УДК 621.3.049.771.14

Алгоритм синтеза цифровых микросхем на основе разложения Э.Н. Гильберта

С.И. Гуров, Д.И. Рыжова

Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН),

sgur@cs.msu.ru, ryzhova_d@ippm.ru

***Аннотация* — Проблема оптимизации синтеза комбинационных частей цифровых интегральных микросхем (ИМС) продолжает оставаться актуальной. При автоматизации проектирования методы, ориентированные на регулярные представления комбинационных частей схем, представляются предпочтительными с точки зрения надежности на логическом уровне и легкости тестирования. Такие представления логических блоков микросистем можно получить непосредственно из разложения Э. Гильберта. В статье описан алгоритм синтеза комбинационной схемы, реализующий произвольные булевы функции, основанный на представлении частичной булевой функции f в виде $F = M_0 \,\&\, \overline{M_1 \,\&\, \overline{M_2 \,\&\, \ldots \,\&\, \overline{M_k}}}$, где $M_0, M_1, M_2, \ldots, M_k$ – монотонные функции, а F – оптимальное в некотором смысле доопределение *f*. Для часто используемой на практике системы приоритетов значений таблично заданных функций предложен эффективный способ нахождения их отрицаний. На основе указанного разложения реализован алгоритм синтеза схемы, вычисляющей *f* в реальных проектных базисах микроэлектронных БИС. Приведён пример работы алгоритма в составе системы автоматического синтеза комбинационных логических блоков БИС.**

***Ключевые слова* — булева функция, комбинационная схема, функция Шеффера.**

I. Введение

В статье [1] показано, что любая полностью определенная булева функция (ПБФ) может быть представлена в виде

$$f = M_0 \,\&\, \overline{M_1 \,\&\, \overline{M_2 \,\&\, \ldots \,\&\, \overline{M_k}}}, \qquad (1)$$

где $M_0, M_1, M_2, \ldots, M_k$ — некоторые монотонные булевы функции. Эти функции мы предлагаем называть функциями Э. Гильберта, а формулу (1) — разложением Э. Гильберта для БФ *f*. Хотя по тексту указанной статьи можно восстановить алгоритм нахождения функций $M_0, M_1, M_2, \ldots, M_k$, прямое использование его во многих практически важных случаях может вызвать серьезные затруднения из-за операции взятия отрицания от функции [2-4].

На практике же булевы функции (БФ) и их системы часто задаются таблицами специального вида [5], строки которых описывают значения функций на гранях булева *n*-мерного единичного куба E^n. В этом случае получение таблицы для инверсной функции может оказаться весьма трудоемким. Это связано с тем, что указанные таблицы допускают наличие пересекающихся интервалов E^n, на которых функция может принимать различные значения. При этом для определения значения БФ в области пересечения интервалов необходимо учитывать приоритеты значений функции, с учётом которых построена таблица. При неравных приоритетах значений 0 и 1 инвертирование функции от *n* переменных и заданной на *l* интервалах требует, как будет видно, выполнения $O(n{\cdot}l)$ операций. Указанное обстоятельство, как и опыт работы авторов, показывают, что от того, насколько удачно реализована операция инвертирования таблично заданных функций, в значительной степени зависит эффективность работы алгоритма в целом.

Кроме того, на практике обычно приходится иметь дело с частичными БФ (ЧБФ). Ясно, что можно получить разложение вида (1) для некоторого доопределения ЧБФ, причём функции Э. Гильберта будут определяться, очевидно, неоднозначно. Поэтому желательно иметь алгоритм, генерирующий такие монотонные функции $M_0, M_1, M_2, \ldots, M_k$, что:

$$F = M_0 \,\&\, \overline{M_1 \,\&\, \overline{M_2 \,\&\, \ldots \,\&\, \overline{M_k}}} \qquad (2)$$

для ПБФ F, являющейся одним из возможных и, желательно, в некотором смысле простейших, доопределений *f*. То, что ПБФ F есть (некоторое) доопределение *f*, будем отмечать как $F = Ext(f)$.

Легко заметить, что на основе разложения Э. Гильберта можно предложить простой алгоритм синтеза ПБФ $F = Ext(f)$ в классе схем из функциональных элементов. Действительно, (2) эквивалентно:

$$F = M_0 \,\&\, \big(M_1 \,|\, \big(M_2 \,|\, \big(\ldots \big(M_{k-1} \,|\, M_k\big)\ldots\big)\big)\big), \qquad (3)$$

где | – символ функции Шеффера (отрицание конъюнкции). Ясно, что в указанном классе формула (3) может быть реализована регулярной каскадной схемой, которая будет выглядеть особенно простой при наличии в базисе элемента функции Шеффера.

Наиболее привлекательным представляется использование алгоритма получения представления (3) для автоматического синтеза цифровых комбинационных логических схем. Во-первых, построение схемы из функциональных элементов является первым этапом синтеза в реальных проектных базисах (т.н. этап «абстрактного синтеза»). Во-вторых, условие наличия в базисе элемента функции Шеффера не является обременительным. В-третьих, среди всех функциональных элементов (исключая инвертор) элемент функции Шеффера имеет простейшую реализацию в виде микроэлектронной электрической транзисторной схемы. И, наконец, функции $M_0, M_1, M_2, \ldots, M_k$ могут быть заданы программируемой логической матрицей (ПЛМ), минимизированной с использованием известных эффективных методов и имеющей дополнительный выигрыш в площади за счёт монотонности представляемых функций (т.к. отсутствует необходимость представления их в парафазном виде).

II. Основные определения и обозначения

В статье используются стандартные обозначения величин и основные понятия, ставшие уже общепринятыми в кругу специалистов в области дискретной математики [6].

Множество единиц и нулей БФ f обозначается как N_f^1 и N_f^0; если $N_f^- = E^n \setminus \{N_f^1 \cup N_f^0\} \neq \emptyset$, то БФ f является частичной. Под взятием отрицания (инвертированием) ЧБФ f будем понимать переход к функции $\varphi = \overline{f}$, у которой $N_\varphi^1 = N_f^1 \cup N_\varphi^0 = N_f^0$. Если $N_\vartheta^1 \subseteq N_\varphi^1$, то функция φ называется доопределением ϑ, что обозначается $\varphi = ext(\vartheta)$. Если при этом φ – ПБФ (ЧБФ), то φ – полное (частичное) доопределение ϑ; обозначение для полных доопределений: $\varphi = Ext(\vartheta)$. Множество всевозможных полных доопределений ϑ обозначим $EXT(\vartheta)$. Функции констант (произвольного множества фиктивных переменных) обозначаются 1 и 0. Обозначения двуместных функций традиционны.

Грань E^n наименьшей размерности, содержащую вершины $\tilde{\omega}$ и $(1,\ldots,1)((0,\ldots,0))$ будем называть верхним (нижним) конусом $\tilde{\omega}$ и обозначать $\tilde{\omega}^\Delta$ $\left(\tilde{\omega}^\nabla\right)$. Верхним (нижним) конусом совокупности вершин Ω из E^n будем называть объединение верхних (нижних) конусов всех вершин Ω и обозначать его Ω^Δ $\left(\Omega^\nabla\right)$.

Единичный набор БФ f есть нижняя единица f, если его нижний конус не содержит других единиц f. Множество нижних единиц f обозначается $LU(f)$. Нулевой набор БФ f есть верхний нуль f, если его верхний конус не содержит других нулей f. Множество верхних нулей f обозначается $UZ(f)$.

Множество монотонных функций известного числа аргументов обозначим M. Ясно, что любое из множеств $LU(f)$ и $UZ(f)$ полностью определяет f, если $f \in M$ (при этом $N_f^1 = LU(f)^\Delta$ и $N_f^0 = UZ(f)^\nabla$). Заметим, что, например, функция 1 может быть задана либо пустым множеством верхних нулей, либо множеством $\{\tilde{0}\}$ нижних единиц, а функция 0 – либо пустым множеством нижних единиц, либо множеством $\{\tilde{1}\}$ верхних нулей. Для (возможно частичной) БФ φ ее мажоранта f_φ^1 есть монотонная функция, определяемая соотношением $U\left(f_\varphi^1\right) = LU(\varphi)$.

III. Алгоритм синтеза цифровых микросхем на основе разложения Гильберта

A. *Нахождение функции Гильберта*

Пусть дана ЧБФ f. Будем искать разложение Э. Гильберта для некоторой функции $F = Ext(f) \in EXT(f)$. Функция F будет построена в процессе нахождения указанного разложения, поэтому мы сразу можем пользоваться формулой (2):

$$F = M_0 \,\&\, \overline{M_1} \vee M_0 \,\&\, M_2 \,\&\, \overline{M_3} \vee \ldots$$
$$\ldots \begin{cases} M_0 \,\&\, M_2 \,\&\, \ldots \,\&\, \overline{M_{k-1}} \,\&\, M_k, k - \textit{нечетное}, \\ M_0 \,\&\, M_2 \,\&\, \ldots \,\&\, M_k, k - \textit{четное или } 0. \end{cases}$$

Последнее соотношение эквивалентно совокупности равенств:

$$F = f_0 = M_0 \,\&\left(\overline{M_1} \vee \overline{f_2}\right),$$
$$f_2 = M_2 \,\&\left(\overline{M_3} \vee \overline{f_4}\right),$$
$$\ldots$$
$$f_{k-1} = M_{k-1} \,\&\, \overline{M_k}, k - \textit{нечетное};$$
$$f_k = M_k, k - \textit{четное или } 0.$$

Отсюда ясно, что функции Э. Гильберта могут быть получены в ходе итерационного процесса, на каждом шаге которого для некоторой ЧБФ $\Phi = ext(\varphi)$ находится следующее разложение:

$$\Phi = Me \,\&\left(\overline{Mo} \vee \vartheta\right), \tag{4}$$

где Me и Mo – некоторые подходящие монотонные БФ.

Пусть Me и Mo, удовлетворяющие (4), найдены. Определим функцию $S = Me \,\&\, \overline{Mo}$. Тогда в качестве ϑ можно взять функцию, задаваемую множествами своих единиц и нулей следующим образом:

$$N_\vartheta^1 = N_\varphi^1 \setminus N_S^1, \quad N_\vartheta^0 = N_\varphi^0 \cap N_{Me}^1. \tag{5}$$

В данном процессе на первом шаге $\varphi = f$ и далее Me и Mo – суть функции Э. Гильберта соответствующего чётного и нечётного индексов. Если окажется, что $\vartheta \equiv 1$ или $\vartheta \equiv 0$, то итерации следует прервать, в противном случае ϑ есть φ для следующего шага. Ясно, что выполнение условия $N_{\vartheta}^{1} \subset N_{\varphi}^{1}$ обеспечит последовательность строгих вложений множеств единичных наборов разлагаемых функций на каждом шаге и, следовательно, конечность указанного процесса. Ясно также, что в результате описанного процесса будет найдено разложение Э. Гильберта для некоторой функции $F = Ext(f) \in EXT(f)$.

Множества N_{Me}^{1} и $\left[N_{Mo}^{1}\right]$ есть объединение верхних конусов нижних единиц [нижних нулей] соответствующих функций на каждом этапе алгоритма.

B. Краткое описание алгоритма нахождения разложения Э. Гильберта

Алгоритм нахождения разложения Э. Гильберта определяет функции *Me* и *Mo* из предельных условий $|Me| \to min$ и $|Me| \to max$ из возможных значений [7-9]. Введём обозначение для множества единичных координат булева вектора:

$$\tilde{\alpha} = (\alpha_1,...,\alpha_n) \in E^n : 1(\alpha) = \{i = 1,...,n \mid \alpha_i = 1\}.$$

Обозначим символами *LU*(*f*) и *UZ*(*f*) соответственно множества нижних единиц и верхних нулей ЧБФ $f(x_1,...,x_n)$ и определим функции *Me* и *Mo*:

$$Me = \vee_{\tilde{\alpha} \in N_{LU(f)}^{1}} \wedge_{i \in 1(\tilde{\alpha})} x_i \, , \, Mo = \vee_{\tilde{\beta} \in N_{UZ(f)}^{0}} \wedge_{i \in 1(\tilde{\beta})} x_i \, .$$

Если в множестве $(\tilde{\alpha})^{\Delta} \setminus \tilde{\alpha},\ \tilde{\alpha} \in LU(f)$, окажутся только единицы ЧБФ *f*, то удалим их из списка N_f^1 и дальнейшего рассмотрения и аналогично поступим с нулями f из $(\tilde{\beta})^{\Delta} \setminus \tilde{\beta},\ \tilde{\beta} \in UZ(f)$. Данный шаг приводит к минимизации сложности получаемого представления. Далее переопределяем функции *Me* и *Mo* по формулам (5), пока соответствующие множества не станут пусты. Например, для ЧБФ $f(x_1, x_2, x_3)$ с $N_f^1 = \{(100),(111)\}$, $N_f^0 = \{(110)\}$ на первом этапе получим $Me^1 = x_1$, $Mo^1 = x_1 x_2$, на втором – $Me^2 = x_1 x_2 x_3$ и алгоритм даст $Ext(f) = x_1 \overline{x_2} \vee x_1 x_3$, а для ЧБФ $g(x_1, x_2, x_3)$ с $N_g^1 = N_f^1 \cup \{(001)\}$, $N_g^0 = N_f^0$ получим $Ext(g) = x_1 \overline{x_2} \vee x_3$.

C. Табличное задание систем частичных булевых функций

Значения БФ будем задавать на интервалах (гранях) E^n. Интервалы E^n задаются наборами $\overline{\sigma} = \sigma_1 ... \sigma_n$, где $\sigma_i \in \{1,0,-\}$, $j = \overline{1,n}$. Как известно, каждая грань $\overline{\sigma}$ булева куба соответствуют некоторой элементарной конъюнкции $K_{\overline{\sigma}}(\tilde{x}) = \wedge_{j \in J} x_j^{\sigma_j}$ так, что вершина $\tilde{\alpha}$ из E^n принадлежит $\overline{\sigma}$ тогда и только тогда, когда $K_{\overline{\sigma}}(\tilde{\alpha}) = 1$. Здесь *J* – подмножество таких индексов из $\{1,...,n\}$, что $\sigma_i \in \{1,0\}$. Таким образом, переменная x_j присутствует в конъюнкции $K_{\overline{\sigma}}(\tilde{x})$ в прямой (инверсной) форме, если $\sigma_i = 1$ и отсутствует в последней при $\sigma_i = -$. Это, в свою очередь, означает следующую трактовку символа $-$ в записи интервала $\overline{\sigma}$: интервал $\overline{\sigma}$ может быть заменен двумя интервалами, отличающимися от $\overline{\sigma}$ наличием 1 и 0 в соответствующей позиции.

Для того чтобы покоординатная конъюнкция интервалов соответствовала их теоретико-множественному произведению, доопределим функции ∧ (&) над интервалами. Введём на множестве $\{1,0,-\}$ частичный порядок с помощью рефлексивного несимметричного отношения « ≥ »:1 $- \geq 1$, $- \geq 0$, и 0 несравнимы. Будем трактовать функции ∧ и & как *max* и *min* соответственно на множестве $\{1,0,-\}$ с указанным частичным порядком.

Частичная БФ принимает значения из множества $\{1,0,-\}$, при этом значение $\{-\}$ трактуется как неопределенное. Будем определять ее через множество значений конъюнкций $\{K_{\overline{\sigma}}(\tilde{x})\}$ на некоторой совокупности интервалов $\{\overline{\sigma}\}$. Интервалы $\{\overline{\sigma}\}$ будем называть задающими. Если на всех наборах интервала $\overline{\sigma}$ конъюнкция $K_{\overline{\sigma}}(\tilde{x})$ принимает значение $\omega \in \{1,0,-\}$, то будем писать $K_{\overline{\sigma}}(\tilde{x}) = \omega$. Значения упорядоченного набора конъюнкций $K_{\overline{\sigma}}^{1}(\tilde{x}),...,K_{\overline{\sigma}}^{m}(\tilde{x})$, участвующие в определении ЧБФ $f_1,..., f_m$, на некотором интервале $\overline{\sigma}$ будем задавать кортежем $\overline{\omega} = \omega_1 ... \omega_m$, $\omega_j \in \{1,0,-\}$, $j = \overline{1,m}$. Таким образом, в записи $\overline{\omega}$ символ $\{-\}$ означает, что он может быть заменён на 1 или 0 при соответствующем доопределении данной функции.

Систему из *m* частичных булевых функций от *n* переменных (СЧБФ), заданную на *l* интервалах, будем задавать в табличной форме в виде совокупности *l* строк:

$$N \ \overline{\sigma} : \overline{\omega}, \qquad (6)$$

где *N* – номер строки, $\overline{\sigma}$ – интервал из E^n, $\overline{\omega}$ – кортеж значений функций. Совокупности $\{\overline{\sigma}\}$ и $\{\overline{\omega}\}$ из указанного задания будем называть соответственно левой и правой частью таблицы, а $\overline{\sigma}$ и $\overline{\omega}$ – левой и правой частью соответствующей строки. Заметим, что если $\omega_j \in \{1,0\}$, $j=\overline{1,m}$, то (5) определяет систему ПБФ заданием соответствующей системы дизъюнктивных нормальных форм. Возможность замены $\{-\}$ на 0 или 1 в правой части таблицы (5) есть одна из дополнительных степеней свободы при минимизации ПЛМ, отличающих указанный процесс от задачи минимизации БФ (или систем БФ) в классической постановке.

При указанном задании СЧБФ в случае, если некоторые из задающих интервалов пересекаются, а правые части соответствующих строк не совпадают, то возникает неоднозначность в определении значений функций на грани, общей для обоих интервалов. Поэтому таблицы (5) строятся с учётом выбранной системы приоритетов значений составляющих функций. Точнее, на множестве $\{1,0,-\}$ с помощью несимметричного рефлексивного отношения, трактуемого как «приоритет … больше, чем приоритет …» или «… подавляет …», вводится тот или иной (возможно частичный) порядок. Указанное отношение будем записывать как «>» (правильнее было бы «$\geq$», однако принятое обозначение уже стало традиционным и не приводит к затруднениям, поскольку используется лишь при обнаружении коллизии значений функций).

Если для некоторых интервалов $\overline{\sigma}^1$ и $\overline{\sigma}^2$, участвующих в задании данной функции, имеют место следующие равенства:

$$\overline{\sigma} = \overline{\sigma}^1 \cap \overline{\sigma}^2 \neq \emptyset,$$
$$K_{\overline{\sigma}^1}(x) = \omega_1,$$
$$K_{\overline{\sigma}^2}(x) = \omega_2,$$
$$\omega_1 \neq \omega_2,$$

то говорят, что имеет место коллизия значений функций. При этом считается, что $K_{\overline{\sigma}}(\tilde{x}) = \omega_1(\omega_2)$, когда $\omega_1 > \omega_2$ $(\omega_2 > \omega_1)$, и говорят, что принята интерпретация коллизий в соответствии с данным введенным частичным порядком на множестве значений функции. Появление в описанной выше ситуации несравнимых значений ω_1 и ω_2 (противоречие в определении функции) означает ошибку в задании таблицы. Заметим, что появление ошибочных таблиц вполне возможно на практике, и поэтому каждая таблица должна быть проверена на отсутствие противоречий, если нет уверенности в обратном. Факт несравнимости значений ω_1 и ω_2 можно записать $\omega_1 = \omega_2$, понимая под этим, что приоритеты соответствующих значений равны.

Системы приоритетов значений удобно задавать пятёрками символов следующего вида:

$$a \circ b \circ c,$$

где $a,b,c \in \{0,1,-\}$, $\circ \in \{>,=\}$. Нетрудно подсчитать, что всего возможно 10 различных систем приоритетов.

Систему $a = b = c$ (приоритеты всех значений равны) обычно называют инвариантной. В построенных по данной системе таблицах строки, соответствующие пересекающимся интервалам, должны иметь совпадающие правые части. Название системы связано с тем, что алгоритмы преобразования СЧБФ (например, поглощения и склеивания интервалов) и реализации элементарных операций с таблицами, разработанные для других систем приоритетов, будут применимы и для инвариантных систем. По этой причине инвариантная система используется, например, для того, чтобы обеспечить возможность выполнения алгоритмов, предназначенных для работы с таблицами, записанными в различных системах приоритетов, и избежать при этом преобразовании таблиц из одной системы в другую (ясно, что такое преобразование есть переборная задача). С другой стороны, легко видеть, что таблицы, построенные по инвариантной системе, требуют наибольшего числа задающих интервалов, по сравнению с таблицами, использующими другие системы приоритетов.

Три системы вида $a > b = c$ (значение a подавляет значения b и c, приоритеты которых равны) не интересны с точки зрения задания функций таблицами, не используются и, по-видимому, никогда не будут использоваться на практике. Причина заключается в наличии двух значений с наименьшим приоритетом. Наличие лишь одного такого значения (для определённости – c), как во всех других типах систем, исключая инвариантную, даёт возможность не включать в таблицу строки вида $\overline{\sigma}: c..c$, что часто приводит к существенному сокращению числа задающих интервалов. Заметим, что принятый порядок на множестве задающих интервалов мог бы быть записан как $- > 1 = 0$.

Из систем приоритетов типа $a = b > c$ и $a > b > c$ практически исключительно используются $1 = 0 > -$ (так называемая «листопадовская», примененная в системе синтеза ПЛМ «Листопад» [10]) и $1 > - > 0$ (так называемая «со слабым нулём»). Последняя система удобна тем, что соответствующие таблицы, как и все таблицы, построенные по системам типа $a > b > c$, не требуют проверки на наличие противоречий в определении значений. Недостатком системы со слабым нулём является невозможность быстрого получения инверсных значений функций. Действительно, в этом случае некоторые нули той или иной функции будут подавляться её неопределёнными или единичными значениями и не будут представлены в таблице явно, так же как и интервалы $\overline{\sigma}$ с нулевыми значениями всех функций (строки $\overline{\sigma}: 0...0$).

Алгоритм нахождения разложения Э. Гильберта использует две основных операции: определение мажоранты и взятие отрицания от функции. Его реализация для таблично заданных функций имеет свои особенности: если алгоритм определения мажоранты функции элементарен для любой системы приоритетов значений, то инвертирование функции в системе приоритетов $a > b > c$ может вызвать трудности в силу трудоёмкости явного определения соответствующих задающих интервалов и неоднозначности представления результата.

Для получения совокупности нулевых интервалов функции $\varphi = \overline{f}$ необходимо найти множество $E^n \setminus \{N_f^1 \cap N_f^-\}$ в явном виде. Это можно сделать, если имеется возможность выполнять операцию «вычитания интервалов», т.е. явного определения множества $\overline{\sigma}^1 \setminus \overline{\sigma}^2$.

Результат такой операции может быть записан в различной форме. Мы предлагаем записывать его в виде совокупности p интервалов возрастающей размерности $r, r+1, ..., r+p-1$, где p – мощность множества $\left\{ j : \sigma_j^1 = -,\ \sigma_j^1 \,\&\, \sigma_j^2 \neq -,\ j = \overline{1,n} \right\}$, а r – размерность интервала $\sigma_j^1 \,\&\, \sigma_j^2$. Наш опыт показывает, что такое представление обеспечивает, с одной стороны, компактность представления, а с другой стороны, необходимое быстродействие алгоритма. Единичные интервалы функции φ получаются как результат вычитания всех строк вида $\overline{\sigma}:1$ и $\overline{\sigma}:-$ таблицы функции *f* из строки $-\cdots-:1$.

D. Применение разложения Гильберта к задаче синтеза комбинационных блоков ИМС

Описанный алгоритм нахождения функции Э. Гильберта, преобразованной для СЧБФ, был запрограммирован и в качестве одного из методов был включён в состав системы LORD автоматического многоуровневого синтеза комбинационно-логических схем цифровых блоков интегральных микросхем. Система LORD создавалась авторами в НИИ молекулярной электроники МЭП СССР как одна из компонент САПР БИС «Arc/ws» [11].

Алгоритм был реализован в виде отдельного программного модуля, при этом монотонные функции представлялись в простейшем виде «сумма произведений» с некоторой минимизацией представляющих интервалов. В целях исследования свойств алгоритма были проведены численные эксперименты на наборе тестовых схем программного комплекса SIS [12, 13]. Было проведено сравнение результатов синтеза комбинационных схем средствами SIS с выходными данными алгоритма Гильберта по занимаемой площади *S* и величине критического пути *T* (в условных единицах). Результат сравнения приведен в таблице 1 [14].

Таблица 1

Сравнение результатов синтеза схем средствами SIS и результатов, полученных с помощью алгоритма Гильберта

Схема	SIS		Gilbert		
	S	T	S	T	Число каскадов
Fout	2,3	10,7	1,8	8,9	4
Life	2,57	11,2	2,7	11,4	2
P18	1,9	11,09	2,1	11,06	17
P38	3,1	11,12	2,8	11,01	20
Sqn 7	1,35	9,0	1,6	7,0	8

Данные в таблице имеют демонстрационный характер, поскольку при апробировании метода вне реальной практической системы синтеза ИС не проверялись другие способы доопределения ЧБФ, не проводились ни минимизация представления монотонных функций, ни использование так называемого «фазирования» входных и выходных векторов систем (работа с инверсными представлениями некоторых аргументов всех функций и значений некоторых функций).

Использование при реализации входного и выходного фазирования открывает дополнительные возможности для улучшения характеристик синтезируемых комбинационных схем. Теоретически данные операции не исследованы, так как лежат вне сферы вопросов, традиционно рассматриваемых при исследованиях проблем минимизации БФ.

Применение входного фазирования с вектором $\tilde{\gamma} \in E^n$ геометрически эквивалентно «повороту» булева куба E^n, на котором задана БФ. Происходит преобразование $E^n \to E^n_{\tilde{\gamma}}$ и, соответственно, $f \to f_{\tilde{\gamma}}$ по правилу $\tilde{x} \mapsto \tilde{x} + \tilde{\gamma}$ (здесь «+» – это сложение по модулю 2). Данное преобразование является одним из образующих для классов эквивалентности Шеннона-Поварова.

Можно легко продемонстрировать улучшение сложности представления функций в виде (2) при использовании входного фазирования. Пусть, например, ЧБФ $f(x_1, x_2, x_3)$ задана своими множествами единичных и нулевых значений: $N_f^1 = \{(011),(010),(001)\}$, $N_f^0 = \{(111),(110)\}$. Тогда по предложенному алгоритму имеем $F = Ext(f) = (x_2 \vee x_3) \cdot \overline{x_1 x_2}$. Очевидно, что $F = \overline{x_1} x_2 \vee \overline{x_1} x_3 \vee \overline{x_2} x_3$ и сложность *l*-представления функции в виде суммы произведений равна 8. Если же использовать входной вектор фазы $\tilde{\gamma} = (100)$, то получим $N_{f\tilde{\gamma}}^1 = \{(111),(110),(101)\}$, $N_{f\tilde{\gamma}}^0 = \{(011),(010)\}$ и $F = Ext(f_{\tilde{\gamma}}) = x_1 x_2 \vee x_1 x_3$, т.е. количество интервалов

$l = 3$. Эффективность применения выходного фазирования очевидна и не требует иллюстраций.

Применение фазирования в сочетании с предложенным подходом на основе разложения Э. Гильберта приводит к значительному улучшению параметров синтезируемых схем. Имеются простые эвристики подбора входного вектора фазы, ориентированные на применение рассматриваемого разложения функций по монотонным. Фазирование также можно отдельно производить на каждом этапе рассматриваемого разложения (нахождения очередной монотонной функции Э. Гильберта). Наш опыт показывает, что использование векторов фазы приводит к заметному выигрышу в сложности реально синтезируемых комбинационных схем.

Ниже приведён пример работы алгоритма автоматического синтеза в произвольной логике схемы цифрового блока по его функциональному (поведенческому) описанию. Монотонные функции Э. Гильберта реализованы в виде сумм логических произведений.

На рис. 1 представлен файл kluch.fdt, задающий поведенческое описание работы некоторого разрабатываемого устройства (в данном случае ключа видеоконтроллера). Работа устройства описывается системой двух частичных булевых функций от восьми переменных (строки kinp и kout), которая задаётся в табличной форме. Для описания логики работы данного устройства заданы значения соответствующих функций на 36 интервалах (строка klin) 8-мерного единичного булева куба. Интервалы и значения функций на них указаны 36 строками таблицы; каждая строка предваряется порядковым номером. Принятая интерпретация коллизий значений функций на интервалах указана в строке Interpretation. Она указывает на равные приоритеты значений 1, 0 и - (инвариантная кодировка). Это, в частности, означает, что непустое пересечение могут иметь лишь те интервалы левой части таблицы, которые имеют идентичные правые части (например, интервалы 1 и 2). Строка Phase показывает, что все аргументы и все функции заданы в прямой форме, т.е. указаны их значения, а не отрицания.

На рис. 2 представлен листинг файла kluch.rm, содержащий табличное задание системы из двух ДНФ, являющихся возможными доопределениями исходных функций из файла kluch.fdt. Интерпретация коллизий значений правых частей таблицы производится в соответствии с приоритетом $1 > - > 0$. Число задающих интервалов сократилось до двух.

Рис. 3. представляет собой листинг файла kluch.sht, описывающего схемное решение логики разрабатываемого ключа контроллера в виде списка цепей (netlist).

Строки списка цепей имеют следующий вид:

$$< имя\_цепи >=< имя\_ключа > (< список\_цепей >).$$

Имена цепей состоят из буквы и нескольких (не менее одной) цифр, разделённых точками. Входы схемы имеют имена от x.1 до x.8, выходы – y.0 и y.1, все остальные цепи – внутренние. Мнемоника букв имён:

1) nc означает, что данная цепь есть выход элемента NAND;

2) m означает, что данная цепь реализует монотонную функцию (одну из функций Э. Гильберта);

3) z – прочие цепи.

```
Interpretation        : 1 = 0 = −
kinp =      8
kout =      2
klin =      36
  1 00---1--:00
  2 00----1-:00
  3 11---1--:00
  4 11----1-:00
  5 0110-000:10
  6 0110-001:00
  7 01100010:00
  8 0110-011:00
  9 01100100:10
 10 0110-101:00
 11 01100110:00
 12 0110-111:00
 13 01101000:10
 14 0110-001:00
 15 01101010:00
 16 0110-011:00
 17 01101100:10
 18 0110-101:00
 19 01101110:00
 20 0110-111:00
 21 01010000:01
 22 0101-001:00
 23 01010010:01
 24 0101-011:00
 25 01010100:00
 26 0101-101:00
 27 01010110:00
 28 0101-111:00
 29 01011000:01
 30 0101-001:00
 31 01011010:01
 32 0101-011:00
 33 01011100:00
 34 0101-101:00
 35 01011110:00
 36 0101-111:00
```

Рис. 1. Фрагмент файла kluch.fdt

```
Interpretation        : 1 > − >  0 >

kinp =      8
kout =      2
klin =      2
  1 0101-0-0:01
  2 0110--00:10
```

Рис. 2. Листинг файла kluch.rm

```
1 nc.0 = NAND (x.1, x.3);
2 nc.1 = NAND (x.1, x.2); 3 nc.2 = NAND (x.0, x.1, x.3, x.6);
4 nc.3 = NAND (x.0, x.1, x.2, x.5);
5 nc.4 = NAND (x.1, x.3, x.7);
6 nc.5 = NAND (x.1, x.3, x.5);
7 nc.6 = NAND (x.1, x.2, x.7);
8 nc.7 = NAND (x.1, x.2, x.6);
9 m.0 = NOT (nc.1);
10 m.1 = NOT (nc.0);
11 m.2 = NAND (nc.3, nc.6, nc.7);
12 m.3 = NAND (nc.2, nc.4, nc.5);
13 z.0.1 = NOT (m.2);
14 z.0.0 = NAND (z.0.1, m.0);
15 y.0 = NOT (z.0.0);
16 z.1.1 = NOT (m.3);
17 z.1.0 = NAND (z.1.1, m.1);
18 y.1 = NOT (z.1.0);
```

Рис. 3. Листинг файла kluch.sht

В схеме использовались ключи (функциональные элементы) с именами NAND («многоместный» штрих Шеффера, т.е. отрицание конъюнкций входов) и NOT (отрицание). Список цепей, заключённый в скобки после имени ключа, есть перечисленные через запятую имена цепей, являющиеся входами для данного ключа (список имён аргументов функционального элемента).

В результате для схемной реализации ключа видеоконтроллера, фрагмент описания которого приведен на рис.1. в виде таблицы размером $(8+2)\cdot 36$, потребовалось всего 18 функциональных элементов при «прямой» реализации на элементах NOT и NAND. Учитывая реализацию данной схемы в КМОП-технологии, где (в простейшем случае) приходится по 2 транзистора на каждый вход элемента, функция ключа видеоконтроллера может быть реализована с использованием только 40 транзисторов. Столь значимое сокращение числа задающих интервалов в примере – с 36 до 2 – получено за счёт разработанного авторами алгоритма минимизации СЧБФ для крайне удобной системы приоритетов «со слабым нулём» (исходная система задана в самой неэкономной – «инвариантной» системе).

IV. Заключение

В данной работе предложен перспективный метод и алгоритм синтеза комбинационных логических микросхем на основе разложения Э.Н. Гильберта. Полученные результаты численных экспериментов показывают эффективность разработанного метода для синтеза цифровых схем в реальных проектных базисах, в том числе и для задач проектирования различных подсистем ASIC на логическом уровне.

Литература

[1] Gilbert E.N. Lattice theoretic properties of frontal switching functions // J. Math. Phys. 33, 1954. No. 1. Pp. 57-67 (Русск. пер.: Э.Н. Гильберт. Теоретико-структурные свойства замыкающих переключательных функций // Кибернетический сборник, 1960. – №1. – С. 175-188).

[2] George Grätzer, B.A. Davey, R. and other. General Lattice Theory: Second edition // Springer Science & Business Media, 2002. 663 p.

[3] Касим-Заде О.М. Об одном методе получения оценок сложности схем над произвольным бесконечным базисом // Дискретный анализ и исследование операций, 2004. Сер. 1. С. 41-65.

[4] Kochergin V.V., Mikhailovich A.V. Some Extensions of the Inversion Complexity of Boolean Functions // Cornell University Library, 2015.

[5] Закревский А.Д. Логический синтез каскадных схем / М.: Наука, 1981. 416 с.

[6] Дискретная математика и математические вопросы кибернетики / Под ред. С.В. Яблонского и О.Б. Лупанова. – М.: Наука, 1974. – Т. I. – 313 с.

[7] Гуров С.И. Алгоритм получения разложения Э.Н.Гильберта и его реализация для задач синтеза схем // Труды факультета Вычислительной математики и кибернетики. М.: МАКС Пресс, 2004. – № 18 – С. 108-121.

[8] Gurov S.I. An Algorithm to Construct Gilbert's Decomposition and Its Implementation for the Circuit Design Problem / Computational Mathematics and Modeling, Springer US, vol. 16, no. 4, pp. 370-378.

[9] Гуров С.И. Приведение произвольных булевых функций к монотонным // Журнал «Вычислительная математика и математическая физика», 1991. – Т. 31. – № 1. – С. 143-150.

[10] Бобошко Ю.Г. Об одном подходе к алгоритмам минимизации не полностью определённых булевых функций и его применение к кодированию программируемых логических матриц // Микроэлектроника и полупроводн. приборы, 1979. С. 33-38.

[11] Авдеев Ю.В., Гаврилов С.В., Гуров С.И. и др. САПР заказных БИС на открытых вычислительных системах // Электронная техника. Сер. 3. «Микроэлектроника», 1992. №1. – С. 12-21.

[12] Sentovich E.M., Singh K.J., Lavagno L., Moon C., Murgai R., Saldanha A., Savoj H., Stephan P.R., Brayton R.K. and Sangiovanni-Vincentelli A.L. SIS: A System for Sequential Circuit Synthesis // EECS Department University of California, Berkeley. 1992. P. 52.

[13] Vasicek Z., Sekanina L. A global postsynthesis optimization method for combinational circuits // Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011. Pp. 1-4.

[14] Гуров С.И., Долотова Н.С., Фатхутдинов И.Н. «Некомпактные» задачи распознавания. Синтез схем по Э. Гильберту // Spectral and Evolution Problems (International scientific journal). Международный научный журнал. Simferopol: Taurida National V. Vernadsky University, 2007. Т. 17. С. 37-44.

The algorithm for synthesis of digital ICs based on the Gilbert decomposition

S.I. Gurov, D.I. Ryzhova

Institute for Design Problems in Microelectronics of Russian Academy of Sciences,

sgur@cs.msu.ru, ryzhova_d@ippm.ru

***Keywords* — Boolean function, combinational circuit, stroke (Sheffer) function.**

Abstract

Today the problem of digital IC synthesis optimization is still relevant. In the process of design automation, methods oriented towards the regular representations of combinational circuit parts are preferable in terms of reliability at logical level and testing. Such representations of microsystem logical blocks can be obtained directly from the Gilbert decomposition. This article describes the algorithm of synthesis for combinational circuit that implements an arbitrary Boolean function. It is based on the representation of partial Boolean function *f* in the form $F = M_0 \& \overline{M_1 \& \overline{M_2 \& \ldots \& \overline{M_k}}}$, where $M_0, M_1, M_2, \ldots, M_k$ are monotonic functions, and *F* is the optimal extension for the definition of *f*, in a certain sense. We propose to call functions $M_0, M_1, M_2, \ldots, M_k$ Gilbert functions, and formula $F = M_0 \& \overline{M_1 \& \overline{M_2 \& \ldots \& \overline{M_k}}}$ – Gilbert decomposition for Boolean function (BF) *f*.

Although algorithm for finding Gilbert functions can be restored from the text of this article, the direct use of this algorithm in many practically important cases can cause serious difficulties because of the operation of taking the function negation. In practice, BF and their systems are often specified as tables of a special type, rows of which describe values of the functions on the edges of the Boolean n-dimensional unit cube E^n. In this case, receiving a table for the inverse function can be very laborious. This is due to the fact that these tables allow the presence of intersecting intervals at which the function can take different values. At the same time, to determine the BF value in the area of the intervals intersection, it is necessary to consider the priorities of the values on which the table is based. If the priorities of the values 0 and 1 are not equal, then the inverting of function with *n* variables (function is defined on the *l* intervals) requires the fulfillment of $O(n \cdot l)$ operations. This circumstance and the experience of the authors show that the efficiency of algorithm operation largely depends on realization of inverting operation for table-defined functions.

For this reason, the efficient way of finding negations for commonly used system of priorities of table-defined function values is proposed. On the basis of this decomposition, algorithm of circuit synthesis is implemented. It calculates *f* in the actual design basis of microelectronic LSI circuits.

References

[1] Gilbert E.N. Lattice theoretic properties of frontal switching functions – J. Math. Phys. 33, 1954. No. 1. Pp. 57-67.

[2] George Grätzer, B.A. Davey, R. and other. General Lattice Theory: Second edition // Springer Science & Business Media, 2002. 663 p.

[3] Kasim-Zade O.M. One method for obtaining evaluations of circuit complexity over an arbitrary infinite basis // Diskretnyj analiz i issledovanie operacij, 2004. Vol. 1. Pp. 41-65.

[4] Kochergin V.V., Mikhailovich A.V. Some Extensions of the Inversion Complexity of Boolean Functions // Cornell University Library, 2015.

[5] Zakrevskij A.D. The logical synthesis of cascade circuits. Moscow, Nauka, 1981. 416 p.

[6] Discrete mathematics and mathematical problems of cybernetics. Pod red. S.V. Jablonskogo i O.B. Lupanova. Moscow, Nauka, 1974. Vol. I. 313 p.

[7] Gurov S.I. Algorithm for obtaining E.N.Gilbert expansion and its implementation for the circuits synthesis // Trudy fakul'teta Vychislitel'noj matematiki i kibernetiki. Moscow, MAKS Press, 2004. No. 18. Pp. 108-121.

[8] Gurov S.I. An Algorithm to Construct Gilbert's Decomposition and Its Implementation for the Circuit Design Problem / Computational Mathematics and Modeling, Springer US, vol. 16, no. 4, pp. 370-378.

[9] Gurov S.I. Reduction of arbitrary Boolean functions to monotonic functions // Zhurnal «Vychislitel'naja matematika i matematicheskaja fizika», 1991. Vol. 31. No 1. Pp. 143-150.

[10] Boboshko Ju.G. One approach to the minimization of incompletely defined Boolean functions and its application to coding programmable logic arrays – Mikrojelektronika i poluprovodn. pribory, 1979. Pp. 33-38.

[11] Avdeev Ju.V., Gavrilov S.V., Gurov S.I. i dr. CAD systems for the custom LSI on the open computing systems // Jelektronnaja tehnika. «Mikrojelektronika», 1992. No. 1. Pp. 12-21.

[12] Sentovich E.M., Singh K.J., Lavagno L., Moon C., Murgai R., Saldanha A., Savoj H., Stephan P.R., Brayton R.K. and Sangiovanni-Vincentelli A.L. SIS: A System for Sequential Circuit Synthesis // EECS Department University of California, Berkeley. 1992. P. 52.

[13] Vasicek Z., Sekanina L. A global postsynthesis optimization method for combinational circuits // Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011. Pp. 1-4.

[14] Gurov S.I., Dolotova N.S., Fathutdinov I.N. «Non-compact» recognition problem. Circuit synthesis by E. Gilbert // Spectral and Evolution Problems (International scientific journal). Mezhdunarodnyj nauchnyj zhurnal. Simferopol: Taurida National V. Vernadsky University, 2007. Vol. 17. Pp. 37-44.