УДК 004.312: 004.052.42: 621.3.049.771.14 DOI: 10.31114/2078-7707-2020-3-73-78

Разработка моделей специальных логических элементов для анализа быстродействия реконфигурируемых систем на кристалле

А.С. Михмель, И.А. Мкртычан, Д.В. Тельпухов

Институт проблем проектирования в микроэлектронике РАН, г. Москва, artem@ippm.ru

***Аннотация* — В данной работе предложены методы временного анализа схем, реализованных в базе реконфигурируемых систем на кристалле (РСнК). Методы основаны на идее кластеризации и частичной характеризации с последующим использованием инструментов статического временного анализа. РСнК относится к новой архитектуре СБИС, которая содержит реконфигурируемые логические блоки на основе ПЛИС и IP-блоки, такие как ОЗУ, PLL, арифметические блоки и другие. Частичная характеризация одной ячейки занимает времени значительно меньше, чем время обычного моделирования всей схемы в электрическом симуляторе. В статье предложены методы автоматического распараллеливания вычислительного процесса для дальнейшего ускорения процесса временного анализа. Проведенные эксперименты демонстрируют высокую точность расчета временных задержек схемы за приемлемое расчетное время, что позволяет использовать методы в задаче поиска оптимального размещения и маршрутизации.**

***Ключевые слова* — реконфигурируемая система на кристалле, РСнК, программируемая логика, частичная характеризация, статический временной анализ, кластеризация.**

I. Введение

Современные коммерческие реконфигурируемые системы на кристалле (РСнК) в настоящее время стремятся занять большую часть рынка интегральных микросхем. Их архитектура содержит наряду с комбинационной и последовательной логикой программируемой логической интегральной схемы (ПЛИС): блоки памяти, умножители, блоки цифровой обработки сигналов и др. Каждое семейство РСнК имеет свою собственную уникальную архитектуру и технические особенности, что влечет за собой совершенно разные проблемы и требует применения индивидуального подхода к каждой из архитектур при автоматизации проектирования заказных интегральных схем на основе таких систем.

Конфигурирование ПЛИС в составе РСнК осуществляется загрузкой пользователем необходимой информации в конфигурационную память (конфигурационное ОЗУ), которое управляет коммутационными ресурсами ПЛИС. Коммутационные элементы распределены по всему устройству для обеспечения необходимых соединений между блоками РСнК и установки режимов их функционирования.

Но коммутационными ресурсами кристалла могут быть не только стандартные ключи, но и целый ряд дополнительных логических элементов для конфигурирования и трассировки, расположенных как снаружи, так и внутри логических элементов (ЛЭ) [1]. Также для дополнительных возможностей конфигурирования и трассировки межсоединений, ПЛИС обладают еще одной характерной особенностью – их сигнальные шины могут приходить на входы логических элементов с инверсией.

Наличие широкого спектра дополнительных элементов для конфигурирования, наличие инверсий и других архитектурных особенностей РСнК приводят к тому, что стандартные методы оценки быстродействия становятся неприменимыми [2]. В связи с этим появляется необходимость в разработке специальных методов временного анализа, учитывающих данные архитектурные особенности.

II. Разработанный алгоритм

Первым этапом классического маршрута проектирования схем в базисе ПЛИС является загрузка исходных данных проекта (архитектура ПЛИС, режим работы, описание схемы).

На основе входных данных выполняются логический синтез и трансляция проекта в базис ячеек логических блоков ПЛИС. На последующих этапах происходит размещение, трассировка и генерация конфигурационных данных для описания исходной схемы в базисе логических блоков ПЛИС (netlist) [3].

В дополнение к классическому маршруту проектирования коллективом авторов статьи были разработаны алгоритмы и программные средства для обеспечения высокой точности оценки быстродействия. Для схемы, размещенной на ПЛИС, генерируется файл в SPICE-формате с учетом информации о межсоединениях, элементах и логических блоках. На основе этого файла выполняется группировка в макроячейки. Каждая макроячейка является объединением групп элементов межсоединений, входных или логических блоков. На следующем этапе выполняется характеризация сформированных макроячеек посредством собственного программного

обеспечения (ПО) на базе какого-либо электрического симулятора [4]. Далее используется собственное ПО для проведения статического временного анализа схем.

На рис. 1 представлен маршрут работы программы статического временного анализа.

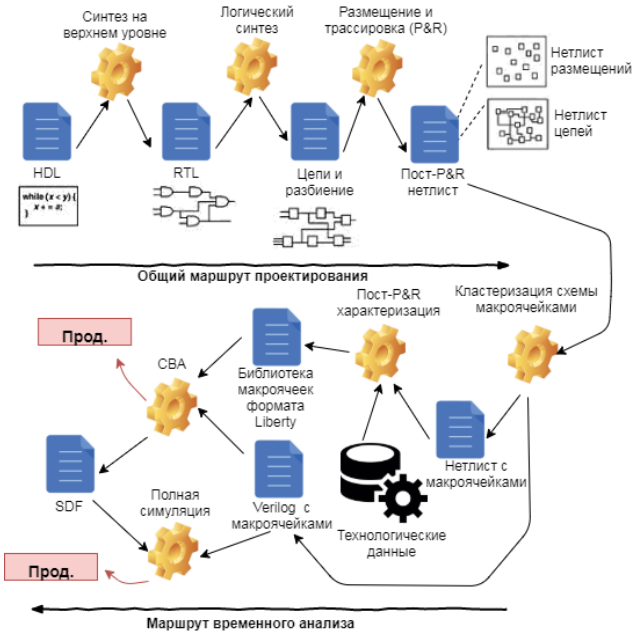

Рис. 1. Маршрут работы программы статического временного анализа

Разработанные и описанные в статье алгоритмы и модели позволяют обеспечить высокую точность оценки быстродействия за счет учета информации о межсоединениях, элементах и логических блоках. Учитывая то, что это происходит еще и достаточно быстротечно, это является ключом к оптимальному размещению и трассировке.

III. Реализация алгоритма

После этапов логического синтеза схемы, размещения блоков и трассировки получается файл соединений формата SPICE, с которым работает данный маршрут. Этот файл содержит в плоском виде соединения логических LE и входных/выходных IO блоков, а также трассировочных элементов [5]. В именах цепей и инстансов сохранена иерархия. При формировании макроячеек в схему добавляется дополнительный уровень иерархии. В данной статье под макроячейкой понимается единица кластеризации схемы и содержит в себе часть файла соединений.

Для проведения упрощенного статического временного анализа [6-9] схемы производится частичная характеризация макроячеек и формирование Synopsys библиотеки. В статье рассмотрено два способа частичной характеризации макроячеек: характеризация блока и трассировочных элементов (метод SP4) или только трассировочных элементов (метод SP5).

A. Описание метода SP4

Формирование макроячейки для метода SP4 происходит следующим образом. Вся схема разбивается на ячейки, включающие в себя один базовый элемент: логический элемент LE (logic element) или входной IO (input/output) блок – и элементы трассировки: ключи, буферы, инверторы и мультиплексоры [1]. На рис. 2 показано разбиение на примере схемы C17 [10].

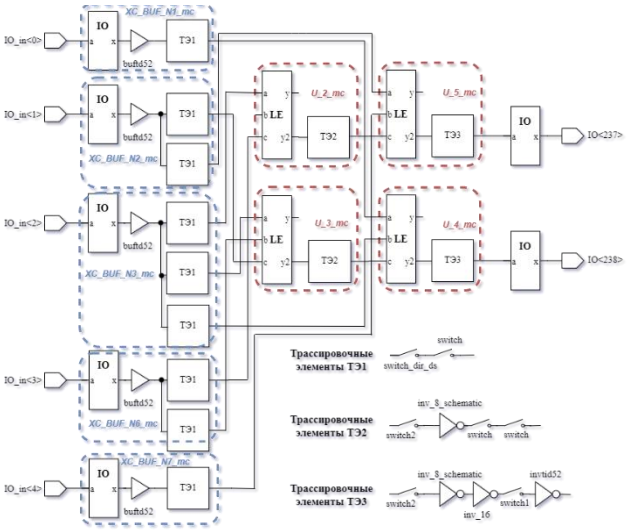

Рис. 2. Структура схемы C17 с учетом межсоединений

В данном случае не происходит формирования макроячеек с базовым элементом выходным IO блоком в виду отсутствия за ним в схеме последовательности из трассировочных элементов. Если базовый элемент внутри новой ячейки всегда один, то число элементов трассировки может доходить до 50 штук в зависимости от сложности схемы. На рис. 3 структурно показан новый добавленный уровень иерархии для схемы C17.

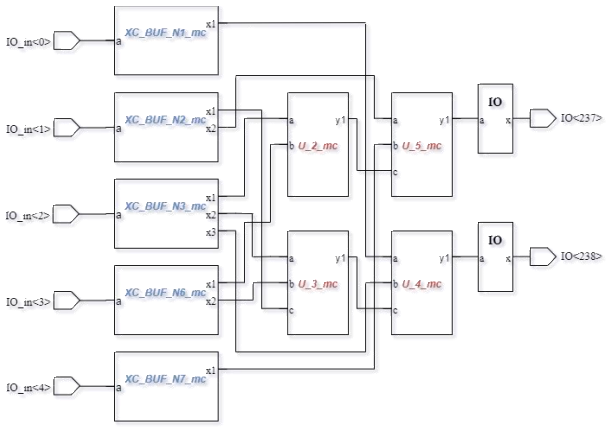

Рис. 3. Структура схемы C17 в базисе макроячеек

Внутри новой ячейки формируется следующее расположение элементов: базовый элемент находится в начале, входы макроячейки являются входами базового элемента; от выходов базового элемента внутри макроячейки следуют элементы трассировки, которые идут на выходы макроячейки. Таким образом, входы

макроячейки взаимно-однозначно соответствуют входам базового элемента. Набор выходов макроячейки формируется за счет сложности соединений трассировочных элементов и обычно пропорционален количеству трассировочных элементов. В случае макроячейки с входным IO блоком мы имеем только один вход, количество выходов также зависит от трассировочного решения.

Логическая функция, которую реализует макроячейка, напрямую зависит от базового элемента. Если берется в рассмотрение макроячейка с базовым элементом входным IO блоком и входом “a”, то все выходы этой ячейки будут иметь функцию “a”, так как в такой макроячейке не используются трассировочные элементы с инверсией. Если базовый элемент макроячейки является логическим LE блоком, то функция на каждом выходе макроячейки зависит от трех составляющих: функции базового блока, инверсии на входах макроячейки, инверсии на выходах макроячейки. В свою очередь, инверсии на входах и выходах макроячейки зависят от наличия инверторов среди последовательности трассировочных элементов на входе и выходе.

Входные данные для частичной характеризации формируются исходя из описанных выше особенностей макроячейки. Так, воздействия для регистрации переключений состояний генерируются на основе логической функции и заносятся в специальный-файл.

Во время характеризации используются следующие предположения:

1) Нагрузочная емкость на выходе текущей макроячейки равна емкости на входе следующей за ней макроячейки. Это достигается за счет соединения одной цепью. Значение входной емкости извлекается из библиотеки Synopsys стандартных ячеек для конкретной базовой ячейки, формирующей макроячейку, и конкретного входного порта. За счет этого можно уменьшить количество характеризационных циклов в 5-7 раз. Следует отметить, что для выполнения статического временного анализа нужно иметь в наличии сформированную библиотеку стандартных ячеек и ячеек входа/выхода, которая формируется после характеризации ячеек LE и IO и входит в поставку ПЛИС. За счет изменения прошивки блоков LE и IO происходит вариация входов, выходов и получаются разные логические функции на выходе. Входные емкости ячеек в библиотеке Synopsys отличаются незначительно между собой, что позволяет использовать некоторую усредненную величину выходной емкости. Тестами также подтверждено, что этой разницей можно пренебречь.

2) Для уменьшения времени характеризации и в виду того, что нас интересуют только временные характеристики, не будет производиться расчет статической и динамической мощностей.

3) Для макроячеек, реализующих последовательную логику, в составе которых есть такие ячейки, как триггер или защёлка (latch), не производится расчет setup/hold. Эти значения извлекаются из стандартной библиотеки Synopsys, сформированной для всех возможных вариаций прошивок логических элементов LE. Это положение основано на том факте, что элементы трассировки никак не влияют на внутренние характеристики самого триггера или защелки. Благодаря этому ожидается, что характеризация такой макроячейки не рискует превратиться в многочасовой процесс, как в случае с полной характеризацией базовой ячейки, реализующей последовательную логику. Естественно, время исполнения процесса зависит от аппаратных мощностей компьютера. Напомним, что данные характеризации базовой ячейки предоставляются вместе с программным обеспечением самой ПЛИС.

4) Также описываемый программный продукт позволяет распараллелить процесс характеризации согласно установкам при запуске, по умолчанию 10 запусков симулятора работают одновременно на одном сервере. Каждая макроячейка требует отдельного запуска симулятора. Так, для схемы ISCAS’s C17, синтезированной в базисе логических элементов, сформировано 9 макроячеек (4 макроячейки на базе логических ячеек LE и 5 макроячеек на базе IO ячеек). Среднее время характеризации логической макроячейки составляет 2-3 минуты, макроячейки на базе IO ячейки – 5-7 минут. Таким образом, при распараллеливании процесса характеризации схемы C17 на 9 подзадач общее время характеризации будет составлять 7 минут. Если же характеризацию проводить последовательно, используя только одно ядро, то время исполнения составит 35 минут.

На базе характеризационных данных формируется собственная библиотека макроячеек формата Liberty.

Завершающими этапами являются поиск, оценка и формирование отчета о максимальном критическом пути схемы в базисе библиотеки макроячеек в собственной программе статического временного анализа.

Программное обеспечение STA может выполнять DSTA (детерминированный статический временной анализ). Для статического временного анализа время прибытия сигнала и время перехода распространяются от первичных входов к первичным выходам каждого комбинационного блока, а требуемое время прибытия сигнала распространяется от первичных выходов к первичным входам. Поддерживается расчет как наибольшего, так и самого раннего реального/требуемого времени прибытия. Для схем на базе триггеров и защелок (latch) реализованы анализ дерева синхронизации и расчет минимального времени установки/удержания (setup/hold, recover/removal time).

B. Описание метода SP5

Формирование макроячейки для метода SP5 происходит следующим образом. В схеме все трассировочные элементы разбиваются на части, соответствующие блокам IO или LE. Каждая такая часть образует макроячейку и располагается между

двумя блоками IO-LE или LE-LE. По сути, макроячейка SP5 отличается от макроячейки SP4 только отсутствием базового элемента: IO или LE.

Количество входов макроячейки равно количеству используемых выходов базового элемента. Макроячейка содержит только трассировочные элементы. Таким образом, входы макроячейки взаимно-однозначно соответствуют выходам базового элемента. Набор выходов макроячейки формируется за счет сложности соединений трассировочных элементов и пропорционален их количеству. Макроячейка, следующая за входным блоком IO, имеет только один вход, количество ее выходов также зависит от трассировочного решения.

Логическая функция, которую реализует макроячейка по методу SP5, не зависит от ведущего блока. Если мы рассматриваем макроячейку с ведущим входным блоком IO и входом макроячейки “a”, то все выходы этой макроячейки будут также иметь функцию “a”, так как в такой макроячейке не используются трассировочные элементы с инверсией. Функция на каждом выходе макроячейки на базе LE зависит от инверсии на выходах макроячейки и может быть либо “a”, либо “!a” (или “b” и “!b”). Также инверсии на выходах макроячейки зависят от наличия инверторов среди последовательности трассировочных элементов внутри макроячейки, выходящих на данный выход. По сути, такая макроячейка работает как буфер или инвертор.

Входные данные для частичной характеризации формируются исходя из описанных выше особенностей макроячейки. Нюансы характеризации макроячейки SP5 такие же, как и у описанной выше SP4.

IV. Экспериментальные результаты

Маршрут расчета временных характеристик на базе метода частичной характеризации был проведен для 7 схем LGSynth’89/91 и ISCAS’85 [11-12]. Табл. 1 показывает их характеристики.

Таблица 1

Характеристики тестовых схем

Схема	Количество			
	входов	***выходов***	***ЛЭ***	***ТЭ***
b1_synth	3	4	5	58
c17	5	2	4	41
con1	7	2	9	111
decod_synth	5	16	24	385
majority_synth	5	1	5	50

В табл. 2 приведены результаты работы программы по расчету временных задержек для алгоритмов SP4 и SP5 и их сравнение с результатами работы электрического симулятора Spectre, где производился полный перебор сигналов на входах схемы. Из таблицы видно, что отклонение критического пути между традиционным расчетом Spectre и методом SP4 не превышает в среднем 11%, а между Spectre и методом SP5 не превышает в среднем 14%. Разница между SP4 и SP5 не превышает 7%. Полученные результаты говорят о приемлемой точности метода для использования в задачах оптимизации размещения и трассировки схем, реализуемых в РСнК.

Таблица 2

Статистика по временным задержкам

Схема	Макс. задержка, нс		
	Spectre	***SP4***	***SP5***
b1_synth	9,97	12,03	12,13
c17	10,29	9,39	9,69
con1	13,25	11,50	12,02
decod_synth	11,81	12,97	15,37
majority_synth	11,24	11,12	11,55

Табл. 3 дает сравнение времени работы реализаций алгоритмов SP4 и SP5 (в секундах), запущенных на 1-ом ядре и на 10-ти ядрах. Для сравнения, симуляционные циклы Spectre на полном переборе для схем с 11 входами требуют времени порядка 200 часов и больше.

Таблица 3

Время характеризации схем

Схема	Затраченное время, сек			
	SP4	***SP5***	***SP4 #10***	***SP5 #10***
b1_synth	896	41,3	232,4	7,33
c17	1376,8	31,32	226,3	5,59
con1	2028,4	64,64	281,9	11,03
decod_synth	2850,8	360,62	484,4	81,22
majority_synth	1309,7	31,04	237,8	4,67

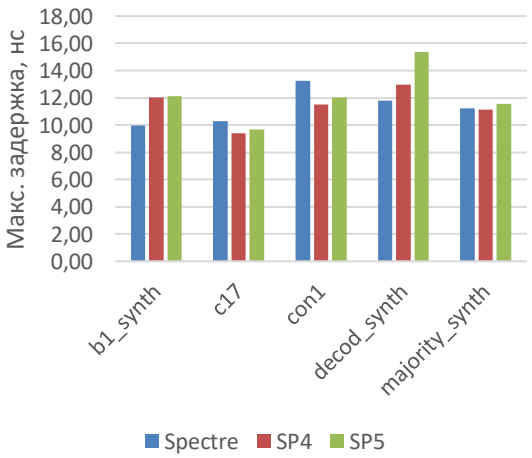

Рис. 4. Результаты по временным задержкам из табл. 2

Анализ данных, представленных в табл. 3, показывает очевидное преимущество параллельного выполнения частичной характеристики даже в случае метода SP4. Что касается времени работы методов SP4 и SP5, то наблюдается 30-кратная разница во времени. Ускорение за счет параллельного запуска составляет 5-6 раз при использовании 10 ядер.

В табл. 4 показано среднее время характеризации одной макроячейки (двух типов: на базе LE и IO) для каждой схемы. Расчет временных задержек производился по методу SP4 и SP5.

Таблица 4

Усредненное время характеризации одной макроячейки

Схема	**Затраченное время, sec**			
	SP4		***SP5***	
	LE	***IO***	***LE***	***IO***
b1_synth	42,48	227,87	4,76	5,83
c17	63,73	224,38	2,56	4,22
con1	42,72	234,84	2,54	5,97
decod_synth	45,64	351,10	5,86	43,98
majority_synth	29,30	232,64	2,23	3,98

Из таблицы видно, что для метода SP4 существует большая разница во времени характеризации макроячейки на основе логического блока LE и блока ввода-вывода IO. В то же время, в методе SP5 этого различия практически не существует, поскольку он не включает базовые элементы, а только элементы маршрутизации.

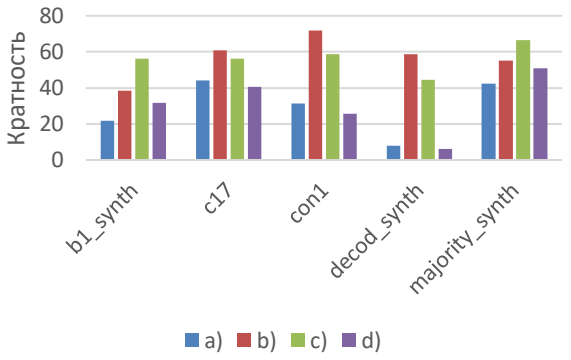

Рис. 5. Результаты тестов времени исполнения из табл. 3. а) SP5 (#1 ядро) и SP4 (#1 ядро); б) SP4 (#1 ядро) и SP4 (#10 ядер); в) SP5 (#1 ядро) и SP5 (#10 ядер); г) SP5 (#10 ядер) и SP4 (#10 ядер)

V. Заключение

Разработана модель частичной характеризации макроячеек, позволяющая при достаточной точности получаемых результатов производить расчет временных задержек схем достаточно быстро. Алгоритм SP4 и его модификация SP5 реализованы на языке C и имеют интерфейс Tcl. Предложенные алгоритмы SP4 и SP5 являются перспективным решением и будут иметь дальнейшее развитие в рамках проектов в ИППМ РАН. Разработанный алгоритм показал свою эффективность на схемах, проектируемых на основе отечественных ПЛИС, и включающих сложные конструкции, такие как параметризованные макроблоки и блоки памяти.

Разработанный алгоритм успешно используется в ИППМ РАН в задаче оптимизации маршрутизации схемы и анализа полного моделирования.

Литература

[1] Gavrilov S. V., Zheleznikov D. A., Khvatov V. M. Solving the Problems of Routing Interconnects with a Resynthesis for Reconfigurable Systems on a Chip, Russian Microelectronics, Vol 47, No. 7, 2018, pp 516–521.

[2] Lu J., Xu N., Yu J., Weng T., Research of timing graph traversal algorithm in static timing analysis based on FPGA. 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, 2017, pp. 334-338.

[3] Verilog-to-Routing. FPGA Architecture Description. Architecture Reference. Wire Segments. (2016). [Электронный ресурс]. Режим доступа: https://docs.verilogtorouting.org/en/latest/arch/reference (дата обращения: 30.04.2020).[4] Gavrilov S. V., Gudkova O. N. and Egorov Yu.B. Methods of accelerated characterization of VLSI cell libraries with prescribed accuracy control // Russian Microelectronics, Vol.40, N7, 2011, pp.476-482. DOI: 10.1134/S1063739711070067

[5] Gavrilov S., Zheleznikov D., Khvatov V., Chochaev R. Clustering optimization based on simulated annealing algorithm for Reconfigurable Systems-on-Chip // 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering. 2018. Pp. 1492-1495.

[6] Gavrilov S., Ivanova G., Ryzhova D., and Volobuev P. Multi-Interval Static Timing Analysis Accounting Logic Compatibility // 14th IEEE EAST-WEST DESIGN & TEST SYMPOSIUM (EWDTS). 2016. pp. 440-443. DOI: 10.1109/EWDTS.2016.7807650

[7] Ivanova G.A., Korshunov A.V., Tiunov I.V., Zhukova T.D. Development of Mathematical Models for Standard Cells Timing Analysis at the 28 nm Technology and Below // 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering. 2017. pp. 525-529. DOI: 10.1109/EIConRus.2017.7910832

[8] Stempkovsky A., Glebov A., Gavrilov S. Calculation of Stress Probability for NBTI-Aware Timing Analysis // Proc. of ISQED, 2009, pp.714-718. DOI: 10.1109/ISQED.2009.4810381

[9] Sundareswaran S., Gavrilov S., Soloviev R., Panda R. A Timing Methodology Considering Within-Die Clock Skew Variations // IEEE International SOC conference, Sep 17-20, 2008, Newport Beach, CA. pp. 351-356.

[10] D. Bryan, “The ISCAS ‘85 Benchmark Circuits and Netlist Format”, North-Carolina State University, 1985.

[11] F. Brglez, D. Bryan, K. Kozminski, “Notes on the ISCAS ‘89 Benchmark Circuits”, North-Carolina State University, 1989.

[12] Collection of Digital Design Benchmarks. [Электронный ресурс]. Режим доступа: http://ddd.fit.cvut.cz/prj/Benchmarks/ (дата обращения: 30.04.2020).

Development of Special Logic Element Models for Timing Analysis of Reconfigurable System-on-a-Chip

A.S. Mikhmel, I.A. Mkrtchan, D.V. Telpukhov

Institute for Design Problems in Microelectronics RAS, Moscow, artem@ippm.ru

***Abstract* — In this paper we have proposed and researched several ways to implement the method of partial characterization of special blocks and fast calculations of the time delays of a circuit implemented in a reconfigurable system on chip (RSoC) based on the method. The RSoC under consideration is a new VLSI architecture that contains reconfigurable logic blocks and IP cores based on FPGA, such as RAM, PLL, LVDS, multipliers, and others. Special blocks include the basic element and routing elements. The base element implements combinational or sequential logic, or it may be an I/O device. Partial characterization of such cell takes much less time than the time of simulation the entire circuit by an electrical simulator. At the same time the maximum possible complexity of a special macrocell is known in advance for a specific RSoC. Based on this we can estimate the total time of characterization, special cells library generation and calculation of time delays. Moreover, it is possible to automatically generate characterization tasks on different server cores, which allows to parallelize the process and to significantly reduce the overall runtime. The method allows you to achieve a fairly accurate calculation of the circuit time delays in acceptable time, which in turn makes it possible to use the method for the task of searching optimal routing solution. The method is also optimized over time due to partial characterization of special cells that include only routing elements. Time delays are calculated from the data described above and presented in the provided library of standard cells. The method has shown excellent practical results in the CAD flow for domestic RSoC.**

***Keywords* — reconfigurable system-on-a-chip, RSoC, programmable logic, partial characterization, static timing analysis, clustering.**

REFERENCES

[1] Gavrilov S. V., Zheleznikov D. A., Khvatov V. M. Solving the Problems of Routing Interconnects with a Resynthesis for Reconfigurable Systems on a Chip, Russian Microelectronics, Vol 47, No. 7, 2018, pp 516–521.

[2] Lu J., Xu N., Yu J., Weng T., Research of timing graph traversal algorithm in static timing analysis based on FPGA. 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, 2017, pp. 334-338.

[3] Verilog-to-Routing. FPGA Architecture Description. Architecture Reference. Wire Segments. (2016). Available at: https://docs.verilogtorouting.org/en/latest/arch/reference (accessed date: 30.04.2020).

[4] Gavrilov S. V., Gudkova O. N. and Egorov Yu.B. Methods of accelerated characterization of VLSI cell libraries with prescribed accuracy control // Russian Microelectronics, Vol.40, N7, 2011, pp.476-482. DOI: 10.1134/S1063739711070067

[5] Gavrilov S., Zheleznikov D., Khvatov V., Chochaev R. Clustering optimization based on simulated annealing algorithm for Reconfigurable Systems-on-Chip // 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering. 2018. Pp. 1492-1495.

[6] Gavrilov S., Ivanova G., Ryzhova D., and Volobuev P. Multi-Interval Static Timing Analysis Accounting Logic Compatibility // 14th IEEE EAST-WEST DESIGN & TEST SYMPOSIUM (EWDTS). 2016. pp. 440-443. DOI: 10.1109/EWDTS.2016.7807650

[7] Ivanova G.A., Korshunov A.V., Tiunov I.V., Zhukova T.D. Development of Mathematical Models for Standard Cells Timing Analysis at the 28 nm Technology and Below // 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering. 2017. pp. 525-529. DOI: 10.1109/EIConRus.2017.7910832

[8] Stempkovsky A., Glebov A., Gavrilov S. Calculation of Stress Probability for NBTI-Aware Timing Analysis // Proc. of ISQED, 2009, pp.714-718. DOI: 10.1109/ISQED.2009.4810381

[9] Sundareswaran S., Gavrilov S., Soloviev R., Panda R. A Timing Methodology Considering Within-Die Clock Skew Variations // IEEE International SOC conference, Sep 17-20, 2008, Newport Beach, CA. pp. 351-356.

[10] D. Bryan, “The ISCAS ‘85 Benchmark Circuits and Netlist Format”, North-Carolina State University, 1985.

[11] F. Brglez, D. Bryan, K. Kozminski, “Notes on the ISCAS ‘89 Benchmark Circuits”, North-Carolina State University, 1989.

[12] Collection of Digital Design Benchmarks, [online] Available at: http://ddd.fit.cvut.cz/prj/Benchmarks/ (accessed date: 30.04.2020).