УДК 519.7

Ю.В. Поттосин

ЭВРИСТИЧЕСКИЙ МЕТОД ЭНЕРГОСБЕРЕГАЮЩЕГО ПРОТИВОГОНОЧНОГО КОДИРОВАНИЯ СОСТОЯНИЙ АСИНХРОННОГО АВТОМАТА

Рассматривается задача кодирования состояний дискретного автомата с минимизацией переключательной активности элементов памяти в реализующей логической схеме. Предлагается эвристический метод решения этой задачи для асинхронного автомата, где при кодировании состояний устраняются опасные состязания между элементами памяти реализующей схемы.

Введение

Одним из важных критериев оптимизации при проектировании дискретных устройств является величина потребляемой энергии. Это обусловлено, с одной стороны, стремлением увеличить время действия источника энергии в портативных приборах, а с другой – снизить остроту проблемы отвода тепла при проектировании сверхбольших интегральных схем.

Как отмечено в работах [1, 2], потребляемая мощность схемы, построенной на основе КМОП-технологии, пропорциональна интенсивности переключений логических элементов и элементов памяти. В частности, снижения энергопотребления можно добиваться на этапе кодирования состояний автомата, т. е. когда абстрактным символам состояний приписываются булевы векторы. Очевидно, кодировать состояния при этом надо таким образом, чтобы при переходе автомата из одного состояния в другое меняли свое состояние как можно меньше элементов памяти. Этой задаче посвящены, например, работы [3–5], где она решалась для синхронной реализации автомата. Асинхронным автоматам давно уделяется большое внимание [6–8]. Задача энергосбережения для асинхронных автоматов также может быть частично сведена к уменьшению переключательной активности элементов схемы. Энергосберегающее противогоночное кодирование состояний асинхронного автомата рассматривалось в статье [9], где эта задача сводилась к нахождению минимального взвешенного покрытия, и тем самым обеспечивалось получение кода минимальной длины. Поскольку задача нахождения минимального покрытия имеет неполиномиальную сложность, данный метод не всегда позволяет получить решение за практически приемлемое время. Представляемый здесь эвристический метод решения данной задачи не гарантирует получения минимума длины кода состояния и минимума интенсивности переключений элементов памяти, но в ряде случаев позволяет получать решение, близкое к оптимальному по данным критериям, за приемлемое время. Случаев получения наихудших решений по указанным критериям на практических примерах применения предлагаемого метода не выявлено.

1. Модель поведения асинхронного автомата

Моделью поведения логической схемы с памятью является конечный автомат, представляющий собой пятерку (A, B, Q, Ψ, Φ), где A, B и Q – соответственно множества входных сигналов, выходных сигналов и состояний автомата, а Ψ и Φ – функции $\Psi : A \times Q \to Q$ и $\Phi : A \times Q \to B$, называемые соответственно *функцией переходов* и *функцией выходов*. Для состояний $q_i, q_j \in Q$ и входного сигнала $a \in A$ состояние $q_j = \Psi(a, q_i)$ является тем состоянием, в которое автомат переходит из состояния q_i под воздействием входного сигнала a. Конечный автомат функционирует в дискретном времени, т. е. время разбивается на конечные промежутки, называемые *тактами*, в течение каждого из которых автомат переходит из одного состояния в другое и выдает соответствующий выходной сигнал. Рассматриваемая задача позволяет игнорировать функцию выходов Ф, поэтому в дальнейшем она не будет упоминаться.

Здесь рассматривается асинхронная реализация конечного автомата, называемая *асинхронным автоматом*, которая в отличие от синхронной реализации не имеет внешнего источника тактирующих сигналов. Переход от такта к такту происходит в момент изменения входного сигнала. При действии любого входного сигнала асинхронный автомат приходит в некоторое устойчивое состояние, из которого он не выходит до конца действия данного сигнала. При этом должно выполняться требование прямого перехода, которое формально выражается следующим образом: если $\Psi(a, q_i) = q_j$ для фиксированного входного сигнала a и некоторых состояний q_i и q_j, то $\Psi(a, q_j) = q_j$.

Задача кодирования состояний автомата заключается в присвоении каждому состоянию определенного булева вектора $(z_1, z_2, \ldots, z_k)$, называемого *кодом состояния*, который соответствует набору состояний двоичных элементов памяти (триггеров) в логической схеме, где каждый переход из состояния в состояние представляется переключением одного или нескольких триггеров. Естественно, что в реальной электронной схеме такое переключение не может происходить мгновенно и одновременно. Явление неодновременного переключения элементов памяти называется *состязаниями* или *гонками* элементов памяти [10]. Принято называть состязания *неопасными*, если все промежуточные состояния, в которых автомат может оказаться при переходе из одного состояния в другое под воздействием некоторого входного сигнала a, являются неустойчивыми для сигнала a, т. е. при любом порядке переключений элементов памяти автомат из некоторого состояния q_i под воздействием входного сигнала a переходит всегда в состояние $q_j = \Psi(a, q_i)$. Если же при этом автомат может оказаться в некотором устойчивом состоянии q_k, отличном от q_j, то состязания называются *опасными*.

Кодирование состояний, обеспечивающее отсутствие опасных состязаний (гонок), называется *противогоночным*. Естественно, здесь возникает задача минимизации длины кода состояния, приводящая к наименьшему числу элементов памяти в реальной схеме.

Другим критерием оптимизации схемы, как указано выше, является величина потребляемой энергии.

2. Условия отсутствия опасных состязаний

Прежде всего в решении задачи противогоночного кодирования состояний асинхронного автомата необходимо установить условия отсутствия опасных состязаний в реализуемой схеме. Существование опасных состязаний для пары переходов $q_i \to q_j$, $q_k \to q_l$ $(q_j \neq q_l)$ при одном и том же входном сигнале a может привести к тому, что автомат вместо перехода в состояние q_j из состояния q_i окажется в состоянии q_l, которое также является устойчивым при входном сигнале a. Условие отсутствия опасных состязаний для этой пары можно выразить троичным вектором, в котором компоненты соответствуют состояниям автомата и компоненты i и j имеют одно значение, 0 или 1, а компоненты k и l – противоположное ему значение [11]. Остальным компонентам приписывается значение «–». В схеме, реализующей заданный автомат, это условие выполняется триггером, который в процессе одного из переходов рассматриваемой пары хранит состояние 0, а в процессе другого перехода – состояние 1.

Пусть, например, табл. 1 представляет функцию переходов $\Psi(a, q)$ заданного автомата, т. е. является его таблицей переходов. Строкам ее соответствуют состояния автомата, а столбцам – входные сигналы. В клетках таблицы показаны значения функции $\Psi(a, q)$ при соответствующих значениях аргументов a и q. Устойчивые состояния для каждого входного сигнала выделены.

Условие отсутствия опасных состязаний для пары переходов $q_3 \to q_1$, $q_4 \to q_2$ при входном сигнале a_1 выражается вектором (0 1 0 1 – –) либо покомпонентной инверсией этого вектора (1 0 1 0 – –).

На множестве векторов, представляющих условия отсутствия опасных состязаний, устанавливается отношение импликации: троичный вектор **a** имплицирует троичный вектор **b**, если **b** получается из **a** заменой некоторых нулей или единиц значением «–» и, возможно, инвертированием полученного результата. Например, вектор (1 0 – – 1 0 1) имплицирует вектор (1 0 – – – 0 1), а также вектор (0 1 – – – 1 –). Смысл этого отношения в том, что условие, представленное вектором **b**, автоматически выполняется при соблюдении условия, представленного вектором **a**.

Таблица 1

Переходы автомата

	a_1	a_2	a_3	a_4
q_1	$\boldsymbol{q}_1$	q_5	$\boldsymbol{q}_1$	q_5
q_2	$\boldsymbol{q}_2$	q_6	$\boldsymbol{q}_2$	$\boldsymbol{q}_2$
q_3	q_1	$\boldsymbol{q}_3$	q_4	$\boldsymbol{q}_3$
q_4	q_2	q_5	$\boldsymbol{q}_4$	$\boldsymbol{q}_4$
q_5	q_2	$\boldsymbol{q}_5$	q_1	$\boldsymbol{q}_5$
q_6	q_7	$\boldsymbol{q}_6$	q_2	q_3
q_7	$\boldsymbol{q}_7$	q_5	$\boldsymbol{q}_7$	q_3

Все условия отсутствия опасных состязаний в виде описанных векторов составляют троичную матрицу, в которой отсутствуют имплицируемые строки. Эта матрица называется *матрицей условий* [11]. Для автомата, поведение которого описывает табл. 1, матрица условий имеет следующий вид:

$$\boldsymbol{C} = \begin{array}{c} \begin{matrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & – & – & – \\ 0 & 1 & 0 & – & 1 & – & – \\ 0 & – & 0 & – & – & 1 & 1 \\ – & 0 & – & 0 & – & 1 & 1 \\ – & 0 & – & – & 0 & 1 & 1 \\ 0 & 1 & – & – & 0 & 1 & – \\ – & 0 & – & 1 & 1 & 0 & – \\ – & 0 & – & – & 1 & 0 & 1 \\ – & – & 0 & 1 & 1 & – & – \\ – & – & 0 & – & 1 & – & 1 \\ 0 & – & 1 & 1 & 0 & – & – \\ – & 0 & 1 & 1 & – & 0 & – \\ – & – & 0 & 0 & – & – & 1 \\ 0 & – & 1 & – & 0 & 1 & – \\ 0 & – & 1 & – & 0 & – & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix} .$$

Задача противогоночного кодирования состояний асинхронного автомата с минимизацией числа внутренних переменных $z_1, z_2, \ldots, z_k$ сводится к получению минимального числа векторов, таких, что для каждой строки матрицы $\boldsymbol{C}$ среди полученных векторов найдется вектор, имплицирующий данную строку [11]. Результатом кодирования состояний автомата является *матрица кодирования*. Ее строкам соответствуют состояния автомата, а столбцам – внутренние переменные, строки этой матрицы представляют коды соответствующих состояний, а столбцы – указанные выше векторы.

3. Вычисление весов строк матрицы условий

Для снижения интенсивности переключений элементов памяти можно воспользоваться следующими соображениями. Каждому i-му столбцу матрицы кодирования можно поставить в соответствие множество переходов. Данное множество составляют те переходы, которыми связаны состояния автомата с кодами, где переменная z_i имеет различные значения. При таких переходах i-й триггер в реальной схеме, реализующей заданный автомат, меняет свое состояние.

Если удается вычислить вероятности переходов между состояниями, то i-му столбцу матрицы кодирования состояний ставится в соответствие в виде веса w_i вероятность события, заключающегося в том, что происходит некоторый переход, при котором меняется значение перемен-

ной z_i. Поскольку переходы между состояниями автомата являются несовместимыми событиями, эта вероятность равна сумме вероятностей отдельных переходов из множества, соответствующего *i*-му столбцу. Для подсчета вероятностей переходов между состояниями в статье [5] используется метод Чэпмена – Колмогорова, где такие вероятности получаются в результате решения системы линейных уравнений с этими вероятностями в качестве неизвестных. Однако данный метод можно применять только тогда, когда автомат является полностью определенным, а его граф поведения представляет собой сильно связный ориентированный граф. В противном случае столбцу матрицы кодирования состояний автомата можно, например, приписывать мощность связанного с ним множества переходов.

Каждый столбец матрицы кодирования рассматривается как вектор, имплицирующий некоторые строки матрицы условий ***C***. Таким образом, каждой строке матрицы ***C*** можно приписать вес в виде суммы вероятностей переходов между состояниями, которым соответствуют компоненты с различными значениями, отличными от «–». Для упрощения вычислений вместо таких сумм можно использовать пропорциональные им величины, например числители при общем знаменателе вероятностей.

Искомое решение рассматриваемой задачи представляется матрицей кодирования с минимальной суммой весов ее столбцов.

Вероятность перехода автомата в состояние q_j, вызываемого входным сигналом a, когда он находится в состоянии q_i, равна вероятности прихода входного сигнала a. Если имеется несколько входных сигналов, переводящих автомат из состояния q_i в состояние q_j, условная вероятность p'_{ij} такого перехода равна сумме вероятностей этих сигналов, поскольку поступления на вход автомата различных входных сигналов – несовместимые события. Условием является то, что автомат находится в состоянии q_i. Пребывание автомата в состоянии q_i и приход сигнала, переводящего его в состояние q_j, – независимые события. Поэтому абсолютная вероятность p_{ij} перехода из состояния q_i в состояние q_j в течение всего времени работы автомата равна $P_i p'_{ij}$, где P_i – вероятность того, что автомат находится в состоянии q_i.

Вероятности P_i ($i = 1, 2, \ldots, |Q|$) находятся путем решения системы уравнений Чэпмена – Колмогорова, которая имеет следующий вид:

$$\sum_{i=1}^{|Q|} P_i p'_{ij} = P_j, \quad j = 1, 2, \ldots, |Q|,$$

$$\sum_{i=1}^{|Q|} P_i = 1.$$

Вероятности p'_{ij} должны быть известны. Таким образом, решив эту систему уравнений, получим вероятности P_i. Как было сказано раньше, абсолютная вероятность p_{ij} определяется как $p_{ij} = P_i p'_{ij}$.

Асинхронный автомат, поведение которого описывает табл. 1, является полностью определенным, а граф его поведения представляет собой сильно связный ориентированный граф. Следовательно, вероятности переходов между состояниями можно определять по методу Чэпмена – Колмогорова. Допустим, что вероятности входных сигналов данного автомата имеют равномерное распределение. Тогда условная вероятность p'_{ij} перехода (перехода в состояние q_j, когда автомат находится в состоянии q_i) представлена в табл. 2, где строки и столбцы соответствуют состояниям автомата и на пересечении строки q_i и столбца q_j расположена вероятность p'_{ij}. Пустые клетки означают отсутствие перехода между соответствующими состояниями (вероятность такого перехода равна нулю).

Для нахождения вероятностей состояний (вероятностей попадания автомата в те или иные состояния) надо решить следующую систему линейных уравнений (для упрощения вычислений используем величины, пропорциональные условным вероятностям):

$$2P_1 + P_3 + P_5 = 4P_1;$$

$$3P_2 + P_4 + P_5 + P_6 = 4P_2;$$

$$2P_3 + P_6 + P_7 = 4P_3;$$

$$P_3 + 2\,P_4 = 4\,P_4;$$

$$2\,P_1 + P_4 + 2\,P_5 + P_7 = 4\,P_5;$$

$$P_2 + P_6 = 4\,P_6;$$

$$P_6 + 2\,P_7 = 4\,P_7;$$

$$P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 = 1.$$

В результате решения этой системы уравнений получаем $P_1 = 19/135$, $P_2 = 48/135$, $P_3 = 12/135$, $P_4 = 6/135$, $P_5 = 26/135$, $P_6 = 16/135$, $P_7 = 8/135$. Абсолютные вероятности переходов представлены в табл. 3, где строки и столбцы соответствуют состояниям автомата и на пересечении строки q_i и столбца q_j расположена вероятность p_{ij}. Пустые клетки так же, как в табл. 2, соответствуют нулевым вероятностям.

Таблица 2

Условные вероятности переходов

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_1	1/2				1/2		
q_2		3/4				1/4	
q_3	1/4		1/2	1/4			
q_4		1/4		1/2	1/4		
q_5	1/4	1/4			1/2		
q_6		1/4	1/4			1/4	1/4
q_7			1/4		1/4		1/2

Таблица 3

Абсолютные вероятности переходов

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_1	19/270				19/270		
q_2		72/270				24/270	
q_3	6/270		12/270	6/270			
q_4		3/270		6/270	3/270		
q_5	13/270	13/270			26/270		
q_6		8/270	8/270			8/270	8/270
q_7			4/270		4/270		8/270

В табл. 4 представлены веса строк матрицы условий $\boldsymbol{C}$ для рассматриваемого примера, где верхняя строка содержит номера строк, а нижняя – веса соответствующих строк. В качестве веса строки взята величина, пропорциональная вероятности любого из переходов, связывающих состояния, которым соответствуют элементы строки с несовпадающими значениями, отличными от «–». В данном случае взяты числители при общем знаменателе вероятностей.

Таблица 4

Веса строк матрицы условий

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	32	12	40	36	13	20	21	6	4	9	11	4	6	10

4. Получение матрицы кодирования состояний

После построения матрицы условий $\boldsymbol{C}$ требуется получить по возможности минимальное количество совместимых множеств строк этой матрицы, покрывающих все ее строки. Множество строк матрицы $\boldsymbol{C}$ называется совместимым, если существует вектор, имплицирующий все

строки из этого множества [11]. Естественно, две строки матрицы ***C*** следует назвать несовместимыми, если не существует вектора, имплицирующего обе эти строки. Матрица совместимости ***T***, где строки и столбцы соответствуют строкам матрицы ***C*** и элемент на пересечении *i*-й строки и *j*-го столбца имеет значение 0, если *i*-я и *j*-я строки матрицы ***C*** несовместимы, и значение 1 – в противном случае, имеет следующий вид:

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix}.$$

Процесс получения искомого семейства совместимых множеств состоит из двух этапов. На первом этапе предлагаемого процесса находится как можно большее множество попарно несовместимых строк матрицы ***C***. При этом достаточно использовать «жадный» алгоритм, выбирающий для внесения в данное множество каждый раз строку, имеющую наибольшее число несовместимых строк. Если таких строк несколько, выбираем ту, которая обладает минимальным весом. Назовем *степенью совместимости* строки матрицы условий ***C*** число строк, совместимых с этой строкой. В табл. 5 представлены степени совместимости строк матрицы ***C***, которые легко определяются по матрице ***T***.

Таблица 5

Степени совместимости строк матрицы условий

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	8	10	10	9	11	9	9	11	12	9	9	11	10	8

Из табл. 5 видно, что самой низкой степенью совместимости обладают строки 1, 2 и 15. Из них выбираем строку 1, обладающую наименьшим весом. После удаления из рассмотрения строки 1 и совместимых с ней строк получаем матрицу совместимости оставшихся строк:

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} 7 & 11 & 12 & 13 & 14 & 15 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} 7 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix},$$

из которой видно, что все оставшиеся строки обладают одной и той же степенью совместимости, равной 4. Из них выбираем строку 15 с наименьшим весом. Строка 13 осталась единственной строкой, не совместимой ни со строкой 1, ни со строкой 15. Упомянутые строки образуют три одноэлементных множества {1}, {13} и {15}, на основе которых строится искомая совокупность совместимых множеств.

Текущая ситуация в процессе, выполняемом на втором этапе, характеризуется совокупностью совместимых множеств $C_1, C_2, \ldots, C_k$. Каждому множеству C_i ($i = 1, 2, \ldots, k$) соответствуют:

– вектор $\boldsymbol{c}_i$, имплицирующий все строки из C_i, с весом w_i, определяемым, как описано в разд. 3;

– множество строк B_i матрицы условий, совместимых со всеми строками из C_i и не принадлежащих ни одному из множеств $C_1, C_2, \ldots, C_k$;

– множество строк D_i матрицы условий, каждая из которых не совместима хотя бы с одной строкой из C_i и не принадлежит ни одному из множеств $C_1, C_2, \ldots, C_k$.

В начале второго этапа все совместимые множества являются одноэлементными, состоящими из строк матрицы $\boldsymbol{C}$, выделенных на первом этапе. Очередной шаг процесса представляет собой выбор пары $(C_i, \mathbf{b}_i)$, где $\mathbf{b}_i \in B_i$, включение вектора $\mathbf{b}_i$ в множество C_i и коррекцию вектора $\boldsymbol{c}_i$, связанную с данным включением. После этого $\mathbf{b}_i$ удаляется из всех $B_1, B_2, \ldots, B_k$ и $D_1, D_2, \ldots, D_k$. Из тех же множеств удаляются все строки, которые оказались имплицируемыми вектором $\boldsymbol{c}_i$. Они также вносятся в C_i. Если $B_i = \varnothing$, то C_i вносится в решение и исключается из дальнейшего рассмотрения. Этот шаг повторяется, и процесс заканчивается, когда все множества $B_1, B_2, \ldots, B_k$ окажутся пустыми.

При коррекции вектора $\boldsymbol{c}_i$ в множество D_i могут переноситься строки из множества B_i, оказавшиеся не имплицируемыми вектором $\boldsymbol{c}_i$. Если при этом некоторая строка будет присутствовать во всех множествах $D_1, D_2, \ldots, D_k$, то из нее образуется одноэлементное совместимое множество C_{k+1} и соответственно множества B_{k+1} и D_{k+1}, т. е. k увеличивается на единицу.

Выбор пары $(C_i, \mathbf{b}_i)$ осуществляется по следующим правилам:

1. Выбирается вариант с наименьшим числом строк, вносимых в D_i.

2. Выбирается вариант с наименьшим увеличением веса w_i вектора $\boldsymbol{c}_i$.

При этом второе правило применяется только тогда, когда согласно первому правилу имеется несколько вариантов выбора.

Как отмечено в работе [6], число k, которое становится длиной кода состояния, растет пропорционально величине $\log_2|Q|$. Выбор пары $(C_i, \mathbf{b}_i)$ происходит за время, пропорциональное произведению $k\,l$, где l – число строк матрицы условий, не вошедших в множества $C_1, C_2, \ldots, C_k$. С каждым шагом число l уменьшается, по крайней мере, на единицу. Если m – число строк и n – число столбцов матрицы условий, т. е. $l < m$ и $n = |Q|$, то, как нетрудно подсчитать, время выполнения второго этапа ограничено величиной, пропорциональной произведению $m^2 \log_2 n$.

В рассматриваемом примере начальная ситуация представляется следующими множествами и векторами:

$C_1 = \{1\};$	$\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$;	$B_1 = \{2,3,4,5,6,8,9,10\};$	$D_1 = \{7,11,12,14\};$	$w_1 = 6;$
$C_2 = \{13\};$	$\boldsymbol{c}_2 = (- - 0\ 0 - - 1)$;	$B_2 = \{2,3,4,5,6,7,8,10,11,12,14\};$	$D_2 = \{9\};$	$w_2 = 4;$
$C_3 = \{15\};$	$\boldsymbol{c}_3 = (0 - 1 - 0 - 1)$;	$B_3 = \{4,5,6,7,9,11,12,14\};$	$D_3 = \{2,3,8,10\};$	$w_3 = 10.$

По первому правилу выбирается пара $(C_3, 14)$. Этот вариант является единственным, когда из B_i в D_i переносится всего одна строка. Вес вектора $\boldsymbol{c}_3$ при этом не меняется. В результате соответствующего шага ситуация представится следующим образом:

$C_1 = \{1\};$	$\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$;	$B_1 = \{2,3,4,5,6,8,9,10\};$	$D_1 = \{7,11,12\};$	$w_1 = 6;$
$C_2 = \{13\};$	$\boldsymbol{c}_2 = (- - 0\ 0 - - 1)$;	$B_2 = \{2,3,4,5,6,7,8,10,11,12\};$	$D_2 = \{9\};$	$w_2 = 4;$
$C_3 = \{14,15\};$	$\boldsymbol{c}_3 = (0 - 1 - 0\ 1\ 1)$;	$B_3 = \{4,5,6,7,9,11\};$	$D_3 = \{2,3,8,10,12\};$	$w_3 = 10.$

На следующем шаге тоже удается воспользоваться только первым правилом, поэтому приходим к ситуации

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5,6,8,10\}$; $D_1 = \{7,11,12\}$; $w_1 = 6$;
$C_2 = \{13\}$; $\boldsymbol{c}_2 = (- - 0\ 0 - - 1)$; $B_2 = \{2,3,4,5,6,7,8,10,11,12\}$; $D_2 = \varnothing$; $w_2 = 4$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,6,7\}$; $D_3 = \{2,3,8,10,11,12\}$; $w_3 = 16$.

Далее по первому правилу выбираются три варианта $(C_1, 2)$, $(C_2, 3)$ и $(C_2, 10)$, когда должны быть перенесены из множества совместимых в множество несовместимых строк две строки. Последний вариант обладает минимальным приращением веса имплицирующего вектора. Применяется второе правило, и очередная ситуация выглядит так:

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5,6,8\}$; $D_1 = \{7,11,12\}$; $w_1 = 6$;
$C_2 = \{10,13\}$; $\boldsymbol{c}_2 = (- - 0\ 0\ 1 - 1)$; $B_2 = \{2,3,4,6,8,11,12\}$; $D_2 = \{5,7\}$; $w_2 = 7$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,6,7\}$; $D_3 = \{2,3,8,11,12\}$; $w_3 = 16$.

На следующем шаге после применения первых двух правил приходим к ситуации вида

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5,6,8\}$; $D_1 = \{7,12\}$; $w_1 = 6$;
$C_2 = \{10,11,13\}$; $\boldsymbol{c}_2 = (0 - 1\ 1\ 0 - 0)$; $B_2 = \{4,6,8,12\}$; $D_2 = \{2,3,5,7\}$; $w_2 = 7$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,6,7\}$; $D_3 = \{2,3,8,12\}$; $w_3 = 16$.

Далее, применяя первые два правила, выбираем вариант $(C_2, 6)$, при котором получается вектор $\boldsymbol{c}_2$, имплицирующий, кроме строк, присутствующих в C_2, и строки 6, еще строку 8. Рассматриваем следующую ситуацию:

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5\}$; $D_1 = \{7,12\}$; $w_1 = 6$;
$C_2 = \{6,8,10,11,13\}$; $\boldsymbol{c}_2 = (0\ 1\ 1\ 1\ 0\ 1\ 0)$; $B_2 = \varnothing$; $D_2 = \{2,3,4,5,7,12\}$; $w_2 = 34$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,7\}$; $D_3 = \{2,3,12\}$; $w_3 = 16$.

Данная ситуация отличается от предыдущих ситуаций тем, что $B_2 = \varnothing$ (вектор $\boldsymbol{c}_2$ не имплицирует ни одну из строк, не вошедших в множества C_1, C_2, C_3), а строка 12 вошла во все множества D_1, D_2, D_3. Поэтому множество C_2 вносим в решение, исключаем его из дальнейшего рассмотрения и вводим $C_4 = \{12\}$ вместе с соответствующими B_4 и D_4. Текущая ситуация примет тогда вид

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5\}$; $D_1 = \{7,12\}$; $w_1 = 6$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,7\}$; $D_3 = \{2,3,12\}$; $w_3 = 16$;
$C_4 = \{12\}$; $\boldsymbol{c}_4 = (- 0\ 1\ 1 - 0 -)$; $B_4 = \{2,3,7\}$; $D_4 = \{4,5\}$; $w_2 = 11$.

Применяя первые два правила, получаем приращения множеств в следующем порядке: $C_4 = \{3,12\}$, $C_1 = \{1,2\}$, $C_4 = \{3,7,12\}$, $C_3 = \{4,5,9,14,15\}$. Результатом описанного процесса являются следующие множества и соответствующие им векторы с весами:

$C_1 = \{1,2\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1\ 1 - -)$; $w_1 = 38$;
$C_2 = \{6,8,10,11,13\}$; $\boldsymbol{c}_2 = (0\ 1\ 1\ 1\ 0\ 1\ 0)$; $w_2 = 34$;
$C_3 = \{4,5,9,14,15\}$; $\boldsymbol{c}_3 = (0\ 0\ 1\ 0\ 0\ 1\ 1)$; $w_3 = 48$;
$C_4 = \{3,7,12\}$; $\boldsymbol{c}_4 = (0\ 1\ 0\ 0\ 0\ 1\ 1)$; $w_2 = 32$.

Из четырех вариантов замены символов «–» в векторе $\boldsymbol{c}_1$ выбираем комбинацию 1 1, дающую минимум веса w_1. Окончательным решением рассматриваемого примера является матрица кодирования с суммарным весом 164:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \end{array}.$$

В результате применения описанного в работе [11] точного метода противогоночного кодирования состояний с минимизацией длины кода без учета интенсивности переключений элементов памяти могут быть получены следующие матрицы кодирования:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array},$$

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array}.$$

Первая из этих шести матриц совпадает с точностью до перестановки столбцов с матрицей, полученной с помощью описанного эвристического метода. Остальные пять матриц имеют соответственно веса 154, 178, 156, 196 и 171. Вес матрицы кодирования можно рассматривать как оценку качества решения. Чем меньше вес, тем лучше решение. В работе [5] применена оценка качества решения рассматриваемой задачи по критерию, который выражается формулой $\sum w_{ij}(d_{ij} - 1)$. Здесь d_{ij} – расстояние по Хэммингу между кодами состояний q_i и q_j; w_{ij} – вес перехода между состояниями q_i и q_j, суммирование ведется по всем парам состояний. Чем меньше эта оценка, тем лучше результат. Для приведенных матриц она равна соответственно 39, 35, 91, 37, 120 и 43. Обе эти оценки не противоречат друг другу, и отсюда следует, что описанный метод дает не худший результат.

Для оценки времени выполнения метода можно использовать следующие соображения. В работе [6] в качестве верхней границы длины кода состояния при отсутствии опасных состязаний приведена величина $2\log_2\lceil |Q| \rceil$, где $|Q|$ – число состояний автомата, а $\lceil a \rceil$ – ближайшее сверху целое число, не меньшее a. Эту величину можно принять за верхнюю границу числа множеств $C_1, C_2, \ldots, C_k$. Число вариантов, перебираемых на каждом шаге, меньше чем

$m \times 2 \log_2\lceil |Q| \rceil$, где m – число строк матрицы условий, уменьшающееся на каждом шаге не менее чем на единицу. Таким образом, по грубой оценке, время выполнения метода не достигает величины, пропорциональной $m^2 \times 2 \log_2\lceil |Q| \rceil$. Время выполнения точного метода растет экспоненциально относительно роста числа m. Такой трудностью обладает задача о покрытии, решение которой используется в рассматриваемом методе.

Заключение

Предлагаемый эвристический метод энергосберегающего противогоночного кодирования состояний асинхронного автомата рассчитан на использование его в автоматизированной системе логического проектирования. На примере показано, что описанный метод может дать решение, сравнимое по качеству с решением, получаемым точным методом. При этом время, затрачиваемое на получение решения, значительно меньше времени, за которое выполняется точный метод, поскольку точный метод сводит данную задачу к задаче о покрытии, имеющей неполиномиальную сложность.

Список литературы

1. Мурога, С. Системное проектирование сверхбольших интегральных схем : в 2 кн. Кн. 1 / С. Мурога. – М. : Мир, 1985. – 288 с.

2. Pedram, M. Power minimization in IC design: Principles and applications / M. Pedram // ACM Trans. Design Automat. Electron. Syst. – 1996. – Vol. 1. – P. 3–56.

3. Kashirova, L. State assignment of finite state machine for decrease of power dissipation / L. Kashirova, A. Keevallik, M. Meshkov // Second Intern. Conf. Computer-Aided Design of Discrete Devices, CAD DD'97, Minsk, Republic of Belarus, November 12–14, 1997. – Minsk : National Academy of Sciences of Belarus, Institute of Engineering Cybernetics, 1997. – Vol. 1. – P. 60–67.

4. Sudnitson, A. Partition search for FSM low power synthesis / A. Sudnitson // Fourth Intern. Conf. Computer-Aided Design of Discrete Devices, CAD DD'2001, Minsk, November 14–16, 2001. – Minsk : National Academy of Sciences of Belarus, Institute of Engineering Cybernetics, 2001. – Vol. 1. – P. 44–49.

5. Закревский, А.Д. Алгоритмы энергосберегающего кодирования состояний автомата / А.Д. Закревский // Информатика. – 2011. – № 1(29). – С. 68–78.

6. Ангер, С. Асинхронные последовательностные схемы / С. Ангер. – М. : Наука, 1977. – 400 с.

7. Синтез асинхронных автоматов на ЭВМ ; под ред. А.Д. Закревского. – Минск : Наука и техника, 1975. – 184 с.

8. Автоматизированное проектирование цифровых устройств ; под ред. С.С. Бадулина. – М. : Радио и связь, 1981. – 240 с.

9. Поттосин, Ю.В. Энергосберегающее противогоночное кодирование состояний асинхронного автомата / Ю.В. Поттосин // Информатика. – 2015. – № 2(46). – С. 94–101.

10. Закревский, А.Д. Алгоритмы синтеза дискретных автоматов / А.Д. Закревский. – М. : Наука, 1971. – 512 с.

11. Закревский, А.Д. Логические основы проектирования дискретных устройств / А.Д. Закревский, Ю.В. Поттосин, Л.Д. Черемисинова. – М. : Физматлит, 2007. – 592 с.

Поступила 13.06.2016

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: pott@newman.bas-net.by

Yu.V. Pottosin

A HEURISTIC METHOD FOR LOW POWER RACE-FREE

STATE-ASSIGNMENT OF AN ASYNGHRONOUS AUTOMATON

The problem of race free state-assignment of an asynchronous automaton minimizing the switching activity of memory elements in an implementing circuit is considered. A heuristic method to solve this problem for an asynchronous automaton is suggested where the critical races between memory elements of the implementing circuit are eliminated.