УДК 681.32

СФЕРИЧЕСКИЙ МУЛЬТИПРОЦЕССОР PRUS ДЛЯ РЕШЕНИЯ БУЛЕВЫХ УРАВНЕНИЙ

ГАЙДУК С., ХАХАНОВ В.И., ОБРИЗАН В.И., КАМЕНЮКА Е.А.

Предлагается высокопроизводительная технология решения булевых уравнений, основанная на использовании сферического однобитного мультипроцессора PRUS (Programmable Unlimited Systems), реализуемого на кристалле ASIC. Он позволяет осуществлять параллельную, последовательную и конвейерную обработку булевых уравнений, записанных в базисе операций AND, OR, NOT, XOR. Мультипроцессор экономичен в аппаратурном исполнении – для обработки системы уравнений, насчитывающей 20 миллионов вентилей, необходимо иметь всего 256 Мбайт оперативной памяти.

1. Состояние рынка микроэлектроники

Рынок современной микроэлектроники поделен на секторы, где достаточно жестко обозначены границы использования типов кристаллов для имплементации цифровых систем. Здесь лидируют заказные и стандартизованные микросхемы (ASIC, ASSP), общий уровень продаж которых составляет 78% от всего рынка производимых чипов (рис. 1) (EDA Industry survey. By Dr. Jack Horgan, http://www10.edacafe.com).

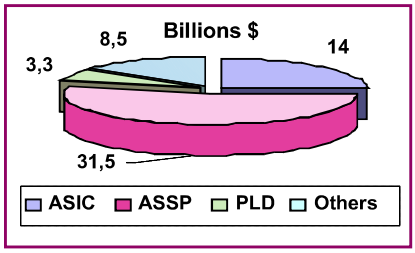

Рис. 1. Рынок типов кристаллов

Постепенно возрастает доля программируемых логических интегральных схем (ПЛИС) [1], которая сегодня составляет 6% от общего объема продаж или 3,3 млрд долларов. Этому способствуют определенные преимущества PLD (рис.2), связанные с перепрограммируемостью цифровых систем, реализуемых в ПЛИС, низкой стоимостью, незначительным временем проектирования и тестирования по сравнению с ASIC.

Characteristic	FPGA	ASIC
Время выхода на рынок	Short	Long
Минимальная партия	None	High
Цена единицы в партии	High	Low
Затраты на подложку	None	High
Реконфигурируемость	High	None
Производительность	Medium	Very High
Плотность вентилей	Medium	Very High
Энергопотребление	High	Low
Сложность разработки	Medium	Very High
Сложность тестирования	Low	High
Длительность цикла разработки	Hours	Months

Рис. 2. Сравнительный анализ кристаллов

Тем не менее имеются очевидные преимущества применения ASIC для имплементации цифровой системы, поскольку она имеет низкую стоимость и энергопотребление, самую высокую производительность. Естественно, что отслеживая далее существующие тенденции рынка микроэлектроники, производители все более склоняются к созданию цифровых систем, имплементируемых в кристаллы, сочетающие преимущества ASIC и PLD. Так появляются интересные интегральные решения [2-6] у ведущих фирм мира (рис. 3): {PLD + ASIC} – Lucent Technologies (OR3TP12), Quick Logic (RAM, PCI, DSP), Cyprus Semicomductor (PCI), Altera (Stratix), Xilinx (Virtex II). Также интегрируются вместе такие компоненты как {CPU + PLD} – Atmel (FPSLIC), Triscend (E5, A7), Xilinx (Virtex II Pro), Altera (Excalibur);{ CPU + ASIC}– микроконтроллеры; {CPU + PLD + ASIC}. Упомянутые сочетания типов электронных чипов позволяют получать кристаллы, имеющие быстродействие и энергопотребление от ASIC, перепрограммируемость и реконфигурируемость от PLD, гибкость и управляемость от CPU.

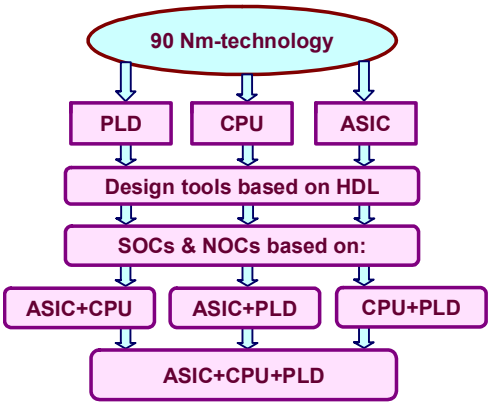

Рис. 3. Комбинированные микросхемы

Что касается внедрения или продвижения новых технологий на рынке электроники, то этот процесс можно определить как аппаратуризация (hardwaring). Изначально апробация любого метода или алгоритма осуществляется в программном исполнении. После достаточно полной верификации проект может быть выставлен на рынке в виде IP-core, что гарантирует авторские права разработчика. Следующим этапом может быть имплементация IP-core в кристалл программируемой логики, что дает возможность существенно повысить быстродействие, оставляя возможность перепрограммирования отдельных функций в процессе доработки и выявления ошибок. Исключительная важность технологического решения является критерием, чтобы оно было имплементировано в кристалл ASIC. В этом случае быстродействие реализованной системы становится максимальным, но здесь уже невозможно исправить ошибки, поэтому проект должен быть идеально корректным. Таким образом, аппаратуризация проекта (рис. 4) есть последовательность стадий его имплементации от Software к Hardware, который становится менее гибким для модификации, но более производительным и экономичным для пользователя.

2. Актуальность создания мультипроцессора

Чтобы решить систему булевых уравнений, содержащую порядка миллиона линий или эквивалентных вентилей, существует несколько практически ориентированных технологий.

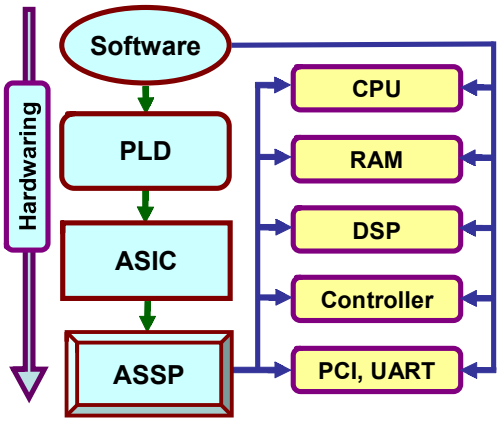

Рис. 4. Аппаратуризация: от software к hardware

1. Можно использовать персональный компьютер или рабочую станцию на основе микропроцессора фирмы Intel. Здесь каждое уравнение будет решаться программным путем и последовательно, поскольку существует только один процессор, хотя и достаточно мощный. Стоимость решения проблемы, а также временные затраты очень высоки.

2. Реализовать специализированный параллельный процессор на основе PLD. В данном случае высокий параллелизм обработки уравнений компенсирует сравнительно низкую (по сравнению с CPU) тактовую частоту. Такое схемотехническое решение [7], к тому же с возможностью репрограммирования, является по производительности абсолютно выигрышным. Но здесь существенным недостатком есть отсутствие гибкости, присущей программным методам решения уравнений. Кроме того, реализация системы на кристалле PLD имеет высокую стоимость, если объем будущих продаж составит десятки тысяч штук.

3. Третье решение связано с объединением достоинств CPU, PLD и ASIC [8], таких как:

– гибкость программирования системы уравнений, которая позволяет оперативно корректировать спецификацию в виде исходных кодов;

– минимально возможная мощность системы команд, приводящая к простым схемотехническим решениям аппаратурной реализации процессора;

– распараллеливание процесса решения уравнений, благодаря идеологии PLD, но с элементами CPU, что означает иметь много простых (однобитовых) взаимосвязанных между собой процессоров с собственной системой команд для параллельного программирования;

– имплементация мультипроцессора в кристалл ASIC, что позволит иметь максимально возможную тактовую частоту, минимальную стоимость одной микросхемы при большом объеме (более 10 000) выпуска продукции, низкое энергопотребление;

– конвейеризация решения системы уравнений есть исключительное свойство, присущее только мультипроцессорной системе, где конвейерная обработка данных есть один из основных режимов наряду с параллельным и последовательным способами.

Таким образом, необходимость появления мультипроцессора для решения булевых уравнений со свойствами, упомянутыми выше, обусловлена существующими тенденциями на рынке электронных технологий [9-13]. Что имеется здесь в виду? В ближайшем будущем актуальные и вычислительно сложные проблемы должны будут решаться специализированными вычислительными микро- и мультипроцессорами. На рынке микроэлектроники возрастает показатель специализированной ориентации на решение конкретных задач – customization. Когда возможности универсальных компьютеров начинают превышать потребности данной области рынка, конкуренция смещает акценты с повышения общей производительности на усовершенствование свойств, за которые пользователь готов платить дополнительные деньги. Это – не только надежность и удобство, но и быстродействие решения сложной проблемы, заказные свойства, энергосбережение, автономность, миниатюризация, гибкость и репрограммируемость.

3. Постановка задач проектирования мультипроцессора PRUS

PRUS (Programmable Unlimited Systems) есть ad hoc технология и специализированное вычислительное устройство, реализуемое в кристалле ASIC, для быстрого решения булевых уравнений. Архитектура PRUS представляет собой матрицу параллельных процессоров, каждый из которых связан с восемью другими линиями передачи данных. Структура PRUS и модель обработки уравнений разработаны доктором Stanley Hyduke [10] и носят название “Compiler Synchronized Parallel-processor Network-based Logic Device” – сеть параллельных компилятивно синхронизированных процессоров. PRUS-технология и логический процессор, имеющий порядка 90 вентилей, защищены патентами США, что подтверждает оригинальность архитектурных и процессных решений, ориентированных на эмуляцию системы булевых уравнений большой размерности.

Проектирование на основе PRUS подобно классической схеме (design flow) за исключением того, что стадия place and route заменяется фазой распределения булевых уравнений между сотнями (тысячами) логических процессоров, работающих параллельно. Boolean Equation Compiler (BECom) обеспечивает размещение уравнений по процессорам, задает время формирования решения на выходе каждого из них, а также планирует передачу полученных результатов другому процессору.

PRUS есть эффективная сеть процессоров, которая обрабатывает систему логических уравнений и обеспечивает обмен данными между компонентами сети в процессе их решения. Простая схемотехника каждого процессора, требующая 48 бит памяти для эмуляции двухвходового вентиля, позволяет обрабатывать сверхбольшие проекты, насчитывающие более 20 миллионов вентилей, затрачивая на это 256 Мбайт оперативной памяти. При этом ее затраты на описание триггера находятся в интервале 48 - 64 бит. Триггерные структуры могут располагаться в любой части комбинационной схемы, обеспечивая полную свободу проектирования.

Базовый вариант процессора PRUS может быть синтезирован на 90 вентилях, что дает возможность легко имплементировать, например, сеть, содержащую 4096 вычислителей, в ASIC, используя современную силиконовую технологию. Учитывая, что затраты памяти для эмуляции вентилей и триггеров весьма незначительны, PRUS может представлять интерес для проектирования систем управления в таких областях человеческой деятельности, как: индустрия, медицина, защита информации, геология, прогнозирование погоды, искусственный интеллект, космонавтика. Это представляет особый интерес для цифровой обработки данных, распознавания образов, криптоанализа. Одним из основных приложений PRUS в EDA (Electronic Design Automation) технологиях является эмуляция больших проектов [8], имплементируемых в ASICs и FPGA. Учитывая изложенное выше, далее необходимо сформулировать проблему, цель и задачи исследования с учетом основного предназначения мультипроцессора.

Сферический мультипроцессор PRUS предназначен для решения системы булевых уравнений большой размерности за приемлемое для пользователя время.

Проблема обработки булевых уравнений, насчитывающих порядка миллиона вентилей, связана с организацией вычислений на мультипроцессорной системе путем введения параллелизма и конвейеризации. Это позволяет уменьшить время получения решения в сотни раз по сравнению с последовательной обработкой данных.

Цель исследования – разработать однобитовый мультипроцессор, имплементируемый в ASIC, с матрично-сферической организацией взаимных связей и минимальным набором команд для повышения быстродействия решения систем булевых уравнений большой размерности в сотни раз.

Задачи исследования:

1) Разработать структуру сферического мультипроцессора PRUS, имплементируемого в кристалл ASIC и ориентированного на решение системы булевых уравнений большой размерности.

2) Спроектировать структуру секвенсера, как базовой ячейки PRUS, предназначенной для эмуляции примитивного элемента, выполняющего одну из следующих функций: AND, OR, NOT, XOR.

3) Разработать систему команд, достаточную для параллельного программирования и решения булевых уравнений на основе мультипроцессора PRUS.

4) Реализовать компилятор для преобразования HDL-описания системы булевых уравнений в язык машинных кодов мультипроцессора PRUS.

5) Разработать программный модуль оптимизации распределения системы булевых уравнений между секвенсерами PRUS в целях максимального использования параллельного и конвейерного режимов работы мультипроцессора для получения решения за минимально возможное время.

6) Разработать стратегию (System Design Flow) совместного использования современных средств проектирования и мультипроцессора PRUS для эффективного решения системы булевых уравнений большой размерности.

7) Выполнить верификацию и тестирование мультипроцессора PRUS на доказательных примерах систем булевых уравнений, представляющих промышленный и научный интерес, в целях оценивания быстродействия и аппаратурных затрат на реализацию проекта.

4. Структура и функции мультипроцессора PRUS

Мультипроцессор PRUS в качестве основного составляющего компонента включает секвенсер, как hardware модуль, содержащий однобитовый процессор (Single Bit Processor – SBP) и оболочку (wrapper) в виде памяти для хранения программного кода и данных, а также декодеров и мультиплексора. Собственно процессор педставлен в виде модуля для реализации операций {AND, OR, NOT, XOR}. Далее мы будем считать секвенсер и процессор синонимами, если это не приведет к неверному толкованию излагаемого материала.

Для оптимизации распределения булевых уравнений по частям мультипроцессора необходимо иметь представление о его топологической структуре. Она отчасти подобна карте Карно (Karnaugh) [14], поскольку каждый секвенсер имеет восемь соседей (в карте Карно – 4). Планарная структура PRUS представлена на рис. 5. Такое представление удобно для решения оптимизационных задач на плоскости, к которым можно отнести: размещение булевых уравнений по процессорам, минимизацию функциональных связей между секвенсерами при эмуляции уравнений. Поэтому естественным представляется вопрос измерения логических путей и соединений.

Для определения расстояния между двумя компонентами мультипроцессора необходимо задать метрику. Однозначно ясно, что здесь не может быть использована манхеттенская планарная система [14,15], поскольку расстояние между диагональными соседями равно расстоянию между соседними элементами по горизонтали и вертикали. Поэтому введем следующую модификацию.

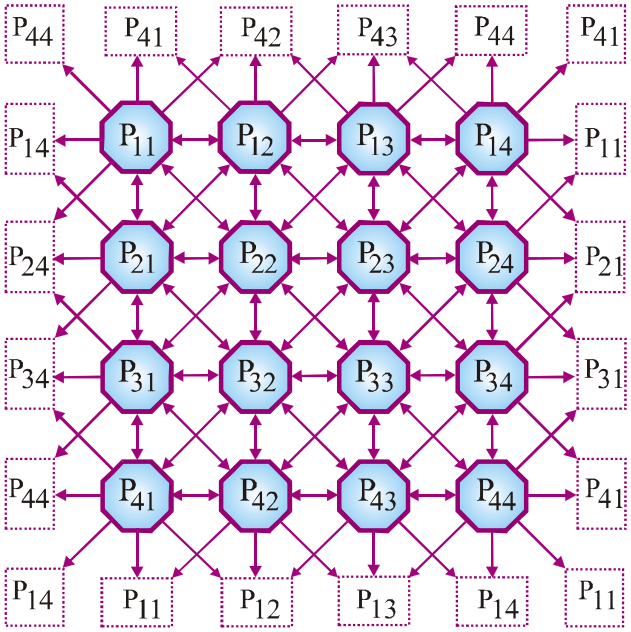

Рис. 5. Планарная структура процессора PRUS

Определение. Манхэттен*2 есть топологическое, дискретное, геометрическое пространство, каждая точка которого имеет восемь связей с соседними, по вертикали и горизонтали, а также по двум диагоналям, а метрика определяет минимальное в данном простанстве расстояние между двумя точками P_{ij}, P_{pq} (i,j и p,q – целочисленные декартовы координаты) в соответствии со следующим выражением:

$$d(P_{ij}, P_{pq}) = |i-p| + |j-q| - \min(|i-p|, |j-q|). \qquad (1)$$

Данная метрика удовлетворяет условиям эквивалентности отношений (рефлексивность, симметричность, транзитивность) [14], которые трансформируются с точностью до изоморфизма в следующие правила:

$$1)\ d(P_{ij}, P_{pq}) = 0 \leftrightarrow (P_{ij} = P_{pq});$$
$$2)\ d(P_{ij}, P_{pq}) = d(P_{pq}, P_{ij});$$
$$3)\ d(P_{ij}, P_{pq}) + d(P_{pq}, P_{st},) \geq d(P_{ij}, P_{st}), \forall \{P_{ij}, P_{pq}, P_{st}\} \in P.$$

Логическая структура мультипроцессора PRUS может быть представлена в виде квадратичной матрицы:

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1j} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2j} & \dots & P_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{i1} & P_{i2} & \dots & P_{ij} & \dots & P_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{n1} & P_{n2} & \dots & P_{nj} & \dots & P_{nn} \end{bmatrix} \qquad (2)$$

При n=4 матрица, размерностью 4x4, будет иметь 16 однобитовых процессоров. Каждый компонент P_{ij} имеет своими соседями 8 процессоров:

$$P_{ij} \rightarrow \{P_{i-1j}, P_{i+1j}, P_{ij-1}, P_{ij+1}, P_{i-1j-1}, P_{i+1j+1}, P_{i-1j+1}, P_{i+1j-1}\}.$$

Отсюда следует, что каждый крайний элемент также имеет 8 сеседних компонентов, поэтому в целом логическая структура процессора может быть представлена в сферической форме, изображенной на рис.6.

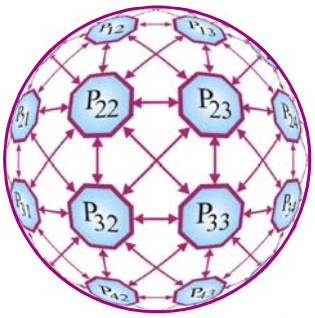

Рис. 6. Логическая структура процессора PRUS

Это означает, что все процессоры в сферической логической структуре идентичны и не имеют приоритетов при распределении булевых уравнений для их последующего решения.

5. Организация вычислительных процессов в PRUS

После определения логической математической модели мультипроцессора PRUS (2) необходимо задать модель системы уравнений, подлежащих решению, а затем совместить обе модели в целях поиска оптимального, а точнее сказать квазиоптимального размещения уравнений по процессорам. В общем случае система булевых уравнений представляется в скобочной форме [14]. Это означает, что объект или процесс описывается k уравнениями, структурная глубина которых равна n, где $n \geq 2$. С учетом сказанного будем представлять систему булевых уравнений в виде матрицы:

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1j} & \dots & Y_{1n} \\ Y_{21} & Y_{22} & \dots & Y_{2j} & \dots & Y_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ Y_{i1} & Y_{i2} & \dots & Y_{ij} & \dots & Y_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ Y_{k1} & Y_{k2} & \dots & Y_{kj} & \dots & Y_{kn} \end{bmatrix}, \qquad (3)$$

где $Y_{ij} = f_{ij}(X_1^{ij}, X_2^{ij}, \dots, X_r^{ij}, \dots, X_{m_{ij}}^{ij})$ – i-е уравнение j-го ранга; k – общее количество таких уравнений внутри ранга j; n – максимальный ранг подмножества уравнений.

Здесь Y_{ij} – выходная линия для функции f_{ij} от m_{ij} переменных $X_1^{ij}, X_2^{ij}, \dots, X_r^{ij}, \dots, X_{m_{ij}}^{ij}$, связанных между собой операциями AND, OR, NOT, XOR для формирования Y_{ij}. Ранг уравнения j соответствует номеру итерации, на которой формируется состояние выхода Y_{ij}, при этом предполагается, что значения всех входных переменных $X_1^{ij}, X_2^{ij}, \dots, X_r^{ij}, \dots, X_{m_{ij}}^{ij}$ были вычислены в предыду-

щих итерациях. Все уравнения ранга j могут вычисляться параллельно, разных рангов – последовательно или конвейерным способом.

В целях разработки стратегии распределения системы уравнений по секвенсерам с учетом топологии мультипроцессора необходимо сформулировать основные критерии их размещения:

1) Разбиение системы булевых уравнений на n^2 (по общему числу процессоров в матрице) максимально независимых и примерно одинаковых подсхем,минимизирующих количество связей между ними.

2) Обеспечение эквивалентности структурной глубины подсхем при разбиении системы уравнений.

3) Обеспечение минимальности межсоединений между секвенсерами для решения системы уравнений.

4) Минимизация общей длины топологических путей прохождения сигналов по матрице процессоров при размещении системы булевых уравнений.

5) Факторизация системы – выделение общих частей в записи уравнений и их упорядочение по мере убывания их мощностей.

6) Минимизация системы булевых уравнений до и после разбиения их на подсхемы.

7) Построение ориентированного графа распределения вычислительных процессов с вершинами-уравнениями, для которых время их обработки примерно одинаковое.

8) Рассмотрение критических случаев: мощность системы уравнений больше, чем количество секвенсеров; система содержит только одно уравнение большой размерности и структурной глубины, типа сходящийся (расходящийся) конус.

Стратегия распределения системы уравнений по секвенсерам основывается на использовании метода дихотомии [14]. Здесь имеется в виду итеративный характер процедуры, на каждом шаге которой рассматриваемое множество уравнений делится на две примерно одинаковые части. Процедура заканчивается, когда каждый секвенсер получает свою часть уравнений для их последующей эмуляции. На i-й итерации процедуры исходное множество уравнений (подсхема) делится на два подмножества $C_i = \{C_i^0, C_i^1\}$, для которых выполняются следующие три правила (объединение разделенных подсхем равно исходной схеме, пересечение – пустому множеству, полученные подсхемы примерно одинаковы):

$$1.C_i^0 \cup C_i^1 = C_i;\ 2.C_i^0 \cap C_i^1 = \varnothing;\ 3.\left|C_i^0\right| \approx \left|C_i^1\right|.$$

Выбор сечения схемы на подсхемы осуществляется по минимуму функции предпочтения:

$$f = \min_i [X(C_i^0) \cap X(C_i^1)],$$

которая позволяет выбрать из совокупности вариантов такой, который минимизирует множество общих для обеих подсхем переменных или межсоединений, где $X(C_i^0)$ и $X(C_i^1)$ – переменные в схемах C_i^0, C_i^1 соответственно.

Стоимость реализации стратегии есть самая большая подсхема, оцениваемая по Квайну [14], получившаяся после 2^m -1 (это суммарные затраты, m – число уровней в бинарном дереве) разбиений исходной схемы или системы уравнений с получением подсхем, количество которых (2^m) соответствует числу секвенсеров процессорной матрицы n^2. При n=8 суммарные затраты на получение бинарного дерева разбиения системы булевых уравнений (исходной схемы) равны n^2 -1 = 256 - 1 = 255.

6. Архитектура секвенсера PRUS

Основным компонентом мультипроцессора является секвенсер, структура которого представлена на рис. 7.

Секвенсер эмулирует булевы уравнения. Все секвенсеры в сети функционируют синхронно и параллельно, обрабатывая однобитовый поток логических данных путем выполнения операций AND, OR, NOT, XOR. Результат моделирования сохраняется в памяти данных. В целях синхронизации всей вычислительной системы программная память каждого секвенсера в сети адресуется общим счетчиком адресов команд.

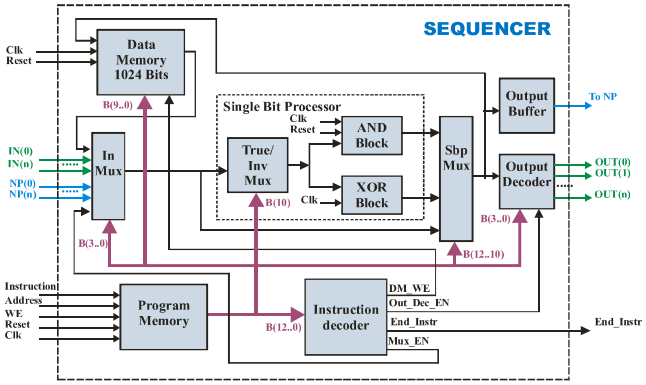

Рис. 7. Функциональная структура секвенсера

Program Memory – предназначена для хранения программы моделирования уравнений в течение всего цикла, необходимого для их решения. Очередная команда выбирается с помощью адреса, формируемого на соответствующей шине. Код команды управляет работой всех блоков секвенсера.

Data Memory – используется для хранения промежуточных результатов обработки уравнений на одном секвенсере. Здесь 10 младших разрядов командного слова используются как адрес памяти данных, если сигнал Data Memory Write Enable = 1. При таком количестве разрядов возможна адресация к 1024 ячейкам памяти данных.

InMux – используется для ввода внешних данных в секвенсер. Он может коммутировать один из восьми соседних процессоров (Mux_En = 1) или выбирать информацию из памяти данных.

Single Bit Processor (SBP) – выполняет две параллельные операции AND и OR над прямым или инверсным значением входной переменной.

SBP Mux – коммутирует выход одного из четырех компонентов (AND block; AND block инвертор; XOR block, InMux) на выход секвенсера. При этом коммутация блока InMux означает режим by pass для данного секвенсера в целях передачи информации ближайшему соседу.

Output Buffer – синхронизирует работу двух соседних процессоров и используется для передачи результирующей информации с выхода к соседу, после чего последний продолжает дальнейшую обработку уравнения.

Output Decoder – используется для выбора одного из n внешних выходов мультипроцессора PRUS в целях последующей передачи информации о решении системы уравнений.

Instruction Decoder – используется для декодирования адресов операндов и специальных команд, типа NOP, END. Он обеспечивает подачу управляющих синалов для инициирования работы блоков: Output Decoder, Data Memory, Input Multiplexer и Single Bit Processor. Кроме того, он активизирует сигнал END_INSTR, который управляет работой программного счетчика адресов. При END_INSTR = 1 он обнуляется.

Каждый секвенсер имеет собственную программную память для хранения двоичных инструкций моделирования логических уравнений и память данных для сохранения результатов их решения. Во время программирования системы уравнений в память каждого секвенсера загружается собственное множество машинных команд, предназначенных для обработки логических уравнений. Поскольку все секвенсеры и соответствующие им памяти управляются одним общим регистром адреса, то они функционируют и обрабатывают логические уравнения синхронно и параллельно (рис. 8).

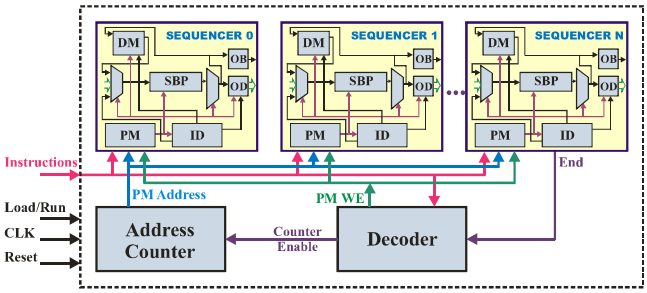

Рис. 8. Соединение секвенсеров

Счетчик адреса инкрементирует свое значение под управлением сигналов синхронизации системы CLK и Counter Enable – выходной сигнал декодера (Decoder), который равен 1 при достижении последней команды в программной памяти. Счетчик адреса может выбирать из памяти только последовательно идущие команды в программной памяти.

7. Структура и функции SBP

Однобитовый процессор SBP – основной компонент секвенсера. Его базовая структура представлена на рис. 9.

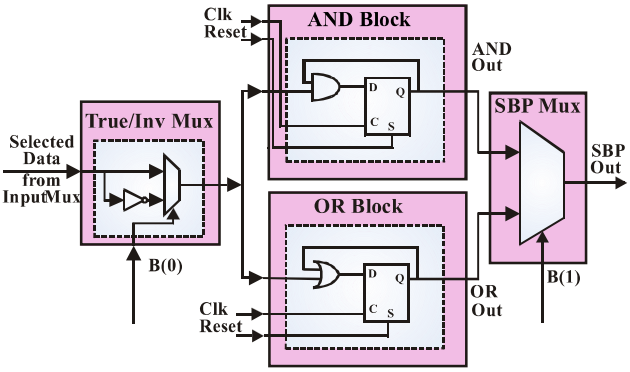

Рис. 9. Функциональная структура SBP

SBP предназначен для выполнения логических операций AND, OR, NOT, которых достаточно для решения любых сколь угодно сложных уравнений. В состав SBP входят также однобитовые OR- и AND-регистры для хранения промежуточных данных. SBP выполняет две операции над содержимым входа и регистра в параллельном режиме. Для выполнения операции AND (OR) на любом из блоков необходимо 2 временных такта: на первом выполняется загрузка первого операнда в однобитовый регистр, второй такт формирует результат на выходе D-триггера, как функцию $Q^t = Q^{t-1} \wedge X; Q^t = Q^{t-1} \vee X$, где X – вход данных для AND- или OR-блока. Один из полученных результатов передается на выход процессора с помощью мультиплексора SBP Mux. Функция мультиплексора True/Inv Mux заключается в передаче значения логической переменной в прямом или инверсном коде, что необходимо для обработки функций И-НЕ, ИЛИ-НЕ, а также простого инвертора или повторителя. Управление мультиплексором осуществляется одним битом командного слова. Данные для SBP могут быть доставлены со входа мультиплексора, от одного из соседних процессоров или из памяти данных. Что касается результата обработки уравнения, то он может быть передан в память данных, соседнему процессору или на внешний выход мультипроцессора.

Базовая структура (см. рис.9) может быть модифицирована путем замены одного из функциональных блоков. Поскольку правила де Моргана [14] позволяют преобразовать любую функцию к базису И-НЕ или ИЛИ-НЕ, то, естественно, что один из блоков AND- или OR-block можно считать избыточным. Поэтому OR-block предлагается исключить из SBP, поставив вместо него модуль, выполняющий логическую операцию XOR (рис.10). Какие преимущества это дает?

В базовой схеме SBP (см. рис.9) AND(OR)-регистр устанавливается в 1(0) перед началом обработки уравнения и он будет иметь в первом такте 1, если на вход поступит сигнал, равный 1. В модифици-

рованной структуре SBP (см. рис.10) регистр XOR не требует первоначальной установки. Его инициализация выполняется единичным сигналом старшего бита командного слова. Обратная связь с выхода XOR на вход регистра сдвига позволяет вычислять XOR-операцию много раз. Такое расширение функций SBP оказывает положительное влияние на быстродействие обработки булевых уравнений, имеющих операции XOR.

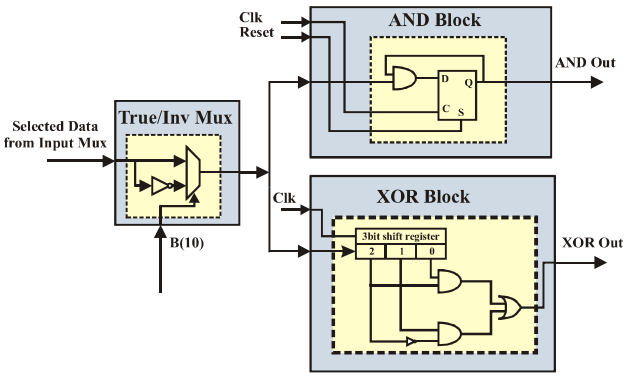

Рис. 10. Модифицированная структура SBP

8. Система команд SBP

Формат команд является функцией от архитектуры однобитного процессора, секвенсера и организации межсоединений SBP в матрице PRUS.

1) **IN** – операция AND над прямым значением входной переменной и содержимым регистра AND, результат которой сохраняется в регистре AND и передается на выход секвенсера.

2) **IN NOT** – операция AND над инвертированным значением входной переменной и содержимым регистра AND, результат которой сохраняется в регистре AND и передается на выход секвенсера.

3) **INI** – операция AND над прямым значением входной переменной и инвертированным содержимым регистра AND, результат которой сохраняется в регистре AND и передается на выход секвенсера.

4) **INI NOT** – операция AND над инвертированными значениями входной переменной и содержимым регистра AND, результат которой сохраняется в регистре AND и передается на выход секвенсера.

5) **AND** – передача состояния регистра AND на выход секвенсера, а далее в память данных или на один из выходов в зависимости от состояния 10-битного кода операции. Регистр AND устанавливается в 1 в целях его инициализации для выполнения следующей операции.

6) **ANDI** – запись инвертированного состояния регистра AND в память данных, а также передача его на выход. Регистр AND устанавливается в логическую единицу.

7) **XOR** – передача состояния XOR-блока на выход секвенсера, а далее в память данных или на один из выходов в зависимости от состояния 10-битного кода операции. Регистр **AND** устанавливается в логическую единицу.

8) **TRANSMIT** – данные со входа SBP подаются на выход секвенсера за один такт синхронизации без изменения.

9) **NOP** – программный счетчик команд ожидает количество тактов синхронизации, указаное в десяти младших битах командного слова.

10) **END_INSTR** – команда, завершающая последовательность выполнения инструкций для каждого процессора и инициирующая обнуление программного счетчика для следующего цикла моделирования.

Примечание. Регистр AND устанавливается в 1 перед началом вычисления уравнения и после выполнения операций: NOP, END_INSTR, AND, XOR и TRANSMIT.

9. Стратегия подготовки проекта для PRUS

Цикл проектирования цифрового устройства по технологии PRUS состоит из трёх основных частей: 1) преобразование проекта в систему булевых уравнений; 2) распределение булевых уравнений среди процессоров PRUS; 3) эмуляция проекта в мультипроцессоре PRUS (рис.11).

Система PRUS моделирует проекты, представленные в виде схемного описания на языках VHDL, Verilog или в форме графического изображения [16,17]. Цифровые модели задаются в виде HDL-кода, таблиц истинности или переходов для цифровых автоматов. Система ввода проекта поддерживает как RTL, так и поведенческие языковые конструкции.

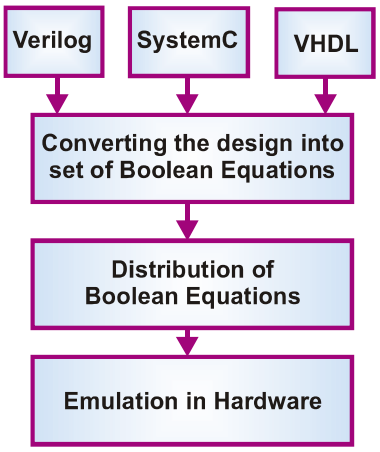

Рис. 11. Схема подготовки проекта для PRUS

На первом этапе HDL-код описания проекта конвертируется в систему булевых уравнений. На втором этапе уравнения обрабатываются с помощью алгоритмов оптимизации и преобразуются в бинарные инструкции. Далее происходит их распределение между секвенсерами. Для оптимального использования ресурсов мультипроцессора в программную память каждого секвенсера загружается примерно равное количество инструкций.

Компилятор BECom может точно определить время появления результата эмуляции булевой функции на выходах секвенсеров, поскольку число

команд в памяти каждого секвенсера известно. BECom, используемый для компиляции проекта, размещаемого в PRUS, имеет простые алгоритмы, которые работают в сотни раз быстрее, чем средства для проектирования FPGA или ASIC. Это связано с тем, что отсутствует необходимость во временном анализе (timing verification) схемы. Здесь необходим только функциональный анализ, что существенно сокращает цикл проектирования цифрового устройства. Частота дискретизации определяется временем выполнения программы максимальной длины, помещенной в память секвенсеров мультипроцессора.

10. Эмуляция и тестирование мультипроцессора

Фирма AllTech разработала и протестировала модель программируемого устройства, представляющего сеть из 64 процессоров (PRUS для PLD), реализованных на HES3-3000 (фирмы Alltech) с FPGA Virtex II и 256МБ ОЗУ. На PRUS были эмулированы десятки реальных цифровых устройств. Для этого был использован Design Compiler (Synopsys), который затратил одинаковое время на синтез проектов для FPGA и PRUS. Размещение логических уравнений (40000 вентилей) в мультипроцессор PRUS заняло 10 минут. Этот процесс может автоматически выполняться при помощи script-файла, что обеспечивает безошибочное функционирование проекта в PRUS. Размещение и трассировка этого же проекта в FPGA занимает несколько часов и требует высокого знания инструментальных средств и процесса проектирования. Разработанная сеть процессоров обеспечивает эффективное использование аппаратурных ресурсов. Имеется в виду, что 90% площади кристалла в PRUS занимает память. Это приводит к эффективному его использованию. Благодаря небольшому количеству регистров в устройстве (2 триггера на процессор), рассеяние мощности является достаточно низким даже при эмуляции больших систем. На рис. 12 представлены основные блоки, расположеные на плате HES. Тестовые воздействия подаются из среды Active-HDL посредством интерфейса PLI.

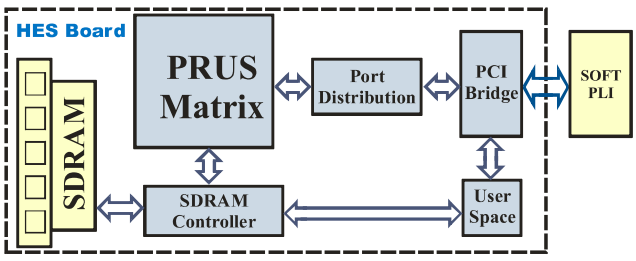

Рис. 12. Эмуляция мультипроцессора в аппаратуре

HES3-3000 имеет дочернюю плату для подачи тестовых воздействий на высоких тактовых частотах, до 10МГц. На плате размерностью 4,67' x 9,18' могут быть эмулированы проекты до 20 миллионов вентилей.

11. Программный код модели для обработки на SBP

В целях иллюстрации подготовки устройства для его эмуляции в системе PRUS необходимо рассмотреть следующий пример, представленный на рис. 13.

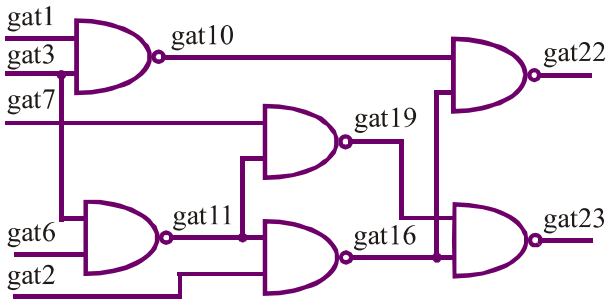

Рис. 13. Схема для подготовки к решению на PRUS

Листинг системы булевых уравнений для схемы 17 (см. рис. 13) из библиотеки ISCAS'85:

```
gat22 = ! (gat10 & gat16);
gat23 = ! (gat19 & gat16);
gat19 = ! (gat11 & gat7);
gat16 = ! (gat2 & gat11);
gat11 = ! (gat3 & gat6);
gat10 = ! (gat1 & gat3);
/* выходные функции схемы*/
gat22 = f1 (gat1, gat2, gat3, gat6, gat7)
gat23 = f2 (gat1, gat2, gat3, gat6, gat7)
/* Программы, распределенные по пяти секвенсерам, для решения булевых уравнений */
Sequencer A
NAND gat1        ;Loading gat1
NAND seq_b       ;Calculating gat10
NOP              ;Storing gat10
NAND seq_b       ;Calculating gat22
Sequencer B
NAND gat3        ;Loading gat3
NAND seq_d       ;Calculating gat11
NAND seq_c       ;Calculating gat16
NAND seq_e       ;Calculating gat23
Sequencer C
NAND gat2        ;Loading gat2
NOP              ;Storing gat2
NOP              ;Storing gat2
NOP              ;Storing gat2
Sequencer D
NAND gat6        ;Loading gat6
NOP              ;Storing gat6
```

```
NOP              ;Storing gat6
NOP              ;Storing gat6
Sequencer E
NAND gat7        ;Loading gat7
NOP              ;Storing gat7
NAND seq_b       ;Calculating gat19
NOP              ;Storing gat19
```

Время обработки системы булевых уравнений равно четырём тактам. Результат вычислений будет содержаться в секвенсерах A и B (gat22 и gat23 соответственно).

12. Заключение

В качестве основных результатов, фигурирующих в предложенной PRUS-технологии, следует выделить два основных компонента: научную и практическую значимость. Первый компонент – научная новизна – определяется формулировкой следующих тезисов:

1. Предложена новая высокопроизводительная PRUS-технология параллельного решения систем булевых уравнений, насчитывающих сотни миллионов эквивалентных вентилей.

2. Разработана архитектура сферического мультипроцессора, где каждый однобитный процессор имеет непосредственные связи с восемью другими.

3. Представлена схемотехнически простая структура однобитного процессорного элемента, ориентированная на выполнение операций AND, OR, NOT, XOR, а также секвенсера, в состав которого входит память программы и данных, а также соответствующие схемы управления.

4. Предложена модель организации вычислительных процессов, заключающаяся в равномерном распределении системы булевых уравнений между вычислительными ресурсами мультипроцессора на основе применения метода половинного деления.

Второй компонент – практическая значимость – показывает преимущества использования PRUS-технологии на мировом рынка EDA в сравнении с существующими:

1. Классические технологии имплементации цифровых систем в силиконовые кристаллы хорошо известны и доступны. Особенно популярными в настоящее время являются такие кристаллы, как ASIC, FPGA, CPLD, в которые могут быть имплементированы цифровые системы, имеющие компонентами ячейки, вентили, триггеры, соединенные между собой жесткими связями для образования функционально законченного проекта. Преимуществом таких технологий является высокое быстродействие выполнения операций, благодаря аппаратурному параллелизму прохождения сигналов через элементы от входов к выходам. Тем не менее, классическая технология имеет и недостатки, которые становятся более ощутимыми, когда геометрические параметры цифровых компонентов уменьшаются, а размерность проекта возрастает. Например, время реализации стадии трассировки и размещения для таких устройств возрастает экспоненциально от параметра размерности проекта. В PRUS-технологии время компиляции имеет линейную зависимость от размерности проекта.

2. Классическая технология проектирования требует существенных затрат для выполнения трудоемкого временного анализа критических путей на этапе размещения и трассировки. Отдельные FPGA-проекты могут повторять эту процедуру более 40 раз для достижения необходимой скорости выполнения операций. PRUS-методология полностью исключает дорогостоящий временной анализ. Первая компиляция сразу же дает готовое и удовлетворительное решение.

3. Современная архитектура устройства, использующая нанометрувую силиконовую технологию, требует сложного анализа компонентов и межсоединений для определения общей производительности системы. Физические явления, возникающие в нанометровом диапазоне, играют существенную роль при высокой плотности размещения элементов, делая анализ слоев все более сложным. При этом стадия силиконовой компиляции требует значительных человеческих ресурсов, которые замедляют процесс проектирования. Использование PRUS в данном случае исключает любые операции по ручному размещению и трассировке.

4. Эффективное тестирование сложных устройств на кристалле требует дополнительных встроенных средств диагностирования, таких как автоматические генераторы и анализаторы тестов, средства граничного сканирования (boundary-scan). Это делает процесс проектирования достаточно сложной процедурой и снижает эффективность использования полезной площади кристалла. С другой стороны, поскольку не существует других эффективных средств тестирования силиконового кристалла, такие технологии широко используются для верификации систем, имплементируемых в ASIC, размерностью более 100 000 вентилей. Это еще один аспект, где использование PRUS обеспечивает эффективное тестирование проекта без каких-либо дополнительных средств тестопригодности.

5. Учитывая случайный характер загрузки ячеек на стадии синтеза, большие площади силиконового кристалла остаются в стороне при организации связей между компонентами и могут быть не задействованы для выполнения функций в чипах CPLD и FPGA. Это уменьшает эффективность использования площади кристалла. PRUS имеет только регулярную архитектуру памяти и поэтому область межсоединений не является здесь проектнозависимой, что положительно влияет на улучшение использования полезной площади силикона.

6. Репрограммируемые устройства CPLD и FPGA устанавливают жесткие ограничения на соединения между вентилями и триггерами. Их количество даже в больших FPGA строго ограничено. PRUS-технология дает полную свободу для эмуляции как вентилей, так и триггеров в оперативной памяти. Он способен хранить в памяти миллионы триггеров и особенно эффективен для обработки последовательностных схем.

7. Современные технологии рассеивают большое количество тепловой энергии, благодаря паралаллелизму выпонения операций в схемах. Это ограничивает размерность проекта, который может быть размещен на кристалле. PRUS использует параллельно-последовательный режим выполнения операций, который снижает мощность рассеивания и позволяет повысить плотность размещения компонентов на кристалле. Более того, базовый процессор PRUS представлен только двумя триггерами и менее чем 90 вентилями, поэтому он рассеивает незначительное количество энергии.

8. Современные силиконовые компиляторы более ориентированы на использование эвристических средств, чем на строгие математические методы и алгоритмы. Поэтому каждый проектировщик должен вручную улучшать отдельные схемные решения с целью повысить производительность используемой площади кристалла. Это требует высокого уровня послесинтезной экспертизы, постоянного повышения квалификации работников, проведения испытаний и знания методов тестирования и верификации. PRUS, основанный на использовании детерминированных методов и алгоритмов, полностью обеспечен автоматическими средствами проектирования, что исключает ручную доводку проекта и снижает требования к уровню экспертизы со стороны проектировщика.

9. Современные технологии и средства проектирования чувствительны к физическим явлениям в силиконе, поэтому проектировщики постоянно нуждаются в приобретении более усовершенствованных программных приложений. PRUS ограждает проектировщика от необходимости отслеживать постоянные изменения в силиконовой технологии. Однажды приобретенный набор средств проектирования для сопровождения PRUS в состоянии поддерживать будущие улучшения в технологии кристаллов в течение длительного времени.

10. Современный прирост силиконовой продукции опережает развитие средств проектирования с большим преимуществом. В настоящее время силиконовые технологии позволяют создавать устройства, имеющие сотни миллионов транзисторов, но современные коммерческие средства синтеза ограничивают мощность проекта в пределах 100 миллионов транзисторов. Поэтому средства и методологии проектирования являются главным препятствием на пути расширения и вредрения силиконовых устройств. Однако здесь ошибка не в средствах проектирования, а в пути имплементации цифровых систем в кристалл. Современные силиконовые архитектуры делают средства проектирования чрезвычайно медленными, громоздкими и сложными в использовании. В этом случае PRUS-технология обеспечиват такую силиконовую архитектуру, которая способствует появлению быстрых и простых средств и методологий проектирования, требующих минимального времени обучения для использования мультипроцессора при разработке новых устройств.

Литература: 1. *Nick Tredennock, Brion Shimamoto*. Go Reconfigure. IEEE Spectrum. December, 2003. P. 37-40. **2.** *Manfred Glesner, Thomas Hollstein, Tudor Murgan*. System Design Challenges in Ubiquitous Computing Environments. Proceedings of the 9th Biennual Baltic Electronic Conference. Tallinn. Estonia. 2004. P. 343 - 346. **3.** *Alberto Vincentelli, Luca Carloni, Fernando De Bernardinis, Marco Sgroi*. Benefits and Challenges for Platform-Based Design.Design Automation Conference. Proceedings. San Diego, California. 2004. P. 409 - 414. **4.** *Wayne Wolf*. The Future of Multiprocessor System-on-Chips. Design Automation Conference. Proceedings. San Diego, California. 2004. P. 681 - 685. **5.** *Bass M.J. and Cristensen C.M*. The future of the Microprocessor Business.- IEEE Spectrum, April, 2002. P. 34-39. **6.** *Don Edenfeld, Andrew B Kahng, Mike Rodgers, and Yervant Zorian*. 2003 Technology Roadmap for Semiconductors, Computer, January, 2004. P. 47-56. **7.** *Yoo S. and Jerraya A*. Introduction to Hardware Abstraction Layers for SoC // Design Automation and Test in Europe. DATE 03. 2003. P. 336-338. **8.** *Nikolaos S. Voros, Luis Sanchez, Aleandro Alonso, Alexios N. Birbas, Michael Birbas, Ahmed Jerraya*. Hardware-Software Co-Design of Complex Embedded Systems. Design Automation for Embedded Systems. Boston: Kluwer Academic Publishers. 2003. P. 5-34. **9.** *Cocke John*. U.S. patent 4,306,286 December 15, 1981. **10.** *Hyduke Stanley*. U.S. Patent 6,578,133, June 10, 2003. **11.** *Srihari Cadambi, Chandra S. Mulpuri, Pranav N. Ashar*. Fast, Inexpensive and Scalable Hardware Acceleration Technique for Functional Simulation. *DAC 2002, June* 10-14, 2002. P. 570-575. **12.** *Incisive* Palladium Family with Incisive XE Software Datasheet, Cadence Inc, 2004. 244p. **13.** *David Baneres, Jordi Cortadella, Mike Kishinevsky*. A Recursive Paradigm to Solve Boolean Relations. Design Automation Conference. Proceedings. San Diego, California. 2004. P. 416 - 421. **14.** *Richard J.* Discrete Mathematics, Prentice Hall 2001, 621 p. **15.** *Jing Li, Tan Yan, Bo Yang, Juebang Yu, Chunhui Li*. A Packing Algorithm for Non-Mahattan Hexagon/Triangle Placement Design by Using an Adaptive O-Tree representation. Design Automation Conference. Proceedings. San Diego, California. 2004. P. 646 - 651. **16.** *Active-HDL* User's Guid. Second Edition. Copyright.– Aldec Inc. 2003. 213p. **17.** *Samir Palnitkar*. Verilog HDL. A Guide to digital design and synthesis. Sunsoft Press. A prentice Hall Title. 2002. 396p.

Поступила в редколлегию 11.11.2004

Рецензент: д-р техн. наук, проф. Кривуля Г.Ф.

Гайдук Стенли, доктор, президент фирмы Aldec, США. Научные интересы: тестирование и верификация цифровых систем на кристаллах. Увлечения: антиквариат, книги, путешествия, садоводство. Адрес: США, Лас-Вегас, Гендерсон, Aldec Inc. E-mail: Stanley@aldec.com

Хаханов Владимир Иванович, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua

Обризан Владимир Игоревич, студент кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и верификация цифровых систем на кристаллах. Увлечения: английский язык, книги, путешествия. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua

Каменюка Евгений Александрович, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и верификация цифровых систем на кристаллах. Увлечения: автопутешествия. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua