УДК 621.3.049.771.14:621.3.062 DOI: 10.31114/2078-7707-2020-1-26-32

Применение SAT подхода к трассировке блоков коммутации для реконфигурируемых систем на кристалле

Д.В. Жуков, Д.А. Железников, М.А. Заплетина,

Институт проблем проектирования в микроэлектронике РАН, Зеленоград, Москва,

zhukov_d@ippm.ru, zheleznikov_d@ippm.ru, zapletina_m@ippm.ru

***Аннотация* — В статье предлагается метод трассировки коммутационных блоков реконфигурируемых систем на кристалле островного типа на основе SAT подхода. Целью процедуры трассировки является достижение полной трассируемости списка межсоединений при стремлении к их минимальной длине и величине задержки. При SAT подходе для каждого блока коммутации создаются два списка ограничений. Базовые ограничения формируются, исходя из схемотехнической конфигурации коммутационного блока. Конфликтные ограничения задаются по мере итерационной трассировки списка цепей, назначенных на блок коммутации по завершению глобального этапа трассировки. Впоследствии ограничения обоих видов транслируются в булеву систему уравнений, передаваемую в SAT решатель. Задача детальной трассировки была решена для коммутационных блоков двух типов связности. На основании вычислительных экспериментов показано, что предложенный алгоритм, реализованный на языке программирования C, отвечает требованиям полноты трассируемости и способствует минимизации времени, затрачиваемого на процедуру трассировки.**

Ключевые слова: реконфигурируемая система на кристалле, островная архитектура, смешанный граф трассировки, детальная трассировка, SAT.

I. Введение

Маршрут топологического синтеза реконфигурируемых систем на кристалле (РСнК) состоит из трех главных этапов: декомпозиции, размещения логических элементов проектной схемы на кристалле и трассировки списка проектных цепей. В условиях оптимального размещения решающее влияние на быстродействие и корректность работы итоговой проектной схемы оказывает выбор метода трассировки [1].

Существуют два важных критерия оценки эффективности методов и алгоритмов трассировки. Обязательное условие корректности их работы представляет собой полнота трассируемости списка проектных цепей. Уровень быстродействия конечной интегральной схемы является вторым важным аспектом и определяется по большей части двумя факторами: степенью минимизации суммарной длины цепей и величиной задержек на критических путях. Достичь оптимальных результатов по этим критериям возможно лишь благодаря эффективному использованию коммутационных ресурсов кристалла с учетом всех его архитектурных особенностей, преимуществ и недостатков.

Архитектура островного типа часто используется в учебных и коммерческих ПЛИС [2]. Рассматриваемая в данной работе РСнК также построена по принципам этой архитектуры. Программируемые логические элементы расположены на кристалле в узлах прямоугольной сетки и могут быть соединены между собой с помощью сети коммутации. Она состоит из многочисленных дорожек металлизации и программируемых коммутационных (трассировочных) элементов, а также их групп, объединенных в составе блоков коммутации. По периметру программируемой логики расположены ячейки ввода-вывода.

РСнК имеет ряд особенностей, связанных с архитектурой коммутационных ресурсов. Помимо стандартных транзисторных МОП-переключателей в нее могут входить мультиплексоры, инверторы, усиленные буферы и др. Смешанная графовая модель для плоской трассировки обсуждалась ранее в [3]. Предлагаемый подход на основе задачи выполнимости булевых формул (Boolean satisfiability problem, SAT) исключает влияние этих особенностей на детальную трассировку, а вопрос специальной математической модели для глобальной маршрутизации выходит за рамки данной работы.

SAT задача – это задача определения для заданной булевой формулы, существует ли такая комбинация переменных, при которой формула обращается в единицу. Если такой набор существует, то формула является выполнимой [4].

Хотя задача SAT является NP-полной и в настоящее время известны универсальные алгоритмы её решения только с экспоненциальной сложностью, в течение последних двух десятилетий были разработаны эффективные узкоспециальные алгоритмы на базе SAT для решения многочисленных задач автоматизации проектирования. Среди них – канальная трассировка [5], размещение элементов ПЛИС [6], логическая оптимизация [7], автоматическое формирование тестовых шаблонов [8], комбинаторная оптимизация при физическом проектировании логических блоков [9]. Задача детальной трассировки блоков коммутации ранее не рассматривалась в контексте SAT проблемы.

Рассматриваемая в данной работе РСнК подразумевает наличие двух типов коммутационных блоков, схемы которых представлены на рис. 1а, б. Их главным различием является количество терминалов на противоположной стороне (или сторонах) блока, с которыми может быть соединен каждый терминал путем программирования внутренних трассировочных элементов. Это количество отражено в параметре связности *flexibility* [10]. Он равен 1 для блока *SB* и 7 - для *CB*. В блоках коммутации терминалы располагаются на четырех границах блока для модели *SB* и на двух - для модели *CB*. Для модели *SB* доступна возможность соединения выбранного терминала на одной границе блока с одним терминалом того же разряда на каждой из трех других границ. Терминал коммутационного блока модели *CB* может быть скоммутирован со всеми остальными терминалами, включая находящиеся с ним на одной границе. Также блоки коммутации имеют параметр ширины канала *W*, величина которого определяет количество терминалов на каждой границе блока.

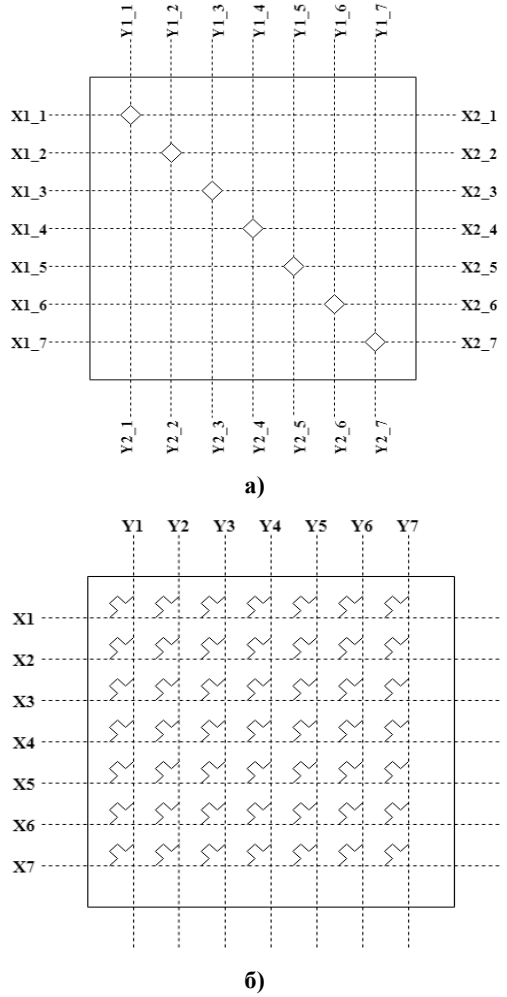

Рис. 1. Блоки коммутации: а) модели *SB*; б) модели *CB*

Дальнейшее содержание работы организовано следующим образом. В разделе II рассматривается проблема детальной трассировки блоков коммутации и определяется задача трассировки в терминах SAT подхода. В разделе III приведен разработанный алгоритм, пояснения к нему и пример работы. Раздел IV демонстрирует результаты численных экспериментов. Раздел V содержит заключительные выводы.

II. Формулировка задачи детальной трассировки

После этапа глобальной трассировки каждому блоку коммутации установлено в соответствие множество из n цепей $\mathbf{E} = \{e_1, e_2, \ldots, e_n\}$, в котором каждой цепи $e_i \in \mathbf{E}$ соответствует множество терминалов $\mathbf{T} = \{t_0, t_1, \ldots, t_k\}$, где терминал $t_0 \in \mathbf{T}$ является источником, а подмножество $\{t_1, .., t_k\} \subset \mathbf{T}$ приемниками сигнала (рис. 2).

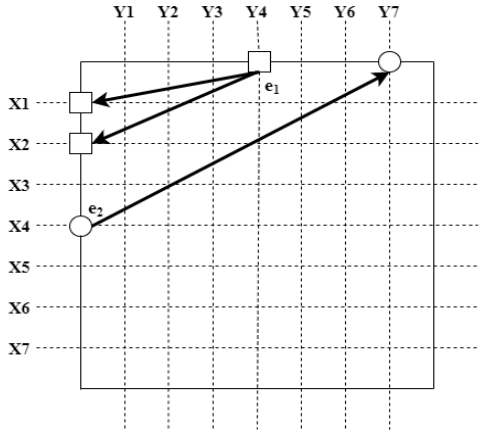

Рис. 2. Входные данные процедуры трассировки блоков коммутации: e1, e2 - проектные цепи, стрелка указывает направление к приемникам сигнала

Таким образом, множество терминалов $\mathbf{T} \subset e_i$ представляет собой цепь, которая в результате должна быть трассирована путем открытия соответствующих коммутационных элементов. Конфликт возникает, когда магистраль или коммутационный элемент оказывается перегруженным несколькими цепями. Задача трассировки в этом случае заключается в том, чтобы найти *бесконфликтное* решение для множества всех цепей в коммутационном блоке.

A. SAT трассировка

Решение SAT задачи заключается в поиске комбинации булевых переменных, которая будет удовлетворять набору условий, обычно представленному в конъюнктивной нормальной форме (КНФ, англ. CNF). При таком подходе SAT трассировка заключается в описании архитектуры коммутационного блока и дополнительных ограничений набором булевых уравнений таким образом, что любой набор значений этих переменных, удовлетворяющий всем ограничениям, обеспечит корректную трассировку соответствующего множества проектных цепей. Таким образом, задача трассировки представляется в форме эквивалентной SAT задачи, которая затем обрабатывается SAT решателем, позволяющим найти решение для системы булевых уравнений.

Обычно в качестве входных данных SAT решатель [11] принимает данные в формате DIMACS [12], стандартном представлении булевых формул в КНФ. Они записываются в текстовом файле, имеющем следующую структуру. Первая строка начинается с зарезервированных слов «p» (от англ. problem) и «cnf» (КНФ) и объявляет количество переменных (литералов) и дизъюнктов, которые используются в системе:

$$p \quad cnf \quad numVar \quad numClause,$$

где *numVar* – количество переменных, *numClause* – количество дизъюнктов. Каждая из последующих строк описывает один дизъюнкт: литерал обозначается присвоенным ему номером, а операция отрицания добавляет перед номером литерала знак «–». Последний символ в строке – «0», что обозначает конец дизъюнкта. Таким образом, для выражения $(A \vee \overline{B}) \wedge (B \vee C \vee D)$ условия в формате DIMACS запишутся следующим образом:

p cnf 4 2
1 -2 0
2 3 4 0,

где 1, 2, 3, 4 - номера литералов *A*, *B*, *C* и *D*, соответственно.

При описании коммутационного блока в КНФ каждому терминалу этого блока присваиваются уникальные индексы, с помощью которых задается назначение цепей, полученное на этапе глобальной трассировки.

III. Алгоритм SAT трассировки коммутационного блока

Рассмотрим алгоритм для трассировки одного блока коммутации. Он состоит из нескольких этапов. Алгоритм перебирает по очереди множество присвоенных цепей $\mathbf{E} = \{e_1, e_2, \ldots, e_n\}$, и для каждого $e_i \in \mathbf{E}$ выполняет:

1. создание системы уравнений:
   a. добавление базовых ограничений блока;
   b. добавление ранее найденных решений;
   c. добавление терминалов цепи $\{t_0, t_1, \ldots, t_k\}$ из множества $\mathbf{T}$;
   d. добавление исключений.
2. Передача составленной системы в SAT решатель.
   a. Если трассировка блока *CB* невозможна, выполняется:
      i. проверка на наличие свободных магистралей;
      ii. проверка на нужные магистрали;
      iii. переработка системы ограничений;
      iv. переход обратно к трассировке первой цепи $e_1 \in \mathbf{E}$.

Примечание: если невозможно трассировать блок *SB*, т.е. решение отсутствует, выполняется переход к шагу 4.

3. Переход к e_{i+1} и шагу 1 до тех пор, пока $i \le n$.
4. Трассировка блока коммутации закончена.

A. *Пояснения к работе алгоритма*

Основным этапом детальной трассировки с применением SAT подхода является создание системы уравнений. Её принцип описывается формулой трассировки:

$$\text{Routable}(x) = \text{conn}(x) \wedge \text{live} \wedge \text{exc}(b),$$

где $\text{conn}(x)$ – это терминалы цепи, которые используются на текущей итерации, $\wedge$ - операция конъюнкции, live – уравнения которые задают границы и внутренние магистрали блока, использующиеся в данной итерации, $\text{exc}(b)$ – система булевых уравнений, составленных из ограничений на коммутационные ресурсы блока.

Для описания блока *CB* необходимо ввести ограничения на использование нескольких магистралей на одной границе. Исключения составляют цепи, у которых входной и выходной терминал находятся на одной границе, в этом случае ограничивающие условия для этой пары удаляются. Для описания блока *SB* требуется намного больше условий, поскольку для него параметр *flexibility* = 1, а для блока *CB flexibility* = 7. Необходимо ограничить все маршруты сигнала для каждого терминала кроме тех, которые обеспечивают доступные (легальные) пути.

В систему уравнений входят следующие элементы: начальные ограничения блока; ограничения исключительности, вызванные ранее разведенными цепями; конфликтные ограничения; набор терминалов и границ для текущей цепи.

Начальные ограничения блока коммутации формируются исходя из его схемотехники. Ограничения исключительности гарантируют, что различные цепи с пересекающимися вертикальными или горизонтальными участками в одном блоке будут назначены разным магистралям. Изначально для *CB* блока с *W*=4, уравнение будет следующим:

$$\text{exc}(b) = (\overline{x}_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3) \wedge$$
$$\wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_4) \wedge (\overline{y}_1 \vee \overline{y}_2) \wedge (\overline{y}_1 \vee \overline{y}_3) \wedge (\overline{y}_1 \vee \overline{y}_4) \wedge$$
$$\wedge (\overline{y}_2 \vee \overline{y}_3) \wedge (\overline{y}_2 \vee \overline{y}_4) \wedge (\overline{y}_3 \vee \overline{y}_4),$$

где $x_1, x_2, ..., x_n,\ y_1, y_2, ..., y_n$ - одноименные терминалы коммутационного блока.

Если текущая итерация не первая, и одна из назначенных на блок цепей уже имеет трассировочное решение, необходимо добавить его в систему уравнений для того, чтобы избежать конфликта. Для этого на каждой итерации формируются дополнительные ограничения.

Если в *CB* блоке не трассируется одна из цепей, то необходимо найти цепь, в которой содержится конфликтующее решение, и добавить его в

конфликтные исключения. После этого следует начать трассировку заново. Этот пункт применяется только для *CB* блоков, поскольку из-за различий архитектуры блоков *CB* и *SB* у *CB* блока можно найти несколько решений для одной цепи, в то время как в *SB* коммутационные возможности не позволяют этого сделать. Если трассировка хотя бы одной цепи невозможна в *SB* блоке, то полная трассировка всей схемы также невозможна. Для исправления этой ситуации необходимо повторное проведение глобальной трассировки или размещения элементов схемы.

Чтобы переработать ограничения системы уравнений, необходимо определить проблему, из-за которой трассировка данной цепи невозможна. Она может быть двух видов. Первый – отсутствие свободных магистралей. В этом случае производится проверка наличия свободных магистралей в блоке. Если обнаруживается их нехватка хотя бы по одному из ортогональных направлений, дальнейший поиск решений и, соответственно, трассировка цепи становятся невозможны. Второй вид проблемы возникает, когда нужная магистраль уже занята другой парой терминалов. В этом случае алгоритм ищет магистраль, которая мешает нахождению решения, и цепь, в которой эта магистраль задействована. После этого генерируются ограничения на эту магистраль, и попытка трассировки блока повторяется. Если развести цепь вновь не удается, работа алгоритма на этом заканчивается.

Если трассировка цепи прошла успешно, алгоритм переходит к трассировке следующего межсоединения. Когда все цепи коммутационного блока успешно разведены, программа детальной трассировки завершается или переходит к следующему блоку коммутации.

B. Пример работы алгоритма

Ниже представлен пример работы алгоритма для *CB* блока для следующих входных значений (рис. 3).

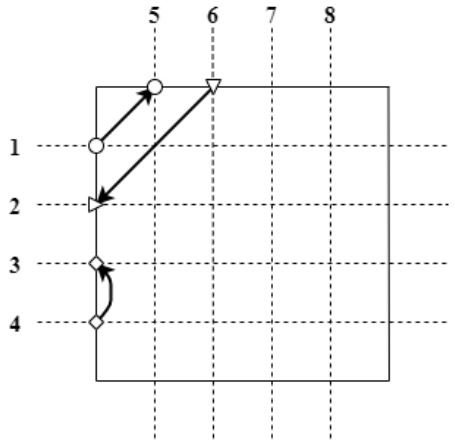

Рис. 3. Начальные данные блока коммутации

1) Трассируется цепь 1 5. Поскольку это первая цепь, конфликты исключены. Терминалы находятся на разных границах блока, поэтому занимаются магистрали, соответствующие входным данным (рис. 4).

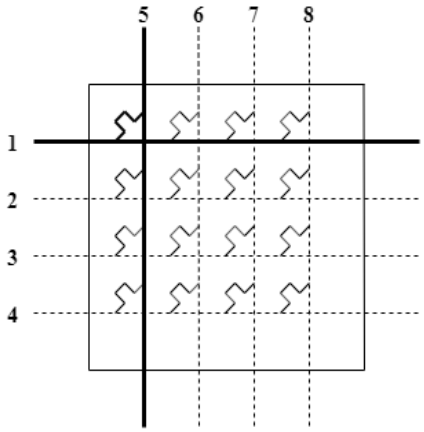

Рис. 4. Состояние блока после первой итерации, жирными линиями выделены магистрали, занятые первой цепью

2) Трассируется цепь 6 2. Первая разведенная цепь не мешает текущей, и они также находятся на разных границах блока. Алгоритм выбирает для решения соответствующие магистрали (рис. 5).

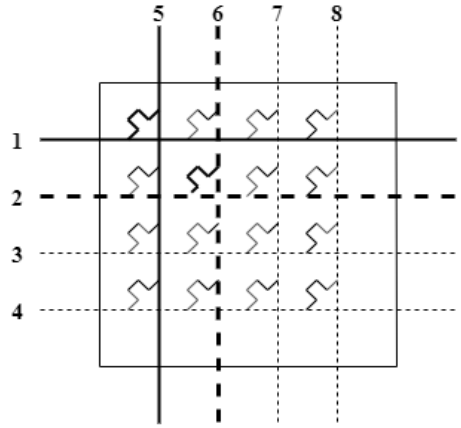

Рис. 5. Состояние блока после второй итерации: жирным выделены магистрали, занятые первой цепью, а пунктиром - занятые второй

3) Трассируется цепь 4 3. Поскольку её терминалы находятся на одной границе, для нее необходимо использовать три магистрали. При создании системы уравнений в неё добавляются имеющиеся решения в виде дополнительных ограничений. В данном примере не имеет значения, какая дополнительная магистраль будет использоваться – 7 или 8 (рис. 6). Это последняя итерация работы алгоритма, так как список цепей подошел к концу. Информация о задействованных коммутационных элементах блока коммутации может быть передана далее в процедуру формирования прошивки программируемой логики.

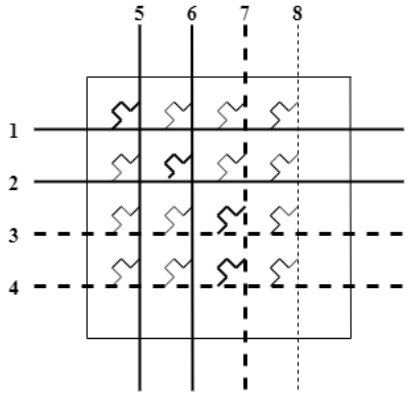

Рис. 6. Состояние блока коммутации после третьей итерации: сплошной жирной линией выделены магистрали, занятые ранее разведенными цепями

IV. Результаты вычислительных экспериментов

Численные эксперименты для разработанного алгоритма были выполнены на наборе тестовых схем ISCAS'89 и LGSynth'89, выборка из которых представлена в табл. 1. Расчеты были произведены на персональном компьютере с процессором Intel Core i5-7400 с тактовой частотой 3.0 ГГц и 8Гб ОЗУ.

Таблица 1

Количественные характеристики тестовых проектных схем

Имя схемы	**Количество проектных цепей**	**Количество логических элементов**	**Количество коммут. блоков**
c17	9	4	19
c432	117	81	303
c499	168	127	587
c880	210	150	628
c1355	162	121	597
c1908	181	148	623
c3540	554	504	1477
c6288	765	733	2072
misex3	584	570	1880
x4_syn	260	166	882

Данные табл. 2 свидетельствуют о том, что максимальное ускорение времени трассировки в большинстве случаев наблюдается у схем, которые имеют наименьшее количество проектных цепей и задействованных при глобальной трассировке коммутационных блоков. Кроме того, наилучшие результаты по времени были получены для проектных схем с наименьшим числом сильно разветвленных цепей (от 32 приемников).

Таблица 2

Сводная временная статистика для плоской и иерархической трассировки

Имя схемы	**Плоская трасси-ровка, с**	**Иерархическая трассировка, с**		**Разница в n, раз**
		Глобаль-ная	**Деталь-ная**	
c17	2.867	0.285	0.013	9.62
c432	51.737	10.421	0.482	4.74
c499	115.628	26.464	0.968	4.21
c880	99.658	24.451	0.985	3.91
c1355	99.812	22.051	1.02	4.32
c1908	98.908	21.881	1.121	4.29
c3540	149.093	37.4	4.27	3.57
c6288	102.201	23.12	4.118	3.75
misex3	578.344	236.158	5.011	2.39
x4_syn	207.58	43.075	2.209	4.58

Время трассировки коммутационного блока зависит от загруженности блока. В табл. 3 и 4 приведено среднее время работы алгоритма в зависимости от количества задействованных цепей в коммутационных блоках обоих типов.

Таблица 3

Зависимость времени трассировки от количества цепей в SB блоке

Количество цепей	**Время, мс**	**Количество цепей**	**Время, мс**
1	1.255	8	9.602
2	2.475	9	10.402
3	3.945	10	11.281
4	4.915	11	12.463
5	5.948	12	13.500
6	7.045	13	15.240
7	8.097	14	16.590

Таблица 4

Зависимость времени трассировки от количества цепей в CB блоке

Количество цепей	**Время, мс**
1	0.2650
2	0.5275
3	0.7575
4	0.9250
5	1.1000
6	1.2852
7	1.2950

Графики на рис. 7, построенные по данным табл. 3 и 4, свидетельствуют о том, что вычислительная сложность разработанного алгоритма близка к линейной. График времени трассировки *Tm* блока модели *SB* в зависимости от числа назначенных цепей *n* может быть аппроксимирован прямой $Tm = 1.1083n + 0.3556$ со среднеквадратичным отклонением 2.7%. Аналогичная зависимость для модели *CB* аппроксимируется прямой $Tm = 0.157n + 0.2315$ со среднеквадратичным отклонением 9%.

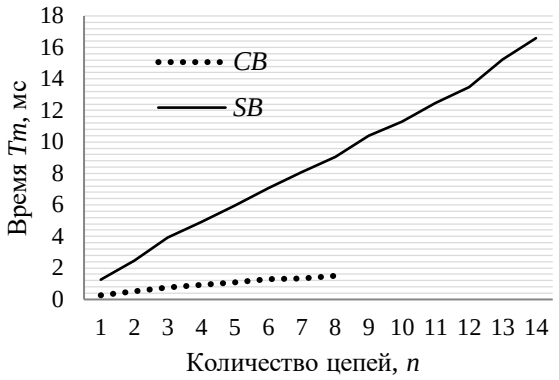

Рис. 7. Зависимость времени трассировки *Tm* в миллисекундах от количества цепей *n* в блоках типа *CB* и *SB*

V. Заключение

Предложенное в работе решение задачи детальной трассировки блоков коммутации с применением итерационного алгоритма на основе SAT подхода доказало свою эффективность. Статистические данные, собранные по результатам вычислительных экспериментов, свидетельствуют о линейной сложности разработанного алгоритма. Представлена универсальная методика формирования задачи детальной трассировки блоков коммутации в терминах задачи SAT, подробно рассмотренная на примере двух архитектур коммутационных блоков РСнК.

Важное преимущество представленной SAT трассировки над иными подходами заключается в том, что если решение построенной системы для трассировки коммутационного блока не найдено, то это является доказательством его отсутствия.

Литература

[1] Romanov A.Yu., Ivannikov A.D., Romanova I.I. Simulation and synthesis of networks-on-chip by using NoCSimp HDL library // 2016 IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO 2016 Conference Proceedings. 7493072. PP. 300-303.

[2] Gort M., Anderson J. H. Combined Architecture/ Algorithm Approach to Fast FPGA Routing // IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2013. Vol. 21. No. 6. PP. 1067-1079G.

[3] 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, 2018. Vol. 47. No. 7. PP. 516–521.

[4] H. Zhang. SATO: An efficient propositional prover. In Proceedings of the International Conference on Automated Deduction, p. 272-275, July 1997. 72

[5] Nam G.-J., Sakallah K. A., Rutenbar R.A. A comparative study of two Boolean formulations of FPGA detailed routing constraints // IEEE Transactions on Computers, 2004. Vol. 53. No. 6. PP. 688-696.

[6] Xiu-qin W., Yang Y. New approach of exploiting symmetry in SAT-based Boolean matching for FPGA technology mapping // IEEE International Conference on Vehicular Electronics and Safety, 2013. PP. 282-285.

[7] Mishchenko A., Brayton R., Roland J., Jang J. S.SAT-based logic optimization and resynthesis // Proc. IWLS '07, 2007. PP.358-364.

[8] Larrabee T., Test pattern generation using Boolean satisfiability // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1992. Vol. 11. No. 1. PP. 4-15.

[9] Taylor B., Pileggi L., Exact Combinatorial Optimization Methods for Physical Design of Regular Logic Bricks // 44th ACM/IEEE Design Automation Conference, San Diego, 2007. PP. 344-349.

[10] Prado D. F. G. Tutorial on FPGA Routing // Electronica, 2006. No. 17. PP. 23-33.

[11] Schrans F. SAT solver project // URL: https://github.com/franklinsch/SATSolver. Дата обращения: 08.04.2020.

[12] Babic D. Satisfiability Suggested Format // 1993. - URL: https://www.domagoj-babic.com/uploads/ResearchProjects/Spear/dimacs-cnf.pdf. 8 p. Дата обращения: 08.04.2020.

Application of SAT Approach to Switch Blocks Routing for Reconfigurable Systems-on-Chip

D. V. Zhukov, D. A. Zheleznikov, M. A. Zapletina

Institute for Design Problems in Microelectronics RAS (IPPM RAS), Zelenograd, Moscow, zhukov_d@ippm.ru, zheleznikov_d@ippm.ru, zapletina_m@ippm.ru

Abstract – The paper presents the SAT based approach to detailed routing of the switch blocks in an island-style reconfigurable system-on-a-chip. The goal of the routing procedure is to find such a combination of internal switching elements of a switch block that guarantees all project nets are completely routed without any conflicts. An important advantage of the presented SAT approach in comparison with other routing algorithms is the possibility to prove mathematically if the current design is routable or not. In the SAT approach a system of Boolean equations is formed for each switch block. It consists of the intrinsic (basic) and conflict constraints. The basic constraints occur from the architecture of the switch block. The previously routed nets cause the conflict constraints that are produced by the early routing iterations. The constraints are converted into a Boolean system of equations that is processed by the SAT solver. The paper presents the detailed routing task solved for the switch blocks of two types. Their main difference, shown in the flexibility parameter, is the quantity of tracks the current pin of the switch block can be connected to. The computational tests confirmed that the presented algorithm meets the requirements of the netlist full routability and minimization of routing runtime. The algorithm implemented in C has linear complexity.

Keywords: reconfigurable system-on-chip, SAT, detailed routing, island-style architecture, hierarchical routing.

REFERENCES

[1] Romanov A.Yu., Ivannikov A.D., Romanova I.I. Simulation and synthesis of networks-on-chip by using NoCSimp HDL library // 2016 IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO 2016 Conference Proceedings. 7493072. PP. 300-303.

[2] Gort M., Anderson J. H. Combined Architecture/ Algorithm Approach to Fast FPGA Routing // IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2013. Vol. 21. No. 6. PP. 1067-1079G.

[3] 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, 2018. Vol. 47. No. 7. PP. 516–521.

[4] H. Zhang. SATO: An efficient propositional prover. In Proceedings of the International Conference on Automated Deduction, p. 272-275, July 1997. 72

[5] Nam G.-J., Sakallah K. A., Rutenbar R.A. A comparative study of two Boolean formulations of FPGA detailed routing constraints // IEEE Transactions on Computers, 2004. Vol. 53. No. 6. PP. 688-696.

[6] Xiu-qin W., Yang Y. New approach of exploiting symmetry in SAT-based Boolean matching for FPGA technology mapping // IEEE International Conference on Vehicular Electronics and Safety, 2013. PP. 282-285.

[7] Mishchenko A., Brayton R., Roland J., Jang J. S.SAT-based logic optimization and resynthesis // Proc. IWLS '07, 2007. PP.358-364.

[8] Larrabee T., Test pattern generation using Boolean satisfiability // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1992. Vol. 11. No. 1. PP. 4-15.

[9] Taylor B., Pileggi L., Exact Combinatorial Optimization Methods for Physical Design of Regular Logic Bricks // 44th ACM/IEEE Design Automation Conference, San Diego, 2007. PP. 344-349.

[10] Prado D. F. G. Tutorial on FPGA Routing // Electronica, 2006. No. 17. PP. 23-33.

[11] Schrans F. SAT solver project // URL: https://github.com/franklinsch/SATSolver. Checking date: 08.04.2020.

[12] Babic D. Satisfiability Suggested Format // 1993. - URL: https://www.domagoj-babic.com/uploads/ResearchProjects/Spear/dimacs-cnf.pdf. 8 p. Checking date: 08.04.2020.