ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2010, № 6, с. 70–79

ДИСКРЕТНЫЕ СИСТЕМЫ

УДК 681.3.14./21

ПРЕОБРАЗОВАНИЕ АВТОМАТА ТИПА МИЛИ В АВТОМАТ ТИПА МУРА ПУТЕМ РАСЩЕПЛЕНИЯ ВНУТРЕННИХ СОСТОЯНИЙ

© 2010 г. А. С. Климович, В. В. Соловьев

Белоруссия, Минск, Высший государственный колледж связи,

Польша, Белосток, Белостокский технический ун-т

Поступила в редакцию19.08.09 г., после доработки 24.02.10 г.

Рассматривается задача преобразования автомата типа Мили в эквивалентный автомат типа Мура. Данная задача часто встречается в практике инженерного проектирования, когда необходимо исключить непосредственную зависимость значений выходных переменных от изменения значений входных переменных. Особенностью предлагаемого подхода является использование операции расщепления внутренних состояний конечного автомата, а также представление конечного автомата в виде списка переходов. Результаты экспериментальных исследований показывают, что при переходе от автоматов типа Мили к автоматам типа Мура число внутренних состояний увеличивается в среднем в 1.96 раза, а число переходов – в 2.05 раза; стоимость реализации автоматов типа Мура с помощью индустриального пакета приблизительно в 2 раза больше, чем автоматов Мили. Указываются актуальные направления исследований методов минимизации и синтеза автоматов типа Мура.

Введение. В практике инженерного проектирования последовательностных устройств наибольшее распространение получили две модели конечного автомата: типа Мили [1] и типа Мура [2]. Существенное отличие между этими двумя моделями заключается в том, что выходные переменные автомата Мура определяются только внутренним состоянием конечного автомата и не зависят от входных переменных, а в автомате Мили выходные переменные формируются под влиянием как внутреннего состояния конечного автомата, так и значений входных переменных. Другими словами, при нахождении конечного автомата в некотором устойчивом состоянии значения выходных переменных автомата Мура не изменяются, а значения выходных переменных автомата Мили могут корректироваться с учетом значений входных переменных. В ряде случаев практического использования конечных автоматов последнее свойство автомата Мили является недопустимым. Поэтому возникает необходимость перехода от автомата типа Мили к эквивалентному автомату типа Мура [3]. Кроме того, функционирование автомата типа Мура более прозрачно для понимания, поэтому он чаще применяется в практике инженерного проектирования разработчиками цифровой аппаратуры.

Новым толчком повышенного интереса к модели автомата Мура послужил переход в качестве элементной базы к программируемым логическим интегральным схемам (ПЛИС). В самом общем виде ПЛИС можно представить в виде совокупности макроячеек или логических элементов, организованных определенным образом. Одна из архитектурных особенностей ПЛИС – возможность программирования либо комбинационного, либо регистрового выхода каждой макроячейки или логического элемента [4]. Еще одна архитектурная особенность ПЛИС проявляется в том, что каждая макроячейка (логический элемент) имеет обратную связь с внутренней логикой. Эти две особенности позволяют выходные сигналы автомата Мура буферизировать в регистрах, а формируемые выходные наборы использовать в качестве некоторой части кодов внутренних состояний конечного автомата. Такая модель конечного автомата получила название автомата класса С (соответственно автомат типа Мили стали называть автоматом класса А, а автомат Мура – автоматом класса В) [5].

Исследование модели автомата класса С показало, что он значительно превосходит автоматы классов А и В как по стоимости реализации, так и по быстродействию [6]. Еще одним преимуществом подобного автомата по сравнению с другими моделями конечных автоматов [4] является то, что его реализация возможна на любых типах ПЛИС. Однако не всякий конечный автомат есть автомат типа Мура, поэтому использование модели автомата класса С требует необходимости преобразования автомата типа Мили в эквивалентный автомат типа Мура.

Традиционные методы перехода от автомата типа Мили к автомату типа Мура ориентированы на преобразование вручную конечных автоматов небольшого размера [7]. В качестве исходного представления конечного автомата в традиционных подходах обычно применяются таблицы переходов и выходов, которые весьма неудобны и громоздки при программной реализации, а также

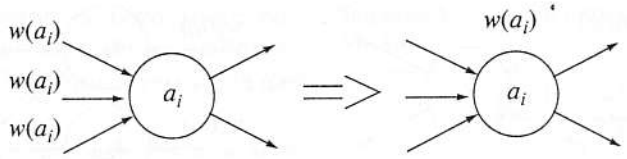

Рис. 1. Преобразование внутреннего состояния автомата типа Мили в состояние автомата типа Мура в случае формирования одинаковых наборов выходных переменных

при рассмотрении автоматов большого размера. (Чего, например, стоит определение столбцов таблицы переходов, которые соответствуют всем возможным наборам входных переменных.)

В предлагаемом методе для перехода от автомата типа Мили к автомату типа Мура привлекается операция расщепления внутренних состояний конечного автомата. Отметим, что эта операция относится к операциям эквивалентных преобразований конечного автомата и не изменяет алгоритм его функционирования [8–10]. Кроме того, в разработанном методе в качестве представления конечного автомата рассматривается список переходов (а не таблицы переходов, как в традиционных методах), который широко применяется при описании конечных автоматов в большинстве известных языков проектирования (например, VHDL, Verilog, AHDL, Abel, KISS2 и др.). Еще одна особенность развиваемого подхода состоит в том, что во всех алгоритмах последовательно рассматривается только одно внутреннее состояние конечного автомата, что позволяет избежать построения структур данных большой размерности и, как следствие, делает возможной работу с конечными автоматами достаточно большого размера.

Отличие обсуждаемого подхода от метода [8] заключатся в том, что в [8] решается задача расщепления внутренних состояний с целью уменьшения числа аргументов функций возбуждения элементов памяти конечного автомата, тогда как нами операция расщепления применяется для преобразования автомата типа Мили в автомат типа Мура.

В последнее время конечные автоматы по-прежнему привлекают внимание специалистов. Ограничимся перечислением основных направлений данных исследований. В [11, 12] задача кодирования внутренних состояний конечных автоматов решается с целью снижения потребляемой мощности, а в [13] изложены различные стили кодирования, допустимые в языке проектирования VHDL. В [14] снижение потребляемой мощности осуществляется путем отображения конечного автомата на блоки внутренней памяти ПЛИС. В [15, 16] решается задача минимизации неполностью определенных конечных автоматов. Метод [17] декомпозиции конечных автоматов ориентирован на минимизацию стоимости реализации и повышения быстродействия, а метод функциональной декомпозиции [18] полезен при реализации конечных автоматов на блоках памяти ПЛИС. С увеличением размеров реализуемых устройств актуальной становится задача минимизации описания конечных автоматов [19, 20]. В [21] затронуты вопросы использования внутренних микропроцессоров ПЛИС при реализации конечных автоматов, которые также применяются для решения отдельных задач вычислительных систем [22] и при проектировании систем на кристалле [23].

1. Суть предлагаемого подхода. Конечный автомат будем характеризовать набором L входных переменных множества $X = \{x_1, ..., x_L\}$, числом N выходных переменных множества $Y = \{y_1, ..., y_N\}$, количеством M внутренних состояний, образующих множество $A = \{a_1, ..., a_M\}$, а также параметром $R = \text{intlog}_2 M$ – минимальным числом разрядов кода, достаточных для кодирования внутренних состояний конечного автомата. Функционирование конечного автомата описывается с помощью списка переходов, который представляет собой таблицу, состоящую из четырех столбцов: a_m, $X(a_m, a_s)$, a_s и $Y(a_m, a_s)$. Каждая строка списка переходов соответствует одному переходу конечного автомата. В столбце a_m списка переходов присутствует состояние, в котором начинается переход (present state); столбец $X(a_m, a_s)$ содержит набор входных переменных, который инициирует данный переход; в столбце a_s записывается состояние перехода (next state); столбец $Y(a_m, a_s)$ включает набор выходных переменных, принимающих единичное значение на данном переходе. Поскольку в автомате типа Мура выходные переменные зависят только от состояния, в котором находится конечный автомат, то для него последний столбец часто обозначают как $Y(a_m)$. Множество переходов из одного и того же состояния будем называть группой переходов. В основу предлагаемого подхода положены следующие две идеи:

1) если на всех переходах в некоторое состояние a_i, $a_i \in A$, формируется один и тот же выходной набор $w(a_i)$, то такое состояние можно считать состоянием автомата Мура, а выходной набор $w(a_i)$ приписать непосредственно состоянию a_i (рис. 1);

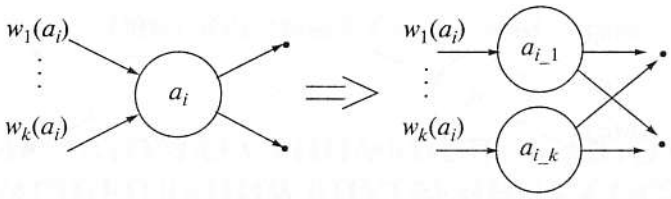

Рис. 2. Расщепление внутреннего состояния автомата типа Мили при переходе к автомату типа Мура в случае формирования различных наборов выходных переменных

2) если на переходах в некоторое состояние a_i, $a_i \in A$, формируются различные выходные наборы $w_1(a_i)$, ..., $w_K(a_i)$, то состояние a_i можно расщепить на K состояний a_{i_1}, ..., a_{i_K} таким образом, чтобы переходу в a_{i_k}, $k = \overline{1, K}$, соответствовал только один выходной набор $w_k(a_i)$ (рис. 2).

Кроме того, чтобы не работать с массивами большого размера и упростить программную реализацию, все алгоритмы метода будем строить таким образом, чтобы в каждый момент времени выполнения некоторого алгоритма рассматривалось только одно внутреннее состояние конечного автомата.

2. Метод преобразования автомата типа Мили в автомат типа Мура. Пусть $P(a_i)$ – множество переходов из состояния a_i, а $C(a_i)$ – множество переходов в состояние a_i, $a_i \in A$. Тогда с учетом сделанных замечаний алгоритм перехода от автомата типа Мили к автомату типа Мура выглядит следующим образом.

Алгоритм 1 (расщепления состояний конечного автомата для преобразования автомата типа Мили в автомат типа Мура).

Ш а г 1. Последовательно перебираются состояния конечного автомата. Если рассмотрены все его состояния, то выполняется переход к шагу 5, иначе – к шагу 2.

Ш а г 2. Для очередного состояния a_i, $i = \overline{1, M}$, по списку переходов определяется множество $W(a_i) = \{w_1(a_i), ..., w_K(a_i)\}$ наборов выходных переменных, получаемых на переходах в состояние a_i.

Ш а г 3. Если $K = 1$, то осуществляется переход к шагу 1; иначе – к шагу 4.

Ш а г 4. Состояние a_i расщепляется на K состояний a_{i_1}, ..., a_{i_K} таким образом, чтобы на переходах в каждое состояние a_{i_k} формировался только один выходной набор $w_k(a_i)$, $k = \overline{1, K}$.

Ш а г 4.1. Вводится K новых состояний a_{i_1}, ..., a_{i_K}.

Ш а г 4.2. Определяются подмножества переходов $C(a_{i\_1})$, ..., $C(a_{i\_K})$ в состояния a_{i_1}, ..., a_{i_K}. Для этого множество $C(a_i)$ разбивается на подмножества $C(a_{i\_1})$, ..., $C(a_{i\_K})$ образуемые так, чтобы всем переходам каждого подмножества $C(a_{i\_k})$ соответствовали выходные наборы $w_k(a_i)$, $k = \overline{1, K}$.

Ш а г 4.3. Находятся подмножества переходов $P(a_{i\_1})$, ..., $P(a_{i\_K})$ из состояний a_{i_1}, ..., a_{i_K}. Для этого полагается $P(a_{i\_k}) := P(a_i)$ для всех $k = \overline{1, K}$.

Ш а г 4.4. Корректируется множество состояний конечного автомата, следуя правилам.

$$A := A \backslash \{a_i\};$$
$$A := A \cup \{a_{i\_1}, ..., a_{i\_K}\};$$
$$M := M + K.$$

Ш а г 4.5. Выполняется соответствующая корректировка списка переходов с помощью *алгоритма 2.*

Ш а г 4.6. Реализуется переход к шагу 1.

Ш а г 5. Формируется список переходов автомата Мура с помощью *алгоритма 3.*

Ш а г 6. Конец.

Алгоритм 2 (корректировка списка переходов, выполняемая на шаге 4.5 алгоритма 1).

Ш а г 1. В столбце a_s списка переходов состояние a_i заменяется одним из состояний a_{i_1}, ..., a_{i_K}. Для этого последовательно просматриваются строки списка переходов. Если в столбце a_s записано состояние a_i, то вместо него вводится состояние a_{i_k}, для которого выполняется $Y(a_m, a_s) = w_k(a_i)$.

Ш а г 2. Группа переходов из состояния a_i повторяется K раз, всякий раз заменяя в столбце a_{im} состояние a_i на одно из состояний a_{i_1}, ..., a_{i_K}.

Ш а г 3. Из списка переходов исключается группа переходов из состояния a_i.

Ш а г 4. Конец.

Алгоритм 3 (формирование списка переходов автомата Мура на шаге 5 алгоритма 1).

Ш а г 1. Для каждого состояния a_i, $a_i \in A$, определяется набор выходных переменных $w(a_i)$, соответствующий переходам в состояние a_i. (В результате выполнения шагов 2–4 алгоритма 1 каждому состоянию a_i будет отвечать только один выходной набор $w(a_i)$.)

Ш а г 2. Последовательно рассматриваются состояния конечного автомата a_i, $a_i \in A$.

Ш а г 3. В группе переходов из состояния a_i содержимое столбца $Y(a_m, a_s)$ заменяется набором $w(a_i)$ выходных переменных, формируемых на переходах в состояние a_i.

Ш а г и 4. Пункты 2–3 повторяются для всех состояний конечного автомата.

Ш а г 5. Конец.

П р и м е р. Пусть исходный автомат типа Мили представлен списком переходов, приведенном в табл. 1. В результате выполнения шага 2 алгоритма 1 имеем

$$W(a_1) = \{w_1(a_1), w_2(a_1), w_3(a_1)\},$$

где $w_1(a_1) = \{y_1\}$, $w_2(a_1) = \{y_3\}$ и $w_3(a_1) = \{y_4\}$. Поскольку $K = |W(a_1)| = 3 > 1$, то осуществляется расщепление состояния a_1 на состояния a_{1_1}, a_{1_2} и a_{1_3}. В результате выполнения шага 4 алгоритма 1 получается список переходов, приведенный в табл. 2. Поскольку для состояния a_2 имеем $K = |W(a_2)| = 1$, то реализуется шаг 5 алгоритма 1. После того, как отработал шаг 1 алгоритма 3, получим $w(a_{1\_1}) = \{y_1\}$, $w(a_{1\_2}) = \{y_3\}$, $w(a_{1\_3}) = \{y_4\}$ и $w(a_2) = \{y_2\}$. Проведя замены в списке переходов содержимого столбца $Y(a_m, a_s)$ в результате выполнения шагов 2, 3 алгоритма 3, имеем список переходов автомата Мура (табл. 3).

Отметим, что точно такой же итог возникает и в случае применения традиционных методов перехода от автомата типа Мили к автомату типа Мура.

3. Результаты экспериментальных исследований. В качестве исходных данных для оценки эффективности рассмотренного метода перехода от конечного автомата типа Мили к автомату типа Мура были выбраны эталонные примеры, разработанные в центре MCNC (Microelectronics Center of North Carolina) [24]. Результаты экспериментальных исследований приведены в табл. 4, где FSM – название эталонного примера; s_0 – число внутренних состояний исходного конечного автомата; p_0 – число переходов исходного конечного автомата; s_1 – число внутренних состояний автомата Мура; p_1 – число переходов автомата Мура; s_1/s_0 и p_1/p_0 – отношения соответствующих параметров; Mid – среднеарифметическое значение параметров.

Анализ табл. 4 показывает, что шесть исходных конечных автоматов Donfile, Modulo12, S1, S1a, S27 и S8 уже являются автоматами типа Мура и для этих примеров значения параметров s_1 и p_1 совпадают со значениями s_0 и p_0 для исходного конечного автомата. При переходе от конечного автомата типа Мили к автомату типа Мура с помощью предложенного метода среднее увеличение числа состояний (отношение s_1/s_0) составляет 1.96 раза, а среднее увеличение числа переходов – 2.05 раза. Наибольшее возрастание числа состояний (в 6.75 раза) наблюдается для примера Tav, а максимальный рост числа переходов (в 7.04 раза) – для примера Ex1.

Таблица 1. Список переходов исходного автомата типа Мили

a_m	$X(a_m, a_s)$	a_s	$Y(a_m, a_s)$
a_1	$\bar{x}_1$	a_1	y_1
a_1	x_1	a_2	y_2
a_2	$\bar{x}_2$	a_1	y_3
a_2	x_2	a_1	y_4

Таблица 2. Список переходов после расщепления состояния a_1

a_m	$X(a_m, a_s)$	a_s	$Y(a_m, a_s)$
a_{1_1}	$\bar{x}_1$	a_{1_1}	y_1
a_{1_1}	x_1	a_2	y_2
a_{1_2}	$\bar{x}_1$	a_{1_1}	y_1
a_{1_2}	x_1	a_2	y_2
a_{1_3}	$\bar{x}_1$	a_{1_1}	y_1
a_{1_3}	x_1	a_2	y_2
a_2	$\bar{x}_2$	a_{1_2}	y_3
a_2	x_2	a_{1_3}	y_4

Таблица 3. Список переходов автомата типа Мура

a_m	$X(a_m, a_s)$	a_s	$Y(a_m)$
a_{1_1}	$\bar{x}_1$	a_{1_1}	y_1
a_{1_1}	x_1	a_2	y_1
a_{1_2}	$\bar{x}_1$	a_{1_1}	y_3
a_{1_2}	x_1	a_2	y_3
a_{1_3}	$\bar{x}_1$	a_{1_1}	y_4
a_{1_3}	x_1	a_2	y_4
a_2	$\bar{x}_2$	a_{1_2}	y_2
a_2	x_2	a_{1_3}	y_2

В большинстве примеров имеет место пропорциональное увеличение числа состояний и переходов (исключением является пример Ex1). При этом средняя разница между увеличениями этих показателей составляет 11%.

4. Сравнение сложности реализации конечных автоматов типа Мили и типа *Мура*. Может возникнуть естественный вопрос: как переход от автомата типа Мили к автомату типа Мура отражается на сложности реализации конечного автомата? Прежде всего, рост числа состояний конечного автомата Мура по сравнению с аналогичным автоматом типа Мили

Таблица 4. Результаты экспериментальных исследований алгоритма преобразования автомата типа Мили в автомат типа Мура

FSM	L	N	Mealy		Moore		s_1/s_0	p_1/p_0
			s_0	p_0	s_1	p_1		
Bbara	4	2	10	60	12	72	1.20	1.20
Bbsse	7	7	16	56	24	94	1.50	1.68
Bbtas	2	2	6	24	9	36	1.50	1.50
Beecount	3	4	7	28	10	40	1.43	1.43
Cse	7	7	16	91	29	163	1.81	1.79
Dk14	3	5	7	56	26	208	3.71	3.71
Dk15	3	5	4	32	17	136	4.25	4.25
Dk16	2	3	27	108	75	300	2.78	2.78
Dk17	2	3	8	32	16	64	2.00	2.00
Dk27	1	2	7	14	10	20	1.43	1.43
Dk512	1	3	15	30	24	48	1.60	1.60
Donfile	*2*	*1*	*24*	*96*	*24*	*96*	*1.00*	*1.00*
Ex1	9	19	18	233	78	1641	4.33	**7.04**
Ex2	2	2	19	72	23	84	1.21	1.17
Ex3	2	2	10	36	13	40	1.30	1.11
Ex4	6	9	14	21	18	28	1.29	1.33
Ex5	2	2	9	32	13	44	1.44	1.38
Ex6	5	8	8	34	14	61	1.75	1.79
Ex7	2	2	10	36	14	48	1.40	1.33
Keyb	7	2	19	170	20	183	1.05	1.08
Lion	2	1	4	11	5	14	1.25	1.27
Lion9	2	1	9	25	11	31	1.22	1.24
Mc	3	5	4	10	8	20	2.00	2.00
Modulo12	*1*	*1*	*12*	*24*	*12*	*24*	*1.00*	*1.00*
Planet	7	19	48	115	95	224	1.98	1.95
Pma	8	8	24	73	49	132	2.04	1.81
S1	*8*	*6*	*20*	*107*	*20*	*107*	*1.00*	*1.00*
S1488	8	19	48	251	168	912	3.50	3.63
S1494	8	19	48	250	168	1030	3.50	4.12
S1a	*8*	*6*	*20*	*107*	*20*	*107*	*1.00*	*1.00*
S208	11	2	18	153	37	309	2.06	2.02
S27	*4*	*1*	*6*	*34*	*6*	*34*	*1.00*	*1.00*
S298	3	6	218	1096	332	1669	1.52	1.52
S386	7	7	13	64	23	127	1.77	1.98
S420	19	2	18	137	37	282	2.06	2.06
S510	19	7	47	77	73	133	1.55	1.73
S8	*4*	*1*	*5*	*20*	*5*	*20*	*1.00*	*1.00*
S820	18	19	25	232	70	613	2.80	2.64
S832	18	19	25	245	70	707	2.80	2.89
Sand	11	9	32	184	84	611	2.63	3.32
Shiftreg	1	1	8	16	16	32	2.00	2.00
Sse	7	7	16	56	24	94	1.50	1.68
Styr	9	10	30	166	57	366	1.90	2.20
Tav	4	4	4	49	27	322	**6.75**	6.57
Tbk	6	3	32	1569	60	2942	1.88	1.88
Tma	7	6	20	44	38	72	1.90	1.64
Train11	2	1	11	25	13	31	1.18	1.24
Train4	2	1	4	14	5	18	1.25	1.29
Mid							**1.96**	**2.05**

влечет повышение значения параметра R, который влияет на число элементов памяти и обратных связей конечного автомата. В табл. 5 приведены значения параметра R для исходного конечного автомата (r_0) и автомата типа Мура (r_1), а также отношение (r_1/r_0) и разность ($r_1 - r_0$) этих параметров.

Анализ значений r_0 и r_1 показывает, что переход от автомата типа Мили к автомату типа Мура не приводит к увеличению параметра R в 14 примерах из 48 или приблизительно в 29% случаев. В остальных примерах значение параметра R изменяется от 1 до 3. Наибольшее увеличение параметра R ($r_1 - r_0 = 3$) наблюдается для примеров Dk15 и Tav. Среднее же увеличение параметра R (r_1/r_0) при переходе от автомата Мили к автомату типа Мура составляет 1.27 раза или в абсолютном значении ($r_1 - r_0$) 0.96. Другими словами, при переходе от автомата Мили к автомату Мура параметр R в среднем возрастает на единицу. С другой стороны, поскольку выходные функции автомата типа Мура не зависят от входных переменных, то при реализации конечного автомата значительно упрощаются логические выражения выходных функций автомата типа Мура по сравнению с автоматом типа Мили. Поэтому можно предположить, что в отдельных случаях для конечных автоматов с большим числом выходных функций реализация автомата типа Мура будет менее сложной, чем автомата типа Мили.

Для сравнения сложности реализации конечных автоматов типа Мили и Мура был выполнен синтез рассматриваемых примеров с помощью пакета MAX + PLUSII на ПЛИС EPM9320ARC208-10 семейства MAX9000. Результаты экспериментальных исследований приведены также в табл. 5. Здесь c_0 и c_1 – стоимости реализации конечного автомата (измеряемые числом используемых макроячеек ПЛИС) для исходного конечного автомата и автомата Мура; d_0 и d_1 – максимальные задержки сигналов в наносекундах на комбинационной части исходного конечного автомата и автомата Мура; c_1/c_0 и d_1/d_0 – отношения этих параметров.

Анализ табл. 5 показывает, что стоимость реализации автомата Мура по сравнению с автоматом Мили в среднем повышается в 2.02 раза, а величина задержки возрастает в 1.72 раза. Причем наибольшее увеличение числа состояний в 6.75 раза для примера Tav также повлек и максимальный рост стоимости реализации (в 11 раз) для автомата типа Мура по сравнению с автоматом типа Мили. В то же время наибольшее увеличение числа переходов в 7.04 раза для примера Ex1 сопровождалось ростом стоимости реализации только в 3.74 раза. Следовательно, на увеличение стоимости реализации автомата типа Мура по сравнению с автоматом типа Мили большее влияние оказывает расширение числа состояний, чем повышение числа переходов конечного автомата. Можно также заметить, что при увеличении параметра R на 2 и более наблюдается пропорциональное изменение стоимости реализации в среднем в 3 раза и более (примеры Dk14, Dk15, Dk16, Ex1, S1488, S1494, S820, S832, Sand и Tav). Исключение составляют примеры S208 и S420, для которых при возрастании параметра R на единицу стоимость реализации выросла более чем в 5 раз.

Очевидно, что для исходных автоматов типа Мура (примеры Donfile, Modulo12, S1, S1a, S27 и S8) стоимость реализации не меняется. Более того, в четырех случаях (примеры Ex2, Ex6, Keyb и Pma) стоимость реализации автомата типа Мура меньше стоимости реализации аналогичного автомата типа Мили. Отметим также, что предположение об уменьшении стоимости реализации конечных автоматов типа Мура с большим числом выходных функций не подтвердилось (примеры Ex1, Planet, S1488, S1494, S820, S832, Sand и Styr). Последнее обстоятельство можно объяснить особенностями метода синтеза, реализованного в пакете MAX + PLUSII. По-видимому, данный метод синтеза не ориентирован на реализацию автоматов типа Мура, а снижение стоимости реализации для отдельных автоматов Мура объясняется чистой случайностью. В общем случае повышение стоимости реализации автоматов типа Мура, по сравнению с автоматами типа Мили, сопровождается также ростом величины задержки сигналов, однако в несколько меньшем абсолютном значении. Так в среднем величина задержки выросла в 1.72 раза, а максимально – в 6.16 раза для примера S1488.

Следует отметить, что иногда при значительном увеличении стоимости реализации наблюдается умеренное возрастание задержки (примеры Dk14, Dk15, S208, S420 и Tav). В отдельных случаях существенный рост стоимости реализации приводит к еще большому росту задержки (примеры Ex1, S1488 и S1494). (Напомним, что в примере Ex1 имело место максимальное увеличение числа переходов в 7.04 раза.) В примере Planet проявилось значительное (в 3.20 раза) возрастание задержки при умеренном повышении стоимости реализации (в 1.36 раза). В то же время в двух случаях (примеры Dk27 и Train11) наблюдается уменьшение времени задержки. Таким образом, при переходе от автомата типа Мили к автомату типа Мура увеличение величины задержки менее предсказуемо, чем рост стоимости реализации. Это можно объяснить тем, что на время задержки сигналов конечного автомата при его реализации на ПЛИС влияет больше факторов: кроме числа внутренних состояний, числа переходов и метода синтеза также оказывает влияние метод трассировки внутренних соединений ПЛИС.

5. Переход от автоматов типа Мили к автоматам типа Мура при минимизации конечных автоматов.

Таблица 5. Результаты экспериментальных исследований сложности реализации автоматов типа Мили и типа Мура

FSM	r_0	r_1	r_1/r_0	$r_1 - r_0$	Mealy		Moore		c_1/c_0	d_1/d_0
					c_0	d_0	c_1	d_1		
Bbara	4	4	1.00	0	8	17.6	10	18.2	1.25	1.03
Bbsse	4	5	1.25	1	13	19.2	24	25.7	1.85	1.34
Bbtas	3	4	1.33	1	5	13.3	9	17.9	1.80	1.35
Beecount	3	4	1.33	1	7	17.3	10	18.1	1.43	1.05
Cse	4	5	1.25	1	25	31.1	41	34.5	1.64	1.11
Dk14	3	5	1.67	2	12	18.2	42	27.2	3.50	1.49
Dk15	2	5	**2.50**	**3**	7	16.5	20	26.7	2.86	1.62
Dk16	5	7	1.40	2	33	22.1	102	65.8	3.09	2.98
Dk17	3	4	1.33	1	7	17.7	11	22.2	1.57	1.25
Dk27	3	4	1.33	1	5	16.9	6	13.3	1.20	***0.79***
Dk512	4	5	1.25	1	7	17.3	13	18.2	1.86	1.05
Donfile	*5*	*5*	*1.00*	*0*	*27*	*25.8*	*27*	*25.8*	*1.00*	*1.00*
Ex1	5	7	1.40	2	53	25.7	198	155.3	3.74	6.04
Ex2	5	5	1.00	0	16	21.7	15	22.1	***0.94***	1.02
Ex3	4	4	1.00	0	7	16.9	8	17.8	1.14	1.05
Ex4	4	5	1.25	1	13	17.6	15	18.0	1.15	1.02
Ex5	4	4	1.00	0	7	16.9	11	17.9	1.57	1.06
Ex6	3	4	1.33	1	21	17.9	19	21.5	***0.90***	1.20
Ex7	4	4	1.00	0	7	16.9	9	17.9	1.29	1.06
Keyb	5	5	1.00	0	24	30.1	23	30.7	***0.96***	1.02
Lion	2	3	1.50	1	3	12.5	4	13.3	1.33	1.06
Lion9	4	4	1.00	0	7	17.7	7	17.7	1.00	1.00
Mc	2	3	1.50	1	7	13.3	8	13.4	1.14	1.01
Modulo12	*4*	*4*	*1.00*	*0*	*4*	*16.5*	*4*	*16.5*	*1.00*	*1.00*
Planet	6	7	1.17	1	84	22.8	114	72.9	1.36	3.20
Pma	5	6	1.20	1	70	35.4	62	52.3	***0.89***	1.48
S1	*5*	*5*	*1.00*	*0*	*59*	*39.3*	*59*	*39.3*	*1.00*	*1.00*
S1488	6	8	1.33	2	93	35.4	277	218.0	2.98	***6.16***
S1494	6	8	1.33	2	90	35.4	287	172.2	3.19	4.86
S1a	*5*	*5*	*1.00*	*0*	*40*	*42.1*	*40*	*42.1*	*1.00*	*1.00*
S208	5	6	1.20	1	10	21.9	52	45.2	5.20	2.06
S27	*3*	*3*	*1.00*	*0*	*4*	*13.3*	*4*	*13.3*	*1.00*	*1.00*
S298	8	9	1.13	1	375	174.9	581	266.2	1.55	1.52
S386	4	5	1.25	1	16	18.1	27	31.0	1.69	1.71
S420	5	6	1.20	1	10	21.9	51	34.5	5.10	1.58
S510	6	7	1.17	1	45	23.6	68	54.9	1.51	2.33
S8	*3*	*3*	*1.00*	*0*	*4*	*16.5*	*4*	*16.5*	*1.00*	*1.00*
S820	5	7	1.40	2	47	36.6	154	94.7	3.28	2.59
S832	5	7	1.40	2	52	41.3	164	101.1	3.15	2.45
Sand	5	7	1.40	2	74	43.3	167	99.2	2.26	2.29
Shiftreg	3	4	1.33	1	4	12.5	6	17.7	1.50	1.42
Sse	4	5	1.25	1	13	19.2	24	25.7	1.85	1.34
Styr	5	6	1.20	1	73	43.9	110	80.0	1.51	1.82
Tav	2	5	**2.50**	3	6	13.3	66	43.4	**11.00**	3.26
Tbk	5	6	1.20	1	87	85.1	186	158.4	2.14	1.86
Tma	5	6	1.20	1	26	26.7	31	30.3	1.19	1.13
Train11	4	4	1.00	0	8	21.0	8	17.8	*1.00*	***0.85***
Train4	2	3	1.50	1	3	12.5	4	13.3	1.33	1.06
Mid			**1,27**	**0.96**					**2.02**	**1.72**

Таблица 6. Результаты экспериментальных исследований алгоритма преобразования автомата типа Мили в автомат типа Мура в случае минимизации

FSM	Mealy (min)		Moore (min)		s_3/s_2	p_3/p_2	s_3/s_0	p_3/p_0	s_1/s_3	p_1/p_3
	s_2	p_2	s_3	p_3						
Bbara	7	42	9	54	1.29	1.29	***0.90***	***0.90***	1.33	1.33
Bbsse	13	*208*	21	*300*	1.62	1.44	1.31	**5.36**	1.14	0.31
Beecount	4	20	7	35	1.75	1.75	1.00	1.25	1.43	1.14
Ex2	14	56	19	76	1.36	1.36	1.00	1.06	1.21	1.11
Ex3	5	20	10	40	2.00	2.00	1.00	1.11	1.30	1.00
Ex5	4	16	8	32	2.00	2.00	***0.89***	1.00	1.63	1.38
Ex7	4	16	8	32	2.00	2.00	***0.80***	***0.89***	1.75	1.50
Lion9	4	16	6	24	1.50	1.50	***0.67***	***0.96***	1.83	1.29
S298	135	681	224	1129	1.66	1.66	1.03	1.03	1.48	1.48
S820	24	230	69	*614*	**2.88**	2.67	2.76	2.65	1.01	1.00
S832	24	243	69	*708*	**2.88**	**2.91**	2.76	2.89	1.01	1.00
Sse	13	*208*	21	*300*	1.62	1.44	1.31	**5.36**	1.14	0.31
Tbk	16	1024	30	1920	1.88	1.88	***0.94***	1.22	2.00	***1.53***
Tma	18	*89*	36	*166*	2.00	1.87	1.80	3.77	1.06	0.43
Train11	4	15	6	23	1.50	1.53	***0.55***	***0.92***	**2.16**	1.35
Mid					**1.86**	**1.82**	**1.25**	**2.02**	**1.43**	**1.08**

Эталонные примеры конечных автоматов, разработанные в центре MCNC [24], в основном предназначены для проверки эффективности методов минимизации и синтеза конечных автоматов. По этой причине рассматриваемые эталонные примеры изначально не являются минимальной формой описания конечных автоматов. В практике инженерного проектирования каждый разработчик стремиться выразить конечный автомат в форме, которая максимально приближается к его минимальному представлению. Поэтому полезно проверить эффективность равзиваемого метода перехода от автоматов типа Мили к автоматам типа Мура для случая, когда к исходным эталонным примерам применен один из известных методов минимизации.

Для минимизации исходных эталонных примеров была выбрана широко используемая программа минимизации числа внутренних состояний конечных автоматов STAMINA системы SIS [25]. После применения программы STAMINA к исходным эталонным примерам число внутренних состояний уменьшилось для 20 примеров из 48, что составляет приблизительно 42%. В результате минимизации числа внутренних состояний с помощью программы STAMINA примеры Donfile, Modulo12, S1a и S8 стали иметь только одно внутреннее состояние, т.е. перешли из класса последовательностных в класс комбинационных схем, поэтому данные примеры исключаются из дальнейшего рассмотрения. Кроме того, анализу не подвергались "чистые" конечные автоматы типа Мура: примеры S1 и S27.

В табл. 6 приведены параметры эталонных примеров, для которых с помощью программы STAMINA удалось сократить число внутренних состояний, а также параметры эквивалентных автоматов типа Мура. В табл. 6 приняты следующие обозначения: s_2 и p_2 – число внутренних состояний и переходов соответственно автомата типа Мили после минимизации; s_3 и p_3 – число внутренних состояний и переходов эквивалентного автомата типа Мура; s_3/s_2, p_3/p_2, s_3/s_0, p_3/p_0, s_1/s_3 и p_1/p_3 – отношения указанных параметров.

Отметим, что для отдельных исходных примеров в результате применения программы STAMINA, несмотря на уменьшение числа внутренних состояний, увеличилось число переходов конечного автомата, т.е. $p_2 > p_0$ (примеры Bbsse, Sse и Tma). Далее, в результате перехода от минимизированной формы автомата типа Мили к автомату типа Мура число переходов стало больше, чем число переходов автомата Мура без минимизации, т.е. $p_3 > p_1$ (примеры Bbsse, S820, S832, Sse и Tma).

Отношение s_3/s_2 показывает изменение числа состояний при переходе от минимизированной формы автомата типа Мили к автомату типа Мура. В среднем, число состояний возросло в 1.86 раза, при этом наблюдалось минимальное увеличение в 1.29 раза для примера Bbara, а максимальное – 2.88 раза для примеров S820 и S832. Отношение

p_3/p_2 отражает рост числа переходов конечного автомата при замене минимизированной формы автоматом типа Мили автоматом типа Мура. В среднем, число переходов увеличилось в 1.82 раза, при этом имело место минимальное увеличение в 1.29 раза для примера Bbara, а максимальное – в 2.91 раза для примера S832.

Отношение s_3/s_0 характеризует расширение числа состояний при переходе от исходного автомата типа Мили к автомату типа Мура с использованием программы STAMINA. В среднем, число состояний выросло в 1.25 раза, при этом наблюдалось минимальное увеличение в 0.55 раза для примера Train11, а максимальное – в 2.76 раза для примеров S820 и S832. Отметим, что благодаря применению программы STAMINA для ряда примеров (Bbara, Ex5, Ex7, Lion9, Tbk и Train11) число состояний автомата Мура меньше, чем у исходного конечного автомата.

Отношение p_3/p_0 показывает увеличение числа переходов исходного автомата типа Мили по сравнению с автоматом типа Мура при использовании программы STAMINA. В среднем, число переходов увеличилось в 2.02 раза, при этом минимальное значение этого показателя составило 0.89 раза для примера Ex7, а максимальное – 5.36 раза для примеров Bbsse и Sse. Отметим, что в результате применения программы STAMINA для ряда примеров (Bbara, Ex7, Lion9 и Train11) число переходов автомата Мура меньше, чем у исходного конечного автомата.

Естественно предположить, что программа STAMINA позволяет получить автомат типа Мура с меньшим числом состояний по сравнению с конечным автоматом Мура, построенным без ее использования. Ответ на данный вопрос дает отношение s_1/s_3, которое демонстрирует, что число состояний в среднем уменьшается в 1.43 раза, при этом минимальное значение этого отношения составляет 1.01 раза для примеров S820 и S832, а максимальное – 2.16 раза для примера Train11. Аналогичное уменьшение числа переходов конечного автомата показывает отношение p_1/p_3, при этом среднее сокращение числа переходов конечных автоматов составляет только 1.08 раза, а максимальное снижение в 1.53 раза наблюдается для примера Tbk. В то же время для трех примеров Bbsse, Sse и Tma имеет место значительное увеличение числа переходов в 3.19, 3.19 и 2.31 раза соответственно.

Заключение. Переход от автомата типа Мили к автомату типа Мура, как правило, сопровождается ростом числа внутренних состояний (в среднем в 1.96 раза) и числа переходов (в среднем в 2.05 раза) конечного автомата. При этом во многих случаях наблюдается пропорциональное увеличение числа внутренних состояний и числа переходов конечного автомата. Возрастание числа состояний при переходе от автомата типа Мили к автомату типа Мура влечет за собой также увеличение параметра R – минимального числа разрядов кодов внутренних состояний конечного автомата. При этом параметр R увеличивается в среднем на единицу. Приблизительно в 29% случаев параметр R вообще не изменяется.

При реализации конечных автоматов на ПЛИС с помощью пакета MAX+PLUSII стоимость реализации автомата Мура приблизительно в 2 раза больше, чем автомата Мили. При этом задержка сигналов на комбинационной части конечного автомата Мура по сравнению с автоматом Мили возрастает в среднем в 1.72 раза. Однако для отдельных примеров стоимость реализации автомата Мура может быть даже меньше, чем для автомата Мили. На повышение стоимости реализации автомата Мура по сравнению с автоматом Мили наибольшее влияние оказывает увеличение числа внутренних состояний и, особенно, увеличение параметра R на два разряда и более.

При переходе от минимизированной с помощью программы STAMINA формы автомата Мили к автомату типа Мура число внутренних состояний растет в среднем в 1.86 раза, а число переходов – в 1.82 раза. Поэтому перед использованием алгоритма перехода от автомата Мили к автомату Мура следует применять программу STAMINA для минимизации числа состояний. В некоторых случаях это позволяет получить автомат Мура с числом состояний даже меньшим, чем у исходного конечного автомата. Однако применение программы STAMINA в отдельных случаях приводит к значительному возрастанию числа переходов конечного автомата.

Проведенные экспериментальные исследования показали, что реализованные в индустриальных пакетах методы синтеза, как правило, не учитывают тип конечного автомата (Мили или Мура). Поэтому актуальной является задача разработки специальных методов синтеза, которые позволяют учитывать особенности реализации автоматов типа Мили и типа Мура. В некоторых случаях при преобразовании автомата типа Мили к автомату типа Мура с помощью рассмотренного подхода может наблюдаться значительное увеличение числа внутренних состояний и числа переходов конечного автомата. Поэтому целью дальнейших исследований может быть разработка таких методов минимизации внутренних состояний и числа переходов автомата типа Мура, которые не изменяют тип конечного автомата, т.е. автомат Мура после минимизации должен оставаться автоматом типа Мура.

СПИСОК ЛИТЕРАТУРЫ

1. *Mealy G.H.* A method for synthesizing sequential circuits // Bell System Technical J. 1955. V. 34. P. 1045–1079.

2. *Moore E.F.* Gedanhen-experiments on sequential machines / Eds C. Shannon and J. McCarthy. Automata Studies Princeton University Press, 1956. P. 129–153.
3. *Глушков В.М.* Синтез цифровых автоматов. М.: Физматгиз, 1962.
4. *Соловьев В.В., Климович А.* Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. М.: Горячая линия – Телеком, 2008.
5. *Programmable Logic.* Intel, 1994.
6. *Соловьев В.В.* Проектирование цифровых систем на основе программируемых логических интегральных схем. М.: Горячая линия – Телеком, 2001.
7. *McCluskey E.* Logic design principles. New Jersey. Prentice-Hall, Englewood Cliffs, 1986.
8. *Соловьев В.В.* Расщепление внутренних состояний для снижения числа аргументов функций конечных автоматов // Изв. РАН. ТиСУ. 2005. № 5. С. 113–119.
9. *Avedillo M., Quintana J., Huertas J.* State merging and splitting via state assignment: a new FSM synthesis algorithm // IEE Proc. Part E, Computers and Digital Techniques. 1994. V. 141. № 4. С. 229–237.
10. *Yuan L., Qu G., Villa T., Sangiovanni-Vincentelli A.* An FSM reengineering approach to sequential circuit synthesis by state splitting // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems. 2008. V. 27. № 6. P. 1159–1164.
11. *Aiman M., Sadiq S.M., Nawaz K.F.* Finite state machine state assignment for area and power minimization // Proc. of IEEE Internat. Symp. Circuits and Systems (ISCAS). Institute of Electrical and Electronics Engineers Inc., 2006. P. 5303–5306.
12. *Wen-Tsong S.* Novel state minimization and state assignment in finite state machine design for low-power portable devices // Integration, the VLSI J. 2005. V. 38. № 4. P. 549–570.
13. *Nader R., Brett D.* A study of state machine coding styles for implementation in FPGAs // Proc. 49th Midwest Sympos. on Circuits and Systems (MWSCAS'06), Institute of Electrical and Electronics Engineers Inc., 2006. V. 1. P. 337–341.
14. *Anurag T., Karen T.* Saving power by mapping finite state machines into embedded memory blocks in FPGAs // Proc. Design, Automation and Test in Europe Conf. and Exhibition. Institute of Electrical and Electronics Engineers Computer Society, 2004. V. 2. P. 916–921.
15. *Imtiaz A., Shobà D.* Heuristic algorithm for the minimization of incompletely specified finite state machines // Computers and Electrical Engineering. 2001. V. 27. № 2. P. 159–172.
16. *Perkowski M., Jozwiak L., Zhao W.* Symbolic two-dimensional minimization of strongly unspecified finite state machines // J. Systems Architecture. 2001. V. 47. № 1. P. 15–28.
17. *Rupesh S., Madhav D., Narayanan H.* Decomposition of finite state machines for area, delay minimization // Proc. IEEE Internat. Conf. on Computer Design (ICCD'99): VLSI in Computers and Processors. Institute of Electrical and Electronics Engineers Inc., 1999. P. 620–625.
18. *Rawski M., Selvaraj H., Luba T.* An application of functional decomposition in ROM-based FSM implementation in FPGA devices // J. Systems Architecture. 2005. V. 51. № 6–7. P. 424–434.
19. *Irith P., Sudhakar R.* On finding a minimal functional description of a finite-state machine for test generation for adjacent machines // IEEE Trans. Computers. 2000. V. 49. № 1. P. 88–94.
20. *Christoph M., Thorsten T.* Local encoding transformations for optimizing OBDD-representations of finite state machines // Formal Methods in System Design. 2001. V. 18. № 3. P. 285–301.
21. *Aritz S.* Analysis of the FSMs implementation with mini-microprocessors in FPGAs // Proc. IEEE Internat. Symp. on Industrial Electronics (ISIE). Institute of Electrical and Electronics Engineers Inc., 2007. P. 2290–2294.
22. *Seinstra F., Koelma D., Bagdanov A.* Finite state machine-based optimization of data parallel regular domain problems applied in low-level image processing // IEEE Trans. Parallel and Distributed Systems. 2004. V. 15. № 10. P. 865–877.
23. *Vesa L., Kimmo K., Timo H.* Optimizing finite state machines for system-on-chip communication // Proc. of IEEE Internat. Symp. on Circuits and Systems. Institute of Electrical and Electronics Engineers Inc., 2002. V. 1. P. I/485–I/488.
24. *Yang S.* Logic synthesis and optimization benchmarks user guide. Version 3.0. Technical Report. North Carolina: Microelectronics Center of North Carolina, 1991.
25. *Sentovich E.M., Singh K.J., Lavagno L. et al.* SIS: A system for sequentional circuit synthesis // Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory. Berkley. Department of Electrical Engineering and Computer Science. University of California, 1992.