УДК 004.023 DOI 10.31114/2078-7707-2018-1-30-37

Алгоритмы синтеза схем-заплаток для решения задачи ресурсо-ориентированной функциональной коррекции схем из функциональных элементов

В.В. Жуков, Л.И. Высоцкий, М.С. Шуплецов

Московский государственный университет им. М.В. Ломоносова – факультет вычислительной математики и кибернетики, zhvv117@gmail.com, vysotskylev@yandex.ru, shupletsov@cs.msu.ru

Аннотация — При обнаружении ошибок или изменении спецификации проектируемой сверхбольшой интегральной схемы (СБИС) на поздних этапах маршрута проектирования откат на более ранние этапы проектирования и их повторное выполнение очень часто становится непрактичным в силу существенных временных затрат. Для целей сокращения времени проектирования в современные маршруты проектирования интегрируют специальные этапы функциональной коррекции схемы (англ. Engineering Change Order, ECO). В основе указанного подхода лежит анализ уже спроектированной схемы и построение небольшой подсхемы-заплатки, внедрение которой в уже синтезированную схему позволяет исправить все обнаруженные логические несоответствия. Таким образом, не требуется заново полностью повторять уже пройденные этапы логического и физического синтеза интегральной схемы.

Ключевым аспектом данного подхода является разработка эффективных алгоритмов построения схем-заплаток, оптимизированных по ряду таких параметров как размер (число функциональных элементов), число входов и сложность интеграции заплатки в уже синтезированную схему. Стоит отметить, что последний параметр довольно сложно формализовать, так как он должен учитывать целый ряд факторов и физических ограничений, связанных со структурой интегральной схемы, полученной в результате выполнения соответствующих этапов логического и физического синтеза. Таким образом, разработка методов решения задачи функциональной коррекции, ориентированной на учет доступных физических ресурсов для внедрения схемы-заплатки, становится актуальной задачей.

***Ключевые слова* — схемы из функциональных элементов, логический синтез, функциональная коррекция, генерация схем-заплаток.**

I. Введение

Первые работы [1], в которых были предложены алгоритмы решения некоторых классов задач функциональной коррекции для схем из функциональных элементов, появились в 1995 году. На данный момент, существуют как коммерческие [2], так и академические [3, 4] комплексы программ для решения различных задач функциональной коррекции, возникающих на практике. Кроме того, изучению задач функциональной коррекции схем из функциональных элементов и разработке алгоритмов их решения был посвящен целый ряд научных работ.

При этом можно выделить несколько основных классов алгоритмов, которые были предложены для решения указанных задач. Наибольшее распространении получил подход, основанный на разбиении схемы на эквивалентные подсхемы с целью локализации неэквивалентной части уже спроектированной схемы с некоторой эталонной схемой. Так, например, в [3] был предложен метод поиска разрезов на основе двоичных решающих диаграмм (BDD), которые порождают функционально эквивалентные соответствующие подсхемы, а в работе [4] описана система для эффективного поиска ECO для логических схем среднего размера с использованием техники разрезов. В свою очередь, в работе [5] методы построения разбиений были обобщены на случай поиска неточных соответствий и применены для повышения эффективности поиска решений задачи ECO. Наконец, в работе [6] представлена система, которая использует решение задачи выполнимости булевых формул (SAT) для выявления логически эквивалентных фрагментов схем, которые используются для последующей оптимизации решения задачи проверки эквивалентности исходных схем.

Другой подход, который также получил довольно широкое распространение, основан на построении схемы-заплатки при помощи интерполянта булевой функции, которая должна реализовываться в заданной вершине, построенном на основе набора базовых вершин. В работе [7] предложен инкрементальный подход к поиску интерполянтов для внутренних вершин схемы на основе решения нескольких задач SAT. При этом было показано, что указанный подход может быть также применен для формирования нескольких подсхем-заплаток. В [9,10] были предложены эффективные алгоритмы поиска интерполянта, а также методов решения соответствующей задачи SAT.

Наконец, в работе [11] впервые была поставлена задача функциональной коррекции интегральной схемы с учетом структурных ограничений уже спроектированной схемы и наличия физических ресурсов для построения подсхемы-заплатки. Для учета указанных параметров каждой внутренней вершине,

которая могла бы выступать в качестве основы для построения подсхемы-заплатки, приписывался специальный вес, который рассчитывался на основе специально заданной функции, учитывающей указанные ранее факторы.

Стоит также отметить, что актуальность рассматриваемой задачи также подтверждает тот факт, что задачи, связанные с решением конкретных задач в области функциональной коррекции схем из функциональных элементов регулярно появляются [12, 13, 14] в рамках международного соревнования по разработке алгоритмов автоматизации проектирования интегральных схем «CAD Contest at ICCAD».

В данной работе представлены алгоритмы решения задачи функциональной коррекции схем, описанной в работе [14]. Представленные алгоритмы позволяют синтезировать подсхемы-заплатки, оптимизированные по их размеру, а также по сложности их интеграции в синтезированную схему. При этом полученные при помощи указанных алгоритмов подсхемы-заплатки по своим параметрам сопоставимы с теми подсхемами-заплатками, которые были получены при помощи алгоритмов, вошедших в тройку призеров соревнования [14], а в ряде случаев превзошли их.

Статья имеет следующую структуру. В части II приводятся формальное описание задачи рассматриваемой задачи функциональной коррекции схем и набор вспомогательных определений. Последующие части описывают предложенные алгоритмы, а также методы решения некоторых подзадач, возникающих в ходе работы указанных алгоритмов. В заключительной части приводятся результаты тестирования предложенного решения задачи.

II. Основные понятия и постановка задачи

Дадим формальное определение задачи ресурсо-ориентированной функциональной коррекции, предложенной в работе [14]. Пусть даны две произвольные схемы из функциональных элементов (СФЭ) Σ и Σ' в заданном базисе A из функциональных элементов, реализующих функции алгебры логики (ФАЛ) от конечного множества переменных. Введем следующие обозначения: V и V' — множества вершин, I и I' — множества входов, а O и O' — множества выходов схем Σ и Σ', соответственно. Стоит отметить, что рассматриваемая модель СФЭ является классической математической моделью, которая используется для моделирования комбинационных схем.

Рассмотрим разбиение множества I' входов схемы Σ' на два подмножества: $I' = \hat{I}' \cup T$. Входы из множества T назовем *целевыми*. Пусть также заданы:

1. взаимно-однозначные соответствия $\mu_{in}: I \to \hat{I}'$ и $\mu_{out}: O \to O'$;
2. функция стоимости $c: V' \cup I' \to \mathbb{R}$, где $\mathbb{R}$ - множество вещественных чисел.

Функция стоимости c задает сложность создания новых соединений с внутренними вершинами и не целевыми входами СФЭ Σ'.

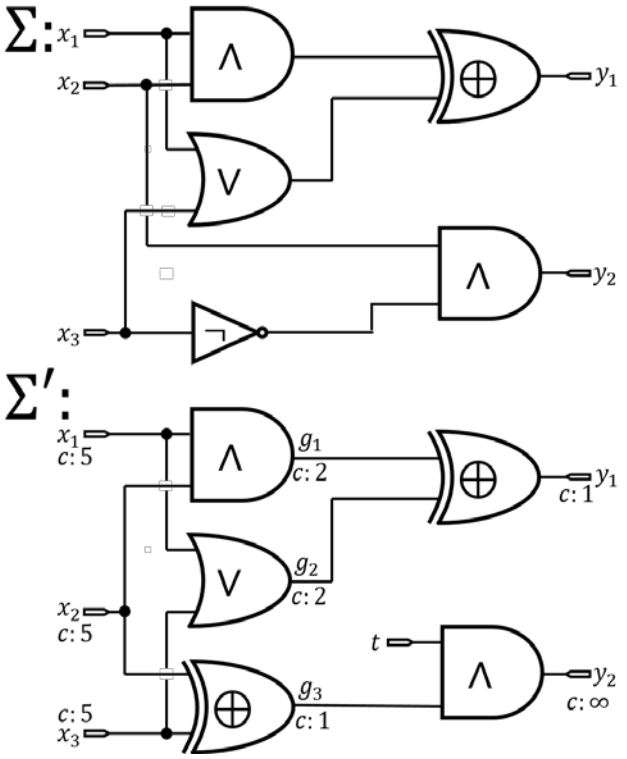

Рис. 1. Пример пары схем, иллюстрирующих введенную задачу функциональной коррекции

Задачей *функциональной коррекции* СФЭ Σ' относительно СФЭ Σ будем называть задачу построения СФЭ P в базисе A с множеством входов I_P и множеством выходов O_P, обладающую следующими свойствами:

1. существует взаимно-однозначное соответствие $\pi_{\text{in}}: I_P \to W'$, где $W', W' \subset V'$, некоторое подмножесто вершин СФЭ Σ', причём никакая вершина $w, w \in W'$ не достижима из вершин множества T;
2. существует взаимно-однозначное соответствие $\pi_{\text{out}}: O_P \to T$;
3. СФЭ $\hat{\Sigma}$, получающаяся в результате композиции СФЭ Σ' и P за счет подключения каждого входа $v_{in}, v_{in} \in I_P$, к $\pi_{\text{in}}(v)$ и каждого выхода $v_{out}, v_{out} \in O_P$, к $\pi_{\text{out}}(v_{\text{out}})$, функционально эквивалентна схеме Σ (с учётом взаимно-однозначных соответствий μ_{in} и μ_{out}).

Построенную СФЭ P будем называть *заплаткой* (от английского patch), а вершины из множества W' - *базовыми*. Стоит отметить, что не для каждой пары СФЭ Σ и Σ' заплатка P существует. При этом для оценки качества получаемой заплатки P будем использовать следующий функционал качества:

$$c(P) = \sum_{w \in W'} c\,(w).$$

Указанный функционал качества характеризует сложность интеграции заплатки в уже синтезированную СФЭ Σ' при использовании множество базовых вершин W'. В качестве дополнительного критерия оценки качества найденной заплатки P будем использовать ее размер, то есть число её функциональных элементов.

Проиллюстрируем введенную задачу функциональной коррекции на простом примере. Рассмотрим СФЭ Σ и Σ', представленные на рис. 1. Схема Σ представляет эталон (например, полученный в результате изменения спецификации проектируемого устройства или в результате обнаружения и исправления некоторой выявленной логической ошибки), который будет использоваться для функциональной коррекции схемы Σ'. Нетрудно видеть, что схема Σ реализует ФАЛ f_{y_1} = (01011100) и ФАЛ $f_{y_2} = (00100010)$. В свою очередь, схема Σ' на своем выходе y_1 реализует ФАЛ f_{y_1}, вход t является целевым, а переменные x_1, x_2, x_3 формируют множество I'. Вершины y_1 и g_3 схемы Σ' имеют стоимость, равную 1, вершины g_1 и g_2 имеют стоимость, равную 2, и каждый нецелевой вход $x_i, i = 1,2,3$, имеет стоимость, равную 5. Наконец, вершина y_2 не может быть использована для построения заплатки, поэтому имеет бесконечную стоимость.

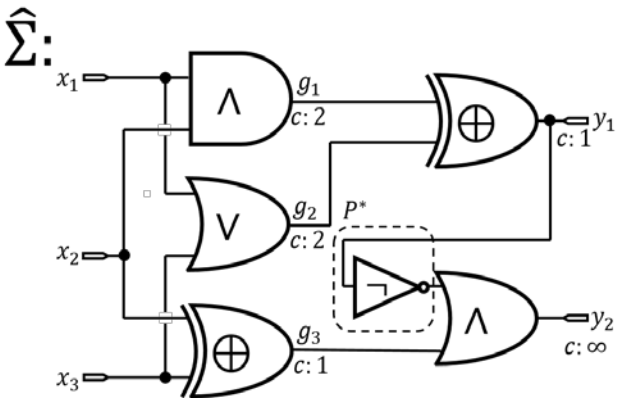

Рис. 2. Оптимальная функциональная коррекция схемы Σ'

В данном случае рассматриваемая задача имеет тривиальное решение, когда заплатка строится только с использованием входов схемы Σ' из множества I'. Заплатка P^0, построенная по формуле $y \cdot \overline{z}$, имеет стоимость $c(P^0)$, равную 10, и содержит 2 функциональных элемента. При этом наилучшая заплатка P^*, результат интеграции которой представлен на рис. 2, имеет стоимость $c(P^*)$, равную 1, и содержит всего 1 функциональный элемент. При этом указанная заплатка использует вершину y_1 в качестве базовой, и в вершине y_2 результирующей схемы $\hat{\Sigma}$ реализуется ФАЛ f_{y_2}.

Другие примеры, разъясняющие данную постановку задачи функциональной коррекции, и более подробную информацию о рассматриваемой задаче можно найти в [14].

III. Базовый алгоритм

Пусть $B = \{0,1\}$, а B^k – k-ая декартовая степень множества B. Введём обозначения $n := |I|$, $m := |O|, t := |T|$. Также обозначим вектор-функции, реализуемые схемами Σ и Σ', за F и F', соответственно:

$$F: B^n \to B^m, F': B^{n+t} \to B^m.$$

Определим функцию $\nu: B^t \to \{0, \dots, 2^t - 1\}$, возвращающую число, двоичной записью которого является аргумент. Стоит отметить, что введенная функция ν ставит в соответствие двоичному набору длины t его номер при лексикографическом упорядочивании наборов множества B^t. Так как функция ν является взаимно-однозначной, то существует обратное отображение $\nu^{-1}: \{0, \dots, 2^t - 1\} \to B^t$, которое по номеру t ставит ему в соответствие двоичный набор. Также определим на B^t отношение порядка: $\tau' < \tau'' \Leftrightarrow \nu(\tau') < \nu(\tau'')$. Обозначим через $prev(\tau)$ элемент множества B^t, непосредственно предшествующий $\tau, \tau \in B^t$, по описанному отношению.

Утверждение 1. Если заплатка P для СФЭ Σ' относительно СФЭ Σ существует, то для каждого $\alpha, \alpha \in B^n$, найдётся вектор $\tau, \tau \in B^t$, такой, что $F(\alpha) = F'(\alpha, \tau)$.

Доказательство. Достаточно взять в качестве τ вектор, составленный из значений на выходах элементов из O_P при подаче на вход композиции Σ' и P вектора α.

Утверждение доказано.

Введем вектор-функцию $\Phi: B^n \to B^t$, такую, что для любого $\alpha, \alpha \in B^n$, $\Phi(\alpha) = \tau^*$, где τ^* - лексикографически минимальный элемент множества B^t для которого верно равенство: $F(\alpha) = F'(\alpha, \tau)$. Из утверждения 1 следует, что вектор-функция $\Phi(\alpha)$ определена корректно. При этом очевидно, что любая СФЭ, реализующая вектор-функцию Φ, будет заплаткой стоимости $\sum_{w \in \hat{I}'} c\,(w)$.

Осталось показать, как сравнительно эффективно построить такую СФЭ. Для начала построим вспомогательную СФЭ E, реализующую следующие 2^t ФАЛ $e_0, \dots, e_{2^t-1}$ от n переменных:

$$e_i(\alpha) := [F(\alpha) = F'(\alpha, \nu^{-1}(i))].$$

Далее, построим СФЭ D, реализующую 2^t ФАЛ $d_0, \dots, d_{2^t-1}$ от n переменных:

$$d_0(\alpha) := e_0(\alpha),$$

$$d_i(\alpha) := e_i(\alpha) \vee d_{i-1}(\alpha) \text{ для } i = 1, \dots, 2^t - 1.$$

Заметим, что набор $(e_0(\alpha), \dots, e_{2^t-1}(\alpha))$ имеет вид (0,0,…,0,1,1,…,1), причём первая единица

соответствует минимальному номеру i такому, что $e_i(\alpha) = 1$ (обозначим этот номер $i_{\mathrm{m}}(\alpha)$). Наконец, построим схему R, реализующую 2^t ФАЛ $r_0, \dots, r_{2^t-1}$ от n переменных:

$$r_0(\alpha) := d_0(\alpha),$$

$$r_i(\alpha) := d_i(\alpha) \oplus d_{i-1}(\alpha) \quad \text{для } i = 1, \dots, 2^t - 1.$$

Ясно, что набор $(r_0(\alpha), \dots, r_{2^t-1}(\alpha))$ имеет ровно одну единицу в позиции $i_{\mathrm{m}}(\alpha)$. СФЭ P построим следующим образом: подключим 2^t выходов СФЭ R к входам шифратора с 2^t входами и t выходами (см. рис. 3).

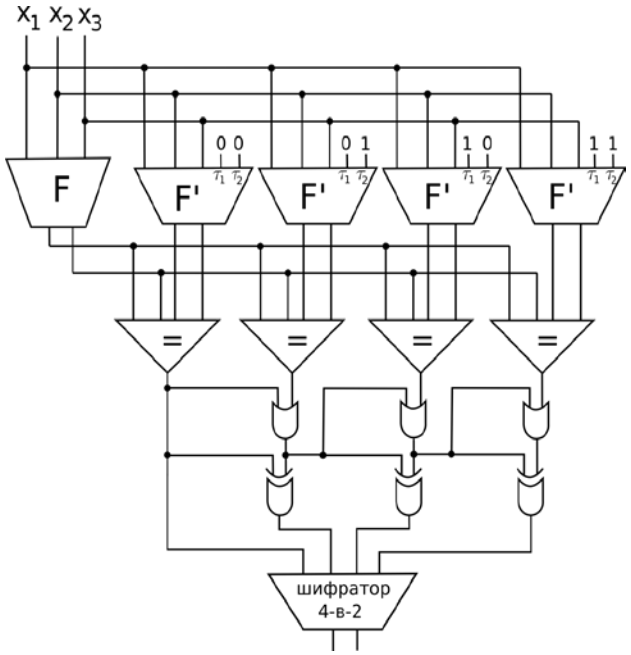

Рис. 3. Пример заплатки, которую строит базовый алгоритм (раздел III) для $n = 3, t = 2, m = 2$. Блок с символом '=' обозначает компаратор, сравнивающий 2-битные вектора

В предположении существования заплатки покажем, что СФЭ P является заплаткой. По определению функций, реализуемых СФЭ E , для каждого набора $\alpha, \alpha \in B^n$, в векторе значений $(e_0(\alpha), \dots, e_{2^t-1}(\alpha))$ есть ровно одна единица, причём $e_i(\alpha) = 1$ именно для того i, который является номером $\nu(\tau^*)$ лексикографически наименьшего набора τ^*, для которого $F(\alpha) = F'(\alpha, \tau^*)$. Шифратор же превратит вектор с единицей в соответствующей позиции в вектор τ, поэтому СФЭ P реализует функцию $\Phi(x)$.

Заметим, что в общем случае (когда мы не знаем, существует ли заплатка) можно построить сначала СФЭ P , используя описанный выше метод синтеза, затем построить СФЭ $\hat{\Sigma}$, являющуюся композицией СФЭ Σ' и P, и проверить эквивалентность СФЭ $\hat{\Sigma}$ и Σ (например, SAT-решателем). Ясно, что неэквивалентность будет означать не существование заплатки.

IV. Проверка кандидатов в базовые вершины

Рассмотрим произвольное подмножество $W' \subset V'$ мощности k, ни одна вершина которого не достижима из T , и поставим следующий вопрос: существует ли вектор-функция $\Phi: B^k \to B^t$ такая, что для любого набора $\alpha, \alpha \in B^n$, выполняется равенство

$$F(\alpha) = F'\left(\alpha, \Phi\left(G(\alpha)\right)\right), \tag{1}$$

где $G: B^n \to B^k$ — вектор функций, реализуемых в вершинах из W'.

Утверждение 2. *Для существования функции Φ, при которой выполнена формула (1), необходимо и достаточно, чтобы невыполнимой была следующая булева формула:*

$$G(\alpha^0) = \dots = G\left(\alpha^{2^t-1}\right) \wedge \wedge (\forall \tau \in B^t)\, F\left(\alpha^{\nu(\tau)}\right) \neq F'\left(\alpha^{\nu(\tau)}, \tau\right), \tag{2}$$

где $\alpha^i \in B^n, i = 0, \dots, 2^t - 1$.

Доказательство. *Необходимость.* Пусть описанная функция Φ существует. Предположим, от противного, существование наборов $\alpha^0, \dots, \alpha^{2^t-1}$, обеспечивающего выполнение формулы (2). Однако для $\tau = \Phi\left(G\left(\alpha^{\nu(\tau')}\right)\right)$, где τ' — произвольный вектор из B^t, верно равенство

$$F\left(\alpha^{\nu(\tau)}\right) = F'\left(\alpha^{\nu(\tau)}, \tau\right).$$

Пришли к противоречию.

Достаточность. Формула (2), очевидно, выполнима тогда и только тогда, когда верна следующая формула:

$$(\exists \alpha_0 \in B^n) \dots (\exists \alpha_{2^t-1} \in B^n)(\exists \nu \in B^k)$$

$$(\nu = G(\alpha_0) = \dots = G(\alpha_{2^t-1}) \wedge \wedge (\forall \tau \in B^t) F\left(\alpha_{\nu(\tau)}\right) \neq F'\left(x_{\nu(\tau)}, \tau\right)\Big).$$

Поэтому условие невыполнимости переписывается следующим образом:

$$(\forall \nu \in E^k)(\forall \alpha^0 \in B^n) \dots \left(\forall \alpha^{2^t-1} \in B^n\right)$$

$$\left(\overline{P_0(\nu, \alpha^0)} \vee \dots \vee \overline{P_{2^t-1}(\nu, \alpha^{2^t-1})}\right),$$

где

$$P_i(\nu, \alpha) \equiv \left(\nu = G(\alpha) \wedge F(\alpha) \neq F'(\alpha, \nu^{-1}(i))\right).$$

Так как $P_i(\nu, \alpha^i)$ не зависит от α^j для $i \neq j$, то получаем эквивалентную формулу:

$$(\forall \nu \in E^k)$$

$$\left(\left((\forall \alpha \in B^n)\overline{P_0(\nu, \alpha)}\right) \vee \dots \vee \left((\forall \alpha \in B_n)\overline{P_{2^t-1}(\nu, \alpha)}\right)\right).$$

Значит можно определить $\Phi(v) = \tau$, где $\tau \in B^t$ таково, что истинна формула $(\forall \alpha \in B^n)\overline{P_{\nu(\tau)}(v, \alpha)}$. Такая функция доставляет равенство в (1), т.к.

$$(\forall \alpha \in B^n)\ \overline{P_{\nu(\tau)}(v, \alpha)} \equiv$$
$$\equiv (\forall \alpha \in B^n)\ (G(\alpha) = v) \to \big(F(\alpha) = F'(\alpha, \tau)\big).$$

Достаточность доказана.

Итак, проверка существования заплатки свелась к проверке выполнимости булевой формулы. Действительно, во-первых, формулу (2) можно реализовать СФЭ размера $O\big(2^t(|\Sigma| + |\Sigma'|)\big)$. Во-вторых, используя преобразование Цейтина [18] для произвольной СФЭ с входами $x_1, \dots, x_n$ можно построить формулу $\varphi(x_1, \dots, x_n, y_1, \dots, y_m)$ (где $m = O(n)$), которая выполнима тогда и только тогда, когда указанная схема реализует нетождественный ноль.

V. Построение заплатки

Для булевой формулы A будем обозначать $Var(A)$ множество её переменных.

Интерполяционной последовательностью для набора формул $P_1, \dots, P_N$ таких, что $P_1 \wedge \dots \wedge P_N \equiv 0$, называется набор формул $I_1, \dots, I_N$ со следующими свойствами:

1. $I_N \equiv 0$;
2. тождественно верна формула $P_1 \Rightarrow I_1$, а также формула $I_i \wedge P_{i+1} \Rightarrow I_{i+1}$ для каждого $i = 1, \dots, N - 1$;
3. для всех $i = 1, \dots, N - 1$

$$Var(I_i) \subset \big(Var(P_1 \wedge \dots \wedge P_i) \cap Var(P_{i+1} \wedge \dots \wedge P_N)\big).$$

В [16] описан алгоритм построения интерполяционной последовательности. Имея его можно построить заплатку следующим образом:

1. Построим интерполяционную последовательность $I_0, \dots, I_{2^t-1}$ для формул $P_0(v, \alpha^0), \dots, P_{2^t-1}(v, \alpha^{2^t-1})$. Заметим, что в соответствии со свойством 3 все I_i зависят только от переменных v.
2. Схему-заплатку построим на основе вектор-функции $\Phi: B^k \to B^t$, такой, что для любого $v, v \in B^n$, $\Phi(v) = \tau^*$, где τ^* – лексикографически минимальный элемент множества B^t, для которого верно равенство:

$$I_{\nu(\tau)}(v) = 0.$$

3. Метод построения такой схемы аналогичен тому, что описан в разделе III.

Утверждение 3. *Любая СФЭ, реализующая вектор-функцию Φ, является заплаткой для пары схем Σ и Σ'.*

Доказательство. Фиксируем произвольное $v, v \in B^k$, и обозначим $\tau = \Phi(v)$, $i = \nu(\tau)$. Если $i > 0$, то по определению интерполяционной последовательности тождественно верна формула

$$P_i(v, \alpha) \wedge I_{i-1}(v) \Rightarrow I_i(v).$$

Однако по выбору τ имеем $I_i(v) = 0$ и $I_{i-1}(v) = 1$. Поэтому $P_i(v, \alpha) \equiv 0$. Иначе говоря,

$$(\forall \alpha \in B^n)\ (G(\alpha) = v) \to \big(F(\alpha) = F'\big(\alpha, \Phi(v)\big)\big).$$

Случай $i = 0$ рассматривается аналогично. Утверждение доказано.

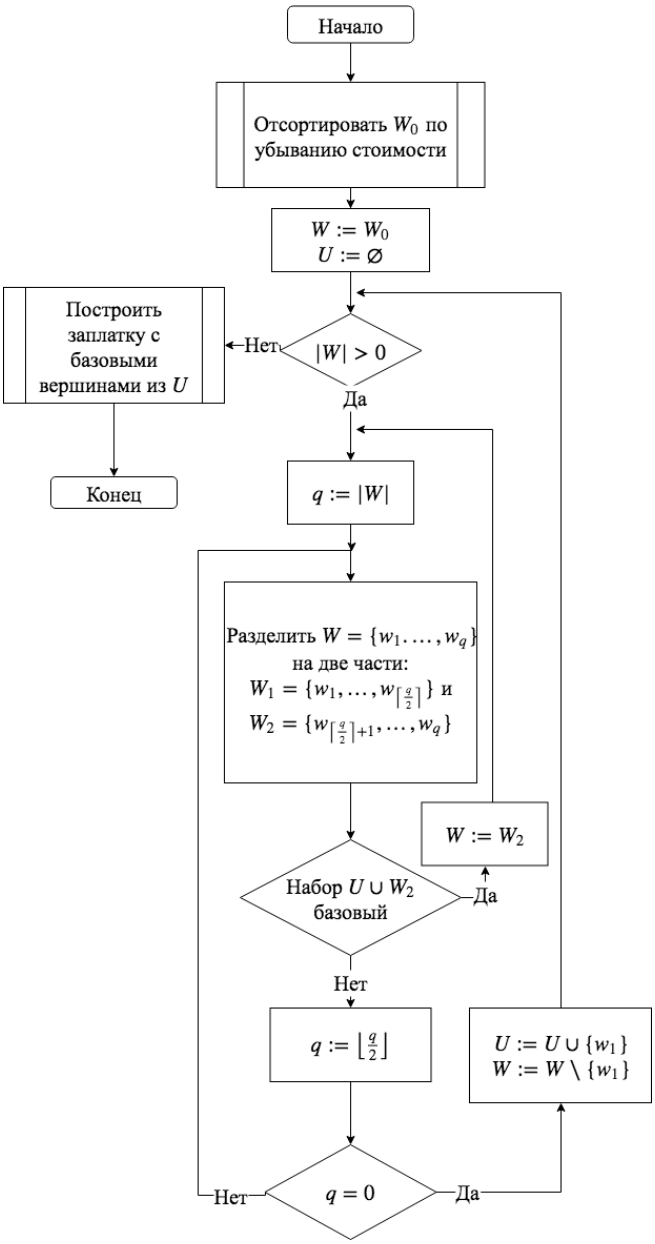

Рис. 4. Блок-схема продвинутого алгоритма поиска набора базовых вершин

VI. Продвинутый алгоритм

В этом разделе мы опишем более сложный алгоритм, который будет пытаться оптимизировать стоимость набора базовых вершин вместо использования подмножества $\hat{I}'$ входов СФЭ Σ'. Обозначим $W_0 \subset V'$ множество всех вершин СФЭ Σ', не достижимых из T. Все эти вершины изначально будут кандидатами в базовые.

Начнём со следующей идеи: отсортируем W_0 по убыванию стоимости. Заведём множество т. н. «зафиксированных» вершин U, изначально пустое. Далее будем последовательно проверять, можно ли построить заплатку, используя только самую дешёвую вершину, а также все зафиксированные вершины. Затем, если заплатку построить нельзя, проверим две самые дешёвые (плюс зафиксированные) и т. д. Так мы найдём минимальный суффикс, для которого можно построить заплатку. Выбросим все «непригодившиеся» вершины. Наконец, зафиксируем одну из «пригодившихся» вершин и запустим процесс сначала.

В описанном методе настораживает только одно: потенциально квадратичное число и так достаточно трудоёмких проверок множества кандидатов в базовые вершины. Однако учитывая «монотонность» проверяемого свойства можно воспользоваться двоичным поиском. А именно, вместо добавления вершин по одной проверим сначала половину всех кандидатов, затем в зависимости от успешности проверки либо вдвое уменьшим количество кандидатов, либо, наоборот, добавим к ним половину ещё не проверенных вершин, и т. д. Блок-схема получившегося алгоритма приведена на рис. 4.

Таблица 1

Результаты тестирования алгоритмов, предложенных в статье. Серым цветом выделены строки, соответствующие тестам, для которых авторское решение превосходит решение как минимум одного из призёров. N/A в ячейке таблицы означает, что решение для данного теста не было построено в отведённое время.

	Авторское решение		**ICCAD 1-ое место**		**ICCAD 2-ое место**		**ICCAD 3-е место**	
№	**Размер заплатки**	**Базовая стоимость**	**Размер заплатки**	**Базовая стоимость**	**Размер заплатки**	**Базовая стоимость**	**Размер заплатки**	**Базовая стоимость**
1	1	4	1	4	1	4	1	4
2	5	17	4	17	5	17	4	17
3	2	80	3	80	3	80	3	80
4	3	32	5	42	2	36	1	32
5	137097	1762	30	47	49	47	N/A	N/A
6	4	118	6605	5660	5	118	5	118
7	3	348	2	284	2	284	2	284
8	6	78	4	78	5	80	4	78
9	45	50	29	50	50	50	55	86
10	3212	135	587	135	310	135	243	135
11	3811	4142	1063	4142	369	760	N/A	N/A
12	1	104	1	104	1	104	1	104
13	114	2871	9	3467	12	3833	16	2656
14	12268630	2219	42	95	65	104	49	501
15	1151	191	11	191	5	168	2	168
16	9268	1762	13	318	13	260	11	262
17	1403	18552	79	434	101	436	N/A	N/A
18	5122	3465	1	18	21	106	1	18
19	270196	501804	7686	501804	N/A	N/A	N/A	N/A
20	49	142	6	136	6	120	11	168

VII. Результаты тестирования

Тестирование предложенных алгоритмов проводилось на тестовой выборке, предложенной в работе [14]. В указанной работе подробно описывается структура тестовых схем и их параметры, а также стратегии назначения базовых стоимостей внутренним вершинам схемы.

Результаты работы алгоритмов (колонки «Авторское решение») сравниваются с тремя наилучшими опубликованными результатами [17] алгоритмов, которые стали призерами соревнования ICCAD 2017 (колонки «ICCAD». При этом сравнение проводилось по двум основным параметрам: размер синтезированной подсхемы-заплатки (колонка «Размер заплатки») и суммарная базовая стоимость вершин, которые использовались для ее построения (колонка «Базовая стоимость»). Результаты сравнения приведены в таблице 1. Анализируя её нетрудно видеть, что во многих случаях авторские алгоритмы позволили построить подсхему-заплатку с суммарной базовой стоимостью, которая не уступает наилучшему результату, полученному в рамках соревнования ICCAD 2017. При этом в ряде случаев авторские алгоритмы проигрывают по размеру синтезируемой заплатки, что можно объяснить тем, что при синтезе самой заплатки не применялось специальных

алгоритмов логического синтеза и оптимизации (напомним, что в первую очередь результаты сравнивались по базовой стоимости, и только при равных стоимостях учитывался размер заплатки). В некоторых тестах авторский алгоритм существенно уступает результатам, которые были получены победителями соревнования ICCAD 2017. Это связано с лимитом времени на поиск интерполяционной последовательности и использованием базового алгоритма, который использует в качестве базовых вершин основные входы схемы, а также в ряде случаев порождает подсхемы-заплатки довольно большого размера. Однако применение базового алгоритма позволило избежать ситуации, когда заплатка вообще не была построена за отведённое время (30 минут), в отличие от решений команд, занявших третье и даже второе место (такие ситуации обозначены N/A в таблице).

Поддержка

Работа выполнена при финансовой поддержке РФФИ – грант № 18-01-00800-а.

Литература

[1] Lin C.-C., Chen K.-C., Chang S.-C., Marek-Sadowska M., Cheng K.-T. Logic synthesis for engineering change // DAC. 1995. P. 647–652.

[2] Cadence Encounter Conformal ECO Designer. URL: https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/functional-eco/conformal-eco-designer.html (дата обращения: 02.04.2018)

[3] S.-L. Huang, W.-H. Lin, P.-K. Huang and C.-Y. Huang. Match and replace: A functional ECO engine for multi-error circuit rectification // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 32, no. 3, pp. 467-478, March 2013.

[4] S. Krishnaswamy, H. Ren, N. Modi and R. Puri, "DeltaSyn: An efficient logic difference optimizer for ECO synthesis," International conference on Computer-Aided Design (ICCAD) - Digest of Technical Papers, 2009, pp. 789-796.

[5] K.-H. Chang, I. L. Markov and V. Bertacco, "Fixing Design Errors With Counterexamples and Resynthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 27, no. 1, pp. 184-188, Jan. 2008.

[6] C.-H. Lin, Y.-C. Huang, S.-C. Chang and W.-B. Jone, "Design and design automation of rectification logic for engineering change," Asia and South Pacific Design Automation Conference (ASP-DAC), 2005, pp. 1006-1009.

[7] K.-F. Tang, C.-A. Wu, P.-K. Huang and C.-Y. Huang, "Interpolation-based incremental ECO synthesis for multi-error logic rectification," Design Automation Conference (DAC), 2011, pp. 146-151.

[8] K.-F. Tang, P.-K. Huang, C.-N. Chou and C.-Y. Huang, "Multi-patch generation for multi-error logic rectification by interpolation with cofactor reduction," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, pp. 1567-1572.

[9] B.-H. Wu, C.-J. Yang, C.-Y. Huang and J.-H. R. Jiang, "A robust functional ECO engine by SAT proof minimization and interpolation techniques," International Conference on Computer-Aided Design (ICCAD), 2010, pp. 729-734.

[10] Petkovska, D. Novo, A. Mishchenko and P. Ienne, "Constrained interpolation for guided logic synthesis," International Conference on Computer-Aided Design (ICCAD), 2014, pp. 462-469.

[11] A.-C. Cheng, I. H.-R. Jiang and J.-Y. Jou, "Resource-aware functional ECO patch generation," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016, pp. 1036-1041.

[12] W. Jong, H. T. Wang, C. Hsieh and K. Y. Khoo, "ICCAD-2012 CAD contest in finding the minimal logic difference for functional ECO and benchmark suite: CAD contest," 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, 2012, pp. 342-344.

[13] C. J. Hsu, C. A. Wu, W. H. Lin and K. Y. Khoo, "ICCAD-2015 CAD contest in large-scale equivalence checking and function correction and benchmark suite," 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, 2015, pp. 916-920.

[14] Ching-Yi Huang, Chih-Jen Hsu, Chi-An Wu, and Kei-Yong Khoo. 2017. ICCAD-2017 CAD contest in resource-aware patch generation. In Proceedings of the 36th International Conference on Computer-Aided Design (ICCAD '17). IEEE Press, Piscataway, NJ, USA, pp. 857-862.

[15] W. Craig, Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory, The Journal of Symbolic Logic 22 (1957), no. 3, 269–285.

[16] Arie Gurfinkel and Yakir Vizel. 2014. DRUPing for Interpolants. In Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design (FMCAD '14), Koen Claessen and Viktor Kuncak (Eds.). FMCAD Inc, Austin, TX, , Article 19 , 8 pages.

[17] Ching-Yi Huang, Chih-Jen Hsu, Chi-An Wu, and Kei-Yong Khoo. ICCAD-2017 Problem A: Resource-aware Patch Generation. URL: http://cad-contest-2017.el.cycu.edu.tw/Problem%20A_Ching-Yi.pdf (дата обращения: 02.04.2018).

[18] Цейтин Г.С. О сложности вывода в исчислении высказываний. Исследования по конструктивной математике и математической логике. Том 8 (1968), С. 234–259. Ленинград: Наука

Resource-Aware Patch Generation of Boolean Circuits

V. V. Zhukov, L. I. Vysotsky, M. S. Shupletsov

Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics,
zhvv117@gmail.com, vysotskylev@yandex.ru, shupletsov@cs.msu.ru

***Abstract* — If some functionality has to be changed or functional bugs are found at late stages, restarting the whole design flow is impractical for modern design flows. To save time and cost, automating Engineering Change Orders (ECOs) is more practical. The idea of this approach lies in careful analysis of already synthesized circuit and generation of a patch circuit, which allows for rectification of all logical errors and functionality changes. Thus, there is no need in the repetition of already accomplished design flow steps of logical and physical synthesis.**

Development of algorithms for optimized patch generation is the main aspect of this approach. The key parameters of the patch are its size (the number of gates), number of primary inputs and effort required for patch integration. The latter is a hard parameter to formalize, since several parameters of the synthesized circuit should be considered (e.g. resource limits imposed by the structure of the circuit). Consequently, the development of algorithms for resource-aware patch generation starts to play an important role in modern ECO research.

Keywords — Boolean circuits, logic synthesis, engineering change order, partitioning, equivalence checking, functional correction, engineering change order, Boolean matching.

REFERENCES

[1] Lin C.-C., Chen K.-C., Chang S.-C., Marek-Sadowska M., Cheng K.-T. Logic synthesis for engineering change // DAC. 1995. P. 647–652.

[2] Cadence Encounter Conformal ECO Designer. URL: https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/functional-eco/conformal-eco-designer.html (accessed: 02.04.2018)

[3] S.-L. Huang, W.-H. Lin, P.-K. Huang and C.-Y. Huang, "Match and replace: A functional ECO engine for multi-error circuit rectification," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 32, no. 3, pp. 467-478, March 2013.

[4] S. Krishnaswamy, H. Ren, N. Modi and R. Puri, "DeltaSyn: An efficient logic difference optimizer for ECO synthesis," International conference on Computer-Aided Design (ICCAD) - Digest of Technical Papers, 2009, pp. 789-796.

[5] K.-H. Chang, I. L. Markov and V. Bertacco, "Fixing Design Errors With Counterexamples and Resynthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 27, no. 1, pp. 184-188, Jan. 2008.

[6] C.-H. Lin, Y.-C. Huang, S.-C. Chang and W.-B. Jone, "Design and design automation of rectification logic for engineering change," Asia and South Pacific Design Automation Conference (ASP-DAC), 2005, pp. 1006-1009.

[7] K.-F. Tang, C.-A. Wu, P.-K. Huang and C.-Y. Huang, "Interpolation-based incremental ECO synthesis for multi-error logic rectification," Design Automation Conference (DAC), 2011, pp. 146-151.

[8] K.-F. Tang, P.-K. Huang, C.-N. Chou and C.-Y. Huang, "Multi-patch generation for multi-error logic rectification by interpolation with cofactor reduction," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, pp. 1567-1572.

[9] B.-H. Wu, C.-J. Yang, C.-Y. Huang and J.-H. R. Jiang, "A robust functional ECO engine by SAT proof minimization and interpolation techniques," International Conference on Computer-Aided Design (ICCAD), 2010, pp. 729-734.

[10] Petkovska, D. Novo, A. Mishchenko and P. Ienne, "Constrained interpolation for guided logic synthesis," International Conference on Computer-Aided Design (ICCAD), 2014, pp. 462-469.

[11] A.-C. Cheng, I. H.-R. Jiang and J.-Y. Jou, "Resource-aware functional ECO patch generation," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016, pp. 1036-1041.

[12] W. Jong, H. T. Wang, C. Hsieh and K. Y. Khoo, "ICCAD-2012 CAD contest in finding the minimal logic difference for functional ECO and benchmark suite: CAD contest," 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, 2012, pp. 342-344.

[13] C. J. Hsu, C. A. Wu, W. H. Lin and K. Y. Khoo, "ICCAD-2015 CAD contest in large-scale equivalence checking and function correction and benchmark suite," 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, 2015, pp. 916-920.

[14] Ching-Yi Huang, Chih-Jen Hsu, Chi-An Wu, and Kei-Yong Khoo. 2017. ICCAD-2017 CAD contest in resource-aware patch generation. In Proceedings of the 36th International Conference on Computer-Aided Design (ICCAD '17). IEEE Press, Piscataway, NJ, USA, pp. 857-862.

[15] W. Craig, Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory, The Journal of Symbolic Logic 22 (1957), no. 3, 269–285.

[16] Arie Gurfinkel and Yakir Vizel. 2014. DRUPing for Interpolants. In Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design (FMCAD '14), Koen Claessen and Viktor Kuncak (Eds.). FMCAD Inc, Austin, TX, , Article 19 , 8 pages.

[17] Ching-Yi Huang, Chih-Jen Hsu, Chi-An Wu, and Kei-Yong Khoo. ICCAD-2017 Problem A: Resource-aware Patch Generation. URL: http://cad-contest-2017.el.cycu.edu.tw/Problem%20A_Ching-Yi.pdf (accessed: 02.04.2018).