для каждой связи производится расчёт времени распространения сигнала между примитивами (Timing Analysis). При наличии временных задержек схема размещения может быть модифицирована некоторое количество раз. Если ни одна из рассмотренных схем размещения не удовлетворяет по целевым параметрам, то дорабатывается или полностью перерабатывается функциональная схема вычислительной структуры с повторением процесса формирования схемы размещения и верификации. Если ни одна из рассмотренных модификаций не приводит к положительному результату, тогда необходимо снижать требования к заполнению и (или) тактовой частоте. Таким образом, время разработки параллельно-конвейерной программы зависит не только от объективных причин, но и субъективных.

Время t_{prog} разработки эффективной параллельно-конвейерной программы можно оценить по формуле

$$t_{prog} = t_{func} + t_{synth}, \quad (2)$$

где t_{func} – время разработки функциональной схемы, t_{synth} – время синтеза параллельно-конвейерной программы.

Время t_{func} может достигать нескольких месяцев в зависимости от квалификации разработчика, сложности реализуемого алгоритма и многих других факторов. Для простоты рассуждений будем считать, что функциональная схема разрабатывается один раз и в процессе синтеза вычислительной структуры не меняется (t_{func} – константа).

Время t_{synth} можно оценить по формуле

$$t_{synth} = k_1 \left(t_{sch} + k_2 \cdot t_{place}(v, U) + t_{route}(v, U) \right), \quad (3)$$

где t_{sch} – время разработки схемы размещения, t_{place} – время этапа компиляции Place Design, t_{route} – время этапа компиляции Route Design, U – заполнение, k_1 – количество итераций получения оптимальной схемы размещения, k_2 – количество итераций исправления технических ошибок.

Время t_{place} в большей степени зависит от заполнения микросхемы. Например, для ПЛИС XCKU095 t_{place} может быть от 0,5 до 4 часов. Время t_{route} зависит от тактовой частоты и заполнения. При высоких значениях v (более 300 МГц) и U (более 50%) t_{route} может быть от 2 до 36 часов. Более того, для современных ПЛИС, например, таких как

XCVU37P и более ресурсоёмких, время синтеза даже при автоматическом размещении средствами САПР (t_{sch}=0, k_1=1, k_2=1) может достигать 2-3 суток и при этом не гарантируется получение требуемых характеристик вычислительной структуры.

Таким образом, верификация схемы размещения и получение работоспособных параллельно-конвейерных программ, реализуется итерационным способом. В ходе верификации может многократно выполняться частичная либо полная компиляция проекта. При больших значениях коэффициентов k_1 и k_2 время синтеза параллельно-конвейерной программы может быть неприемлемым. Поэтому для сокращения времени поиска оптимальной схемы размещения вычислительной структуры и разработки топологических ограничений, обеспечивающих достижение требуемой реальной производительности, важно автоматизировать данные процессы, особенно для вычислительных структур с высоким заполнением ПЛИС и высокой тактовой частотой.

Традиционные алгоритмы размещения электронных компонентов, применяемые в современных САПР, выбирают в качестве критерия оптимизации размещения только топологические и геометрические свойства микросхем. При этом целевая функция может зависеть от длин связей, времени распространения сигнала, занимаемой площади вычислительной структурой и расчётного уровня энергопотребления [19, 20].

Предлагаемый метод генерации топологических ограничений для размещения вычислительных структур на ПЛИС учитывает не только геометрические и архитектурные особенности, но и специфику функциональной зависимости между устройствами (узлами) вычислительной структуры и зависимости от управления потоками данных для решаемых задач различных предметных областей. Это обуславливает в рамках предметной области использование типовых схем размещения. Например, для задач цифровой обработки сигналов характерны линейные и кольцевые схемы расстановки функциональных узлов (рис. 6,а), для задач линейной алгебры – линейные и мультилинейные (рис. 6,б), для криптографических задач и искусственных нейронных сетей – мультилинейные, кольцевые и мозаичные (рис. 6,в,г).

Поэтому для автоматизации процесса размещения необходимо использовать эвристические правила свойственные конкретным классам и подклассам решаемых задач.

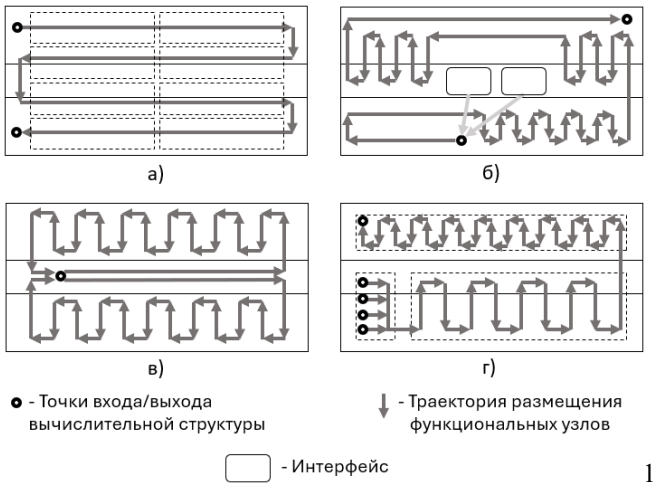

Рис. 6. Примеры схем размещения функциональных узлов некоторых задач

Схема размещения вычислительной структуры на ПЛИС зависит от алгоритма решения задачи, объёма и состава примитивов, связности между примитивами, мощности коммутационной матрицы ПЛИС, обеспечивающей прокладку информационных каналов между примитивами, требуемой производительности, тактовой частоты, требуемой энергоэффективности. Разработка оптимальной схемы размещения сводится к перебору различных вариантов размещения по композиции данных критериев.

Прежде чем приступать к разработке схемы размещения, необходимо задать базовые целевые характеристики, такие как производительность и время синтеза. При условии фиксированной тактовой частоты производительность определяется коэффициентом распараллеливания вычислений, а при условии фиксированного коэффициента распараллеливания – величиной тактовой частоты. Поэтому добиваться целевой производительности можно балансируя одним параметром относительно другого. Для достижения целевых показателей в рамках метода предлагается использовать три стратегии разработки схемы размещения: стратегию жесткого, полугибкого, гибкого размещения.

Стратегия жёсткого размещения может быть применена, когда необходимо получить быстрое, но не самое производительное решение. При жестком размещении геометрия и внутренняя топология масштабируемых функциональных узлов не меняются. Например, такая стратегия может быть использована в локальных участках ПЛИС, когда коммутационный ресурс не позволяет плотно разместить функциональные узлы с большим количеством связей. Время разработки прикладной программы при выборе данной стратегии будет минимальным.

В отличие от предыдущей стратегии, стратегия полугибкого размещения позволяет более эффективно использовать ресурс ПЛИС за счёт ортогональных вращений функциональных узлов в пространстве примитивов ПЛИС и, соответственно, более плотного их размещения. Данная стратегия предполагает баланс между длительностью разработки и производительностью получаемого решения.

Стратегия гибкого размещения, в отличие от предыдущих, позволяет максимально задействовать ресурс ПЛИС для плотного размещения функциональных узлов и является наиболее затратной по времени. При такой стратегии размещения функциональные узлы могут подвергаться ортогональным вращениям, изменять свою геометрию и разделяться примитивами других типов или занятыми ячейками ПЛИС. Однако в ряде случаев применение данной стратегии может приводить к проблемам при трассировке, при высокой связности размещаемых функциональных узлов.

В конечном итоге разработчик решает, какую стратегию использовать для достижения конкретных целей.

Процедура размещения вычислительных структур на ПЛИС. На основе предложенного метода была разработана процедура размещения вычислительных структур на ПЛИС. Данную процедуру с применением указанных стратегий можно выполнять следующим образом:

1. Разработать схему размещения масштабируемого функционального узла.

Так как функциональный узел может занимать в общем случае произвольную область ПЛИС, то при разработке схемы его размещения учитывается совокупность тех же факторов, что и при разработке схемы размещения всей вычислительной структуры.

1.1. Для разработки схемы размещения функционального узла (вычислительной структуры) необходимо проанализировать архитектуру целевой ПЛИС.

1.2. Определить достаточную область для размещения функционального узла.

Необходимо проанализировать совокупность всех примитивов функционального узла на наличие прямых функциональных зависимостей между логическими примитивами и между специализированными примитивами, поскольку вывод вычислительной структуры на требуемые характеристики по производительности и тактовой частоте напрямую зависит от пространственной ориентации примитивов друг относительно друга.

1.3. Сформировать список путей к примитивам (иерархических имен примитивов) функционального узла.

1.4. Проанализировать выбранную область размещения на пересечение специальными областями ПЛИС (областями сопряжения кремниевых пластин, областями физических выводов, областями встроенных интерфейсов).

При наличии таких пересечений необходимо разместить составные части функционального узла таким образом, чтобы базовые арифметико-логические операции были неразрывными.

2. Разместить примитивы функционального узла в выбранной области путём создания набора топологических ограничений.

3. Выполнить компиляцию проекта, проверить его на работоспособность и достижение требуемых характеристик.

В случае выявления ошибки выполнить соответствующий анализ, внести корректировки и повторить цикл компиляции и проверки проекта.

4. Разработать схему размещения вычислительной структуры.

5. Определить границы области размещения вычислительной структуры.

6. Задать условия размещения в соответствии с выбранной стратегией размещения.

6.1. Если выбрана стратегия жёсткого размещения, установить запрет на ортогональные вращения функционального узла в пространстве примитивов ПЛИС, установить запрет на разделение функционального узла занятыми примитивами или примитивами других типов, установить запрет на изменение топологии функционального узла.

6.2. Если выбрана стратегия полугибкого размещения, установить разрешение на ортогональные вращения функционального узла в пространстве примитивов ПЛИС, установить запрет на разделение функционального узла занятыми примитивами или примитивами других типов, установить запрет на изменение топологии функционального узла.

6.3. Если выбрана стратегия гибкого размещения, установить разрешение на ортогональные вращения функционального узла в пространстве примитивов ПЛИС, установить разрешение на разделение функционального узла занятыми примитивами или примитивами других типов, установить разрешение на изменение топологии функционального узла.

7. Определить координаты оптимальных положений функциональных узлов.

Если ячейки, соответствующие полученным координатам, находятся в границах области размещения, не заняты и соответствуют типам примитивов исходного функционального узла, то пометить их занятыми, создать функциональный узел с полученными координатами и перейти к следующему функциональному узлу, в противном случае пересчитать координаты. Если все функциональные узлы размещены, зафиксировать топологические ограничения и перейти к следующему шагу.

8. Сохранить топологические ограничения в файле топологических ограничений.

9. Выполнить компиляцию проекта.

При успешной компиляции перейти к следующему шагу, в противном случае выполнить соответствующий анализ, внести корректировки и повторить цикл компиляции проекта.

10. Проверить проект на работоспособность и достижение целевых характеристик путём анализа отчётов САПР.

Если требуемые характеристики не достигнуты или выявлены ошибки, выполнить соответствующий анализ, внести корректировки и повторить цикл компиляции и проверки проекта.

Пример результата использования стратегии жёсткого размещения при синтезе фрагмента вычислительной структуры показан на рис. 7,а. Данный фрагмент вычислительной структуры строится на основе функционального узла, содержащего три примитива BRAM. Для простоты рассуждений ресурс CLB не показан.

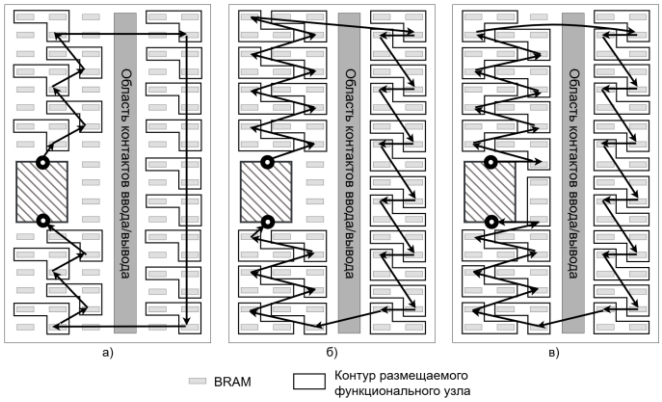

Рис. 7. Примеры результатов применения стратегий жёсткого размещения (а), полугибкого размещения (б) и гибкого размещения (в)

Производительность P при фиксированной тактовой частоте фрагмента вычислительной структуры, приведённого на рис. 7,а можно оценить по формуле

$$P = p \cdot N, \quad (4)$$

где p – производительность одного функционального узла, N – количество функциональных узлов.

Для данного примера $N = 18$, поэтому в соответствии с формулой (4) $P = 18p$.

Для анализа эффективности полученного решения необходимо оценить удельную производительность P_s по формуле

$$P_s = P/R_{all}, \quad (5)$$

где R_{all} – всего ресурса в области размещения.

В данном примере $R_{all} = 82$ ячейки, следовательно $P_s \approx 0{,}22\ p$.

Время размещения вычислительной структуры t можно оценить по формуле

$$t = \varphi \cdot N \cdot \tau, \quad (6)$$

где φ – количество вариантов размещения функционального узла, N – количество функциональных узлов, τ – время размещения одного функционального узла.

Для данного примера $\varphi = 1$, в соответствии с формулой (6) $t = 18\ \tau$.

Применение стратегии жёсткого размещения позволяет получать вычислительные структуры для различных задач с уровнем производительности не ниже 50% от пиковой за время, сопоставимое со временем синтеза, затрачиваемым САПР в автоматическом режиме. Данную стратегию целесообразно применять для оценки эффективности РВС при решении конкретной задачи предметной области либо в случае оперативного изменения алгоритма решения задачи. Однако стратегия жёсткого размещения не направлена на получение высокопроизводительных решений. Для того чтобы получать более производительные вычислительные структуры, следует применять стратегии полугибкого и гибкого размещения.

Пример результата использования стратегии полугибкого размещения при синтезе фрагмента вычислительной структуры показан на рис. 7,б.

Для данного примера $N = 26$, поэтому в соответствии с формулами (4) и (5) $P = 26p$, а $P_s \approx 0{,}317p$.

Стратегия полугибкого размещения предусматривает четыре варианта ориентации функционального узла в пространстве примитивов, следовательно $\varphi = 4$. Таким образом, в соответствии с формулой (6) для данного примера t может достигать 104τ.

Применение стратегии полугибкого размещения в большинстве случаев позволяет получать вычислительные структуры для эффективного решения большинства задач различных предметных областей с достаточным уровнем производительности.

Пример результата использования стратегии гибкого размещения при синтезе фрагмента вычислительной структуры показан на рис. 7,в.

Для данного примера $N = 27$, соответственно $P = 27p$, а $P_s \approx 0{,}329p$.

Для стратегии гибкого размещения φ можно оценить следующим образом:

$$\varphi = \frac{n!}{(n-m)!}, \quad (7)$$

где n – количество ячеек в границах размещения функционального узла, m – количество примитивов в функциональном узле.

Для данного примера $n = 4$, а $m = 3$, тогда в соответствии с формулами (6) и (7) t может достигать 648τ.

Стратегию гибкого размещения следует применять, когда необходимо получить максимально возможную производительность для данного класса (подкласса) задач. Однако недостатком данной стратегии является длительное время поиска оптимальной схемы размещения вычислительной структуры. Как было показано ранее, это связано с перебором различных вариантов размещения по композиции множества критериев.

Предложенный метод был испытан при синтезе вычислительных структур нескольких задач для реконфигурируемого компьютера «Tertius-2» производства «НИЦ СЭ и НК» (рис. 8) [21], содержащего восемь ПЛИС XCKU095.

Рис. 8. Реконфигурируемый компьютер «Tertius-2»

При построении вычислительных структур каждой из задач были задействованы все восемь ПЛИС реконфигурируемого компьютера, в каждой из которых были реализованы примерно равные по функциональной сложности и производительности фрагменты вычислительных структур.

В табл. 2 и 3 приводятся характеристики по удельной производительности и времени синтеза типового фрагмента вычислительной структуры каждой задачи для одной ПЛИС. Исследования эффективности предлагаемого метода проводились при условии, что разработку проекта ведёт один специалист на одном компьютере с одним центральным процессором без распараллеливания процессов проектирования. Естественно, проектом может заниматься коллектив разработчиков и компиляции различных вариантов проекта могут выполняться на нескольких процессорах, что ускорит разработку параллельно-конвейерных конвейерных программ для РВС. Однако предлагаемый метод способствует достижению качественного снижения временных затрат, особенно при создании прикладных программ для сложносоставных вычислительно трудоёмких задач.

В табл. 2 приведены достигаемые значения удельной производительности ПЛИС «Tertius-2» для задач на основе алгоритмов БПФ, AES и LU-разложения при автоматическом размещении вычислительных структур САПР Vivado и размещении с применением предложенного метода.

Удельная производительность ПЛИС рассчитывалась по формуле (5). Число системных логических ячеек (R_{all}) для ПЛИС XCKU095 составляет 1176000 SLC [17].

Таблица 2

Удельная производительность

Задача	САПР Vivado	Жёсткое размещение	Полугибкое размещение	Гибкое размещение
БПФ, kOps/SLC	275,77	451,6	569,71	634,28
AES, kVarps/SLC	4,49	7,43	9,44	10
LU, kFlops/SLC	14,45	39,85	50,08	55,27

Как показано в табл. 2, наибольшая удельная производительность достигается при использовании стратегий полугибкого и гибкого размещения предложенного метода. В тоже время удельная производительность, достигаемая стандартными автоматическими средствами размещения САПР без применения топологических ограничений, в лучшем случае меньше возможной на 55%. Безусловно, время синтеза в этом случае будет минимальным, но в контексте максимизации производительности РВС при решении задач, такие параллельно-конвейерные программы не актуальны.

Проведенные исследования показали, что время синтеза параллельно-конвейерной программы пропорционально зависит от целевой производительности, особенно при ручном формировании схемы размещения, что обусловлено наличием ошибок, связан-

ных с человеческим фактором. В табл. 3 приведены показатели времени синтеза (в долях суток) параллельно-конвейерных программ для рассматриваемых задач при размещении вычислительных структур на ПЛИС с применением предложенной процедуры в ручном (Р) режиме формирования топологических ограничений и автоматизированном (А).

Таблица 3

Время синтеза параллельно-конвейерных программ

Задача	Жёсткое размещение		Полугибкое размещение		Гибкое размещение	
	Р	А	Р	А	Р	А
БПФ	3,39	1,25	13,56	5,1	27,12	10,2
AES	3,02	1,23	12,08	4,92	24,16	9,84
LU	4,05	1,45	16,2	5,8	32,4	11,6

Предложенная процедура размещения вычислительных структур на ПЛИС позволяет сократить время разработки эффективных прикладных программ для РВС в 2-3 раза. Кроме того, применение различных стратегий размещения в разработанной процедуре позволяет синтезировать в САПР параллельно-конвейерные программы, при исполнении которых реальная производительность РВС в 1,65 и более раз выше, чем при исполнении программ, синтезированных в автоматическом режиме компиляции САПР.

Заключение. Результаты исследований показали, что применение в рамках предложенной процедуры эвристических правил формирования схем размещения, свойственных классам решаемых задач, а также автоматизация процесса формирования топологических ограничений позволила значительно сократить время синтеза высокопроизводительных параллельно-конвейерных программ. Следует отметить, что в дальнейшем возможно ещё больше сократить время синтеза, если автоматизировать сам процесс разработки схем размещения.

В настоящее время ведётся разработка программы препроцессора «ТС-Creator», предназначенного для генерации в автоматизированном режиме файлов топологических ограничений. Применение данного препроцессора позволяет полностью исключить влияние человеческого фактора. Также развиваются эвристические алгоритмы разработки схем размещения вычислительных структур на поле ПЛИС, дальнейшее внедрение которых в «ТС-Creator» позволит еще больше повысить автоматизацию синтеза вычислительных структур и сократить время разработки высокопроизводительных параллельно-конвейерных программ для РВС.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Каляев И.А., Дордопуло А.И., Левин И.И., Федоров А.М.* Развитие отечественных многокристальных реконфигурируемых вычислительных систем: от воздушного к жидкостному охлаждению // Тр. СПИИРАН. – СПб.: Изд-во СПИИРАН ФГБУН Санкт-Петербургский институт информатики и автоматизации РАН. – 2017. – № 1 (50). – С. 5-31. – DOI: 10.15622/sp.50.1.
2. *Каляев А.В., Левин И.И.* Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.
3. *Каляев И.А., Левин И.И.* Реконфигурируемые вычислительные системы на основе ПЛИС. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2022. – 506 с. – ISBN 978-5-4358-0232-0.
4. Vivado Overview. – URL: https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado.html (дата обращения: 08.11.2024).
5. FPGA Design Software – Quartus Prime. – URL: https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html (дата обращения: 08.11.2024).
6. *Алексеев К.Н., Сорокин Д.А., Леонтьев А.Л.* Методика создания топологических ограничений при высокой утилизации ресурсов ПЛИС // Известия ЮФУ. Технические науки. – 2022. – № 4. – С. 200-212.
7. *Диченко А.А., Сорокин Д.А., Левин И.И.* Принципы размещения вычислительных структур на ПЛИС реконфигурируемых вычислительных систем // Deutsche Internationale Zeitschrift für Zeitgenössische Wissenschaft. – 2024. – № 79. – С. 54-63. – DOI: 10.5281/zenodo.11127391. – EDN TBKWEJ.

8. *Алексеев К.Н., Сорокин Д.А.* Оптимизация вычислительных структур под архитектуру ПЛИС XILINX // FPGA-Systems Magazine: FSM: № ALFA (state_0). – 2023. – 166 с. – С. 75-82. – URL: https://fpga-systems.ru/fs_fsm/state_0/fsm_state_0.pdf (дата обращения: 17.09.2024).
9. Vivado Design Suite User Guide: Using Constraints (UG903). – URL: https://docs.amd.com/r/en-US/ug903-vivado-using-constraints (дата обращения: 20.11.2024).
10. Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906). – URL: https://docs.amd.com/r/en-US/ug906-vivado-design-analysis (дата обращения: 20.11.2024).
11. Intel Quartus Prime Standard Edition User Guide: Design Constraints. – URL: https://www.intel.com/content/www/us/en/docs/programmable/683492/18-1/constraining-designs.html (дата обращения: 21.11.2024).
12. Speed Grade. – URL: https://www.intel.com/content/www/us/en/docs/programmable/683703/17-1/speed-grade.html (дата обращения: 03.12.2024).
13. Xilinx WP380 Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency (WP380). – URL: https://docs.amd.com/v/u/en-US/wp380_Stacked_Silicon_Interconnect_Technology (дата обращения: 03.12.2024).
14. Virtex-6 Family Overview (DS150). – URL: https://docs.amd.com/v/u/en-US/ds150 (дата обращения: 10.12.2024).
15. *Каляев И.А., Левин И.И., Семерников Е.А., Дордопуло А.И.* Реконфигурируемые вычислительные системы на основе ПЛИС семейства Virtex-6 // Параллельные вычислительные технологии (ПаВТ'2011): Тр. международной научной конференции, Москва, 28 марта – 01 2011 года / отв. за выпуск: Л.Б. Соколинский, К.С. Пан. – М.: Издательский центр ЮУрГУ, 2011. – С. 203-210. – EDN TBLWND.
16. 7 Series FPGAs Data Sheet: Overview (DS180). – URL: https://docs.amd.com/v/u/en-US/ds180_7Series_Overview (дата обращения: 10.12.2024).
17. UltraScale Architecture and Product Data Sheet: Overview (DS890). – URL: https://docs.amd.com/v/u/en-US/ds890-ultrascale-overview (дата обращения: 10.12.2024).
18. Versal Architecture and Product Data Sheet: Overview (DS950). – URL: https://docs.amd.com/v/u/en-US/ds950-versal-overview (дата обращения: 11.12.2024).
19. Vivado Design Suite (WP416). – URL: https://docs.amd.com/v/u/en-US/wp416-Vivado-Design-Suite (дата обращения: 10.12.2024).
20. Defining Implementation Strategies • Vivado Design Suite User Guide: Implementa-tion (UG904). – URL: https://docs.amd.com/r/en-US/ug904-vivado-implementation/Defining-Implementation-Strategies (дата обращения: 10.12.2024).
21. Терциус-2 | НИЦ супер-ЭВМ и нейрокомпьютеров. – URL: https://superevm.ru/index.php?page=tertsius-2 (дата обращения: 14.01.2025).

REFERENCES

1. *Kalyaev I.A., Dordopulo A.I., Levin I.I., Fedorov A.M.* Razvitie otechestvennykh mnogokristal'nykh rekonfiguriruemykh vychislitel'nykh sistem: ot vozdushnogo k zhidkostnomu okhlazhdeniyu [Development of domestic multi-crystal reconfigurable computing systems: from air to liquid cooling], *Tr. SPIIRAN* [Proceedings of SPIIRAS]. St. Petersburg: Izd-vo SPIIRAN FGBUN Sankt-Peterburgskiy institut informatiki i avtomatizatsii RAN, 2017, No. 1 (50), pp. 5-31. DOI: 10.15622/sp.50.1.
2. *Kalyaev A.V., Levin I.I.* Modul'no-narashchivaemye mnogoprotsessornye sistemy so strukturno-protsedurnoy organizatsiey vychisleniy [Modularly scalable multiprocessor systems with structural-procedural organization of computations]. Moscow: Yanus-K, 2003, 380 p.
3. *Kalyaev I.A., Levin I.I.* Rekonfiguriruemye vychislitel'nye sistemy na osnove PLIS [Reconfigurable computing systems based on FPGAs]. Rostov-on-Donu: Izd-vo YuNTS RAN, 2022, 506 p. ISBN 978-5-4358-0232-0.
4. Vivado Overview. Available at: https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado.html (accessed 08 November 2024).
5. FPGA Design Software – Quartus Prime. Available at: https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html (accessed 08 November 2024).
6. *Alekseev K.N., Sorokin D.A., Leont'ev A.L.* Metodika sozdaniya topologicheskikh ogranicheniy pri vysokoy utilizatsii resursov PLIS [Methodology for creating topological constraints with high utilization of FPGA resources], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2022, No. 4, pp. 200-212.
7. *Dichenko A.A., Sorokin D.A., Levin I.I.* Printsipy razmeshcheniya vychislitel'nykh struktur na PLIS rekonfiguriruemykh vychislitel'nykh sistem [Principles of placing computing structures on FPGAs of reconfigurable computing systems], Deutsche Internationale Zeitschrift für Zeitgenössische Wissenschaft, 2024, No. 79, pp. 54-63. DOI: 10.5281/zenodo.11127391. EDN TBKWEJ.

8. *Alekseev K.N., Sorokin D.A.* Optimizatsiya vychislitel'nykh struktur pod arkhitekturu PLIS XILINX [Optimization of computing structures for the XILINX FPGA architecture], *FPGA-Systems Magazine: FSM: № ALFA (state_0)*, 2023, 166 p., pp. 75-82. Available at: https://fpga-systems.ru/fs_fsm/state_0/fsm_state_0.pdf (accessed 17 September 2024).
9. Vivado Design Suite User Guide: Using Constraints (UG903). Available at: https://docs.amd.com/r/en-US/ug903-vivado-using-constraints (accessed 20 November 2024).
10. Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906). Available at: https://docs.amd.com/r/en-US/ug906-vivado-design-analysis (accessed 20 November 2024).
11. Intel Quartus Prime Standard Edition User Guide: Design Constraints. Available at: https://www.intel.com/content/www/us/en/docs/programmable/683492/18-1/constraining-designs.html (accessed 21 November 2024).
12. Speed Grade. Available at: https://www.intel.com/content/www/us/en/docs/programmable/683703/17-1/speed-grade.html (accessed 03 December 2024).
13. Xilinx WP380 Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency (WP380). Available at: https://docs.amd.com/v/u/en-US/wp380_Stacked_Silicon_Interconnect_Technology (accessed 03 December 2024).
14. Virtex-6 Family Overview (DS150). Available at: https://docs.amd.com/v/u/en-US/ds150 (accessed 10 December 2024).
15. *Kalyaev I.A., Levin I.I., Semernikov E.A., Dordopulo A.I.* Rekonfiguriruemye vychislitel'nye sistemy na osnove PLIS semeystva Virtex-6 [Reconfigurable computing systems based on FPGAs of the Virtex-6 family], *Parallel'nye vychislitel'nye tekhnologii (PaVT'2011): Tr. mezhdunarodnoy nauchnoy konferentsii, Moskva, 28 marta – 01 2011 goda* [Proceedings of the international scientific conference, Moscow, March 28 – January 2011], Responsible for the release: L.B. Sokolinskiy, K.S. Pan. Moscow: Izdatel'skiy tsentr YuUrGU, 2011, pp. 203-210. EDN TBLWND.
16. 7 Series FPGAs Data Sheet: Overview (DS180). Available at: https://docs.amd.com/v/u/en-US/ds180_7Series_Overview (accessed 10 December 2024).
17. UltraScale Architecture and Product Data Sheet: Overview (DS890). Available at: https://docs.amd.com/v/u/en-US/ds890-ultrascale-overview (accessed 10 December 2024).
18. Versal Architecture and Product Data Sheet: Overview (DS950). Available at: https://docs.amd.com/v/u/en-US/ds950-versal-overview (accessed 11 December 2024).
19. Vivado Design Suite (WP416). Available at: https://docs.amd.com/v/u/en-US/wp416-Vivado-Design-Suite (accessed 10 December 2024).
20. Defining Implementation Strategies • Vivado Design Suite User Guide: Implementa-tion (UG904). Available at: https://docs.amd.com/r/en-US/ug904-vivado-implementation/Defining-Implementation-Strategies (accessed 10 December 2024).
21. Tertsius-2 | NITS super-EVM i neyrokomp'yuterov [Tertsius-2 | Research Center of Supercomputers and Neurocomputers]. Available at: https://superevm.ru/index.php?page=tertsius-2 (accessed 14 January 2025).

Диченко Алексей Андреевич – Южный федеральный университет; e-mail: dichenko@sfedu.ru; г. Таганрог, Россия; тел.: +79525724918; кафедра интеллектуальных и многопроцессорных систем; аспирант.

Левин Илья Израилевич – Южный федеральный университет; e-mail: iilevin@sfedu.ru; г. Таганрог, Россия; тел.: +78634361608; зав. кафедрой интеллектуальных и многопроцессорных систем; д.т.н.; профессор.

Сорокин Дмитрий Анатольевич – ООО «НИЦ СЭ и НК»; e-mail: jotun@inbox.ru; г. Таганрог, Россия; тел.: +79508668253; начальник отдела прикладного программного обеспечения; к.т.н.

Dichenko Aleksey Andreevich – Southern Federal University; e-mail: dichenko@sfedu.ru; Taganrog, Russia; phone: +79525724918; the Department of Intelligent and Multiprocessor Systems; postgraduate student.

Levin Ilya Izrailevich – Southern Federal University; e-mail: levin@superevm.ru; Taganrog, Russia; phone: +78634361608; head of department; dr. of eng. sc.; professor.

Sorokin Dmitry Anatolyevich – Supercomputers and Neurocomputers Research Center; e-mail: jotun@inbox.ru; Taganrog, Russia; phone: +79508668253; chief of department; cand. of eng. sc.