УДК 621.3.049.771.14

Метод оптимизации быстродействия ПЛИС на микроархитектурном уровне с помощью механизма конвейеризации

Д.А. Железников, А.А. Лялинский

Институт проблем проектирования в микроэлектронике
Российской академии наук (ИППМ РАН),

zheleznikov_d@ippm.ru

***Аннотация* — Представлен метод оптимизации быстродействия комбинационных схем, проектируемых на ПЛИС с помощью механизма конвейеризации. В методе используется построение временного графа с помощью статического временного анализа и поиск критического пути с помощью алгоритма Киркпатрика. Метод основывается на итерационной вставке дополнительных регистров с целью уменьшения длины критических путей на временном графе и достижения заданной рабочей частоты. Метод ориентирован на практическое использование и может быть включен в общий маршрут проектирования цифровых схем на FPGA. Описан алгоритм использования метода и приведен практический пример.**

***Ключевые слова* — программируемые логические интегральные схемы, FPGA, комбинационная логика, автоматизация проектирования, конвейеризация.**

I. Введение

Несмотря на широкую популярность ПЛИС, которая является свидетельством уникального сочетания гибкости и простоты использования, их высокая адаптивность все же имеет свою цену. Как было показано в [1] схемы, реализованные на ПЛИС с архитектурой программируемых пользователем вентильных матриц (далее FPGA), оказываются в три-четыре раза медленней, чем их аналоги, реализованные в интегральных схемах специального назначения (ASIC). Для того чтобы свести к минимуму влияние потери производительности присущей ПЛИС, были разработаны специальные методы размещения и трассировки элементов, повышающие быстродействие проектируемого устройства [2], [3]. Однако в некоторых случаях их все же оказывается недостаточно для достижения необходимой рабочей частоты. В таких случаях может помочь использование механизма конвейеризации.

Быстродействие схем, реализуемых на ПЛИС, определяется их тактовой частотой – скоростью, с которой схема может проводить вычисления, а также временем выполнения, т.е. временем, необходимым для выполнения одного полного вычисления. Если предположить что одно вычисление завершается за один тактовый цикл, то тактовая частота схемы будет равна $F = 1/T_c$, где T_c - длительность одного цикла в наносекундах. Пусть N - число циклов, необходимое для завершения одного вычисления. Тогда время выполнения одного вычисления можно определить как $E = N \cdot T_c$. Повышение быстродействия схемы, как правило, означает увеличение тактовой частоты ее работы, даже за счет увеличения времени выполнения. Тем не менее, бывают случаи, когда время выполнения достаточно важно, и необходимо увеличить тактовую частоту так, чтобы время выполнения при этом возросло минимально.

Конвейеризация – это метод, который увеличивает количество тактовых циклов N, в течение которых выполняется одно вычисление, за счет добавления в схему дополнительных регистров (триггеров). Так как большее количество циклов становится доступным, то каждый отдельно взятый цикл становится короче, тем самым увеличивая тактовую частоту работы схемы. Время выполнения при этом может увеличиваться или снижаться в зависимости от того, насколько снижается длительность тактового цикла, которая, в свою очередь, ограничивается длиной критического пути, поэтому эффект от конвейеризации схемы можно усилить с помощью техники коррекции временных интервалов.

Техника коррекции временных интервалов (retiming technique [4]) – это хорошо известный метод, который сводит к минимуму длительность тактового цикла путем перераспределения операций вычисления, выполняемого схемой, оптимально по числу доступных тактовых циклов. Цель коррекции временных интервалов заключается в передвижении регистров по схеме таким образом, чтобы сохранить ее функциональное поведение и при этом сократить длину критического пути в схеме. На рис. 1 показан простой пример использования этого метода.

На показанном графе узлы представляют собой задержку срабатывания логики (рис. 1(а)), а входы и выходы проходят через обязательные фиксированные регистры, которые не могут быть перемещены. В данном примере начальный граф имеет длину критического пути (между внутренним регистром и

регистром на выходе) равную 5 нс. Если внутренний регистр просто переместить вперед, то длину критического пути можно уменьшить до 4 нс, однако в этом случае цикл обратной связи станет некорректен. Чтобы этого избежать помимо перемещения вперед внутреннего регистра необходимо добавить дополнительный регистр в цикл обратной связи (рис. 1(b)). Длина критического пути сократилась с 5 до 4 нс, но семантика работы схемы при этом не изменилась, и по-прежнему необходимо три тактовых цикла для полного прохождения данных от входа к выходу.

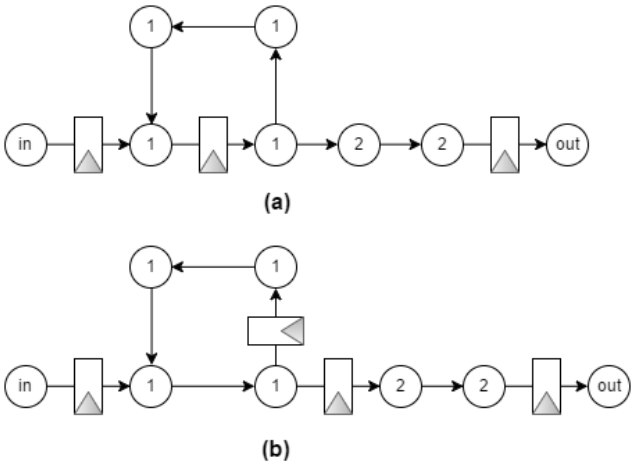

Рис. 1. Пример коррекции временных интервалов на графе. До проведения коррекции (a) и после (b)

Техника коррекции временных интервалов является довольно популярной оптимизацией, однако предоставляет лишь небольшие улучшения для ограниченного класса схем [5]. В частности, она не может улучшить быстродействие схем, чья длительность тактового цикла ограничена циклом обратной связи. Так, на представленном рисунке даже если удалить последний узел перед регистром на выходе, то длина критического пути никогда не сможет стать меньше 4 нс из-за цикла обратной связи. Не существует механизма, который может оптимизировать цикл обратной связи путем перемещения регистров: только дополнительные регистры могут ускорить такой тип схем.

Помимо оптимизации синхронных схем техника коррекции временных интервалов также может применяться и для выполнения конвейеризации комбинационных схем. Для этого необходимо поместить по одному регистру на всех входах или выходах схемы, а затем, применяя коррекцию временных интервалов, найти для них оптимальное положение. Этот метод позволяет добавлять по одному тактовому циклу для проведения вычислений, при этом длина критического пути в схеме будет оставаться минимальной. Однако, как уже упоминалось выше, такой подход ограничен наличием циклов обратной связи, и чем большее их количество содержит исходная схема, тем менее эффективной оказывается техника. Отдельно стоит отметить, что метод используется на фиксированном графе задержек и не учитывает их реальное изменение вследствие изменения нагрузочной и паразитных емкостей, а также длины межсоединений при перемещении регистров, и в результате имеет довольно низкую точность при применении на реальных схемах.

В данной статье предлагается новый подход к оптимизации быстродействия комбинационных схем с помощью выполнения конвейеризации для достижения заданной рабочей частоты.

II. Описание и постановка проблемы

В данном разделе рассматриваются особенности архитектурной модели, а также определены основные понятия. После этого дается формальное описание поставленной задачи.

A. Особенности архитектурной модели

Данная работа основывается на использовании некоторых архитектурных особенностей программируемых логических интегральных схем с архитектурой FPGA.

Обычно ПЛИС с архитектурой FPGA представляют собой совокупность логических элементов на основе функциональных генераторов LUT, объединенных с помощью большого количества программируемых межсоединений. На рис. 2(a) показана структура логического элемента, состоящего из 4-х входового функционального генератора LUT, D-триггера и выходного мультиплексора. Группы 4-х или 5-ти входных логических элементов в свою очередь объединяются в двумерные логические кластеры (рис. 2(b)).

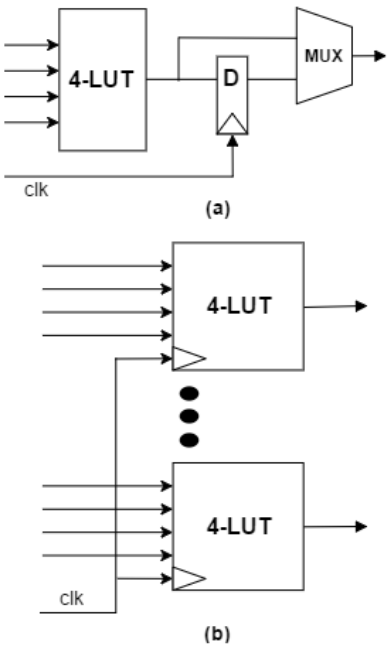

Рис. 2. Структура логического элемента LE (a) и кластера логических элементов LC (b)

Исключительной особенностью логических элементов такого типа является то, что с помощью программирования выходного мультиплексора можно перенаправить выходной сигнал через D-триггер, который, по сути, является регистром и таким образом фактически изменять схему на уровне регистровых передач во время проектирования.

Как упоминалось ранее, потребность в конвейеризации обычно возникает уже после этапов размещения и трассировки элементов микросхемы, когда становится ясным, что все прочие механизмы оптимизации уже задействованы, но достигнутая рабочая частота устройства меньше необходимой. В таких случаях единственный способ увеличить быстродействие – перенаправить выходной сигнал в логических элементах через D-триггер, тем самым конвейеризируя комбинационную схему. При этом для повышения точности вставки конвейеризирующих регистров, становится возможным использовать данные о задержках, полученные в результате проведения статического моделирования.

B. Основные определения и граф задержек

Входной список соединений представляется в виде взвешенного ориентированного (сетевого) графа $G = (V, E)$ с промаркированными вершинами $A(v), v \in V$. Множество вершин V состоит из трех групп узлов: S – множество входов схемы, T – множество выходов и R – множество промежуточных узлов. Во множество ребер $e \in E$ входят все межсоединения схемы, а маркировка $A(v)$ представляет собой задержку до вершины $v \in V$. Для объединения всех входных и выходных сигналов в схеме необходимо определить две дополнительных вершины: S^* (суперисточник) и T^* (суперсток). Вершину S^* необходимо соединить дугами с каждой вершиной из множества S всех входных сигналов схемы, а вершину T^* со всеми вершинами из множества T всех выходов схемы. В качестве примера в данной работе будет использоваться схема, реализующая простую логическую функцию $f = a \& \bar{b} + b \& c$ (рис. 3(a)). На рис. 3(b) показан пример построенного орграфа для этой схемы.

Как известно, общая задержка пути в комбинационных схемах состоит из задержки срабатывания логики элементов (их транзисторов), задержек распространения сигнала в цепях между элементами, и времени рассогласования синхросигнала. Для определения задержек и построения исходного временного графа после этапов размещения и трассировки элементов проводится статический временной анализ с проведением экстракции паразитных сопротивлений, емкостей и индуктивностей.

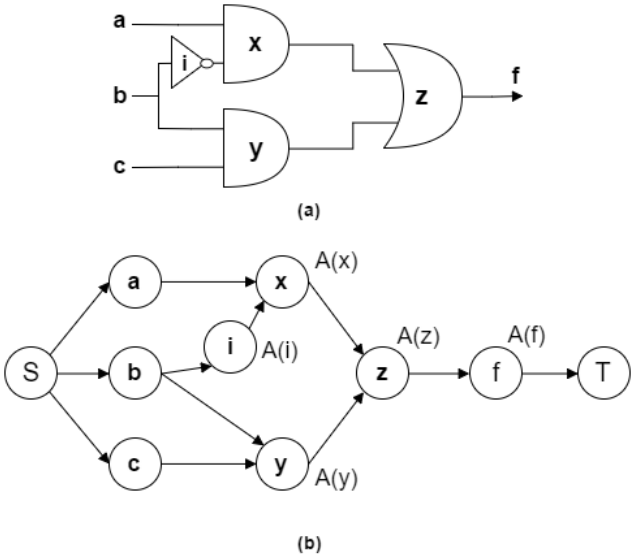

Рис. 3. Схема, использованная в качестве примера (a), и ее орграф (b)

В данной работе не учитывается время рассогласования синхросигнала, то есть разность времени прихода синхросигнала на триггеры, так как трассировка управляющих сигналов происходит уже после добавления конвейеризирующих регистров. При таком подходе для корректирования времени распространения синхросигнала могут быть использованы дополнительные буферы.

Теперь можно дать формальное описание поставленной задачи.

Постановка проблемы: *С учетом архитектурных особенностей ПЛИС с архитектурой FPGA для размещенной и трассированной исходной комбинационной схемы добавить необходимое количество регистров и определить их расположение таким образом, чтобы рабочая частота полученной схемы достигла заданной.*

III. Описание и пример использования предложенного метода

В этом разделе описывается алгоритм конвейеризации исходной комбинационной схемы с целью достижения заданной рабочей частоты, а также приводится пример использования алгоритма.

В предыдущем разделе была показана простая схема, которая будет использоваться далее в качестве примера. В первую очередь для использования метода необходимо провести статический временной анализ исходной схемы и определить задержки срабатывания логики и распространения сигналов (рис. 4(a)). Далее необходимо преобразовать полученные задержки в исходный временной граф (рис. 4(b)). При этом время распространения сигналов от входов схемы до вершины суперстока S^* и от выходов схемы до суперистока T^* полагается равным нулю.

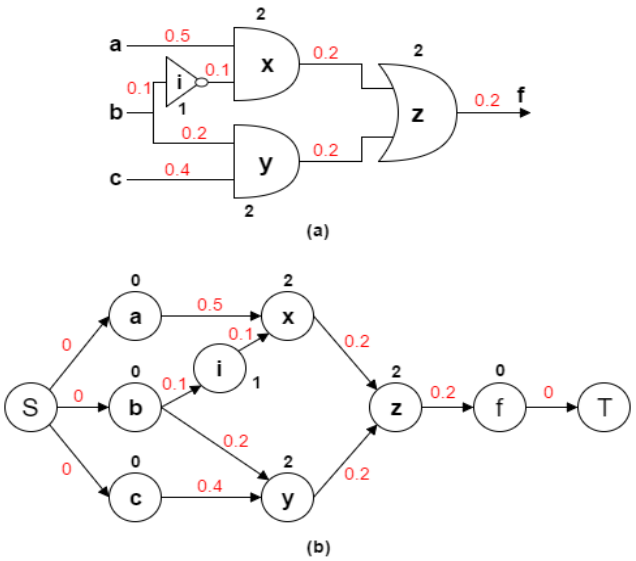

Рис. 4. Задержки срабатывания логики и распространения сигналов схемы из примера (a) и её исходный временной граф (b)

Как упоминалось ранее, быстродействие микросхемы определяется длительностью тактового цикла, который в свою очередь ограничивается длиной критического пути. Это условие можно использовать и в обратную сторону. Так, если положить, что $F_{зад}$ – заданная рабочая частота, то для ее достижения длительность тактового цикла не должна превышать $T_{кр} = 1/F_{зад}$, где $T_{кр}$ – время прохода сигнала по критическому пути. Таким образом, основной целью оптимизации быстродействия с помощью механизма конвейеризации является добавление регистров вдоль всех критических путей в схеме таким образом, чтобы время прохода сигнала вдоль получившихся частей было меньше или равно $T_{кр}$.

Для поиска критического пути на временном графе в данной работе используется классический алгоритм Киркпатрика [6]. Алгоритм обходит исходный временной орграф, один раз посещая каждый узел и маркируя все узлы, смежные с данным, максимальным временем прихода сигнала. Во время работы алгоритма сохраняются указатели на ребра критического пути, и по его завершению для построения критического пути необходимо просто последовательно обойти орграф по указателям в обратном направлении из вершины суперстока T^*.

В случае схемы из примера на первой итерации алгоритма критический путь составил $T_{доc} = 5.6$ нс (рис. 5(a)). Достигнутая тактовая частота в данном случае будет составлять $F_{доc} = 1/T_{доc} = (1/5.6)*10^3 = 176$ МГц. Допустим, что заданная необходимая рабочая частота равна $F_{зад} = 300$ МГц, в таком случае длина критического пути в схеме не должна превышать $T_{кр} \le 1/F_{зад} = (1*10^3)/300 = 3.33 \approx 3.3$ нс. В этом случае достаточно разбить найденный критический путь с помощью вставки регистра всего лишь на две части между узлами x и z.

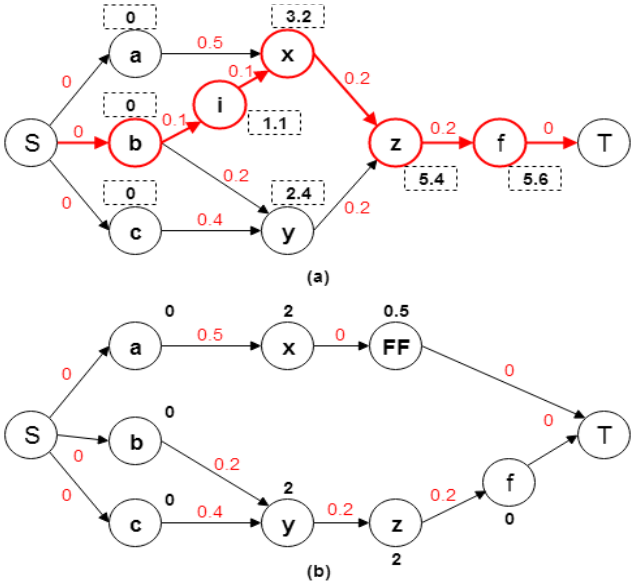

Рис. 5. Критический путь на первой итерации алгоритма (a) и временной граф после проведения редукции (b)

Таким образом, следующим шагом в работе предложенного алгоритма будет определение минимального числа тактовых циклов N, на которые необходимо разбить критический путь и определение места вставки дополнительных регистров. Нужно отметить, что число циклов N, определенное для первого найденного критического пути, далее не изменяется и применяется на всех последующих итерациях алгоритма для сохранения общего количества тактовых циклов. Если очередной критический путь невозможно разбить на необходимое количество частей, алгоритм начинает свою работу с самого начала, при этом увеличив количество тактовых циклов на 1. Если после перезапуска первый критический путь невозможно разбить на $N+1$ количество частей, то это означает, что исходная схема не сможет достичь заданной рабочей частоты и работа алгоритма прекращается.

Далее для каждого добавленного регистра необходимо создать по одному дополнительному узлу в множестве входных узлов S и в множестве выходных узлов T и соединить их дугами с вершиной суперистоком S^* и вершиной суперстоком T^*, соответственно. Для каждого созданного узла необходимо запомнить число M – позицию регистра на конвейере, т.е. тактовый цикл, на котором данные проходят через добавленный регистр.

Для сохранения общего количества тактовых циклов, начиная со второй итерации алгоритма в случае, если выходным узлом критического пути является регистр, то количество частей, на которые его необходимо разбить, равно M. Если же регистр

является входным узлом, то критический путь разбивается на $N-M$ частей. В случае если регистры не входят в состав найденного критического пути, его необходимо разбить на N частей.

Завершающим шагом на каждой итерации алгоритма является редукция временного графа. Для этого необходимо разрезать дугу временного графа в месте добавления регистра и соединить оставшиеся части дуги с вершинами регистров, добавленными во множества входных и выходных узлов (рис. 5(b)). На рисунке узел с маркировкой FF (Flip Flop) представляет собой вершину добавленного регистра. После этого вдоль критического пути можно удалить вершины и дуги, которые не могут быть задействованы для построения новых критических путей. Так, на показанном примере удалению подверглись вершина инвертора i с прилегающими дугами и вновь созданная вершина регистра из множества входных узлов S.

В данной работе время прохождения сигнала через регистр (включая его время установки) определено как 0.5 нс. Время распространения сигнала от функциональной таблицы LUT до регистра в случае перепрограммирования выходного мультиплексора оказывается пренебрежимо мало, поэтому в показанном примере задержка дуг до выходных регистров полагается равной нулю, в то время как задержка дуг после входных регистров сохраняется полностью.

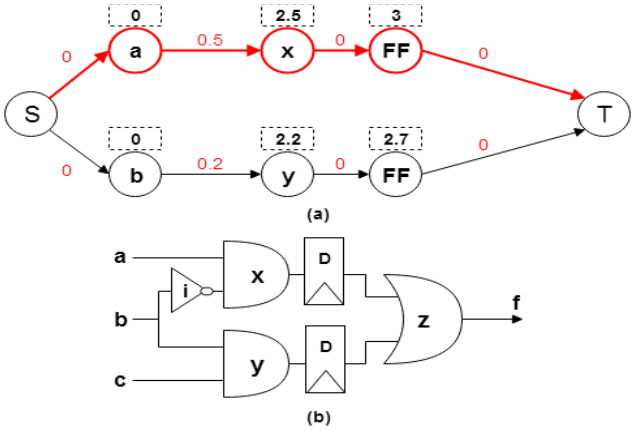

Рис. 6. Временной граф на 3-й итерации алгоритма (a) и схема после завершения его работы (b)

На рис. 6(a) показан временной граф и критический путь в начале третьей итерации алгоритма. Длина критического пути составляет $T_{кр}=3$ нс и выходным узлом для него является регистр. Этот регистр является единственным в построенном конвейере и так как длина критического пути не превышает заданных $T_{зад}=3.3$ нс, то добавлять новые регистры нет необходимости. Алгоритм переходит сразу к проведению редукции временного графа, в результате чего остается всего один критический путь, который будет обработан похожим образом в течение последней итерации. По завершению алгоритма необходимо восстановить позиции добавленных регистров, и в результате схема приобретает вид, показанный на рис. 6(b).

С учетом всего вышеизложенного алгоритм конвейеризации комбинационной схемы выглядит следующим образом.

Алгоритм 1

1. Провести статический временной анализ схемы для определения задержек.
2. Построить исходный временной граф схемы.
3. Найти критический путь в построенном временном графе.
4. Для первого найденного критического пути определить минимальное количество частей N, на которое его необходимо разбить для достижения заданной рабочей частоты.
5. Разбить критический путь на заданное количество частей N и определить место вставки дополнительных регистров.
6. Для каждого добавленного регистра создать пару дополнительных вершин во множествах S и T входных и выходных узлов, для каждой созданной вершины запомнить число M – позицию регистра на конвейере, т.е. тактовый цикл, на котором данные проходят через добавленный регистр.
7. Если выходным узлом в критическом пути является регистр, разбить критический путь на количество частей M. Если же регистр является входным узлом, то критический путь разбивается на $N-M$ количество частей.
8. Провести редукцию временного графа вдоль критического пути. Удалить вершины и дуги, которые не могут быть задействованы для построения новых критических путей.
9. Если критический путь не удается разбить на N частей, то увеличить N на 1 и вернуться в п. 2.
10. Повторять пункты 3-8 до тех пор, пока множество всех входных узлов схемы S не станет пустым в результате редукции временного графа.
11. Восстановить позиции добавленных регистров и перестроить исходную схему.

IV. Заключение

Представленный метод ориентирован на практическое использование и может быть включен в общий маршрут проектирования цифровых схем на ПЛИС с архитектурой FPGA. В следующих работах планируется добавление в алгоритм конвейеризации циклов обратной связи и более точная оценка влияния вставки дополнительных регистров на задержку распространения сигнала. Также в алгоритм планируется включить оптимизацию быстродействия после проведения конвейеризации за счет учета ложных путей и удаления избыточных регистров.

ПОДДЕРЖКА

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 15-07-01778.

ЛИТЕРАТУРА

[1] Kuon I., Rose J. Measuring the Gap between FPGAs and ASICs // IEEE Trans. on Computer-Aided Design. 2007. Vol. 26. №2. P. 203–215.
[2] Marquardt A., Betz V., Rose J. Timing-Driven Placement for FPGAs // In Proc. of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays. 2000. P. 203–213.
[3] McMurchie L., Ebeling C. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs // In Proc. of the Third International ACM Symposium. 1995. P. 111–117.
[4] Leiserson C.E., Rose F., Saxe J.B. Optimizing Synchronous Circuitry by Retiming // In Proc. of the 3rd Caltech Conference on VLSI. 1983.
[5] Ebeling C., Lockyear B. On the Performance of Level-Clocked Circuits // In Proc. of the Sixteenth Conference of Advanced Research in VLSI. 1995. P. 342–356.
[6] Kirkpatrick T.I., Clark N.R. PERT as an aid to logic design // IBM Journal of Research and Development Journal. 1966. Vol. 10. №2. P. 135–141.
[7] Scott Hauck, Andr´e DeHon, Reconfigurable Computing: The theory and practice of FPGA-based computation / San Francisco, CA, Morgan Kaufmann Publishers Inc., 2007. P. 944.

Timing optimization method for FPGA at the microarchitecture level using the pipelining mechanism

D.A. Zheleznikov, A.A. Lyalinsky

Institute for Design Problems in Microelectronics
of Russian Academy of Sciences (IPPM RAS),

zheleznikov_d@ippm.ru

***Keywords* — Field-Programmable Gate Array (FPGA), combinational logic, computer-aided design, pipelining.**

ABSTRACT

In the article the method of timing optimization using pipelining mechanism for combinational circuits designed on FPGA, is presented. Pipelining method using retiming technique for combinational circuits is known [4]. However, this approach is limited by the presence of feedback loops - the more of them in the initial circuit, the less the efficiency of the technique. It should be noted that the method is used for a fixed time graph and it does not consider its changing due to variations of load and parasitic capacitances, as well as the length of the wires because of register movement. Thus, it has a relatively low accuracy on the real circuits.

This paper presents a method of combinational circuits pipelining based on iterative insertion of additional registers to reduce the length of critical paths in the time graph and achieve the desired operating frequency. In the first step, the time graph using static timing analysis is constructed. After that, critical path searching by classical algorithm of Kirkpatrick is performed [6]. The method is focused on the practical use and can be included in the general FPGA design flow. In this article the description of the method usage is given, and a simple practical example is shown.

REFERENCES

[1] Kuon I., Rose J. Measuring the Gap between FPGAs and ASICs // IEEE Trans. on Computer-Aided Design. 2007. Vol. 26. No. 2. P. 203–215.
[2] Marquardt A., Betz V., Rose J. Timing-Driven Placement for FPGAs // In Proc. of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays. 2000. P. 203–213.
[3] McMurchie L., Ebeling C. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs // In Proc. of the Third International ACM Symposium. 1995. P. 111–117.
[4] Leiserson C.E., Rose F., Saxe J.B. Optimizing Synchronous Circuitry by Retiming // In Proc. of the 3rd Caltech Conference on VLSI. 1983.
[5] Ebeling C., Lockyear B. On the Performance of Level-Clocked Circuits // In Proc. of the Sixteenth Conference of Advanced Research in VLSI. 1995. P. 342–356.
[6] Kirkpatrick T.I., Clark N.R. PERT as an aid to logic design // IBM Journal of Research and Development Journal. 1966. Vol. 10. No. 2. P. 135–141.
[7] Scott Hauck, Andr´e DeHon, Reconfigurable Computing: The theory and practice of FPGA-based computation / San Francisco, CA, Morgan Kaufmann Publishers Inc., 2007. P. 944.