Новый подход к модульно-ориентированному проектированию систем на чипах

Бозоян Ш.Е., Егиазарян В.С. (vladimir@ysu.am)

Российско-Армянский (Славянский) государственный университет

Введение

Последние годы в проектировании систем на чипах широко практикуется использование готовых компонентов (модулей). Более того, иногда чип формируется просто интеграцией заранее спроектированных модулей. Такой подход способствует появлению независимых производителей различных модулей, которые в совокупности снабжают соответствующий рынок. Эффекивность такого подхода неоспорима, так как распределение работ между различными фирмами обеспечивает высокую степень параллелизма и, тем, самым сокращает сроки проектирования чипов.

Однако модульная ориентация построения чипов имеет свои проблемы, среди которых главным является обеспечение безотказной работы модулей под “крышу” чипа. Дело в том, что модули спроектированы и проверены на работоспособность в разных, часто независимых, фирмах в условиях “свободы”, когда их входы и выходы являлись абсолютно доступными, в то время как внутри чипа их входы и выходы трудно доступны, а иногда даже практически недоступны. Более того, некоторые желаемые входные наборы сигналов для модулей могут вообще не появляться, поскольку эти наборы должны появляться на выходах той части чипа, которые являются входами модуля. Но как известно, между элементами множеств входных и выходных наборов сигналов схемы, вообще говоря, не существует взаимно-однозначное соответствие. Поэтому, чтобы для модуля внутри чипа создать те же условия, которые были при его проверке вне чипа, и для искусственного создания этих условий внутри чипа, неизбежно введение необходимой избыточности. Хотя эти мероприятия обеспечивают условия проверки модуля, предусмотренными для случая его проверки вне чипа, но по существу теряется возможность проверки модуля в условиях внутри чипа, что совершенно недопустимо.

1. Новый подход интеграции модулей в чипе

Мы предлагаем принципиально другой подход интеграции модулей в чипе, исходя из того, что работоспособность модуля вне чипа и внутры чипа могут существенно отличаться друг от друга. Вне чипа он проверяется в среде, которая абсолютно не учитывает условия той среды чипа, где намечается его использовать. Так, при проверке работоспособности вне чипа на входах модуля устанавливается некоторое определенное распределение вероятностей появления наборов входных сигналов, при котором он, быть может, выдержит испытание, а внутри чипа это распределение, вообще говоря, отличается от вышеуказанного распределения, в результате чего модуль может уже не выдержать испытание. Причина этого явления объясняется тем, что определенный тактовый режим работы цифровой схемы каждому элементу диктует определенную частоту переключения (изменения) состояния его выхода (переходы $\mathbf{1} \rightarrow \mathbf{0}$ или $\mathbf{0} \rightarrow \mathbf{1}$).

Любой такой переход является большой нагрузкой для элемента, и он поглащает энергию, равную

$$\frac{CU^2}{2},$$

где C – емкость на выходе узла, перезаряжаемая в процессе переключения, U – напряжение питания. И если выход элемента в среднем переключается с частотой **n** сек$^{-1}$, то он потребляет мощность

$$\frac{CU^2n}{2}.$$

Вот эта частота переключения выхода элемента зависит не только (и не столько) от частоты работы модуля, а особенно от распределения вероятностей появления наборов входных сигналов. В этой связи вводится понятие **динамической активности** элемента [1] в схеме для данного режима её работы (т.е. при данном распределении вероятностей появления наборов входных сигналов). Она имеет вероятностный смысл: это вероятность того, что в моменты времени **t** и **t**+1 на выходе данного элемента появляются разные сигналы, т.е. в момент времени **t**+1 происходит переключение состояния выхода элемента. Здесь предполагается, что события появления указанных наборов в моменты **t** и **t**+1 независимы. Динамическая активность элемента **e**, выход которого относительно входов схемы реализует функцию $f_e(x_1,\ldots,x_n)$, вычисляется следующей формулой

$$A_e = 2\left\|f_e\right\|\cdot\left\|\bar{f}_e\right\|, \quad \text{где} \quad \left\|f_e\right\| = \sum_{(a_1,\ldots,a_n)} f_e(a_1,\ldots,a_n)\,p(a_1,\ldots,a_n),$$

а $p(a_1,\ldots,a_n)$ является вероятностью появления набора $(a_1,\ldots,a_n)$. Очевидно, динамическую активность элемента **e** можно приблизительно вычислить статистическими методами. Действительно, если схема (модуль) работает с частотой **v,** а выход элемента **e** при этом переключается в среднем с частотой **n** сек $^{-1}$, то отношение **n/v** приблизительно совпадает с динамической активностью элемента. Точность зависит от числа экспериментов при вычислении частоты переключения выхода элемента, которая определяется методами Монте-Карло.

Итак, разные среды для модуля диктуют разные распределения динамических активностей его элементов, а потребляемая мощность элемента пропорциональна его динамической активности, увеличение которой крайне нежелательно. Оно является одним из серьёзных источников ненадежности функционирования модуля.

Таким образом, работоспособность модуля должна проверяться только в режиме “модуль внутри чипа”.

Настоящая работа посвящена решению именно этой проблемы. Она решается двумя, в какой-то степени, эквивалентными способами. Сущность первого способа заключается в том, что из чипа выделяется минимальная его часть, все выходы которой подключены к соответствующим входам модуля. Очевидно, указанная часть чипа на входах модуля создает ту среду, которую создается полным чипом, но проверка работоспособности модуля внутри чипа упрощается тем, что в игру входит не весь чип, а только его часть. Сущность второго способа заключается в том, что статистическими методами определяется распределение вероятностей появления наборов состояний на выходах вышеуказанной части чипа, которое (распределение) затем, с помощью, соответствующим образом настроенным, генератора случайных чисел, используется для подачи наборов состояний с указанным распределением на входы модуля. Эти способы отличаются тем, что первый из них среду для модуля внутри чипа моделирует точно, а второй – несколько снижает точность, но зато работа с частью чипа заменяется работой с генератором случайных чисел, что существенно ускоряет работу.

В качестве основного инструмента описания схем используется специальный язык строчного описания схем **Алех**. Его краткое описание, а также описание некоторых важных процедур над схемами с использованием этого языка, можно найти в [2]. Эти процедуры существенным образом используются в настоящей работе, и мы настоятельно советуем читателя предварительно ознакомиться с этой работой, поскольку все основные понятия и обозначения, используемые здесь, взяты именно из этой работы.

2. Детальное описание процесса интеграции модуля в чипе с использованием языка Alex

Пусть на распоряжении проектировщика имеются частично спроектированный чип и готовый спроектированный модуль, полностью проверенный на работоспособность для определенного режима вне чипа. Схема интеграции модуля в чипе изображена на рис.1.

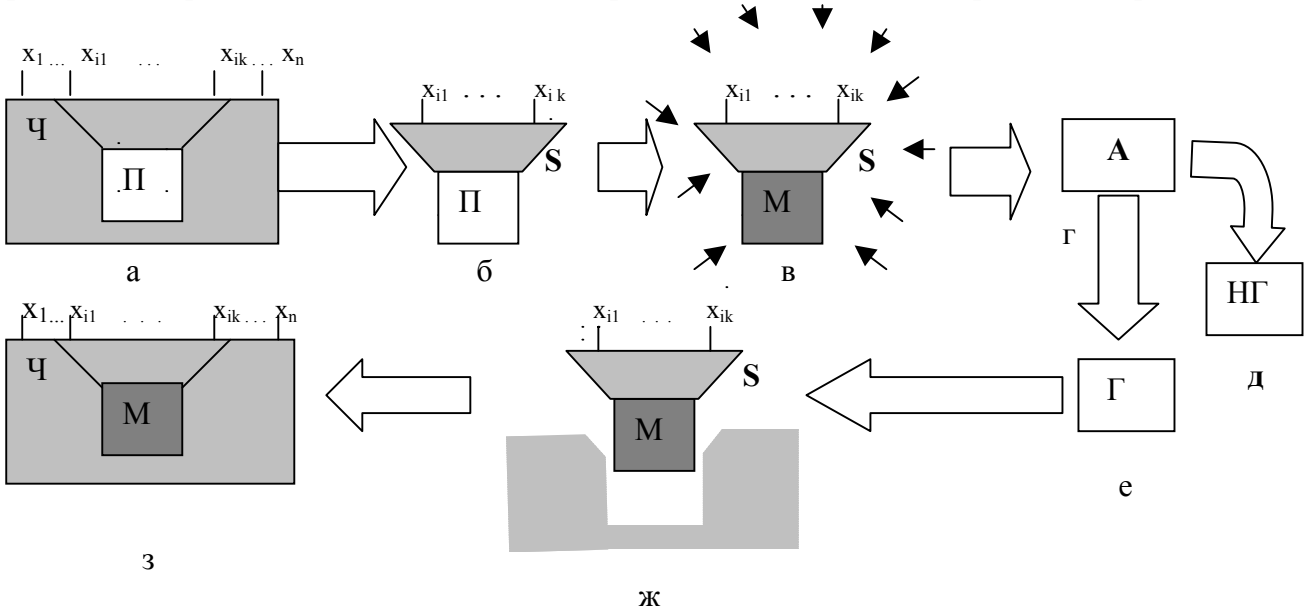

Рис.1. а)Частично спроектированный чип с полостью (**П**), предусмотренной для помещения модуля **М**; б) Выделение из чипа минимальной подсхемы **S,** выходы которой должны соединяться со входами модуля **М;** в)Последовательное соединение схемы **S** и модуля **М** и испытание **М** на работоспособность; г) Анализ годности (**А)** модуля **М** в среде чипа; д) Модуль является не годным (**НГ**) ; е) Модуль является годным (**Г**); ж) Помещение модуля **М** в чипе; з) Модуль **М** в чипе.

Переходные процедуры, указанные на рис.1 стрелками, подробно описаны в [2]. Что касается процедуры “испытание **М** на работоспособность”, указанная на рис.1,в, то она осуществляется тестированием этой схемы. Важной составной частью тестирования является процудура **функционального моделирования** схемы, о которой подробно говорится ниже и от которой существенно зависит время и, следовательно, качество тестирования схемы. Так как по предположению модуль спроектирован и проверен независимой фирмой и принципиально логически спроектирован правильно, под тестированием здесь понимается только вычисление динамических активностей элементов модуля **М** и проверка соблюдения условии относительно потребляемой мощности элементов. Смысл этого условия заключается в том, что потребляемая мощность элемента не должна выходить из допустимого значения, в противном случае безотказная работа элемента не гарантируется.

2.1. Функциональное моделирование комбинационной схемы

Язык **Alex** позволяет эффективно осуществить процедуру функционального моделирования комбинационной схемы. Под функциональным моделированием мы здесь понимаем нахождение значений сигналов на выходах схемы при наличии значений сигналов на её входах. Мы демонстрируем эту процедуру в виде алгоритма на частном примере, однако в нем сущность алгоритма и техника его осуществления обрисовываются четко.

Рассмотрим комбинационную схему **c17** из эталонных схем **ISCAS-85.** Она имеет пять входов и два выхода и описана на языке **Verilog:**

INPUT(G1gat)
INPUT(G2gat)
INPUT(G3gat)
INPUT(G6gat)
INPUT(G7gat)
OUTPUT(G22gat)
OUTPUT(G23gat)

G10gat = nand(G1gat, G3gat)
G11gat = nand(G3gat, G6gat)
G16gat = nand(G2gat, G11gat)
G19gat = nand(G11gat, G7gat)
G22gat = nand(G10gat, G16gat)
G23gat = nand(G16gat, G19gat)

Её графическое изображение приведено на рис.2.

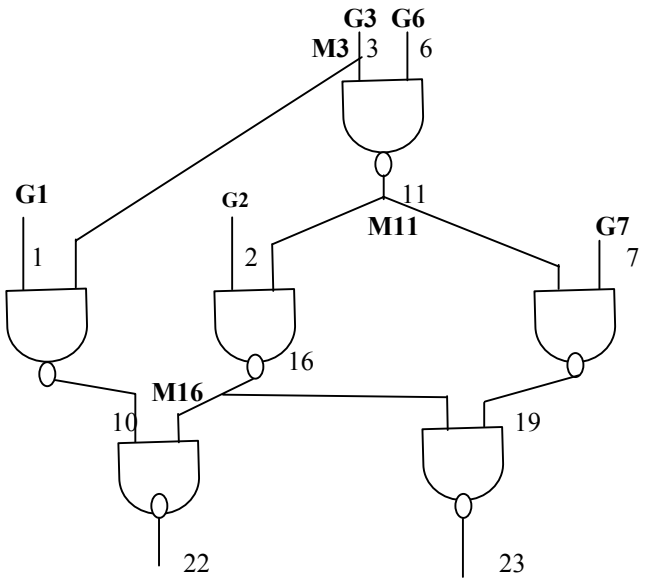

Рис.2

С целью представления схемы на языке **Alex** приведем основную идею построения алгоритма перевода описания схемы с языка **Verilog** на язык **Alex.** Эту идею мы приведем для рассмотренного нами примера. **С** самого начала заметим, что из описания схемы на языке **Verilog** видно, что все элементы (входы также считаются элементами) пронумерованы таким образом, что разным элементам приписаны разные номера, а входы элементов и выходы схемы определенным образом упорядочены. Например, строка **G16gat = nand(G2gat, G11gat)** показывает, что элемент **nand** с номером **16,** выход которого обозначен **G16gat**, имеет два входа, причем к первому входу подключен **G2gat,** т.е. выход элемента с номером **2**, а ко второму входу - **G11gat**, т.е. выход элемента с номером **11.** Что касается выходов схемы, то они упорядочены в описании на языке **Verilog** “сверху-вниз”, т.е. в нашем примере по порядку **OUTPUT(G22gat), OUTPUT(G23gat).** Далее, поскольку построение записи схемы на языке **Alex** осуществляется относительно выходов схемы в том порядке, в каком они расположены по упорядочению выходов, то процесс построения записи начинается с первого выхода и

кончается последним выходом. Первым выходом (**Output(G22gat)**) является выход элемента **nand** с номером **22.** Поэтому первым символом записи схемы на языке **Alex** должен быть **nand22(2).** Первым входом этого элемента является выход элемента **nand** с номером **10.** Поэтому вторым символом записи должен быть **nand10(2).** Далее, первым входом элемента с номером **10** является выход элемента (вход схемы) с номером **1.** Поэтому третьим символом записи будет **G1(0).** Поскольку элемент **G1gat** не имеет входов, мы переходим к рассмотрению второго входа элемента с номером **10.** Этот вход является выходом элемента с номером **3.** Однако выход элемента с номером **3** ветвляется, и только одна из ветвей является вторым входом элемента с номером **10.** Поскольку эта точка ветвления рассматривается впервые, то последующими символами записи схемы будут метка **M3(1)** и её описание, т.е. **G3(0).** Если точка ветвления рассматривается не впервые, то вместо указанной пары символов **M3(1)** и **G3(0)** записывается единственный символ **M3(0).** Наличием факта присутствия точки ветвления на выходе элемента с номером **3** обнаруживается повторением символа **G3(0)** в двух строках **Verilog** – описании схемы, а именно: **G10gat=nand(G1gat, G3gat)** и **G11gat=nand(G3gat, G6gat).** После завершения рассмотрения всех входов данного элемента рассматривается очередной вход того элемента, последний рассмотренный вход которого был выходом данного элемента. Очередным шагом является переход к рассмотрению следующего выхода схемы. Запись схемы полностью завершается с завершением записи относительно её последнего выхода.

Записью схемы **c17** является:

nand22(2)nand10(2)G1(0)M3(1)G3(0)M16(1)nand16(2)G2(0)M11(1)nand11(2)M3(0)G6(0) nand23(2)M16(0)nand19(2)M11(0)G7(0)

Описаниями меток **M3(1), M16(1)** и **M11(1)** в записи схемы являются соответственно:

ОпM3(1)=G3(0), ОпM16(1)=nand16(2)G2(0)M11(1)nand11(2)M3(0)G6(0), ОпM11(1)= nand11(2)M3(0)G6(0).

Поскольку **Mi(1)ОпMi(1)** и **Mi(0)** (**i** = 1,2, …) содержательно эквивалентны, то они взаимнозаменяемы. С целью наиболее эффективного осуществления алгоритма функционального моделирования схемы, её запись преобразуем таким образом, чтобы для любого индекса **i** все метки **Mi(0)** в записи предшествовали **Mi(1).** После такого преобразования записи получим:
nand22(2)nand10(2)G1(0)M3(0)M16(0)nand23(2)M16(1)nand16(2)G2(0)M11(0)nand19(2) M11(1)nand11(2)M3(1)G3(0)G6(0)G7(0)

В том случае, когда после осуществления данной процедуры над схемой, описанной на языке **Alex**, нет необходимости вернуться к описанию на языке **Verilog**, можно запись схемы разгрузить от лишней информации. Такой процедурой, в частности, является функциональное моделирование, для которого лишними являются номера элементов. После исключения номеров элементов, для нашего примера получим соответствующий вариант записи:

nand(2)nand(2)G1(0)M3(0)M16(0)nand(2)M16(1)nand(2)G2(0)M11(0)nand(2)M11(1)nand(2) M3(1)G3(0)G6(0)G7(0)

Если символы (входы) **G1(0), G2(0), G3(0), G6(0)** и **G7(0)** заменить метаобозначениями x_1, x_2, x_3, x_6 и x_7 соответственно, то получим более наглядное представление записи:
nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)M11(1)nand(2)M3(1)x_3 x_6x_7
Функциональное моделирование схемы для данного набора входных сигналов осуществляется последовательным (в любом порядке) применением следующих процедур:

1) f(k)a_1 . . . a_k заменить на **f(k)(a_1, . . . ,a_k),** где **$a_i \in$ {0,1}, i = 1,2, . . ., k.**

2) Mi(1)a и **Mi(0) (I =1, 2 , . . .)** заменить на **a** $\in$ **{0,1}.**

Во избежании многочисленных поисков символов типа **Mi(0)** при замене **Mi(0)** на **a,** если пара символов **Mi(1)a** уже заменена на **a,** дополнительно хранится текущая информация **(i, a, m_i),** где **i –** номер метки **Mi(1), m_i –** текущее значение числа вхождений символов **Mi(0)** в записи после очередного применения п.2. После каждого применения этого пункта число **m_i** уменьшается на единицу. При текущем значении **m_i=0** тройка **(i, a, 0)** исключается из списка.

Ниже шаг за шагом представлена работа алгоритма над схемой с приведенной записью. Пусть на её входы **x_1, x_2, x_3, x_6, x_7** подан набор сигналов **(0,0,1,0,1).** Ход применения алгоритма моделирования следующий:

nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)M11(1)nand(2)M3(1)x_3 x_6x_7
1.
nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)M11(1)nand(2)M3(1)x_3 $x_6$1
2.
nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)M11(1)nand(2)M3(1)$x_3$0 1
3.
nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)M11(1)nand(2)M3(1)10 1
4.
nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)M11(1)nand(2)101
{(3, 1, 1)}
5. nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)M11(1)11
{(3, 1, 1)}
6. nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)nand(2)11
{(3, 1, 1), (11, 1, 1)}
7. nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)x_2M11(0)0
{(3, 1, 1), (11, 1, 1)}
8. nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)$x_2$10
{(3, 1, 1), (11, 1, 0)} = {(3, 1, 1)}
9. nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)nand(2)010
{(3, 1, 1)}
10. nand(2)nand(2)x_1M3(0)M16(0)nand(2)M16(1)10
{(3, 1, 1)}
11. nand(2)nand(2)x_1M3(0)M16(0)nand(2)10
{(3, 1, 1), (16, 1, 1)}
12. nand(2)nand(2)x_1M3(0)M16(0)1
{(3, 1, 1), (16, 1, 1)}
13. nand(2)nand(2)x_1M3(0)11
{(3, 1, 1), (16, 1, 0)} = {(3, 1, 1)}
14. nand(2)nand(2)$x_1$111
{(3, 1, 0)} = ∅
15. nand(2)nand(2)0111
16. nand(2)111
17. 01

Итак, набору входных сигналов **(0,0,1,0,1)** соответствует набор выходных сигналов **(0,1).**

Как видно из рассмотренного примера, функциональное моделирование комбинационной схемы осуществляется одним просмотром записи схемы практически без поисков знаков операций и операндов. Поиски осуществляются только в случае определения значений меток типа **Mi(0)** во вспомогательном списке значений этих меток. Однако объем этого списка динамически меняется за счет прибавления и удаления строк типов **(i, a, m_i) ($m_i \neq 0$)** и **(i, a, 0)** соответственно, и его среднее значение для наиболее часто встречающихся на практике схем, является незначительным.

2.2. Функциональное моделирование цифровых схем из логических и запоминающих элементов

Традиционным представлением цифровых схем из логических и запоминающих элементов является представление, показанное на рис.3. Здесь элемент типа τ является запоминающим элементом, который поступающий на свой вход сигнал задерживает на один такт времени. Символ τ **(1)** в записи схемы имеет вес **- 1.**

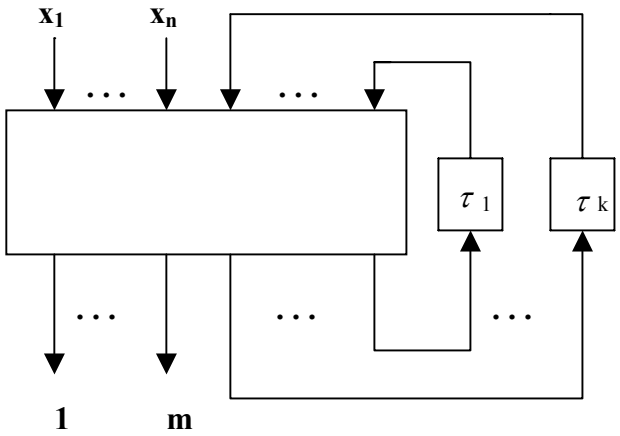

Рис.3

Функциональное моделирование схемы с памятью осуществляется следующим образом. Пусть в момент времени **t** на входы $x_1,..., x_n$ поступает набор сигналов $a_1(t),..., a_n(t)$, а в момент **t–1** на входы элементов $\tau_1,..., \tau_к$ поступил набор сигналов $b_1(t - 1),..., b_k(t – 1)$. Это значит, что в момент времени **t** на последние **k** входы комбинационной части схемы поступает этот же набор $b_1(t – 1),…,b_k (t -1)$ сигналов. Таким образом, указанная на рис.3 схема при функциональном моделировании функционирует как комбинационная схема с **n+k** входами и **m+k** выходами, причем роль последних **k** входов комбинационной схемы выполняют выходы элементов памяти $\tau_1,..., \tau_k$, а роль последних **k** выходов – входы элементов $\tau_1,..., \tau_k$. Итак, в момент времени **t** на выходах указанной условной комбинационной схемы появляется набор из нулей и единиц длины **m+k**, где первый левый набор длины **m** является набором выходных сигналов схемы с памятью, последний набор длины **k -** набор, поступающий на входы запоминающих элементов в момент времени **t.**

Рассмотрим схему из логических и запоминающих элементов **s27** из эталонных схем **ISCAS – 89.** Приведем её запись на языке **Alex.**

not(1)M11(1)nor(2)τ_1(1)nor(2)M14(1)not(1)x_1M11(0)nand(2)or(2)x_4M8(1)and(2)M14(0)τ_2(1) M11(0)or(2)M12(1)nor(2)x_2 τ_3(1)nor(2)x_3M12(0)M8(0)

С целью представления этой записи в виде записи комбинационной схемы выделим области действия символов τ_1, τ_2 и τ_3:

Об τ_1(1) = nor(2)M14(1)not(1)x_1M11(0), Об τ_2(1) =M11(0), Об τ_3(1) = nor(2)x_3M12(0)

Справа записи схемы приписываем **Об τ_1(1), Об τ_2(1)** и **Об τ_3(1),** а затем в полученном выражении отрезки **τ_1(1)Об τ_1(1), τ_2(1)Об τ_2(1)** и **τ_3(1)Об τ_3(1)** соответственно заменим на **b_1, b_2** и **b_3,** а символы **x_1, x_2, x_3** и **x_4** заменим соответственно на **a_1, a_2, a_3** и **a_4**. Получим

not(1)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(1)and(2)M14(0)$b_2$or(2)M12(1)nor(2)a_2b_3M 8(0)
nor(2)M14(1)not(1)a_1M11(0)M11(0)nor(2)a_3M12(0)

Мы получили запись комбинационной схемы с **7** входами и **4** выходами. Как в случае записи схемы **с 17**, запись преобразуем таким образом, чтобы для любого **i** метка **Mi(0)** в записи предшествовала **Mi(1).** В результате получим:

not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1b_2$nor(2)a_3M12(1)nor(2)a_2b_3

Пусть в момент времени **t** на входы схемы **x_1, x_2, x_3** и **x_4** поступили сигналы **a_1=1, a_2=0, a_3=1, a_4=0,** а на выходах элементов памяти τ_1, τ_2, τ_3 образовались сигналы со значениями **b_1=1, b_2=1, b_3=0** соответственно (эти последние сигналы формировались на входах указанных элементов памяти фактически в момент времени **t–1**). Процесс моделирования схемы протекает следующим образом:

not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1b_2$nor(2)a_3M12(1)nor(2)$a_2$0

not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1b_2$nor(2)a_3M12(1)nor(2)00

not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1b_2$nor(2)a_3M12(1)0

not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1b_2$nor(2)$a_3$0
{(12, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1b_2$nor(2)10
{(12, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1b_2$0
{(12, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)$a_1$10
{(12, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)not(1)110
{(12, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)M14(1)010
{(12, 0, 1)}

not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)and(2)010
{(12, 0, 1), (14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
M8(1)00
{(12, 0, 1), (14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)M12(0)
00
{(12, 0, 1), (14, 0, 1), (8, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)or(2)000
{(14, 0, 1), (8, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)a_4M8(0)00
{(14, 0, 1), (8, 0, 1)}

not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)$a_4$000
{(14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)or(2)0000
{(14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$nand(2)000
{(14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)$b_1$10
{(14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)nor(2)110
{(14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)M11(1)10
{(14, 0, 1)}
not(1)M11(0)nor(2)M14(0)M11(0)10
{(14, 0, 1),(11, 1, 2)}
not(1)M11(0)nor(2)M14(0)110
{(14, 0, 1),(11, 1, 1)}
not(1)M11(0)nor(2)0110
{(11, 1, 1)}
not(1)M11(0)110
{(11, 1, 1)}
not(1)1110

0110

Мы получили набор значений из четырех сигналов. Первый из них является значением на выходе схемы в момент времени **t**, остальные три - значения, поступающие на входы элементов памяти в момент времени **t.**

Литература

1. Бозоян Е.Ш. Оценка надежности функциональных схем с учетом частоты переключения её элементов. Изв. НАН и ГИУ Армении (сер. ТН), 1997, № 2, стр. 120 – 125.

2. Бозоян Ш.Е., Егиазарян В.С. Некоторые процедуры над логическими схемами и их реализация на языке **ALEX.** Электронный журнал «ИССЛЕДОВАНО В РОССИИ», htt://zhurnal.ape.relarm.ru/2003/073.pdf, стр. 817 – 824.