1973 год ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ Выпуск 54

УДК 681.326

МЕТОД СИНТЕЗА ПОЛНОГО ПРОВЕРЯЮЩЕГО ТЕСТА ДЛЯ ЛОГИЧЕСКИХ СХЕМ С ПАМЯТЬЮ

Ю.П. Сахаров

В настоящей работе рассматривается контроль больших интегральных схем (БИС) произвольной структуры на одиночные ошибки типа логических констант (*const* 0 и *const* 1).

Описывая метод автоматической генерации полной тестовой последовательности, мы остановились лишь на основных его принципах, опустив всё, что относится к программной реализации и, следовательно, в большой степени зависит от имеющейся ЭВМ, её математического обеспечения, искусства программиста и т.д.

При реализации алгоритма предусматриваются обращения к двум уже действующим программам: логического моделирования схем и контроля полноты теста. Эти же программы существенно использовались в работе над самим алгоритмом.

Авторам этих программ выражаю искреннюю признательность.

§ 1. Общая постановка задачи

Проверяемая логическая схема предварительно декомпозируется на функционально законченные фрагменты. Эта неформальная операция выполняется разработчиком схемы, однако устройства памяти выделяются непременно. При декомпозиции каждому выделенному фрагменту приписывается выход схемы, через который ведется его контроль.

Сначала каждый фрагмент рассматривается отдельно; построение тестовой последовательности для комбинационных схем и

схем памяти излагается соответственно в § 2 и 3. Затем происходит формирование полученных последовательностей в единый тест для всей схемы. Некоторые вопросы, возникающие при этом, рассмотрены в § 4. Наконец, происходит обращение к программе контроля теста на полноту; в случае обнаружения неполноты синтезированного теста (появление ее обусловлено неформальностью декомпозиции) предусмотрено вмешательство инженера-разработчика: он может либо вручную дополнить тест, либо произвести повторную декомпозицию, переназначив фрагменты и выходы.

Эта работа инженера облегчается тем, что программа, контролирующая полноту, указывает непроверенные тестом контакты.

§ 2. Булеворазностный алгоритм синтеза контролирующего теста для комбинационных фрагментов

Рассмотрим логическую функцию $F(x_1, x_2, \dots, x_n)$. Пусть $F = F(f_1, f_2, \dots, f_m)$ – одна из возможных декомпозиций F. В общем случае при этом $f_k = f_k(x_{k1}, \dots, x_{ki}, \dots)$.

Будем считать, что

$$\bigcup_{k=1}^{m} \{arg f_k\} = \{arg F\} \text{ и } \forall_{i,j} \{arg f_i\} \cap \{arg f_j\} = \phi, \quad (1)$$

где $\{arg \psi\}$ – множество аргументов булевой функции ψ.

ТЕОРЕМА 1. Для независимости функции F от некоторой входной переменной $x_{ki} \in \{arg f_k\}$ необходимо и достаточно, чтобы

$$\frac{\partial F}{\partial f_k} \cdot \frac{\partial f_k}{\partial x_{ki}} = 0, \quad (2)$$

где $\frac{\partial \psi(y_1, \dots, y_i, \dots, y_n)}{\partial y_i} = \psi(y_1, \dots, y_i, \dots, y_n) \oplus \psi(y_1, \dots, \bar{y}_i, \dots, y_n)$ – булева разность*) логической функции ψ.

ДОКАЗАТЕЛЬСТВО. Пусть F не зависит от x_{ki}. Возможны два случая:

1. $f_k(x_{k1}, \dots, x_{ki}, \dots, x_{kr}) = f_k(x_{k1}, \dots, \bar{x}_{ki}, \dots, x_{kr})$.

*) Понятие булевой разности введено и впервые применено в [1].

2. $F(f_1,\dots,f_k,\dots,f_m)=F(f_1,\dots,\bar{f}_k,\dots,f_m)$.

Оба случая, по определению булевой разности, с очевидностью приводят к (2). Положим теперь, что (2) верно. Возникающие ситуации сводятся к двум:

I. $\frac{\partial F}{\partial f_k}=0 \Longrightarrow F$ не зависит от f_k, а в силу условия (I) и от $\{arg f_k\}$ и, в частности, от x_{ki};

2. $\frac{\partial f_k}{\partial x_{ki}}=0 \Longrightarrow f$ не зависит от $x_{ki} \xrightarrow{(1)} F$, не зависит от x_{ki}.

Пусть теперь G -логическая схема, реализующая булеву функцию F и фиксирующая её функциональную декомпозицию в базисе $\{\text{И}, \overline{\text{ИЛИ}}, \text{НЕ}\}$. Упорядочим схему, разбив ее на N уровней: 0-й уровень составляют входные переменные X, I-й - функции $f_{j_1}^{(1)}(X)$, 2-й - $f_{j_2}^{(2)}(f_1^{(1)},\dots,f_{n_1}^{(1)})$, N-й - $f_{j_N}^{(N)}(f_1^{(N-1)},\dots,f_{n_{N-1}}^{(N-1)})$, где $j_1=1,2,\dots,n_1$; $j_2=1,2,\dots,n_2;\dots$; $j_N=1,2,\dots,n_N$ а, $\{f_{j_i}^{(i)}\}$ - функции, принадлежащие принятому базису. Без серьёзного уменьшения общности положим далее $n_N=1$.

ТЕОРЕМА 2. Необходимое и достаточное условие независимой булевой функции от некоторой входной переменной $x_i \in X$ состоит в выполнении равенства:

$$\frac{\Delta F}{x_i}=\frac{\partial F}{\partial f_{j_{N-1}}^{(N-1)}}\cdot\frac{\partial f_{j_{(N-1)}}^{(N-1)}}{\partial f_{j_{N-2}}^{(N-2)}}\cdot\dots\cdot\frac{\partial f_{j_2}^{(2)}}{\partial f_{j_1}^{(1)}}\cdot\frac{\partial f_{j_1}^{(1)}}{\partial x_i}=0,$$

$$f_{j_{N-1}}^{(N-1)}\in\{arg F\}, f_{j_{N-2}}^{(N-2)}\in\{arg f_{j_{N-1}}^{(N-1)}\},\dots,f_{j_1}^{(1)}\in\{arg f_{j_2}^{(2)}\}, x_i\in\{arg f_{j_1}^{(1)}\}. \quad (3)$$

Для доказательства достаточно применить результаты теоремы I N раз к двум соседним уровням, увеличивая каждый раз их номер на I.

СЛЕДСТВИЕ I. Для того, чтобы логическая функция F, реализуемая схемой G, зависела только от входной переменной x_i, необходимо и достаточно соблюдения условия:

$$\frac{\Delta F}{x_i}=1. \quad (4)$$

ДОКАЗАТЕЛЬСТВО получается применением равенства $\frac{\partial F}{\partial f_k}\frac{\partial f_k}{\partial x_i}=1$, выражающего по теореме I условия зависимости F от x_i, способом, описанным при доказательстве теоремы 2.

СЛЕДСТВИЕ 2. Если выполнено условие (4) и имеют место соотношения (3), то ошибка (инверсия правильного значения) в любом одном из аргументов $x_i, f_{j_1}^{(1)}, f_{j_2}^{(2)},\dots,f_{j_{N-1}}^{(N-1)}$ будет вызывать ошибку в F.

ДОКАЗАТЕЛЬСТВО очевидно.

По аналогии с [2] введем понятие "функциональный путь", понимая под ним множество соотношений вида (3), упорядоченное по возрастанию номеров уровней логической схемы. Будем обозначать его P (...), а в скобках записывать все левые части соотношений (3), следуя фиксированному порядку движения по уровням.

Шаг I. По заданной схеме составляется список всех возможных функциональных путей отличающихся друг от друга хотя бы одним элементом.

Шаг 2. Каждый из функциональных путей P (...) перерабатывается в уравнение вида $\frac{\Delta F}{x_i}=1$, где x_i - входная переменная для данного пути.

Шаг 3. Из каждого решения этого уравнения формируются два соседних по переменной x_i набора. В случае неизбыточности исходной схемы G при этом не возникает логического противоречия; получается именно пара конъюнкций, которая и составляет проверочный тест для выделенного функционального пути. Последнее утверждение вытекает из следствия 2 и из того, что аргумент x_i, а вместе с ним и остальные аргументы $f_{j_1}^{(1)},\dots,f_{j_{N-1}}^{(N-1)}$ принимают все возможные значения.

Шаг 4. В полном тестовом множестве, полученном на шаге 3, выполняются все возможные совмещения (поглощения).

Введем теперь язык (форму) описания логической схемы. Будем считать, что она по-прежнему реализуется в базисе $\{\&, \vee, \neg\}$ и все функциональные элементы схемы пронумерованы. Поскольку нашей основной целью является нахождение тестов по структурной (логической) схеме, её описание должно сохранять топологию. Этому требованию удовлетворяет совершенный вложенный формат ([3]), которым мы и воспользуемся.

Основные понятия языка

$[_k$ – помеченная открывающая скобка, k – номер описываемого функционального элемента;

$]_k$ – помеченная закрывающая скобка;

$\&, \vee, \overline{\&}, \overline{\vee}, \neg$ – знаки логических операций "и", "или","не-и" "не-или", "не";

x_i – аргумент 0-го яруса (входная переменная). Аргументы I-го,..., $(N-1)$-го ярусов, определенные при формулировке теоремы 2): $f_{j_1}^{(1)}, f_{j_2}^{(2)}, \ldots, f_{j_{(N-1)}}^{(N-1)}$.

. – знак отделяющий аргументы данного функционального элемента.

Синтаксические правила описания

I) Описание начинается с выходного элемента схемы и проходит методом "спуска" (до 0-го яруса).

2) В описание каждого элемента входят следующие четыре объекта:

- открывающая скобка, помеченная номером;
- знак выполняемой элементом логической операции;
- список аргументов, отделенных знаком ",";
- закрывающая скобка, помеченная тем же номером,что и открывающая.

При этом если какой-либо аргумент не лежит на 0-м ярусе, то сформулированное выше правило применяется рекуррентно.

3) Порядок рассмотрения аргументов в рекурсии п.2 безразличен, но в каждом случае она продолжается до 0-го яруса так, что в результирующем описании ("вложенном формате") фигурируют только входные переменные.

4) Каждый функциональный путь $x_i - f_{j_1}^{(1)} - \ldots - f_{j_{N-1}}^{(N-1)} - F$ записывается в виде упорядоченной по ярусам цепочки десятичных чисел, являющихся номерами функциональных элементов схемы. Для определенности условимся в первой справа позиции записывать номер выходного элемента.

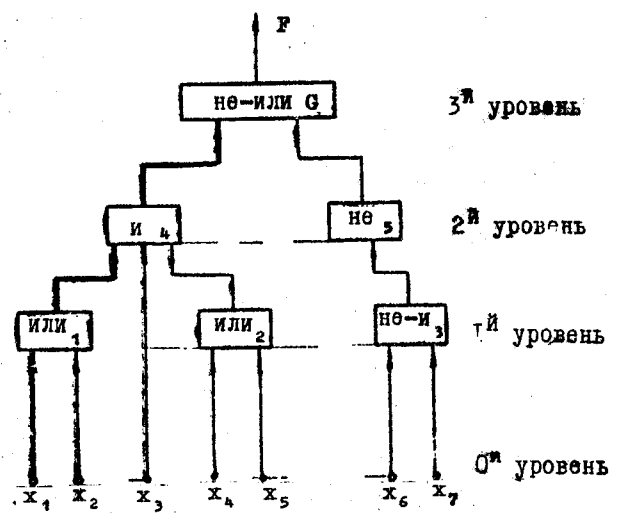


Рис. I

Описание схемы на рис.I по вышеприведенным правилам выглядит так:

$$[_6 \bar{\vee}[_4 \&[_1 \vee x_1, x_2]_1, x_3, [_2 \vee x_4, x_5]_2]_4, [_5 \neg [_3 \& x_6, x_7]_3]_5]_6. \quad (5)$$

Описание одного из путей: $P\ (x_1, 1,4,6)$.

5) Перечень аргументов, заключенный в квадратные скобки, называется в дальнейшем списком аргументов. В (5) $[_4 \ldots]_4$, $[_5 \ldots]_5$ есть список, относящийся к внешней скобке 6; $[_1 \ldots]_1, x_3 [_2 \ldots]_2$ – список, относящийся к внутренней скобке 4,и т.д.

Приведем теперь в виде двух теорем с е м а н т и ч е с к и е п р а в и л а нахождения $\Delta F / \Delta x_i$ на шаге 2.

ТЕОРЕМА 3.

$$\frac{\alpha\,[\,\tilde{\vee}\ (\text{список})\,]}{\alpha\ (\text{аргумент из списка})} = \bar{\vee}\ (\text{список без указанного аргумента}).$$

ДОКАЗАТЕЛЬСТВО. Так как, согласно [I], $\frac{\alpha F}{\alpha(arg)} = \frac{\alpha \bar{F}}{\alpha(arg)}$, то все последующие рассуждения относятся в равной степени как к выражению $[\,\vee\ (\text{список})\,]$,так и к выражению $[\,\bar{\vee}\ (\text{список})\,]$. Рассмотрим для определенности первый случай.Обозначая аргументы списка $arg\,1, arg\,2, \ldots, arg\,i, \ldots, arg\,k$ и выполняя операцию $\vee$, получаем логическое выражение: $\bigvee_{j=1}^{k} arg\,j$ и булеву разность.

$$\frac{\partial(\bigvee_{j=1}^{k} arg\,j)}{\partial(arg\,i)} = (\bigvee_{j=1,\dots,i-1,i+1,\dots,k} arg\,j \vee arg\,i) \oplus (\bigvee_{j=1,\dots,i-1,i+1,\dots,k} arg \vee \overline{arg\,i}).$$

Из этого равенства, и формулы $(x \vee y) \oplus (x \vee \bar{y}) = x$ вытекает, что

$$\frac{\partial(\bigvee_{j=1}^{k} arg\,j)}{\partial(arg\,i)} = \bigvee_{j=1,\dots,i-1,i+1,\dots,k} arg\,j \quad \text{и т.д.}$$

ТЕОРЕМА 4.

$$\frac{\partial[\bar{\&}\,(\text{список})]}{\partial(\text{аргумента из списка})} = \&\ (\text{список без указанного аргумента})$$

ДОКАЗАТЕЛЬСТВО. Повторив все рассуждения, изложенные при доказательстве теоремы 3, получаем булеву разность

$$\frac{\partial(\&_{j=1}^{k} arg\,j)}{\partial(arg\,i)} = (\&_{j=1,\dots,i-1,i+1,\dots,k} arg\,j \cdot arg\,i) \oplus (\&_{j=1,\dots,i-1,i+1,\dots,k} arg\,j \cdot \overline{arg\,i}).$$

Из этого равенства и формулы $xy \oplus x\bar{y} = x$ следует:

$$\frac{\partial(\&_{j=1}^{k} arg\,j)}{\partial(arg\,i)} = \&_{j=1,\dots,i-1,i+1,\dots,k} arg\,j,$$

что и доказывает теорему.

Относительно одноместной операции $\neg$ (отрицание) сформулируем еще одно семантическое правило:

$$\frac{\partial[\neg arg\,i]}{\partial(arg\,i)} = arg\,i \oplus \overline{arg\,i} \equiv 1,$$

и, следовательно, при составлении выражения (4) такие булевы разности могут быть опущены.

З а м е ч а н и е .Правила раскрытия скобок в (4) и перехода к системе пар тестов нетрудно формулируются з языке описания. Однако, по нашему мнению, они относятся более к машинной реализации алгоритма и потому здесь не приводятся. Сказанное, по-видимому, относится и к совмещению тестов (шаг 4).

Продолжая рассмотрение примера, преобразуем запись $D(x_1,1,4,6)$ в уравнение $\Delta_{x_i} F = 1$:

$$\Delta_{x_i} F = \frac{\partial 6}{\partial 4} \cdot \frac{\partial 4}{\partial 1} \cdot \frac{\partial 1}{\partial x_1}$$

Здесь 6,4,I – номера функциональных элементов и соответственно квадратных скобок в описании схемы.

$$\frac{\partial 6}{\partial 4} = \neg[\overline{5}\,[_3\,\bar{\&}\,x_6, x_7]_3]_5 = \overline{\overline{x_6 x_7}} = \bar{x}_6 \vee \bar{x}_7;$$

$$\frac{\partial 4}{\partial 1} = x_3[_2 \vee x_4, x_5]_2 = x_3 \cdot x_4 \vee x_3 \cdot x_5;$$

$$\frac{\partial 1}{\partial x_1} = \bar{x}_2;$$

$$\Delta_{x_1} F = \bar{x}_2(x_3 x_4 \vee x_3 x_5)(\bar{x}_6 \vee \bar{x}_7) = \bar{x}_2 x_3 x_4 \bar{x}_6 \vee \bar{x}_2 x_3 x_4 \bar{x}_7 \vee \bar{x}_2 x_3 x_5 \bar{x}_6 \vee \bar{x}_2 x_3 x_5 \bar{x}_7.$$

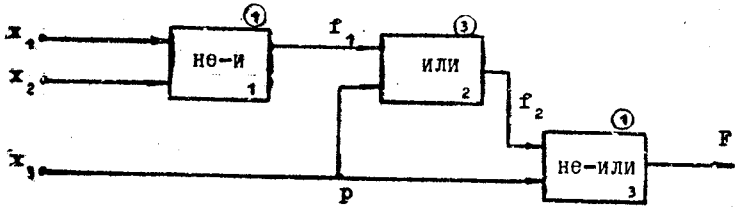
Таким образом, для контроля рассмотренного пути возможны четыре пары тестов:

$$\left.\begin{matrix} x_1\bar{x}_2x_3x_4*\bar{x}_6* \\ \bar{x}_1\bar{x}_2x_3x_4*\bar{x}_6* \end{matrix}\right\} \quad \left.\begin{matrix} x_1\bar{x}_2x_3x_4**\bar{x}_7 \\ \bar{x}_1\bar{x}_2x_3x_4**\bar{x}_7 \end{matrix}\right\} \quad \left.\begin{matrix} x_1\bar{x}_2x_3*x_5\bar{x}_6* \\ \bar{x}_1\bar{x}_2x_3*x_5\bar{x}_6* \end{matrix}\right\} \quad \left.\begin{matrix} x_1\bar{x}_2x_3*x_5*\bar{x}_7 \\ \bar{x}_1\bar{x}_2x_3*x_5*\bar{x}_7 \end{matrix}\right\}.$$

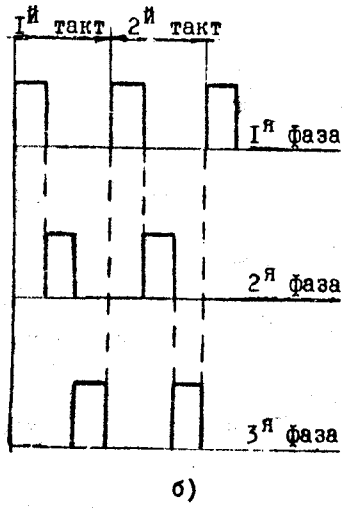
Символ $*$ на месте p -й переменной означает, что ее значение может быть доопределено произвольно.

Обсудим теперь результативность предлагаемого алгоритма и в связи с этим – специфику функционирования тактированных логических схем.

Если учесть, что в тактированных схемах несколькими последовательностями синхроимпульсов осуществлена строгая временная "привязка" логических сигналов так, чтобы распространение сигнала по ветвям сходящегося разветвления занимало разное число тактов, можно заключить, что указанная выше трудность преодолевается фиксацией порядка тестов в синтезируемой тестовой последовательности. Говоря конкретно, необходимо, чтобы в рассматриваемом случае условия проверки (решение уравнения (4)) и сама проверка данного пути занимали бы соседние такты. Поясним эту мысль примером анализа схемы на рис. 2,а. Обычное схемное решение приводит здесь к тому, что оба входа элемента 3 зависят от сигнала x_3 , разветвляющегося в точке P , а именно : $F = \overline{f_2 \vee x_3} = \overline{(f_1 \vee x_3) \vee x_3}$. Если же синхронизировать схему, напри –



а)



б)

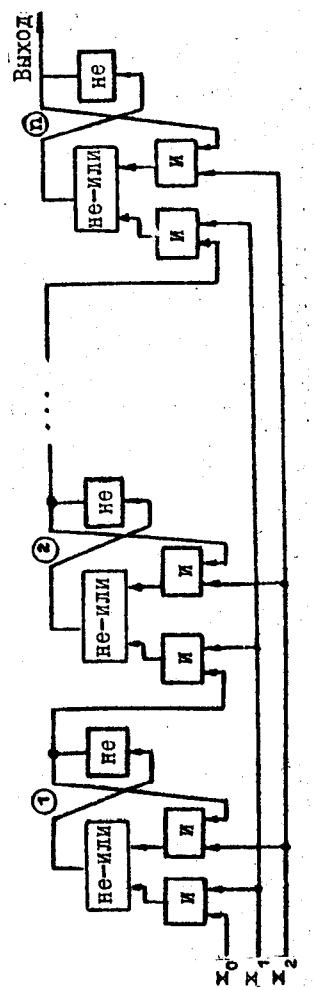
Рис.2

мер,тремя сериями импульсов, изображенными на рис. 2,б так, чтобы элементы I и 3 работали в первой фазе, а элементы 2 – в третьей, то во время работы элемента 3 один из его входов уже не зависит от x_3, т.к. изменение сигнала f_2 возможно лишь в третьей фазе следующего такта. Условие проверки пути $P(x_3,3)$: $x_1 x_2 \bar{x}_3$ противоречит прямому значению переменной x_3, и один из тестов пути $P(x_3,3)$ как бы "невозможен". Осуществить его, как легко видеть,можно подав в Т-м такте сигналы $x_1 x_2 \bar{x}_3$ (при этом $f_2 = 0$), а в $(T+1)$-м – $** x_3$. Так, порядок тестов $x_1 x_2 \bar{x}_3$ и $** x_3$ оказывается фиксированным.

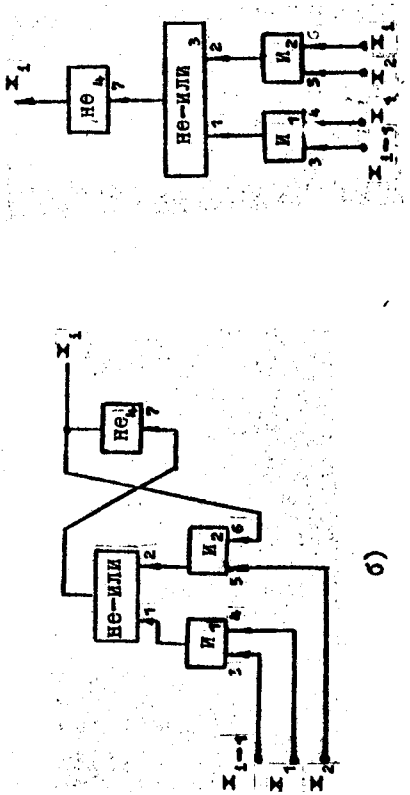
§ 3. Контроль устройств памяти

По нашему мнению, контроль элементов памяти и схем из них (регистры, счетчики и т.п.) не требует развития общей теории. Накопленный опыт показывает целесообразность содержательного подхода с применением простейших следствий из результатов § 2.

Поясним этот подход на примере анализа простейшей схемы регистра сдвига (рис. 3,а). Назначение входов: x_0 – информационный, x_1 и x_2 – управляющие. Рабочие режимы регистра: сброс ($x_1 = x_2 = 0$), хранение информации ($x_1 = 0; x_2 = 1$), сдвиг



а)



б)

в)

Рис. 3

($x_1=1;\ x_2=0$); ситуация $x_1=x_2=1$ запрещена. Регулярный характер схемы позволяет строить рассуждения применительно к какому-либо одному i -му разряду (рис.3,б). Будем считать состояние схемы, при котором $x_i=1$, единичным; противоположное состояние $x_i=0$ – нулевым.

Покажем, что контроль данной схемы на ошибки типа логических констант обеспечивается созданием нескольких режимов ее работы. Для этого представим схему в виде, изображенном на рис. 3,в.

Теперь, получив ее описание:

$$[_4 \neg [_3 \bar{\vee} [_1 \& x_{i-1}, x_i]_1, [_2 \& x_2, x_i]_2]_3]_4 ,$$

можно применить к ней аппарат из § 2. Сам анализ и его результаты удобно представить в виде таблицы, где выполнен один из возможных вариантов совмещения тестов, обеспечивающий, как нетрудно видеть, полноту проверки регистра.

З а м е ч а н и е . Покрытие тестов производилось таким образом, чтобы получающиеся при этом режимы имели естествен - ный физический смысл и легко реализовались. С этой точки зрения, совмещение, приводящее к режиму 3), с условием $x_{i-1}=1$, видимо, нецелесообразно. Оставшуюся проверку пути P (x_1, I, 3, 4) можно выполнить в режиме 5) – сброс из единичного состояния: $\bar{x}_1\,\bar{x}_2\,x_{i-1}\,x_i$.

Если же пренебречь указанным соображением, т.е. усложнить режимы проверки, можно добиться некоторого сокращения длины теста. Например, чередование тактов хранения информации и продвижения сочетания $\bar{0}1$ по регистру позволяет уменьшить длину теста с $(5n+3)$ тактов до $4n+2$.

Изложенное в данном параграфе позволяет сделать вывод о том, что с любой схемой, входящей в логическое устройство, может быть ассоциировано множество режимов работы, обеспечивающих полную проверку. Как показывает наш опыт, это множество исчерпывается уже рассмотренными режимами.

Таблица тестов для регистра сдвига

Пути P	Возможные тесты в T-м такте	Совмещение тестов. Наименование режима	x_i Такт T	x_i Такт T+1	Кол-во тактов для реализации и её проверки
$P(x_{i-1},1,3,4)$	$x_1\bar{x}_2x_{i-1}$* или x_1* $x_{i-1}\bar{x}_i$	1. "бегущая единица"	*	1	n
	$x_1\bar{x}_2\bar{x}_{i-1}$* или x_1* $\bar{x}_{i-1}\bar{x}_i$	$x_1\bar{x}_2:(x_{i-1}=1)\Rightarrow(x_i=1)$			
$P(x_1,1,3,4)$	$x_1\bar{x}_2x_{i-1}$* или x_1* $x_{i-1}\bar{x}_i$	2. "бегущий ноль"	*	0	n
	$\bar{x}_1\bar{x}_2x_{i-1}$* или $\bar{x}_1$* $x_{i-1}\bar{x}_i$	$x_1\bar{x}_2:(x_{i-1}=0)\Rightarrow(x_i=0)$			
$P(x_2,2,3,4)$	*$x_2\bar{x}_{i-1}x_i$ или $\bar{x}_1x_2$* x_i	3. "хранение нулевого состояния"	0	0	1 + n
	$\bar{x}_2\bar{x}_{i-1}x_i$ или $\bar{x}_1\bar{x}_2$ x_i	4. "хранение единичного состояния"	1	1	1 + n
$P(x_i,2,3,4)$	$\bar{x}_1x_2$* x_i или *$x_2\bar{x}_{i-1}x_i$	5. "сброс из единичного состояния"	1	0	1 + n
	$\bar{x}_1x_2$* $\bar{x}_i$ или *$x_2\bar{x}_{i-1}\bar{x}_i$				

§ 4. Объединение тестовых последовательностей фрагментов

Рассмотрим два фрагмента Φ_i и Φ_j (рис.4) с тестовыми последовательностями $T S_i$ и $T S_j$, построенными в соответствии с § 2 и 3. Обозначим через $\mathcal{I}_i$ и $\mathcal{I}_j$ множество входов, а через O_i и O_j – множество выходов этих фрагментов. Часть из них – реальные входовыходы БИС-кристалла, а часть (условные входовыходы) – возникает в связи с декомпозицией схемы. Рассмотрим основные стандартные ситуации.

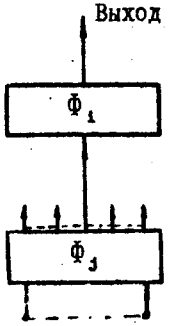


Рис.4

I. $O_j \cap \mathcal{I}_i \neq \phi$. При этом выход схемы, назначенный для проверки фрагмента Φ_j, принадлежит фрагменту Φ_i. Тогда состояние последнего должно быть таким, чтобы удовлетворялось уравнение (4), составленное относительно входа $\alpha \in \mathcal{I}_i$, отождествленного с выходом $b \in O_j$. Решение этой задачи в том случае, когда Φ_i – комбинационная схема, рассмотрено в § 2. Если же Φ_i – схема памяти, тот элемент памяти, который содержит вход $a \in \mathcal{I}_i$ ($\alpha = b \in \mathcal{I}_j$), должен находиться в состоянии, могущем измениться при единичном воздействии по входу α (в предыдущем параграфе это состояние называлось единичным). Легко поверить, что в соответствии с результатами § 2 упомянутое состояние элемента памяти есть условие проверки одного из входов.

Нужно еще учесть, что при $O_j \cap \mathcal{I}_i \neq \phi$ последовательность $T S_j$ распространяется к выходу БИС-кристалла через фрагмент Φ_i и, следовательно, может частично совместиться с его тестовой последовательностью $T S_i$. В этом случае мощность множества $T S_i$ должна быть уменьшена. Ясно, что для выполнения этой операции нужно знать реакцию каждого фрагмента на вычисленную для него тестовую последовательность. Для этого и используется упомянутая программа логического моделирования схемы.

2. $\mathcal{I}_i \cap \mathcal{I}_j \neq \phi$, т.е. элементы тестовых множеств $T S_i$ и $T S_j$ могут оказаться противоречивыми. В этой ситуации разработчику должно быть выдано указание о повторной декомпозиции (переназначении входов).

3. $\mathcal{I}_i \cap \mathcal{I}_j = \phi$, т.е. $T S_i$ и $T S_j$ независимы и могут быть совмещены во времени.

В заключение отметим, что ручной синтез полного множества тестов предложенным методом показал эффективность последнего. При этом анализировались БИС с уровнем интеграции порядка сотен дискретных компонент. В настоящее время данный метод находится в стадии машинной реализации.

Литература

1. SELLERS F.F., HSIAO M.Y., BEARNSON L.W. Analizing errors with the Boolean difference. – "IEEE Trans.on computers", 1968, vol.C-17, N 7, p.676-683.

2. MARINOS P.N. Derivation of minimal complete sets of test-input sequences using Boolean differences. – "IEEE Trans. on computers",1971,vol.C-20, N 1,p.25-32.

3. WHITNEY G.E. Algebraic-fault analisis for constrained networks. – "IEEE Trans. on computers", 1971, vol. C-20, N 2, p.141-148.

4. ARMSTRONG D.B. On finding a nearly minimal set of fault detection tests for combinational logic nets. – "IEEE Trans.on computers",1966,vol.EC-15, N 1,p.66-73.

Поступила в ред.-изд.отд.
19 июля 1972 года