УДК 518.5

А.Д. Закревский

АЛГОРИТМЫ ЭНЕРГОСБЕРЕГАЮЩЕГО КОДИРОВАНИЯ СОСТОЯНИЙ АВТОМАТА

Предлагаются два алгоритма кодирования состояний конечного автомата, при котором сокращаются затраты энергии в реализующей автомат логической схеме. Первый из них основан на оптимальном отображении графа переходов автомата в булево пространство кодирующих переменных, второй учитывает вероятности переходов.

Введение

Рассматриваемая в данной статье проблема возникает при проектировании дискретных устройств и связана с постановкой двух практически важных задач: во-первых, ее положительное решение уменьшает тепловыделение при работе электронных схем, что позволяет увеличивать в них плотность размещения элементов; во-вторых, с уменьшением затрат энергии в дискретном устройстве растет период его исправного функционирования между перезарядками, что существенно для мобильных устройств, а также для устройств с однократной зарядкой.

Как утверждается в работах [1, 2], в современной комплементарной металл-оксидной полупроводниковой технологии (КМОП-технологии) общая мощность P потребления энергии дискретным устройством делится на три главные компоненты:

P_{sc} (short-cut power) обусловлена током, протекающим от источника на землю через пару p-МОП/n-МОП одновременно открытых транзисторов;

P_{lk} (leakage power) обусловлена током утечки – постоянным малым током, протекающим через закрытые транзисторы;

P_{dyn} (dynamic power) является мощностью переключений, при которых заряжаются и разряжаются нагрузочные емкости. При нормальном функционировании корректно спроектированных КМОП-цепей она составляет основную долю всего энергопотребления [3] и определяется по формуле

$$P_{dyn} = ½\, C_l\, A_{sw}\, V_{dd}^2\, f_{clk},$$

где C_l – нагрузочная емкость элементов цепи; V_{dd} – напряжение источника; f_{clk} – частота импульсов синхронизации; A_{sw} – переключательная активность схемы, определяемая ожидаемым числом переключений элементов в течение одного такта синхронизации.

Значения параметров C_l, V_{dd} и f_{clk} определяются используемой КМОП-технологией и являются константами, поэтому минимизация динамической мощности P_{dyn} сводится к минимизации переключательной активности A_{sw}. Эта задача должна решаться при проектировании устройства на различных уровнях, среди которых важную роль играет уровень схемной реализации конечного автомата.

1. Схемная реализация конечного автомата

Конечный автомат является абстрактной математической моделью дискретного устройства и представляет собой совокупность следующих пяти формальных объектов:

A – множество входных символов, или входной алфавит;

B – множество выходных символов, или выходной алфавит;

S – множество состояний, или внутренний алфавит;

Ψ – функция переходов $A \times S \to S$;

Φ – функция выходов $A \times S \to B$.

Слово *конечный* в названии автомата подчеркивает, что все три множества, входящие в состав данной модели, конечны. В том виде, в каком эта модель здесь представлена, она называется *автоматом Мили*. Другой разновидностью данной модели является *автомат Мура*, от-

личающийся от автомата Мили тем, что функция выходов Ф не зависит от входного символа, т. е. представляет собой отображение $S \to B$.

Значением функции $\Psi(a, s)$ является состояние s^+, в которое переходит автомат из состояния s, если на его вход подан символ a. Значением функции $\Phi(a, s)$ автомата Мили является выходной символ b, выдаваемый автоматом в состоянии s при поступлении на его вход символа a, а значением функции $\Phi(s)$ автомата Мура – выходной символ b, который выдается автоматом, находящимся в состоянии s.

Поведение, описанное данной моделью, может быть по-разному реализовано в дискретном устройстве. При *синхронной реализации* моменты времени, когда определяются состояние, в которое переходит устройство, а в случае автомата Мили и выходной символ, зафиксированы. Технически это осуществляется введением генератора синхронизирующих сигналов, которые инициируют соответствующие действия. Период следования таких сигналов не должен быть меньше чем время, необходимое для завершения переходных процессов в устройстве. Реализованный таким образом конечный автомат называется *синхронным автоматом*. Ниже рассматривается именно этот тип конечного автомата.

Реализующее конечный автомат устройство рассматривается как комбинация двух взаимодействующих блоков: логической комбинационной схемы, реализующей некоторую систему булевых функций, и регистра, на котором фиксируется очередное состояние устройства (рис. 1).

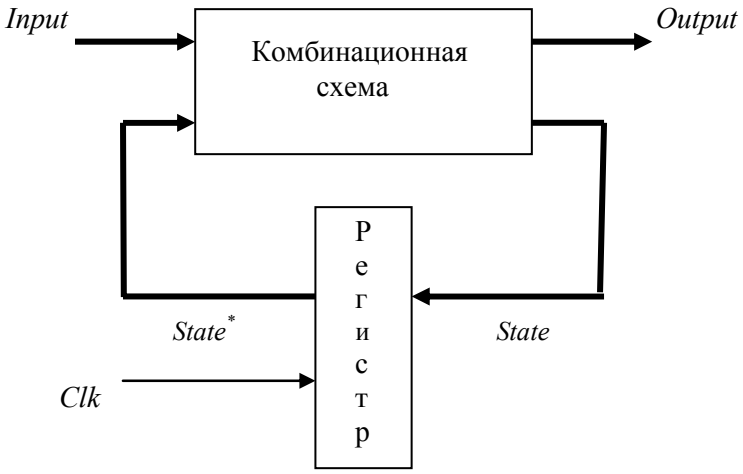

Рис. 1. Двухблочная структура дискретного устройства

Входная информация подается на устройство булевым вектором *Input*, выходная представлена значением булева вектора *Output*. Булевы векторы *State** и *State* задают состояние устройства: предшествующее и последующее. Значения этих векторов фиксируются на регистре соответствующими наборами состояний триггеров. Векторы *Output* и *State* являются функциями векторов *Input* и *State**, реализуемыми комбинационной схемой.

Устройство работает в синхронном режиме, определяемом импульсом синхронизации *Clk*. В момент подачи этого импульса текущее значение вектора *State* передается вектору *State**. При этом изменяются состояния триггеров, соответствующих компонентам векторов *State** и *State* с несовпадающими значениями, на что затрачивается некоторая энергия. Таким образом устройство переходит в следующее состояние.

Средним числом триггеров, изменяющих свое состояние за один такт (при одном переходе), измеряется *переключательная активность* дискретного устройства, играющая существенную роль в оценке его энергопотребления. В свою очередь, эта характеристика зависит от кодирования состояний автомата, т. е. от того, какими значениями вектора *State** представляются текущие состояния.

2. Оценка переключательной активности схемы для заданного кодирования состояний автомата

Состояния автомата, связанные непосредственным переходом, называются *соседними*. Соответствующие им коды (значения булевых векторов *State* и *State**) различаются в некоторых компонентах, число которых принято называть *расстоянием по Хэммингу* между этими кодами. Такие расстояния между соседними состояниями учитываются при оценке затрат энергии на соответствующие переходы.

Ожидаемое число переключений элементов схемы в течение одного такта синхронизации можно оценить, воспользовавшись автоматной моделью схемы – матрицей либо графом переходов синхронного автомата. В каждый такт происходит один переход, при котором очередное состояние S_i меняется на последующее S_k. Обозначим через $\boldsymbol{s}_i$ и $\boldsymbol{s}_k$ булевы векторы, представляющие коды этих состояний, а через $\boldsymbol{d}_{i\,k}$ – вектор, представляющий сумму этих кодов (по модулю 2):

$$\boldsymbol{d}_{i\,k} = \boldsymbol{s}_i \oplus \boldsymbol{s}_k .$$

Компоненты вектора $\boldsymbol{d}_{i\,k}$ принимают значение 0 при совпадении значений соответствующих компонент векторов s_i и s_k и значение 1 в противном случае. Число единиц в векторе $\boldsymbol{d}_{i\,k}$ обозначим через $w(\boldsymbol{d}_{i\,k})$. Оно называется *весом* вектора $\boldsymbol{d}_{i\,k}$ и представляет хэммингово расстояние между векторами $\boldsymbol{s}_i$ и $\boldsymbol{s}_k$.

Значения векторов $\boldsymbol{s}_i$ и $\boldsymbol{s}_k$ фиксируются на двух следующих друг за другом наборах состояний триггеров, образующих регистр – память автомата. Число переключаемых при переходе триггеров оказывается равным $w(\boldsymbol{d}_{i\,k})$. В идеальном случае при каждом переходе будет переключаться максимум один триггер (при $S_k = S_i$ переключений нет), тогда переключательная активность A_{sw} будет минимальна и не превысит числа переходов в автомате.

3. Алгоритм кодирования состояний по методу квадратов

В работе [4] задача энергосберегающего кодирования состояний автомата была сведена к оптимизированному отображению неориентированного графа переходов G с m вершинами в булево пространство $M = \{0, 1\}^n$, размерность которого n равна целому, ближнему сверху к $\log_2 m$. За критерий оптимальности принято число ребер, отображенных на ребра гиперкуба, соответствующего булеву пространству M. Такие ребра связывают соседние элементы булева пространства, т. е. булевы n-векторы, различающиеся значениями ровно в одной компоненте. Положим, что чем больше таких ребер, тем лучше размещение. В идеальном случае все ребра графа проектируются на ребра соответствующего n-мерного гиперкуба. Однако в общем случае это невозможно. Заметим, что информация об ориентации переходов в автомате при этом не учитывается, поскольку она оказывается несущественной.

Был предложен эвристический визуальный метод решения данной задачи, названный *методом квадратов*. Он основан на использовании матрицы смежности графа и карты Карно и заключается в построении последовательности конфигураций из ребер и квадратов, образующих фрагменты гиперкуба. Квадратом называется четырехреберный цикл в графе.

Ниже описывается более формальный алгоритм решения этой же задачи, ориентированный на компьютерную реализацию. Рассматриваемый в данном алгоритме граф переходов представляется симметричной булевой *матрицей смежности* $\boldsymbol{G}$ размером $m \times m$. Элемент матрицы смежности $g_i^{\,k} = 1$, если и только если вершины i и k связаны некоторым ребром – обозначим его через (i–k). Другими словами, соответствующие этим вершинам состояния автомата связаны некоторым переходом.

Например, показанный на рис. 2 граф переходов автомата с 11 состояниями представляется следующей матрицей смежности:

$$G = \begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 3 \\ 0 & \boldsymbol{1} & 0 & 0 & \boldsymbol{1} & 0 & 0 & 1 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 5 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 7 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 8 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 10 \\ 0 & \boldsymbol{1} & 0 & 0 & \boldsymbol{1} & 0 & 0 & 0 & 0 & 1 & 0 & 11 \end{array} .$$

Два ребра графа назовем *параллельными*, если они не содержат общих вершин и принадлежат оба некоторому квадрату. Это условие легко проверяется визуальным методом [5] – достаточно посмотреть на граф. В предлагаемом здесь методе оно представляется более формально, в терминах матрицы смежности.

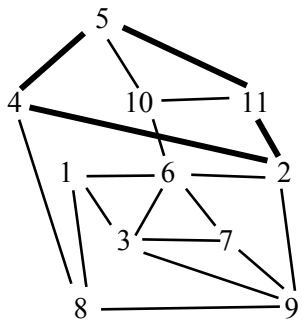

Рис. 2. Граф переходов

Ребра графа (*i*–*j*) и (*k*–*l*), задаваемые матричными элементами $g_i^{\,j}$ и $g_k^{\,l}$, параллельны, если матрица смежности ***G*** принимает значение 1 также на элементах $g_i^{\,l}$ и $g_k^{\,j}$. В этом случае ребра принадлежат квадрату, который удобно задать циклической четверкой вершин (*i*, *j*, *k*, *l*) и в котором оказываются смежными вершины *i* и *j*, *j* и *k*, *k* и *l*, *l* и *i*. Ему принадлежит также пара параллельных ребер (*j*–*k*) и (*l*–*i*) .

В данном примере параллельными оказываются ребра (4–5) и (11–2), принадлежащие отмеченному на рис. 2 квадрату, равно как и ребра (5–11) и (2–4). Они представляются четырьмя матричными элементами на пересечении строк 4 и 11 со столбцами 2 и 5. Это элементы $g_4^{\,2}$, $g_4^{\,5}$, $g_{11}^{\,2}$ и $g_{11}^{\,5}$, они отмечены в матрице курсивом. Квадрат в целом представляется выражением (4–5–11–2).

Предлагаемый алгоритм заключается в последовательном выборе ребер графа (единичных элементов матрицы ***G***) и отображении их в булево пространство $M = \{0, 1\}^n$ (n-мерный булев гиперкуб) путем кодирования соответствующих пар вершин булевыми векторами с n компонентами.

На первом шаге выбирается некоторая пара параллельных ребер. Например, в рассматриваемом графе можно выбрать ребро (1–3) и параллельное ему ребро (7–6). Вершины первого из них кодируются соседними кодами, различающимися значением в одной компоненте. Операцией смены значений другой компоненты из этих кодов получаются коды вершин другого ребра.

Так, вершины 1 и 3 кодируются векторами 0000 и 0001, а вершины 7 и 6 – векторами 0010 и 0011. Эти ребра образуют квадрат (1–3–7–6), куда входят также ребра (3–7) и (6–1). Таким образом, в результате данной операции на булев гиперкуб отображаются сразу четыре ребра графа. Представим эту операцию следующим выражением, в котором звездочкой отмечена компонента с изменяемым значением:

(1–3) → (7–6)

```
1 0000
3 0001
   *
7 0011
6 0010
```

На очередном шаге находится *свободное ребро* (с пока не закодированными вершинами), параллельное некоторому из отображенных ребер. Коды его вершин получаются из кодов соответствующих вершин отображенного ребра сменой значений одной компоненты так, чтобы они были новыми (ранее не встречались). В результате кодируются две новые вершины и отображаются три ребра.

Порядок выбора ребер может быть различным; например, среди подходящих ребер можно выбирать ребро с минимальным номером. В этом случае в данном примере на втором шаге выбираются отображенное ребро (1–3) и параллельное ему свободное ребро (9–8). Вместе с ними в гиперкуб отображаются ребра (1–8) и (9–3).

Аналогично выполняются последующие операции. Эта работа алгоритма представляется следующей цепочкой операций:

(1–3) → (7–6)	(1–3) → (9–8)	(8–9) → (2–4)	(2–4) → (5–11)
1 0000	1 0000	8 0100	2 1100
3 0001	3 0001	9 0101	4 1101
*	*	*	*
7 0011	9 0101	2 1101	11 1110
6 0010	8 0100	9 1100	5 1110

В результате выполнения данной цепочки в гиперкуб отобразились 13 ребер графа и все его вершины, кроме одной с номером 10. Ее можно закодировать, рассматривая коды соседних с ней вершин 5, 6 и 11 и выбирая среди свободных (еще не использованных) кодов ближайший к ним. Таким оказывается булев вектор 0110, соседний с кодами вершин 5 и 6 – векторами 1110 и 0010. Этот выбор приводит к отображению в гиперкуб еще двух ребер – (5–10) и (6–10).

Отображенные ребра отмечены утолщением на рис. 3.

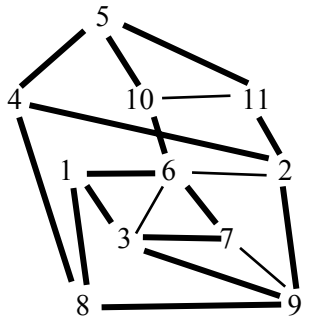

Рис. 3. Граф переходов с отображенными ребрами

Окончательный результат представляется таблицей полученных кодов:

```
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

1 3 6 7 8 9 10          4 2 5 11
```

Остались не отображенными в булев гиперкуб четыре ребра графа: 2–6, 3–6, 7–9 и 10–11. Их веса (хэмминговы расстояния между кодами вершин) равны соответственно 4, 2, 2 и 2.

Назовем *дефектом отображения* разность между суммой хэмминговых расстояний на ребрах и числом ребер. В данном примере он равен 6.

Дефект отображения можно снизить, рандомизируя выбор свободных ребер и кодирование их вершин – выбор переменной, по которой коды этих вершин будут соседними с кодами вершин отображенного ребра, параллельного выбранному.

4. Алгоритм вычисления вероятностей переходов в конечном автомате

Дефект отображения в какой-то степени аппроксимирует переключательную активность дискретного устройства. Более точную оценку ее можно получить, учитывая вероятности каждого перехода. Их можно вычислить методом Чэпмена – Колмогорова [5], исходной информацией для которого служит упрощенное описание автомата, иллюстрируемое примером автомата с шестью состояниями s_0, s_1, s_2, s_3, s_4, s_5 и тремя входными булевыми переменными a, b, c (табл. 1). Заметим, что в более полном табличном описании фигурирует четвертый столбец, в котором представлены соответствующие значения выходного булева вектора *Output*, однако при решении задачи кодирования состояний автомата эта информация не используется.

Таблица 1

Текущее состояние s_i	Условие перехода $f_{i\,k}$	Следующее состояние s_k
s_0	$a \vee b\,c$ $\bar{a}\,(\,\bar{b} \vee \bar{c})$	s_1 s_4
s_1	b $\bar{b}$	s_0 s_5
s_2	$\bar{c} \vee \bar{a}\,b$ $a\,c$ $\bar{a}\;\bar{b}\,c$	s_0 s_3 s_5
s_3	$a\,b\,\bar{c}$ $\bar{b}\;\bar{c}$ $c \vee \bar{a}\,b$	s_1 s_2 s_4
s_4	$\bar{a}$ a	s_2 s_4
s_5	$a\,b\,c$ $\bar{a} \vee \bar{b} \vee \bar{c}$	s_0 s_3

Переход между двумя состояниями совершается, если задающая условие перехода булева функция принимает значение 1. Очевидно, что дизъюнкция таких функций для каждого из текущих состояний автомата равна единице.

Описываемый ниже способ оценки переключательной активности схемы основан на предположении, что вероятности значений аргументов распределены равномерно – каждая входная переменная принимает значение 1 с вероятностью 1/2 независимо друг от друга. Отсюда следует, что вероятность принятия булевой функцией значения 1 можно оценить относительным числом наборов значений аргументов, на которых соответствующая булева функция принимает значение 1.

Например, вероятность выполнения условия $a\,b\,\bar{c} = 1$ полагается равной 1/8 , а условия $a \vee b\,c = 1$ равной 5/8. Такие вероятности выполнения условий $f_{i\,k} = 1$ равны *условным вероятностям* $c(i, k)$ соответствующих переходов из состояния S_i в состояние S_k (необходимым условием перехода является нахождение схемы в состоянии S_i).

Определяемые условные вероятности $c(i, k)$ переходов для рассматриваемого примера представлены в табл. 2. Исходя из них требуется вычислить *абсолютные вероятности* $p(i, k)$ этих же переходов, называемые ниже просто вероятностями.

Таблица 2

	s_0	s_1	s_2	s_3	s_4	s_5
s_0		5/8			3/8	
s_1	4/8					4/8
s_2	5/8			2/8		1/8
s_3		1/8	2/8		5/8	
s_4			4/8		4/8	
s_5	1/8			7/8		

Вероятности находятся в предположении, что автомат функционирует достаточно долго и каждый его активный переход срабатывает достаточно большое количество раз – в этом случае применимы статистические методы. Это значит, что из любого состояния автомат может быть переведен в любое другое при подаче на его вход соответствующей последовательности входных воздействий. Данное условие сводится к другому, эквивалентному ему, но более простому для программной проверки. Достаточно рассмотреть некоторое одно состояние и убедиться, что из него можно прийти в любое другое, а из любого другого прийти к нему. При рассмотрении автомата с одним начальным и одним конечным состояниями данное условие удовлетворяется ведением дополнительного безусловного перехода – из конечного состояния в начальное.

По аналогии с известным в электротехнике законом Кирхгофа можно утверждать, что сумма вероятностей переходов в некоторое состояние равна сумме вероятностей переходов из этого состояния. Очевидно также, что вероятности переходов, исходящих из некоторого состояния, пропорциональны условным вероятностям этих переходов. На основе данных утверждений можно составить систему уравнений

$$p(i, k) = p(i)\, c(i, \underset{k}{k}),\ p(i) = \underset{i}{\Sigma}\, p(i, \underset{k}{k}),\ \Sigma\, p(i) = 1,\ \Sigma\, c(i, k) = 1 \text{ (для любого } i),$$

которая связывает следующие величины:

$p(i)$ – вероятность состояния S_i;

$c(i, k)$ – вероятность перехода в состояние S_k при условии, что автомат находится в состоянии S_i;

$p(i, k)$ – абсолютную вероятность перехода из состояния S_i в состояние S_k.

Поскольку искомые вероятности переходов задаются формулой $p(i, k) = p(i)\, c(i, k)$ с известным значением параметра $c(i, k)$, остается вычислить вероятности $p(i)$ нахождения автомата в состоянии S_i. Они представляют корни системы линейных уравнений, представленной в компактной матричной форме

$$\boldsymbol{C}\, \boldsymbol{p}^{\mathrm{T}} = \boldsymbol{p}^{\mathrm{T}},$$

где $\boldsymbol{C}$ – матрица условных вероятностей переходов; $\boldsymbol{p}$ – вектор вероятностей состояний; T – символ транспонирования. При этом предполагается, что $\Sigma\, p(i) = 1$.

Например, для заданного автомата (см. табл. 1) строится следующая система уравнений, связывающих искомые вероятности состояний:

$$\begin{aligned}
&p(1)\,4/8 + p(2)5/8 + p(5)1/8 = p(0),\\
&p(0)\,5/8 + p(3)1/8 = p(1),\\
&p(3)\,2/8 + p(4)4/8 = p(2),\\
&p(2)\,2/8 + p(5)7/8 = p(3),\\
&p(0)\,3/8 + p(3)5/8 + p(4)4/8 = p(4),\\
&p(1)\,4/8 + p(2)1/8 = p(5).
\end{aligned}$$

Полученная система линейных уравнений решается далее классическим методом Гаусса, планомерно использующим две операции эквивалентного преобразования системы:

– любое уравнение можно умножить на некоторый коэффициент;

– любое уравнение можно заменить его суммой с некоторым другим уравнением.

Перед решением системы уравнений их можно привести к более простой форме уравнений с целочисленными коэффициентами (умножением на 2^n) и нулевой правой частью, а затем представить ее матрицей коэффициентов, принимающей для рассматриваемого примера следующее значение:

```
 0  1  2  3  4  5
 8 -4 -5  0  0 -1
-5  8  0 -1  0  0
 0  0  8 -2 -4  0
 0  0 -2  8  0 -7
-3  0  0 -5  4  0
 0 -4 -1  0  0  8
```

$8p(0) - 4p(1) - 5p(2) - 1p(5) = 0$ и т. д.

По методу Гаусса вычисляются вероятности $p(i)$ состояний автомата $p(0) = 0{,}1878$, $p(1) = 0{,}1326$, $p(2) = 0{,}1768$, $p(3) = 0{,}1215$, $p(4) = 0{,}2928$, $p(5) = 0{,}0884$, затем по формуле $p(i, k) = p(i)\, c(i, k)$ находятся абсолютные вероятности переходов из состояния S_i в состояние S_k. Например, $p(0, 1) = 0{,}1878 \times 5/8 = 0{,}1174$.

Представим абсолютные вероятности таких переходов следующей матрицей:

	0	1	2	3	4	5
0	0,0000	0,1174	0,0000	0,0000	0,0704	0,0000
1	0,0663	0,0000	0,0000	0,0000	0,0000	0,0663
2	0,1105	0,0000	0,0000	0,0442	0,0000	0,0221
3	0,0000	0,0152	0,0304	0,0000	0,0760	0,0000
4	0,0000	0,0000	0,1464	0,0000	0,1464	0,0000
5	0,0110	0,0000	0,0000	0,0773	0,0000	0,0000

Из вероятностей $p(i, k)$ *ориентированных* переходов (из состояния S_i в состояние S_k) легко найти вероятности $p{+}(i, k)$ *неориентированных* переходов между состояниями S_i и S_k :

$$p{+}(i, k) = p(i, k) + p(k, i) .$$

В данном примере они получают следующие значения:

	1	2	3	4	5
0	0,1837	0,1105		0,0704	0,0110
1			0,0152		0,0663
2			0,0746	0,1464	0,0221
3				0,0760	0,0773

5. Кодирование состояний с учетом вероятностей переходов

Представим исходную для рассматриваемой задачи информацию графически, приписав соответствующие веса ребрам неориентированного графа переходов (рис. 4). Они пропорциональны вероятностям неориентированных переходов и могут быть заданы с точностью до двух десятичных знаков, достаточной для изложения предлагаемого ниже алгоритма кодирования состояний.

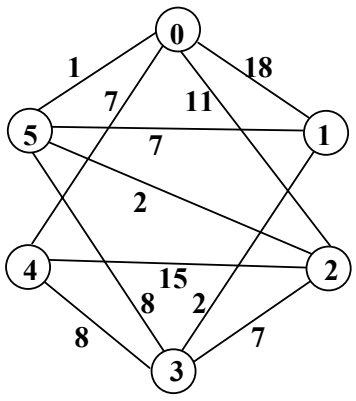

Рис. 4. Граф переходов

Положим, что компоненты используемых кодов нумеруются справа налево (номерами 0, 1, 2, …) и задают значения соответствующих кодирующих переменных z_0, z_1, z_2, … , число которых достаточно и минимально.

Вначале выбирается ребро с максимальным весом и кодируются инцидентные ему вершины – минимальными соседними кодами. Так выбирается ребро (0–1) и вершины 0 и 1 получают соответственно коды 000 и 001.

На каждом из последующих шагов рассматриваются ребра, инцидентные некоторым из ранее закодированных вершин, из них выбирается ребро с максимальным весом. Одна из инцидентных ему вершин уже закодирована. Код второй вершины получается из кода первой путем изменения значения одной компоненты – самой правой среди тех, смена значения которых приводит к еще не использованному коду.

Таким образом на втором шаге выбирается ребро (0–2) и вершина 2 получает код 010. Представим эту операцию символически как Ch(0–2), 2 cod 010. Аналогично выполняются следующие операции. Полная работа алгоритма при решении данного примера выражается последовательностью

Ch(0–1), 0 cod 000,
1 cod 001;
Ch(0–2), 2 cod 010;
Ch(2–4), 4 cod 011;
Ch(4–3), 3 cod 111;
Ch(3–5), 5 cod 110.

В результате закодированы все шесть вершин графа и пять выбранных ребер (показанных на рис. 5 жирными отрезками) отобразились на соответствующие ребра булева трехмерного куба. Кроме них отображено и ребро (2–5), поскольку коды его вершин оказались соседними.

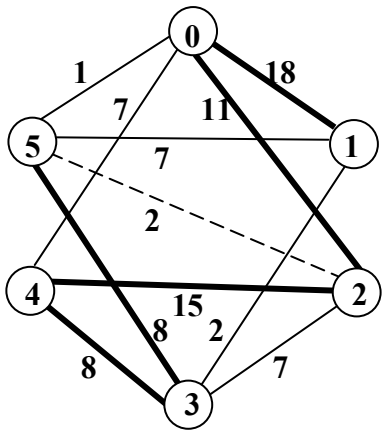

Рис 5. Ребра графа, отображенные в булев куб

Рассмотрим оставшиеся неотображенными ребра, их веса, величину *k* для каждого из них (хэммингово расстояние минус единица) и дефект *d* отображения ребра. Сложив эти дефекты, получим дефект *D* отображения графа в целом:

Ребро	Вес	*k*	*d*
(0–5)	1	1	1
(0–4)	7	1	7
(1–3)	2	1	2
(1–5)	7	2	14
(2–3)	7	1	7
			D = 31

Заметим, что, если последний шаг алгоритма Ch(3–5), 5 cod 110 заменить на Ch(3–5), 5 cod 101, дефект значительно сократится:

Ребро	Вес	k	d
(0–5)	1	1	1
(0–4)	7	1	7
(1–3)	2	1	2
(2–5)	2	2	4
(2–3)	7	1	7
			$D = 21$

Отсюда следует, что эффективность алгоритма можно существенно повысить, используя процедуры рандомизированного кодирования вершин.

6. Сравнение с методом квадратов

Решая данный выше пример методом квадратов, рассмотрим матрицу смежности

	0	1	2	3	4	5
0	0	1	1	0	1	1
1	1	0	0	1	0	1
2	1	0	0	1	1	1
3	0	1	1	0	1	1
4	1	0	1	1	0	0
5	1	1	1	1	0	0

Лучшее решение по этому методу находится цепочкой операций

(0–1) → (3–4)	(0–1) → (5–2)
0 0000	0 0000
1 0001	1 0001
*	*
3 0011	5 0101
4 0010	2 0100

Остаются неотображенными четыре ребра (меньше, чем по другому методу), однако дефект отображения больше:

Ребро	Вес	k	d
(0–5)	1	1	1
(2–3)	7	2	14
(2–4)	15	1	15
(3–5)	8	1	8
			$D = 38$

Видно, что учет вероятностей переходов в автомате приводит к лучшим результатам.

Заключение

Предложенные в статье алгоритмы могут быть использованы в практике энергосберегающего проектирования дискретных устройств, существенно облегчая решение рассматриваемых в этой области задач.

Автор выражает благодарность В.И. Романову за программную реализацию данных алгоритмов.

Список литературы

1. Najm, F. Power estimation in sequential circuits / F. Najm, S. Goel, I.N. Hajj // Proc. of the 32th Design Automation Conf. – USA, 1995. – P. 635–640.

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. Symbolic synthesis of clock-gating logic for power optimization of control-oriented synchronous networks / L. Benini [et al.] // Proc. European Design and Test Conf. – 1997. – P. 514–520.

4. Закревский, А.Д. Энергосберегающее кодирование состояний конечного автомата. Метод квадратов / А.Д. Закревский // Информатика. – 2005. – № 4 (8). – С. 105–113.

5. Macii, E. High-level Power Modeling, Estimation and Optimization / E. Macii, M. Pedram, F. Somenzi // IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems. – 1998. – Vol. 17, № 11. – P. 1061–1079.

Поступила 10.11.10

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: zakrevskij@tut.by

A.D. Zakrevskij

ALGORITHMS OF ENERGY-SAVING CODING OF AN AUTOMATON STATES

Two algorithms are suggested for energy-saving coding of finite automaton states. The first of them is based on optimal mapping of the transition graph onto the Boolean space of coding variables whereas the second one takes into consideration the transition probabilities.