Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра программного обеспечения
информационных технологий

В. Н. Ярмолик, И. А. Мурашко, С. В. Ярмолик

КОНТРОЛЬ И ДИАГНОСТИКА СРЕДСТВ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Методическое пособие по выполнению лабораторных работ
для студентов специальности
1-40 02 01 «Вычислительные машины, системы и сети»
дневной формы обучения

Библиотека БГУИР

Минск БГУИР 2010

УДК 681.518.5(076)
ББК 3стд2-08я73
Я75

Р е ц е н з е н т:
заведующий кафедрой «Вычислительные методы и программирование»
учреждения образования « Белорусский государственный университет
информатики и радиоэлектроники»
кандидат технических наук, доцент А. А. Иванюк

Ярмолик, В. Н.

Я75 Контроль и диагностика средств вычислительной техники : метод. пособие по выполнению лаб. работ для студ. спец. 1-40 02 01 «Вычислительные машины, системы и сети» днев. формы обуч. / В. Н. Ярмолик, И. А. Мурашко, С. В. Ярмолик. – Минск : БГУИР, 2010. – 48 с. : ил.
ISBN 978-985-488-528-5.

Приводятся практические вопросы тестового диагностирования цифровых устройств. Рассматриваются классические вопросы синтеза тестовых последовательностей и их использование для процедур тестирования. Большое внимание уделяется современным методам компактного тестирования произвольных цифровых устройств, а также оперативных запоминающих устройств. По каждой теме приводятся теоретические сведения, практические способы реализации методов тестирования и набор вариантов заданий разной степени сложности.

УДК 681.518.5(076)
ББК 3стд2-08я73

Библиотека БГУИР

ISBN 978-985-488-528-5

© Ярмолик В. Н., Мурашко И. А.,
Ярмолик С. В., 2010
© УО «Белорусский государственный
университет информатики
и радиоэлектроники», 2010

СОДЕРЖАНИЕ

Библиотека БГУИР

1.ТЕСТИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ

Контроль и диагностика цифровых устройств представляют собой область инженерных знаний, исследующую две основные задачи, а именно, задачу определения исправности цифрового устройства и задачу поиска неисправности в случае его неисправного состояния.

Для решения этих задач используются два подхода: *тестовое диагностирование* и *функциональное диагностирование*.

Тестовое диагностирование предполагает, что объект диагностирования не используется в своем рабочем состоянии, т. е. не выполняет своих основных функций, а находится в режиме тестирования. Недостатком такого подхода является необходимость дополнительного ресурса времени, однако в то же время тестовое диагностирование позволяет более полно и с большей достоверностью решить указанные выше две основные задачи диагностики.

Тестовое диагностирование реализуется по алгоритмам, приведенным на рис. 1.1, 1.2 и 1.3.

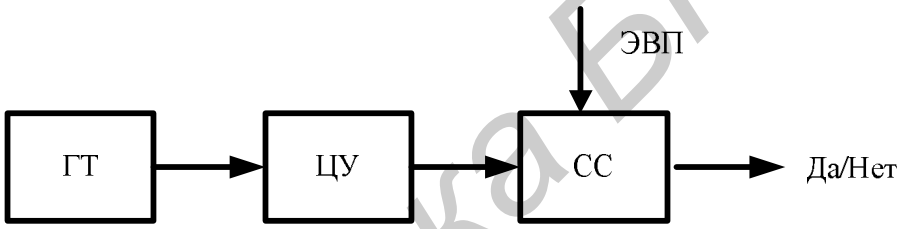


Рис. 1.1. Тестовое диагностирование
с известной эталонной выходной реакцией

На рис. 1.1 использованы следующие сокращения: ГТ – генератор тестов; ЦУ – цифровое устройство; СС – схема сравнения; ЭВП – эталонная выходная последовательность.

Генератор тестов (ГТ) формирует тестовую последовательность цифрового устройства (ЦУ). Выходные реакции на тестовые воздействия сравниваются с эталонными выходными последовательностями (ЭВП) на схеме сравнения (СС). В случае совпадения реальных выходных последовательностей с их эталонными значениями принимается решение об исправном состоянии цифрового устройства, соответственно формируется сигнал «Да». В противном случае устройство находится в одном из возможных его неисправных состоянии, о чем свидетельствует сигнал «Нет».

В случае наличия заведомо исправного эталонного цифрового устройства (ЭЦУ) применяется схема тестирования, приведенная на рис. 1.2. Согласно данной схеме решение об исправности или неисправности цифрового устройства принимается по результатам сравнения выходных реакций проверяемого цифрового устройства и его эталонной версии.

Библиотека БГУИР

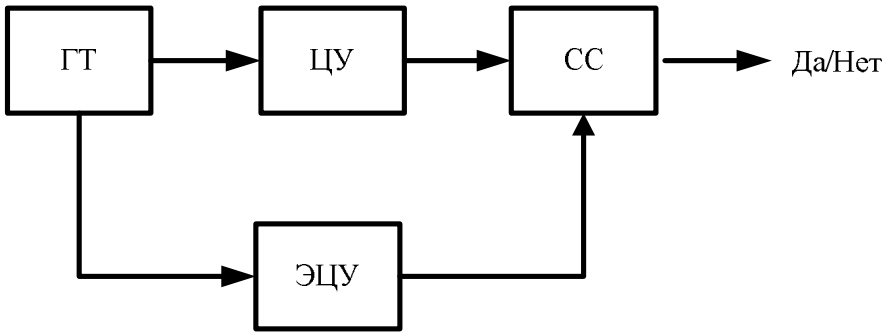


Рис. 1.2. Тестовое диагностирование с эталонным цифровым устройством

Следующий алгоритм тестирования основан на использовании сжатой (компактной) формы как реальной выходной реакции цифрового устройства, так и эталонной реакции представленной в виде эталонной сигнатуры (ЭС). Для получения эталонных реакций используется схема компрессии (сжатия) (СК).

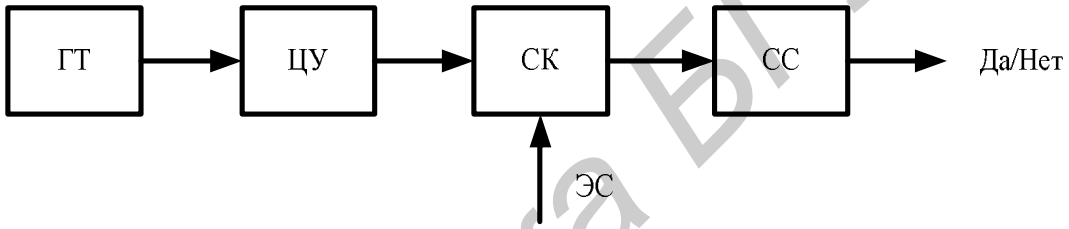


Рис. 1.3. Компактное тестовое диагностирование

Данный алгоритм получил название компактного тестирования цифровых устройств.

Проблема тестового диагностирования цифровых устройств (в дальнейшем ЦУ) возникает на различных этапах их производства и эксплуатации и состоит из решения двух взаимосвязанных задач.

Первая из них заключается в определении, в каком состоянии находится исследуемая схема. Принято различать несколько видов технического состояния цифровых устройств. Основным состоянием ЦУ является *исправное* – это такое техническое состояние устройства, при котором данное устройство удовлетворяет всем требованиям, установленным технической документацией. В противном случае устройство находится в одном из его возможных неисправных состояний.

Если установлено, что ЦУ неисправно, то решается вторая задача: осуществляется поиск неисправности (дефекта) устройства, цель которого – определение места и вида неисправности.

Неисправности цифровых устройств появляются в результате применения неисправных составных компонентов, таких, как логические элементы, элементы памяти и др. Кроме того, причиной неисправностей могут быть возникновение разрывов или коротких замыканий в межкомпонентных соединени-

Библиотека БГУИР

ях, нарушение условий эксплуатации устройства, наличие ошибок при проектировании и производстве и множество других факторов.

Из всего множества различных видов неисправностей выделяют класс *логических* неисправностей, которые изменяют логические функции элементов цифрового устройства. Указанный класс неисправностей является доминирующим среди неисправностей цифровых устройств (схем) (ЦС). Для их описания используются следующие математические модели: константные неисправности, неисправности типа «короткое замыкание», неисправности типа «задержка» и «инверсная неисправность».

Наиболее общей и часто применяемой моделью логической неисправности являются *константные неисправности*: *константный* (*тождественный*) *ноль* (≡0), *константная* (*тождественная*) *единица* (≡1) , что означает наличие постоянного уровня логического нуля или логической единицы на входах или выходе неисправного логического элемента (ЛЭ). Такая модель неисправности часто называется «классической» и широко применяется для описания других типов неисправностей.

Различают *одиночные* и *многократные* константные неисправности. Одиночная неисправность – неисправность, которая присутствует только на одном полюсе ЦС. Существует $2N$ таких неисправностей, где N – количество полюсов ЦС. В случае когда константные неисправности могут присутствовать на нескольких логических полюсах ЦС, т. е. каждый полюс может находиться в одном из трех состояний, исправном и двух неисправных состояниях ≡0 и ≡1, их общее количество определяется как $3^N - 1$.

Неисправности типа «*короткое замыкание*» (мостиковые неисправности) появляются при коротком замыкании полюсов ЦС, в том числе входов или выходов ЛЭ.

Неисправность типа «*задержка*» характеризуется тем, что на конкретном исправном полюсе возникает фиктивный элемент задержки.

«*Инверсная неисправность*» – эквивалентна появлению фиктивного инвертора на неисправном полюсе.

В настоящее время, как правило, все тесты направлены на поиск одиночных *константных* неисправностей, что объясняется невозможностью построения тестов для более сложных типов неисправностей. Кроме того, тесты, синтезированные для класса одиночных константных неисправностей, характеризуются высокой полнотой покрытия (обнаруживающей способностью) других типов неисправностей.

1.1. Методы построения тестов

1.1.1. Метод активизации одномерного пути

Основная идея этого метода заключается в активизации пути от места возникновения неисправности к одному из выходов схемы.

Библиотека БГУИР

Процедура построения теста для комбинационного цифрового устройства по этому методу состоит из следующих этапов:

1. Определяется условие наблюдаемости неисправности в точке ее возникновения.

2. Выбирается путь от неисправного полюса к одному из выходов схемы. Путь задается в виде последовательности элементов, лежащих на этом пути.

3. Определяется условие активности выбранного пути. Путь является активным, если изменение логического уровня в начале пути приводит к инвертированию значений на выходах всех элементов, лежащих на выбранном пути.

4. Решается обратная задача, т. е. условия наблюдаемости неисправности, и активизации выбранного пути определяются в терминах входных переменных комбинационного ЦУ.

В результате получаем множество наборов входных переменных, из которого выбирается один набор, который совместно с соответствующим эталонным выходным значением называется *тестовым набором*.

В качестве примера рассмотрим комбинационную схему, приведенную на рис. 1.4.

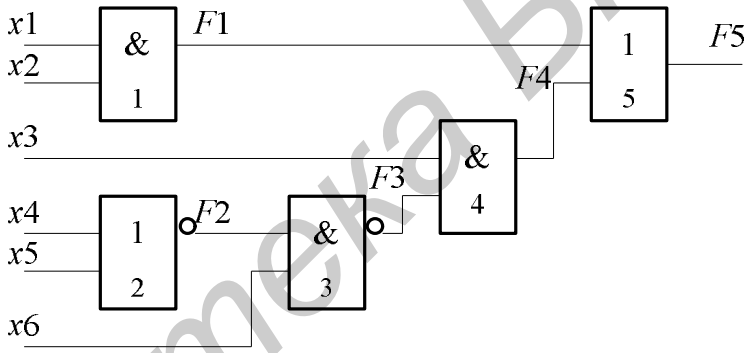


Рис. 1.4. Тестируемая комбинационная схема

Предположим, необходимо построить тест для обнаружения неисправности $F3 \equiv 1$, применив метод активизации одномерного пути.

1. Условием наблюдаемости неисправности $F3 \equiv 1$ в месте ее возникновения будет условие формирования на неисправном полюсе уровня логического нуля, т. е. $F3 = 0$.

2. Путь, по которому неисправность может быть транспортирована на выход схемы, проходит через 4-й и 5-й элементы схемы.

3. Условие активности пути через 4-й элемент обеспечивается равенством $x3 = 1$, а через 5-й – выполнением равенства $F1 = 0$.

4. $F3 = \overline{\overline{(x4 + x5)} \cdot x6} = 0$, отсюда следует, что $\overline{(x4 + x5)} \cdot x6 = 1$.

Используя последнее уравнение и два уравнения $x3 = 1,\ F1 = x1 \cdot x2 = 0$, вытекающие из условий активизации, находим их решения, которые и будут являться тестовыми наборами:

Библиотека БГУИР

$x1$	$x2$	$x3$	$x4$	$x5$	$x6$
0	0	1	0	0	1
0	1	1	0	0	1
1	0	1	0	0	1

Для второго тестового набора 011001 эталонное выходное значение $F5$ равняется 0, а в случае наличия неисправности $F3 \equiv 1$ $F5$ принимает значение 1. Тестовый набор 011001, кроме неисправности $F3 \equiv 1$, также обнаруживает неисправность $F5 \equiv 1$ и $F4 \equiv 1$.

В общем случае тестовые наборы строятся только для характеристических неисправностей, каковыми для случая комбинационной ЦС являются неисправности входных полюсов и неисправности ветвей разветвления.

1.1.2. Метод активизации многомерного пути

Метод активизации многомерного пути (*d*-алгоритм, или алгоритм Рота) является исторически первым методом, который для случая комбинационных ЦС позволяет гарантированно получить тестовый набор, если он существует.

d-алгоритм основан на использовании кубического описания цифровых схем. При реализации *d*-алгоритма применяются *сингулярные кубы*, *d-кубы* и *примитивные d-кубы неисправностей*. Для задания сингулярных кубов используются три символа: 0, 1 и *x*, где символ *x* обозначает безразличное значение 0 либо 1. Сингулярные кубы строятся согласно следующему алгоритму.

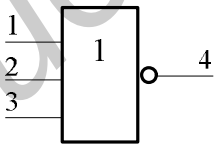
Алгоритм построения *сингулярных кубов* для логических элементов:

1. По выходному полюсу логического элемента задается определенный символ – 0 либо 1.

2. По входным полюсам элемента задается минимальное количество символов, обеспечивающих заданное выходное значение.

3. По остальным входам задается символ безразличного значения x.

В качестве примера рассмотрим логический элемент 3ИЛИ-НЕ и соответствующие ему сингулярные кубы.



1	2	3	4
1	x	x	0
x	1	x	0
x	x	1	0
0	0	0	1

Для задания *d-кубов* используется пять символов: 0, 1, *x*, *d* и $\overline{d}$. Где *x* – означает безразличное значение; символ $d \in \{0, 1\}$ также принимает одно из двух значений. Однако в одном и том же кубе все символы *d* принимают одно и то же значение (в отличие от символов *x*), а $\overline{d}$ – противоположное значение.

Библиотека БГУИР

d-кубы логических элементов строятся на основании сингулярных покрытий элементов. При этом используются следующие правила операции пересечения:

$$0 \cap 0 = 0 \qquad 1 \cap 1 = 1$$
$$0 \cap x = 0 \qquad 1 \cap x = 1$$
$$x \cap 0 = 0 \qquad x \cap 1 = 1$$
$$x \cap x = x \qquad 1 \cap 0 = d$$
$$0 \cap 1 = \bar{d}$$

Для построения *d*-куба элемента из его сингулярного покрытия выбираются два сингулярных куба с противоположными значениями по выходной координате и затем пересекаются по приведенным выше правилам операции пересечения.

Для случая логического элемента 3ИЛИ-НЕ пересечение его первого и четвертого сингулярных кубов показано ниже:

1	2	3	4
1	*x*	*x*	0
0	0	0	1
d	0	0	$\bar{d}$

d-кубы используются для того, чтобы формально решить задачу активизации путей в тестируемой схеме (транспортировки неисправности на выход).

Примитивные d-кубы неисправностей используются для активизации неисправности на неисправном полюсе. Для этого в случае неисправности ≡0 (≡1) выходного полюса элемента на данном полюсе задается значение *d* ($\bar{d}$), что означает наличие на данном полюсе значений 1 (0) при отсутствии неисправности. При наличии неисправности ≡0 (≡1) на данном полюсе фиксируются значения 0 (1). По входным полюсам задается минимальное множество определенных символов, обеспечивающих по выходу элемента значений 1 (0) для неисправности ≡0 (≡1), а по остальным входам задается символ безразличного значения *x*.

Для случая неисправности ≡0 на выходном полюсе логического элемента 3ИЛИ-НЕ примитивный *d*-куб неисправности принимает вид

1	2	3	4
0	0	0	*d*

Для неисправности ≡1 элемента 3ИЛИ-НЕ существует три примитивных *d*-куба неисправности, представленных ниже:

1	2	3	4
1	*x*	*x*	$\bar{d}$
x	1	x	$\bar{d}$
x	x	1	$\bar{d}$

Для реализации *d*-алгоритма используется операция *d-пересечения*, которая определяется для двух исходных кубов.

Библиотека БГУИР

Результатом *d*-пересечения кубов $A=(a_1, a_2, ..., a_n)$; $B=(b_1, b_2, ..., b_n)$, где $a_i, b_i \in \{0, 1, x, d, \bar{d}\}$, а *d*-пересечение кубов *A* и *B* обозначается как $\cap_d$, является результирующий куб $A \cap_d B = a_1 \cap_d b_1, a_2 \cap_d b_2, ..., a_n \cap_d b_n$. Причем $A \cap_d B = \emptyset$, если хотя бы для одного *i* выполняется равенство $a_i \cap_d b_i = \emptyset$. *Покоординатная операция d-пересечения* определяется следующими соотношениями:

$$a_1 \cap_d b_1 = \begin{cases} a_i, \text{ если } b_i = a_i \text{ или } b_i = x; \\ b_i, \text{ если } a_i = b_i \text{ или } a_i = x; \\ \emptyset \text{ в остальных случаях} \end{cases}$$

Суть *d*-алгоритма заключается в том, что примитивный *d*-куб неисправности конкретного элемента схемы расширяется до *d*-куба всей схемы, который описывает исправное и неисправное поведение всей схемы.

Непосредственно *d*-алгоритм состоит из следующих этапов:

1. Выбирается примитивный *d*-куб заданной неисправности элемента схемы.

2. Активизируются возможные пути от элемента, содержащего неисправность к выходу либо нескольким выходам схемы. Для этого применяется операция *d*-пересечения *d*-куба неисправности с *d*-кубами элементов, последовательно включенных между неисправным элементом и выходами схемы. Этот этап называется *d*-проходом, формальным окончанием которого является появление символа *d* или $\bar{d}$ на одном или нескольких выходных схемы.

3. Выполняется обратная фаза как процедура последовательного пересечения *d*-куба, полученного в п. 2, с сингулярными кубами остальных элементов, т. е. элементов, которые не участвуют в выполнении двух предыдущих пунктов. При выполнении данной фазы соблюдается правило обратного хода, заключающееся в последовательном использовании элементов от выходных полюсов схемы до ее входов.

В качестве примера рассмотрим цифровую схему, приведенную на рис. 1.4. Предположим, что для данной схемы необходимо построить тест для обнаружения неисправности $F3 \equiv 1$, применив метод активизации многомерного пути. Предварительно каждому полюсу схемы поставим в соответствие уникальный идентификатор (индекс) в соответствии с табл. 1.1.

Таблица 1.1

*x*1	*x*2	*x*3	*x*4	*x*5	*x*6	*F*1	*F*2	*F*3	*F*4	*F*5
1	2	3	4	5	6	7	8	9	10	11

1. На первом этапе задается примитивный *d*-куб неисправности $F3 \equiv 1$ для третьего элемента схемы, который имеет следующий вид:

Библиотека БГУИР

*F*3

8	**6**	9
1	1	$\overline{d}$

Сингулярные кубы 1-го и 2-го элементов (эти элементы не лежат на пути от неисправного полюса к выходу схемы) и *d*-кубы 4-го и 5-го элементов приведены на рис. 1.5.

*F*1 *F*2

1	2	7	4	5	8
0	*x*	0	1	*x*	0
x	0	0	*x*	1	0
1	1	1	0	0	1

*F*4 *F*5

3	9	10	7	10	11
$\overline{d}$	1	$\overline{d}$	$\overline{d}$	0	$\overline{d}$
1	$\overline{d}$	$\overline{d}$	*d*	0	*d*
1	*d*	*d*	0	$\overline{d}$	$\overline{d}$
d	1	1	0	*d*	*d*

Рис. 1.5. Сингулярные кубы

2. На втором этапе последовательно выполняется операция *d*-пересечения примитивного *d*-куба неисправности с *d*-кубами элементов, лежащих между неисправным полюсом 9 (*F*3) и выходом схемы 11 (*F*5).

1	2	3	4	5	6	7	8	9	10	11	Комментарии
x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	Исходное состояние полюсов не определено.
x	*x*	x	*x*	*x*	1	*x*	1	$\overline{d}$	*x*	*x*	Задается примитивный *d*-куб неисправность *F*3 ≡ 1.
x	*x*	*x*	*x*	*x*	1	*x*	1	$\overline{d}$	*x*	*x*	Результат пересечения.
x	*x*	$\overline{d}$	*x*	*x*	*x*	*x*	*x*	1	$\overline{d}$	*x*	Выполняется пересечение с первым *d*-кубом для *F*4.
x	*x*	$\overline{d}$	*x*	*x*	1	*x*	1	∅	$\overline{d}$	*x*	По 9-му полюсу получен результат ∅.
x	*x*	*x*	*x*	*x*	1	*x*	1	$\overline{d}$	*x*	*x*	Поэтому выполняется возврат на предыдущую итерацию.
x	*x*	1	*x*	*x*	*x*	*x*	*x*	$\overline{d}$	$\overline{d}$	*x*	Выбирается второй *d*-куб *F*4.
x	*x*	1	*x*	*x*	1	*x*	1	$\overline{d}$	$\overline{d}$	*x*	Результат пересечения.
x	*x*	*x*	*x*	*x*	*x*	0	*x*	*x*	$\overline{d}$	$\overline{d}$	Выбирается *d*-куб для *F*5.
x	*x*	1	*x*	*x*	1	0	1	$\overline{d}$	$\overline{d}$	$\overline{d}$	Результат пересечения.

3. На третьем этапе выполняется пересечение с сингулярными кубами.

1	2	3	4	5	6	7	8	9	10	11	Комментарии
x	*x*	1	*x*	*x*	1	0	1	$\overline{d}$	$\overline{d}$	$\overline{d}$	Результат, полученный на 2-м этапе.
0	*x*	x	*x*	*x*	x	0	*x*	*x*	*x*	*x*	Для *F*1 выбираем сингулярный куб.
0	*x*	1	*x*	*x*	1	0	1	$\overline{d}$	$\overline{d}$	$\overline{d}$	Результат пересечения.
x	*x*	*x*	0	0	*x*	*x*	1	*x*	*x*	*x*	Для *F*2 выбираем сингулярный куб.
0	*x*	1	0	0	1	0	1	$\overline{d}$	$\overline{d}$	$\overline{d}$	Результат пересечения.

Библиотека БГУИР

Окончательно имеем результирующий куб, который описывает поведение цифровой схемы в исправном состоянии и в случае наличия неисправности $F3 \equiv 1$.

Так, в случае отсутствия неисправности при подаче на входы схемы одного из двух тестовых наборов 001001, 011001 на выходном полюсе будет получен 0, о чем свидетельствует символ $\bar{d}$ на выходном полюсе 11. При наличии неисправности на данном полюсе будет получен уровень 1.

1.2. Задание для выполнения лабораторной работы

Лабораторная работа выполняется каждым студентом индивидуально согласно приведенной на рис. 1.6 схеме и выданными преподавателем типами логических элементов.

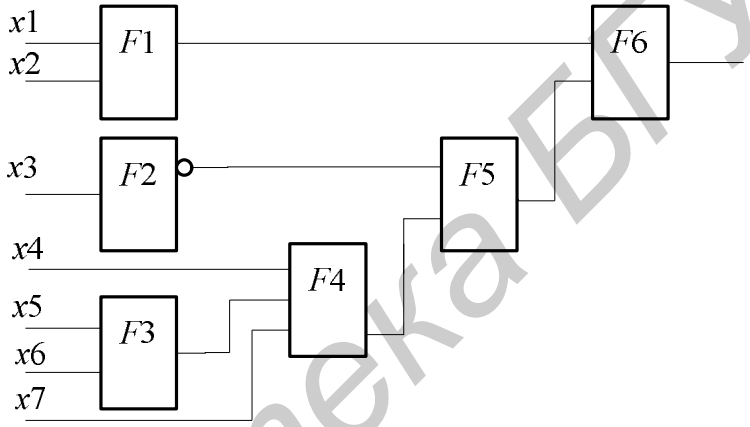


Рис. 1.6. Тестируемая комбинационная схема

Задание для лабораторной работы №1

Разработать программу, которая для любой одиночной константной неисправности находит тестовый набор по методу активизации одномерного пути.

Задание для лабораторной работы №2

Разработать программу, которая находит множество тестовых наборов, покрывающее все одиночные константные неисправности по методу активизации многомерного пути.

Библиотека БГУИР

2. МИНИМИЗАЦИЯ ПОТРЕБЛЕНИЯ ЭНЕРГИИ ПРИ ПРОВЕДЕНИИ ТЕСТИРОВАНИЯ ЦИФРОВЫХ СХЕМ

В настоящее время при проектировании цифровых устройств на основе СБИС одним из главных факторов, ограничивающих плотность компоновки, становится энергопотребление и связанное с ним выделение тепла. Поэтому важной задачей является снижения энергопотребления цифровых устройств при проведении проверки их работоспособности. Можно выделить две основные причины актуальности данной проблемы. Первая причина связана с тем, что в момент тестирования (самотестирования) наблюдаются пик потребления энергии и соответственно пик рассеивания тепла. Потребление энергии может возрастать в 2−3 раза. Это может привести к выходу устройства из строя вследствие перегрева. Вторая причина связана со снижением срока службы автономных источников питания мобильных цифровых устройств.

В данной работе рассматриваются основные источники энергопотребления цифровых интегральных схем (ИС), выполненных по КМОП-технологии, методика оценки рассеиваемой мощности, а также один из подходов к минимизации потребления цифровых устройств при проведении тестирования с использованием детерминированных тестовых наборов.

Рассеиваемую мощность для КМОП-схем можно разделить на два вида: динамическую и статическую. Динамическая рассеиваемая мощность возникает в момент переключения схемы из одного логического состояния в другое и определяется двумя основными источниками – сквозными токами, которые протекают через логический элемент в момент переключения, и токами заряда/разряда паразитных емкостей логических элементов. Следовательно, она зависит от переключательной активности схемы, т. е. чем выше переключательная активность схемы, тем больше рассеиваемая мощность. При отсутствии переключений динамическая мощность равна нулю. Статическая мощность рассеивается тогда, когда логический элемент находится в некотором фиксированном логическом состоянии (0 или 1), и определяется токами утечки канала МОП-транзистора, обратными токами *p*-*n*-переходов и токами внешних выводов ИС. На рис. 2.1 представлена классификация источников рассеиваемой мощности.

В идеальном случае в статическом состоянии КМОП-элемент не потребляет мощности, т. е. ток через него не протекает. Однако в реальных схемах ток через него течет. Величина этого тока составляет порядок пикоампер, соответственно и статическая рассеиваемая мощность, определяемая этими токами, на несколько порядков меньше, чем динамическая.

Динамическая рассеиваемая мощность определяется двумя компонентами: токами заряда/разряда конденсаторов и сквозными токами через вентили.

Большинство цифровых КМОП-схем не требует использования конденсаторов для выполнения своих функций (исключение составляет динамическая память, устройства выборки/хранения, устройства задержки сигналов и т. п.).

Библиотека БГУИР

Конденсаторы образуются за счет паразитных емкостей транзисторов и линий связи, поэтому от них нельзя избавиться. Паразитная емкость оказывает существенное влияние как на время задержки распространения сигнала, так и на рассеиваемую мощность. Оценку паразитной емкости будем проводить на примере инвертора, принципиальная схема которого приведена на рис. 2.2, а. Инвертор состоит из *p*-канального (VT_p) и *n*-канального (VT_n) МОП-транзисторов. Паразитная емкость (рис. 2.2, б) образуется из емкости входных линий связи C_{in}, емкости выходных линий связи C_{out} и паразитных емкостей транзисторов (C_{GD1}, C_{GD2}, C_{GS1}, C_{GS2} – паразитные емкости затвор-сток и затвор-исток соответственно, C_{DB1}, C_{DB2}, C_{SB1}, C_{SB2} – паразитные емкости сток-подложка и исток-подложка соответственно). Величины этих паразитных емкостей зависят от геометрических размеров элементов, технологического процесса изготовления, применяемых материалов и т. п. В данной работе эти емкости будем учитывать в виде эквивалентной переключаемой емкости C_L, расположенной по выходу логического элемента.

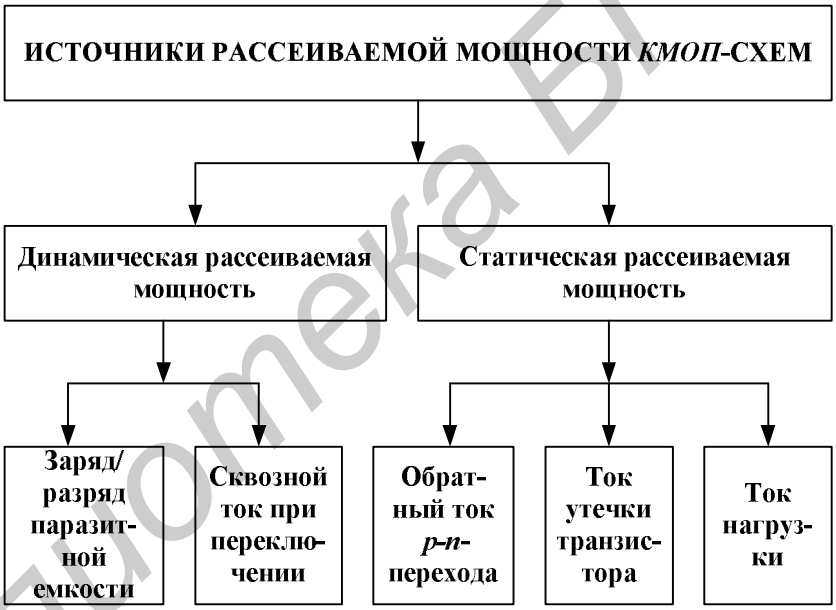


Рис. 2.1. Источники рассеиваемой мощности КМОП схем

Энергия, потребляемая узлом КМОП-схемы при переключении, определяется следующим выражением:

$$E_1 = \frac{1}{2} CV^2, \qquad (2.1)$$

где C – суммарная переключаемая емкость; V – напряжение питания (точнее, амплитуда напряжения переключения).

Библиотека БГУИР

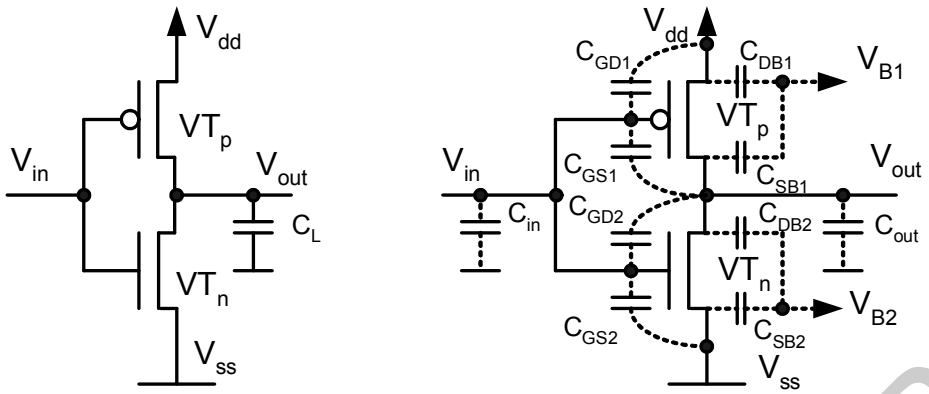


Рис. 2.2. Принципиальная схема КМОП-инвертора (а) и эквивалентная схема инвертора для расчета паразитной емкости (б)

Мощность, рассеиваемая КМОП-схемой вследствие протекания сквозного тока, может составлять 10–60 % от общей динамической мощности. Ее величина зависит от таких параметров, как вольт-амперные характеристики транзисторов (напряжение отпирания и запирания транзисторов), длительность переднего и заднего фронта входного сигнала, емкости нагрузки, напряжения питания и т. п.

Для упрощения расчетов динамической мощности, как правило, используется выражение (2.1), в котором сквозной ток учитывается в виде дополнительной емкости нагрузки C_{SC}. Таким образом, для расчета динамической мощности узла логической схемы будем использовать следующее выражение:

$$P = \frac{1}{2} C_L V^2 f, \tag{2.2}$$

где C_L – переключаемая емкость (которая учитывает как паразитные емкости вентилей, так и дополнительную емкость нагрузки C_{SC}); V – напряжение питания; f – частота изменения логического уровня.

Рассмотрим анализ рассеиваемой мощности более подробно. Энергия, потребляемая j-м узлом схемы при переключении, равна $1/2 C_j V^2_{dd}$, где C_j – емкостная нагрузка j-го узла, V_{dd} – напряжение источника питания. Следовательно, выражение $1/2 f_j C_j V^2_{dd}$ позволяет оценить потребляемую j-м узлом схемы энергию за f_j переключений. Для узлов, к которым подключено несколько входов логических элементов, емкостная нагрузка возрастает пропорционально числу входов s_j. Поэтому выражение для энергии, потребляемой j-м узлом схемы, можно записать следующим образом:

$$E_j = \frac{1}{2} s_j f_j C_0 V^2_{dd}, \tag{2.3}$$

Библиотека БГУИР

где C_0 – номинальная (нормализованная) емкостная нагрузка одного входа, которая является одинаковой для всех логических элементов, выполненных по единой технологии. Обозначим через E_0 энергию одного переключения стандартного входа:

$$E_0 = \frac{1}{2} C_0 \cdot V_{dd}^2. \tag{2.4}$$

Тогда для оценки потребляемой узлом j энергии необходимо знать число переключений f_j и количество входов логических элементов s_j, подключенных к данному узлу. Произведение $s_j f_j$ определим как переключательную активность SA_j (*Switching Activity – SA*) узла j и будем использовать в качестве оценки для рассеиваемой этим узлом энергии. Соответственно переключательная активность всей схемы за один такт синхронизации запишется

$$SA_{CLK} = \sum_{j=1}^{v} s_j \cdot f_j, \tag{2.5}$$

где f_j – число переключений j-го узла за один такт синхронизации; s_j – количество входов логических элементов, подключенных к данному узлу; v – количество узлов логической схемы.

Тогда переключательная активность всей схемы за n тактов работы может быть найдена из следующего выражения:

$$SA = \sum_{i=1}^{n} \sum_{j=1}^{v} s_j \cdot f_j. \tag{2.6}$$

Таким образом, энергия, потребляема схемой, может быть найдена следующим образом:

$$E = SA \cdot E_0. \tag{2.7}$$

Разделив полученное выражение на время t, в течение которого была потреблена энергия (или, что равнозначно, умножив данное выражение на частоту F), получим рассеиваемую мощность схемы:

$$P = \frac{E}{t} = E \cdot F. \tag{2.8}$$

Рассмотрим пример расчета рассеиваемой мощности на примере тестирования схемы на рис. 2.3. В табл. 2.1 представлены все возможные тестовые наборы (ТН) для данной схемы и обнаруживаемые ими неисправности.

Библиотека БГУИР

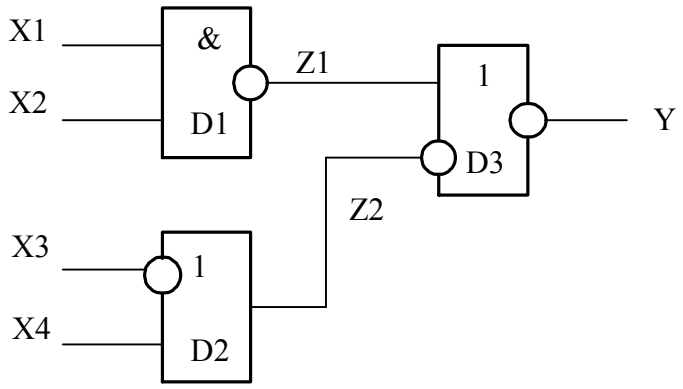


Рис. 2.3. Пример тестируемой схемы

Таблица 2.1

№	*X*1	*X*2	*X*3	*X*4	*Z*1	*Z*2	*Y*	Обнаруживаемые неисправности		Число обнаруженных неисправностей
								Const 0	*Const* 1	
1	0	0	0	0	1	1	0	*Z*1	*Y*	2
2	0	0	0	1	1	1	0	*Z*1	*Y*	2
3	0	0	1	0	1	0	0		*Y*	1
4	0	0	1	1	1	1	0	*Z*1	*Y*	2
5	0	1	0	0	1	1	0	*Z*1	*X*1, *Y*	3
6	0	1	0	1	1	1	0	*Z*1	*X*1, *Y*	3
7	0	1	1	0	1	0	0		*Y*	1
8	0	1	1	1	1	1	0	*Z*1	*X*1, *Y*	3
9	1	0	0	0	1	1	0	*Z*1	*X*2, *Y*	3
10	1	0	0	1	1	1	0	*Z*1	*X*2, *Y*	3
11	1	0	1	0	1	0	0		*Y*	1
12	1	0	1	1	1	1	0	*Z*1	*X*2, *Y*	3
13	1	1	0	0	0	1	1	*X*1, *X*2, *Z*2, *Y*	*X*3, *Z*1	6
14	1	1	0	1	0	1	1	*X*1, *X*2, *Z*2, *Y*	*Z*1	5
15	1	1	1	0	0	0	0	*X*3	*X*4, *Z*2, *Y*	4
16	1	1	1	1	0	1	1	*X*1, *X*2, *X*4, *Z*2, *Y*	*Z*1	6

На основании табл. 2.1 находим минимальное количество тестовых наборов, которое позволит обнаружить все неисправности тестируемой схемы. Для этого сначала выберем тестовый набор №16, который покроет максимальное количество неисправностей №6. Затем из оставшихся тестовых наборов выбираем такой, который покроет максимум из непокрытых неисправностей. Полученные тестовые наборы представлены в табл. 2.2

Найдем число переключений при подаче данного множества тестовых наборов на тестируемую схему, после окончания подачи наборов возвращаемся

Библиотека БГУИР

к исходному набору. Получили, что при тестировании схемы, приведенной на рис. 2.3, происходит 20 переключений (табл. 2.3). При этом используем допущение, что выход схемы подключен к одному логическому входу.

Таблица 2.2

№	*X*1	*X*2	*X*3	*X*4	Const 0							Const 1						
					*X*1	*X*2	*X*3	*X*4	*Z*1	*Z*2	*Y*	*X*1	*X*2	*X*3	*X*4	*Z*1	*Z*2	*Y*
1	1	1	1	1	+	+		+		+	+					+		
2	1	1	1	0			+								+		+	+
3	1	0	1	1					+				+					+
4	1	1	0	0	+	+				+	+			+		+		
5	0	1	0	0					+			+						+
Покрытие					+	+	+	+	+	+	+	+	+	+	+	+	+	+

Таблица 2.3

№	*X*1	*X*2	*X*3	*X*4	*Z*1	*Z*2	*Y*	Число переключений
1	1	1	1	1	0	1	1	
2	1	1	1	0	0	0	0	3
3	1	0	1	1	1	1	0	4
4	1	1	0	0	0	1	1	5
5	0	1	0	0	1	1	0	3
1	1	1	1	1	0	1	1	5
Всего								20

Поставим задачу минимизации числа переключений при тестировании схемы. Для этого построим табл. 2.4, в которой рассчитаем число переключений при подаче различных комбинаций из двух тестовых наборов (ТН).

Таблица 2.4

ТН	1	2	3	4	5
1	–	3	3	2	5
2	3	–	4	3	4
3	3	4	–	5	4
4	2	3	5	–	3
5	5	4	4	3	–

На основании табл. 2.4 находим такую последовательность подачи ТН, при которой число переключений будет минимально. Один из простейших алгоритмов поиска – так называемый «жадный» алгоритм. Суть его заключается в следующем. Из таблицы выбираем пару тестовых наборов, которая имеет минимум переключений, например ТН1 и ТН4. Вычеркиваем 1-й и 4-й столбцы. Затем в 4-ой строке ищем минимальное значение во всех столбцах за исключением вычеркнутых. Таких значений два и они находятся во 2-м и 5-м столбце.

Библиотека БГУИР

Выбираем значение из 2-го столбца и вычеркиваем этот столбец. Затем во 2-й строке ищем минимальное значение среди неудаленных столбцов. Выбираем значение из 3-го столбца и вычеркиваем его. В третьей строке осталось одно не вычеркнутое значение в пятом столбце. Итак, получили следующую последовательность подачи тестовых наборов: ТН1-ТН4-ТН2-ТН3-ТН5-ТН1.

Соответственно число переключений $SA = 2 + 3 + 4 + 4 + 5 = 18$. Получили 18 переключений, что на 2 меньше, чем в предыдущем примере. Однако «жадный» алгоритм не гарантирует достижения абсолютного минимального значения. Например, при подаче ТН в следующей последовательности ТН2-ТН4-ТН5-ТН3-ТН1-ТН2 получим $SA = 3 + 3 + 4 + 3 + 3 = 16$. В общем случае необходимо рассчитать все возможные варианты подачи тестовых наборов.

Для расчета мощности будем считать, что напряжение питания $V_{dd} = 3{,}3$ В, $C_0 = 11$ пФ, частота подачи равна 10 МГц, соответственно длительность одного тактового импульса равна 0,1 мкс. На основании (2.4) найдем энергию одного переключения.

$$E_0 = \frac{1}{2} \cdot 3{,}3^2 \cdot 11 \cdot 10^{-12} = 0{,}59895 \cdot 10^{-10} \text{ Дж.}$$

Для подачи 5-ти тестовых наборов требуется 5 тактов, следовательно, $t = 5 \cdot 0{,}1 = 0{,}5 \text{ мкс} = 5 \cdot 10^{-7} \text{ с}$. Воспользовавшись (2.8), получим

$$P = \frac{E}{t} = \frac{SA \cdot E_0}{t} = \frac{16 \cdot 0{,}59895 \cdot 10^{-10}}{5 \cdot 10^{-7}} = 9{,}5832/5 \;=\; 1{,}91664 \cdot 10^{-3} \text{ Вт.}$$

2.1. Задание для выполнения лабораторной работы

Задание для лабораторной работы №3

Разработать программу, которая минимизирует число переключений при подаче тестовых наборов, при этом необходимо обеспечить 100 %-ное покрытие неисправностей заданной схемы.

Библиотека БГУИР

3. ГЕНЕРАТОРЫ ТЕСТОВ НА БАЗЕ РЕГИСТРОВ СДВИГА С ЛИНЕЙНОЙ ОБРАТНОЙ СВЯЗЬЮ

Одним из ключевых элементов любой системы встроенного самотестирования цифровых устройств является генератор тестовых наборов (ГТН). В современных цифровых устройствах ГТН, как правило, состоит из двух частей: источника тестовой последовательности и устройства транспортировки тестовых наборов на входы тестируемой схемы – цепь сканирования (ЦС). В качестве тестовой последовательности чаще всего используется *M*-последовательность (псевдослучайная последовательность максимальной длины). Это связано с простотой аппаратной реализации генератора *M*-последовательности, который состоит из сдвигового регистра с линейной обратной связью, так называемого LFSR (*Linear Feedback Shift Register*). Работа LFSR определяется характеристическим полиномом $\varphi(x)$, который определяет конфигурацию обратной связи. Пример LFSR с характеристическим полиномом $\varphi(x) = 1 \oplus x^3 \oplus x^4$ представлен на рис. 3.1, а состояние выходов триггеров представлено в табл. 3.1.

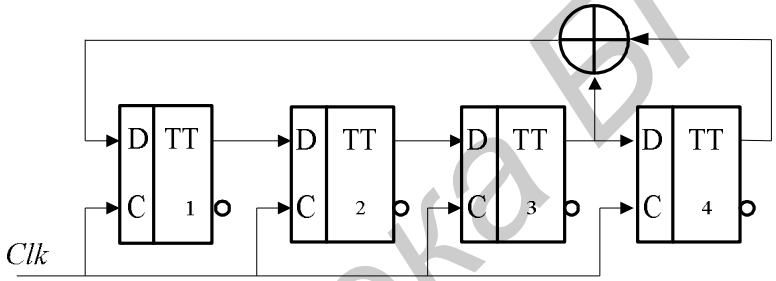


Рис. 3.1. Функциональная схема LFSR

Таблица 3.1

№ такта	Состояние элементов памяти				№ такта	Состояние элементов памяти			
	*Q*1	*Q*2	*Q*3	*Q*4		*Q*1	*Q*2	*Q*3	*Q*4
1	1	0	0	0	9	1	0	1	0
2	0	1	0	0	10	1	1	0	1
3	0	0	1	0	11	1	1	1	0
4	1	0	0	1	12	1	1	1	1
5	1	1	0	0	13	0	1	1	1
6	0	1	1	0	14	0	0	1	1
7	1	0	1	1	15	0	0	0	1
8	0	1	0	1	16	1	0	0	0

Период формируемой псевдослучайной последовательности (ПСП) зависит от свойств характеристического полинома. В рассмотренном примере период равен максимально возможному значению – 15, и соответственно генерируемая последовательность является *M*-последовательностью. Для формирования *M*-последовательности достаточно, чтобы характеристический полином являлся неприводимым. В табл. 3.2 приведено число примитивных полиномов

Библиотека БГУИР

для m от 1 до 19, а в табл. 3.3 – примеры примитивных неприводимых полиномов с минимальным числом ненулевых коэффициентов.

Полином $\varphi(x)$ является примитивным, если полином вида $x^k \oplus 1$ делится на $\varphi(x)$ только при $k = 2^m - 1$, где $m = deg\ \varphi(x)$ – старшая степень полинома $\varphi(x)$. Полином $\varphi(x)$ является неприводимым, если он не делится ни на какой другой полином степени, меньшей чем m, где $m = deg\ \varphi(x)$ – старшая степень полинома $\varphi(x)$.

Для удобства изложения применим следующие обозначения. Пусть a_i есть i-й символ M-последовательности, определяемой порождающим полиномом $\varphi(x)$ степени m. Заданную M-последовательность, начинающуюся с k-го символа, будем обозначать как $\{a_k\} = a_k a_{k+1} a_{k+2} \ldots a_{L-1} a_0 a_1 \ldots a_{k-1}$, где $L = 2^m - 1$ – период M-последовательности. Будем считать, что порождающий полином является примитивным, если не оговорено иное. При анализе многоканальных (параллельных) генераторов M-последовательности через $a_i(k)$ будем обозначать k-й символ M-последовательности, формируемый i-м разрядом генератора.

На рис. 3.2 представлены стандартные структуры реализации LFSR с внешними (*Type* I) и внутренними (*Type* II) сумматорами. Примеры реализации этих генераторов при использовании полинома $\varphi(x) = 1 \oplus x^3 \oplus x^4$ представлены на рис. 3.3.

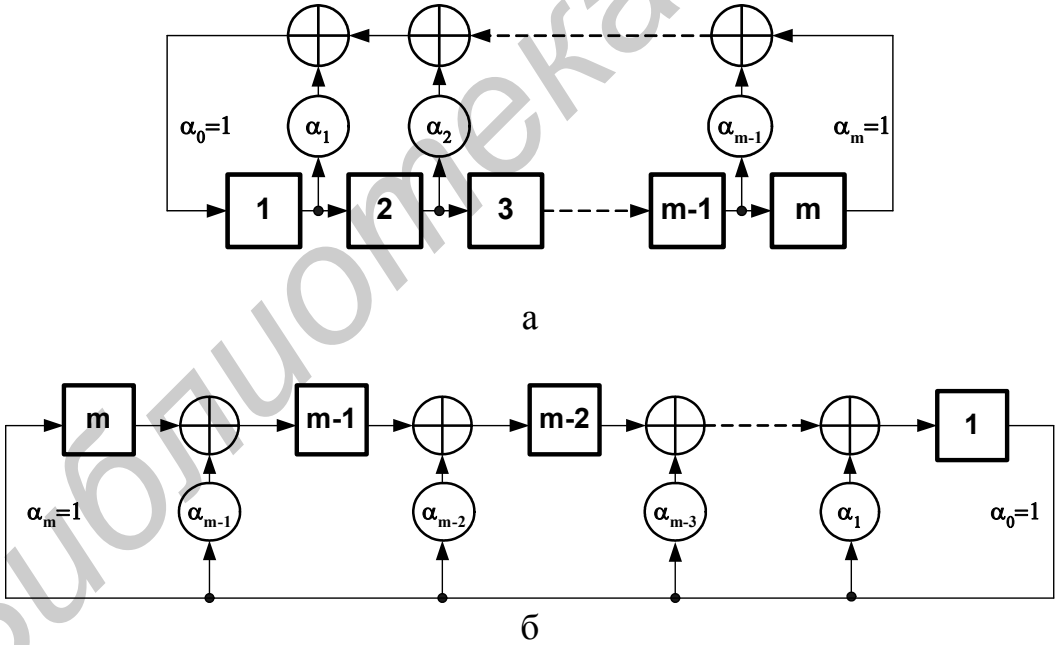


Рис. 3.2. Стандартные структуры LFSR:
а – *Type* I; б – *Type* II

Рассмотрим LFSR, приведенные на рис. 3.1, 3.2, а и 3.3. Регистр сдвига (РС) во всех приведенных структурах выполняет функцию хранения текущего состояния генератора и сдвига этого значения на один разряд вправо. Сумматоры по модулю два в цепи обратной связи выполняют операцию вычисления следующего значения, поступающего на вход первого триггера регистра сдвига.

Библиотека БГУИР

Представим текущее состояние РС в виде двоичного вектора $A(k) = a_1(k)a_2(k)\ a_m(k)$, $k = 0, 1, 2, \ldots$ Тогда в матричной форме функционирование подобного LFSR может быть описано следующим выражением:

$$A(k+1) = VA(k), \tag{3.1}$$

где $A(k)$, $A(k+1)$ предыдущее и последующее состояние LFSR, V – порождающая матрица генератора. Для примера LFSR, представленного на рис. 3.1, порождающая матрица имеет вид:

$$V = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.2}$$

Пусть текущее состояние LFSR задано вектором 1000. Тогда следующее состояние определяется как

$$A(k+1) = VA(k) = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{3.3}$$

По аналогии состояние LFSR через k тактов на основании (3.1) можно записать

$$A(k+s) = V^s A(k). \tag{3.4}$$

Таким образом, полученное выражение позволяет вычислить состояние LFSR в произвольный момент времени.

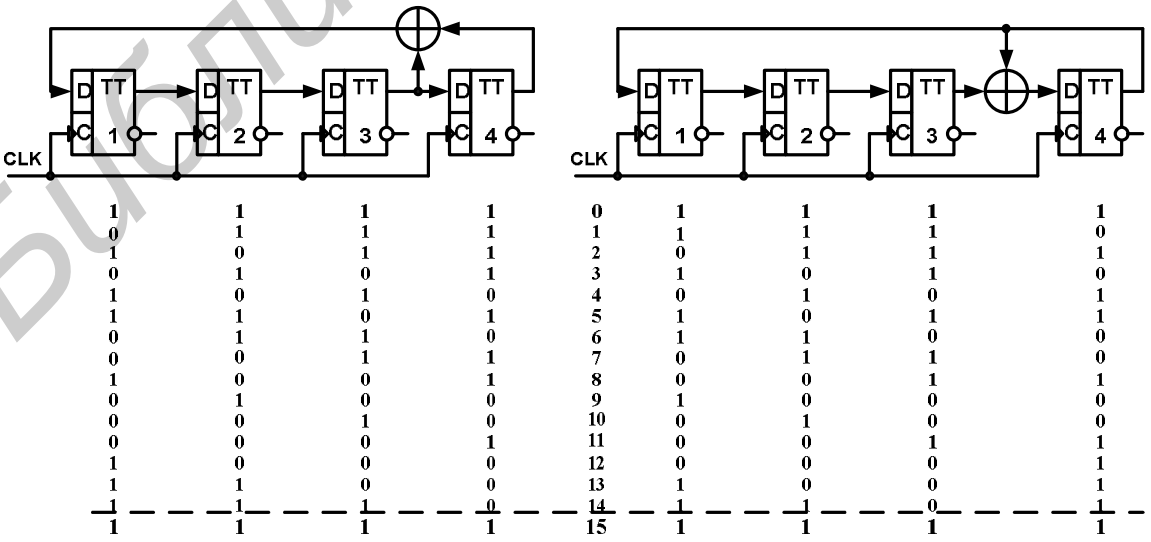


Рис. 3.3. Примеры формирования M-последовательностей на базе LFSR *Type* I и *Type* II

Библиотека БГУИР

Рассмотрим основные свойства *M*-последовательностей.

1. Для заданного m существует $\frac{\Phi(2^m-1)}{m}$ различных примитивных полиномов степени m, где $\Phi(x)$ – функция Эйлера, которая определяет количество целых чисел меньших целого число x и взаимно простых с x. С ростом m значение $(2^m - 1)$ быстро растет, соответственно быстро растет и число примитивных полиномов. Число примитивных полиномов для $m \le 19$ представлено в табл. 3.2.

Таблица 3.2

m	Число примитивных полиномов степени m	m	Число примитивных полиномов степени m
2	1	11	176
3	2	12	144
4	2	13	630
5	6	14	156
6	6	15	1800
7	18	16	2048
8	16	17	7710
9	48	18	7776
10	60	19	27594

2. Для заданного полинома $\varphi(x)$ степени m существует взаимно обратный полином $\varphi^{-1}(x)$, который связан с ним соотношением

$$\varphi^{-1}(x) = x^m\varphi(1/x) \qquad (3.5)$$

Если $\varphi(x)$ является примитивным, то и взаимно обратный ему полином также является примитивным.

Рассмотрим пример. Пусть $\varphi(x) = 1 \oplus x^3 \oplus x^5$. Тогда $\varphi^{-1}(x) = x^5\varphi(1/x) = x^5(1 \oplus (1/x)^3 \oplus (1/x)^5) = x^5 \oplus x^2 \oplus 1$. Получили, что для полинома $\varphi(x) = 1 \oplus x^3 \oplus x^5$ взаимно обратным является полином $\varphi^{-1}(x) = 1 \oplus x^2 \oplus x^5$, который также является примитивным.

3. Период *M*-последовательности L определяется старшей степенью порождающего полинома и равен $2^m - 1$, где $m = \deg \varphi(x)$, т. е. $L = 2^m - 1$.

4. Для заданного полинома $\varphi(x)$ степени m существует $L = 2^m - 1$ различных *M*-последовательностей, различающихся фазовым сдвигом $\{a_0\}$, $\{a_1\}$, $\{a_2\}$,…, $\{a_{L-1}\}$.

В качестве примера рассмотрим пример $\{a_i\} = 111101011001000$. Тогда $\{a_{i+1}\} = 111010110010001$, $\{a_{i+2}\} = 110101100100011$, ...

5. Для заданного полинома $\varphi(x)$ степени m существует такая *M*-последовательность, для которой справедливо соотношение $a_i = a_{2i}$, где $i = 0, 1, 2, \ldots$. Такая М-последовательность называется характеристической.

Библиотека БГУИР

В дальнейшем будем обозначать ее как $\{a_0\}$. На рис. 3.4 приведен пример характеристической последовательности, формируемой LFSR, который представлен на рис. 3.1.

№ такта	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\{a_0\}$	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0
$\{a_0\}^2$	0		1		1		1		1		0		1		0	

Рис. 3.4. Характеристический сдвиг М-последовательности

Характеристический сдвиг *M*-последовательности, формируемой LFSR (см. рис. 3.1), может быть найден путем решения следующей системы линейных уравнений:

$a_0 = a_0$;

$a_1 = a_2$;

$a_2 = a_4 = a_0 \oplus a_3$;

$a_3 = a_6 = a_2 \oplus a_5 = a_2 \oplus a_1 \oplus a_4 = a_2 \oplus a_1 \oplus a_0 \oplus a_1 = a_0 \oplus a_1$.

Решение системы дает $a_0 = 0$, $a_1 = a_2 = a_3 = 1$.

Таким образом, $\{a_0\} = 011110101100100\ldots$.

6. В *M*-последовательности, определяемой характеристическим полиномом $\varphi(x)$ степени m, число единичных символов равно 2^{m-1}, а число нулевых символов равно $2^{m-1} - 1$. Соответственно вероятности появления единичного и нулевого символов равны

$$p(a_k = 1) = \frac{2^{m-1}}{2^m - 1} = \frac{1}{2} + \frac{1}{2^{m+1} - 2};$$

$$p(a_k = 0) = \frac{2^{m-1} - 1}{2^m - 1} = \frac{1}{2} - \frac{1}{2^{m+1} - 2}.$$

7. *Свойство серий*. В *M*-последовательности, описываемой характеристическим полиномом $\varphi(x)$ степени m, серия из одной единицы встретится 2^{m-1} раз, а серия из одного нуля – $2^{m-1} - 1$ раз. Серия из двух идущих подряд единиц встретится 2^{m-2} раза, а серия из двух идущих подряд нулей встретится $2^{m-2} - 1$ раза. Серия 10 (а также серия 01) встретится 2^{m-2} раза. В общем случае серия из r символов встретится в *M*-последовательности 2^{m-r} раза.

8. Автокорреляционная функция *M*-последовательности равна

$$R(\tau) = \begin{cases} 1, & \tau = 0 \bmod L; \\ \dfrac{1}{L}, & \tau \neq 0 \bmod L. \end{cases}$$

Другими словами, среди L фазовых сдвигов *M*-последовательности только в одном случае из L происходит совпадение, а в остальных $L - 1$ случаях они не совпадают.

9. *Свойство сдвига и сложения*. Для любого s ($1 \le s \le L$) существует такое $r \neq s$ ($1 \le r \le L$), для которого истинно соотношение $\{a_k\} \oplus \{a_{k-s}\} = \{a_{k-r}\}$.

Библиотека БГУИР

Для произвольных s и k при суммировании по модулю два сдвинутых копий M-последовательности получается некоторая третья, сдвинутая на r тактов копия той же M-последовательности. Просуммируем поэлементно два фазовых сдвига M-последовательности, представленной на рис. 3.5. В результате получим фазовый сдвиг той же M-последовательности, для которого выполняется равенство $\{a_0\} \oplus \{a_2\} = \{a_9\}$.

№ такта	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\{a_0\}$	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0
$\{a_2\}$	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
$\{a_0\} \oplus \{a_2\} = \{a_9\}$	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1

Рис. 3.5. Пример свойства сдвига и сложения

Это же свойство по отношению к символам М-последовательности можно сформулировать следующим образом: $a_k \oplus a_{k-s} = a_{k-r}$. Таким образом, при суммировании по модулю два двух символов М-последовательности получается некоторый символ той же М-последовательности.

10. *Свойство децимации.* Децимацией M-последовательности $\{a_i\}$ по индексу q ($q = 1, 2, 3, \ldots$) называется выборка q-х элементов данной M-после-довательности. В результате децимации формируется некоторая последовательность $\{b_j\}$. Если период исходной M-последовательности $L = 2^m - 1$ и коэффициент децимации q взаимно просты – $(L, q) = 1$, децимация называется собственной или нормальной. Новая последовательность является M-последова-тельностью, определяемой примитивным полиномом той же степени, что и исходная. В дальнейшем под децимацией будем подразумевать только собственную (или нормальную) децимацию, в результате которой получается M-последовательность того же периода, что исходная M-последовательность. Децимацию $\{a_i\}$ по индексу q будем обозначать через $\{a_i\}^q$. Новую M-пос-ледовательность будем обозначать как $\{b_j\}$. Таким образом, можно записать $\{b_j\} = \{a_i\}^q$.

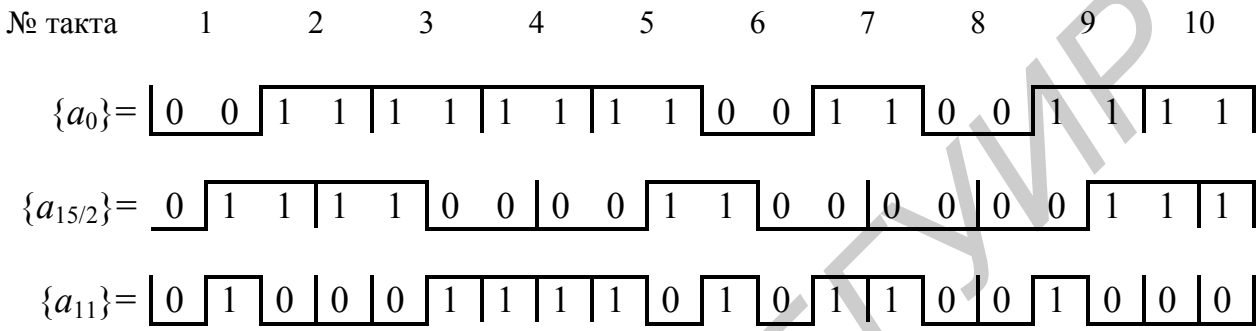
На рис. 3.6 приведен пример децимации M-последовательности $\{a_3\}$, определяемой порождающим полиномом $\varphi(x) = x^4 \oplus x^3 \oplus 1$, по индексу 2 и 3. В первом случай формируется 9-й фазовый сдвиг исходной M-последовательности, т. е. $\{a_3\}^2 = \{a_9\}$. Во втором случае формируется вырожденная последовательность, которая не является M-последовательностью. Это вызвано тем, что индекс децимации и период M-последовательности не являются взаимно простыми числами, т. е. $(15, 3) = 3$.

№ такта	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	0
$\{a_3\}$	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1	1
$\{a_3\}^2$	1		0		0		1		0		0		0		1	
$\{a_3\}^3$	1			1			1			1			0			0

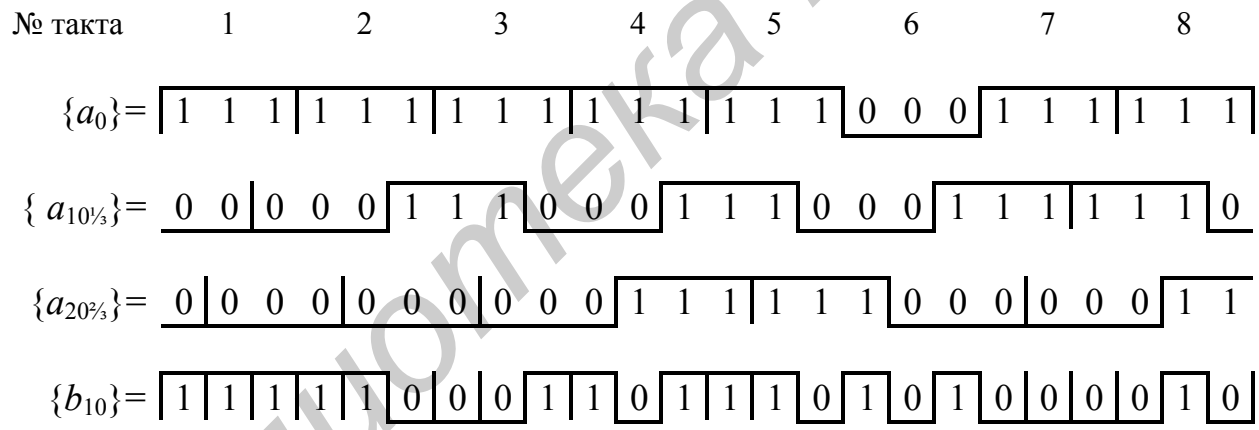
Рис. 3.6. Пример децимации М-последовательности

Библиотека БГУИР

11. Увеличение частоты формирования *M*-последовательности. Если просуммировать по модулю два две *M*-последовательности, сдвинутые ровно на половину периода друг относительно друга, то получим ту же самую *M*-последовательность, но формируемую с удвоенной частотой. На рис. 3.7, а М-последовательность $\{a_0\}$, определяемая порождающим полиномом $\varphi(x) = x^4 \oplus x^3 \oplus 1$, суммируется со своей сдвинутой ровно на половину периода копией $\{a_{L/2}\}$, где $L = 2^4 - 1 = 15$. В результате получаем последовательность $\{a_{11}\}$, формируемую с удвоенной частотой, т. е. $\{a_0\} \oplus \{a_{15/2}\} = \{a_{11}\}$.



а



б

$\{a_0\}$=11111 01110 00101 01101 00001 10010 0
$\varphi(x)=x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$
$\{b_0\}$=10010 11001 11110 00110 11101 01000 0
$\varphi(x)=x^5 \oplus x^3 \oplus 1$

в

Рис. 3.7. Формирование *M*-последовательности с удвоенной (а) и утроенной частотой (б) и характеристические сдвиги исходной и результирующей *M*-последовательностей (в)

Рассмотрим второй пример для *M*-последовательности, определяемой порождающим полиномом $\varphi(x) = x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$, сформируем три сдвинутые

Библиотека БГУИР

ровно на треть периода копии (т. е. на 31/3 = 10⅓ такта) и просуммируем их. В результате получим *M*-последовательность, определяемую порождающим полиномом $\varphi(x) = x^5 \oplus x^2 \oplus 1$ и сдвинутую на 10 тактов относительно характеристического сдвига, символы которой формируются в три раза быстрее, чем символы исходных последовательностей (см. рис. 3.7, б). На рис. 3.7, в приведены характеристические сдвиги *M*-последовательностей, определяемых соответствующими полиномами.

Данное свойство *M*-последовательности является комбинацией свойства сдвига и сложения и свойства децимации. Но если при децимации *M*-последовательности происходит снижение частоты формирования (прореживание последовательности), то в данном случае происходит увеличение частоты ее формирования.

3.1. Задание для выполнения лабораторной работы

Задание для лабораторной работы №4

Разработать программу, которая моделирует работу LFSR с заданным порождающим полиномом. Для заданной тестируемой схемы найти такое начальное состояние LFSR, при котором число тактов работы будет минимально, при этом необходимо обеспечить 100 %-ное покрытие неисправностей.

Таблица 3.3

$n = 3$	$n = 8$	$n = 13$
$x^3 \oplus x^2 \oplus 1$	$x^8 \oplus x^6 \oplus x^5 \oplus x \oplus 1$	$x^{13} \oplus x^4 \oplus x^3 \oplus x \oplus 1$
$x^3 \oplus x \oplus 1$	$x^8 \oplus x^5 \oplus x^3 \oplus x \oplus 1$	$x^{13} \oplus x^9 \oplus x^8 \oplus x^7 \oplus x^2 \oplus x \oplus 1$
	$x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x \oplus 1$	$x^{13} \oplus x^8 \oplus x^5 \oplus x^3 \oplus x^2 \oplus x \oplus 1$
	$x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$	$x^{13} \oplus x^9 \oplus x^7 \oplus x^5 \oplus x^4 \oplus x \oplus 1$
$n = 4$	$x^8 \oplus x^5 \oplus x^3 \oplus x^2 \oplus 1$	$n = 15$
$x^4 \oplus x^3 \oplus 1$	$x^8 \oplus x^6 \oplus x^3 \oplus x^2 \oplus 1$	$x^{15} \oplus x \oplus 1$
$x^4 \oplus x \oplus 1$	$x^8 \oplus x^7 \oplus x^6 \oplus x^3 \oplus x^2 \oplus x \oplus 1$	$x^{15} \oplus x^{10} \oplus x^5 \oplus x \oplus 1$
	$x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$	$x^{15} \oplus x^{12} \oplus x^7 \oplus x \oplus 1$
	$x^8 \oplus x^6 \oplus x^5 \oplus x \oplus 1$	$x^{15} \oplus x^{10} \oplus x^5 \oplus x^4 \oplus x^2 \oplus x \oplus 1$
	$x^8 \oplus x^5 \oplus x^3 \oplus x \oplus 1$	$x^{15} \oplus x^{10} \oplus x^9 \oplus x^7 \oplus x^5 \oplus x^3 \oplus 1$
	$x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$	$x^{15} \oplus x^{14} \oplus x^{13} \oplus x^8 \oplus x^5 \oplus x^4 \oplus 1$
$n = 5$	$x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$	$n = 16$
$x^5 \oplus x^3 \oplus 1$	$x^8 \oplus x^5 \oplus x^3 \oplus x^2 \oplus 1$	$x^{16} \oplus x^5 \oplus x^3 \oplus x^2 \oplus 1$
$x^5 \oplus x^2 \oplus 1$	$x^8 \oplus x^6 \oplus x^3 \oplus x^2 \oplus 1$	$x^{16} \oplus x^{11} \oplus x^8 \oplus x^7 \oplus x^6 \oplus x^2 \oplus 1$
$x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$	$x^8 \oplus x^7 \oplus x^6 \oplus x^3 \oplus x^2 \oplus x \oplus 1$	$x^{16} \oplus x^{13} \oplus x^{11} \oplus x^7 \oplus x^6 \oplus x^2 \oplus 1$
$x^5 \oplus x^4 \oplus x^3 \oplus x \oplus 1$	$x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$	$x^{16} \oplus x^{13} \oplus x^{10} \oplus x^9 \oplus x^8 \oplus x^4 \oplus 1$
$x^5 \oplus x^4 \oplus x^2 \oplus x \oplus 1$		$x^{16} \oplus x^{15} \oplus x^{10} \oplus x^5 \oplus x^4 \oplus x \oplus 1$
$x^5 \oplus x^3 \oplus x^2 \oplus x \oplus 1$		$x^{16} \oplus x^{15} \oplus x^{10} \oplus x^8 \oplus x^5 \oplus x^3 \oplus 1$

Библиотека БГУИР

Окончание табл.3.3

$n = 6$	$n = 9$	$n > 16$
$x^6 \oplus x^5 \oplus 1$	$x^9 \oplus x^5 \oplus 1$	$x^{17} \oplus x^3 \oplus 1$
$x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$	$x^9 \oplus x^8 \oplus x^6 \oplus x^5 \oplus 1$	$x^{18} \oplus x^7 \oplus 1$
$x^6 \oplus x^5 \oplus x^3 \oplus x^2 \oplus 1$	$x^6 \oplus x^7 \oplus x^6 \oplus x^4 \oplus 1$	$x^{19} \oplus x^6 \oplus x^5 \oplus x \oplus 1$
$x^6 \oplus x \oplus 1$		$x^{20} \oplus x^3 \oplus 1$
$x^6 \oplus x^5 \oplus x^4 \oplus x \oplus 1$	$n = 10$	$x^{21} \oplus x^2 \oplus 1$
$x^6 \oplus x^4 \oplus x^3 \oplus x \oplus 1$	$x^{10} \oplus x^7 \oplus 1$	$x^{22} \oplus x \oplus 1$
$x^6 \oplus x^5 \oplus x^2 \oplus x \oplus 1$	$x^{10} \oplus x^9 \oplus x^7 \oplus x^6 \oplus 1$	$x^{23} \oplus x^5 \oplus 1$
$x^6 \oplus x^4 \oplus x^2 \oplus x \oplus 1$	$x^{10} \oplus x^9 \oplus x^8 \oplus x^5 \oplus 1$	$x^{25} \oplus x^3 \oplus 1$
	$x^{10} \oplus x^9 \oplus x^8 \oplus x^7 \oplus x^5 \oplus x^4 \oplus 1$	$x^{28} \oplus x^3 \oplus 1$
	$x^{10} \oplus x^9 \oplus x^7 \oplus x^3 \oplus 1$	$x^{29} \oplus x^2 \oplus 1$
	$x^{10} \oplus x^8 \oplus x^4 \oplus x^3 \oplus 1$	$x^{31} \oplus x^3 \oplus 1$
	$x^{10} \oplus x^8 \oplus x^7 \oplus x^2 \oplus 1$	$x^{32} \oplus x^{28} \oplus x^{27} \oplus x \oplus 1$
	$x^{10} \oplus x^9 \oplus x^8 \oplus x^5 \oplus x^4 \oplus x^3 \oplus 1$	$x^{33} \oplus x^{13} \oplus 1$
	$x^{10} \oplus x^9 \oplus x^5 \oplus x^2 \oplus 1$	$x^{35} \oplus x^2 \oplus 1$
		$x^{36} \oplus x^{11} \oplus 1$
$n = 7$	$n = 11$	$x^{39} \oplus x^4 \oplus 1$
$x^7 \oplus x \oplus 1$	$x^{11} \oplus x^9 \oplus 1$	$x^{89} \oplus x^{38} \oplus 1$
$x^7 \oplus x^5 \oplus x^3 \oplus x \oplus 1$	$x^{11} \oplus x^9 \oplus x^8 \oplus x^6 \oplus 1$	$x^{127} \oplus x^7 \oplus 1$
$x^7 \oplus x^5 \oplus x^4 \oplus x^3 \oplus 1$	$x^{11} \oplus x^9 \oplus x^8 \oplus x^4 \oplus 1$	$x^{50} \oplus x^{49} \oplus x^{19} \oplus x^{18} \oplus 1$
$x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus x \oplus 1$	$x^{11} \oplus x^{10} \oplus x^8 \oplus x^6 \oplus 1$	$x^{60} \oplus x^{57} \oplus x^4 \oplus x \oplus 1$
$x^7 \oplus x^6 \oplus x^5 \oplus x^2 \oplus 1$	$x^{11} \oplus x^{10} \oplus x^9 \oplus x^5 \oplus 1$	$x^{71} \oplus x^{70} \oplus x^7 \oplus x^6 \oplus 1$
$x^7 \oplus x^4 \oplus 1$	$x^{11} \oplus x^{10} \oplus x^6 \oplus x^5 \oplus 1$	$x^{90} \oplus x^{89} \oplus x^{72} \oplus x^{71} \oplus 1$
$x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$	$x^{11} \oplus x^{10} \oplus x^9 \oplus x^2 \oplus 1$	$x^{100} \oplus x^{99} \oplus x^{19} \oplus x^{18} \oplus 1$
$x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$	$x^{11} \oplus x^9 \oplus x^5 \oplus x^3 \oplus 1$	$x^{100} \oplus x^{37} \oplus 1$
$x^7 \oplus x^6 \oplus x^3 \oplus x \oplus 1$	$n = 12$	$x^{1279} \oplus x^{216} \oplus 1$
$x^7 \oplus x^6 \oplus 1$	$x^{12} \oplus x^7 \oplus x^4 \oplus x^3 \oplus 1$	$x^{3217} \oplus x^{67} \oplus 1$
$x^7 \oplus x^6 \oplus x^4 \oplus x^2 \oplus 1$	$x^{12} \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$	$x^{9689} \oplus x^{84} \oplus 1$
$x^7 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$	$x^{12} \oplus x^7 \oplus x^4 \oplus x^3 \oplus 1$	
$x^7 \oplus x^6 \oplus x^5 \oplus x^3 \oplus x^2 \oplus x \oplus 1$	$x^{12} \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$	
$x^7 \oplus x^5 \oplus x^2 \oplus x \oplus 1$		
$x^7 \oplus x^3 \oplus 1$		
$x^7 \oplus x^3 \oplus x^2 \oplus x \oplus 1$		
$x^7 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus x \oplus 1$		
$x^7 \oplus x^6 \oplus x^4 \oplus x \oplus 1$		

Библиотека БГУИР

4. СИГНАТУРНЫЙ АНАЛИЗ

Для описания процедуры сжатия информации используются различные математические модели. Одной из наиболее широко распространенных является модель, которая представляет процедуру сжатия как операцию деления полиномов над полем GF(2).

Пусть с выхода тестируемой схемы поступает последовательность 11110101. В полиномиальной форме она может быть представлена как $h(x) = x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$. Этот полином используется как делимое.

Делителем служит полином сигнатурного анализатора (СА) $\varphi(x) = x^3 \oplus x^2 \oplus 1$.

В результате деления получаем частные $q(x)$ и остаток $s(x)$, т. е. $h(x) = q(x) \cdot \varphi(x) \oplus s(x)$.

Рассмотрим пример деления полиномов в GF(2), который повторяет процедуру сжатия на сигнатурном анализаторе.

На рис. 4.1 приведен пример деления полиномов, который описывает процедуру сжатия данных при помощи сигнатурного анализатора с внутренними сумматорами по модулю два. На разрядах СА формируется двоичный код 110, который соответствует коэффициентам полинома остатка $s(x)$, где $s(x) = 1 \cdot x^2 \oplus 1 \cdot x^1 \oplus 0 \cdot x^0 = x^2 \oplus x$. Для практической реализации сигнатурного анализатора наиболее часто используются структуры с внешними сумматорами по модулю два, пример такого анализатора для полинома $\varphi(x) = x^3 \oplus x^2 \oplus 1$ приведен на рис. 4.2.

$$
\begin{array}{l|l}
\overbrace{x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1}^{h(x)} & \overbrace{x^3 \oplus x^2 \oplus 1}^{\varphi(x)} \\
\underline{x^7 \oplus x^6 \oplus x^4} & \underbrace{x^4 \oplus x^2 \oplus x \oplus 1}_{q(x)} \\
x^5 \oplus x^2 \oplus 1 & \\
\underline{x^5 \oplus x^4 \oplus x^2} & \\
x^4 \oplus 1 & \\
\underline{x^4 \oplus x^3 \oplus x} & \\
x^3 \oplus x \oplus 1 & \\
\underline{x^3 \oplus x^2 \oplus 1} & \\
\underbrace{x^2 \oplus x}_{s(x)} &
\end{array}
$$

Рис. 4.1. Схема деления полиномов

Библиотека БГУИР

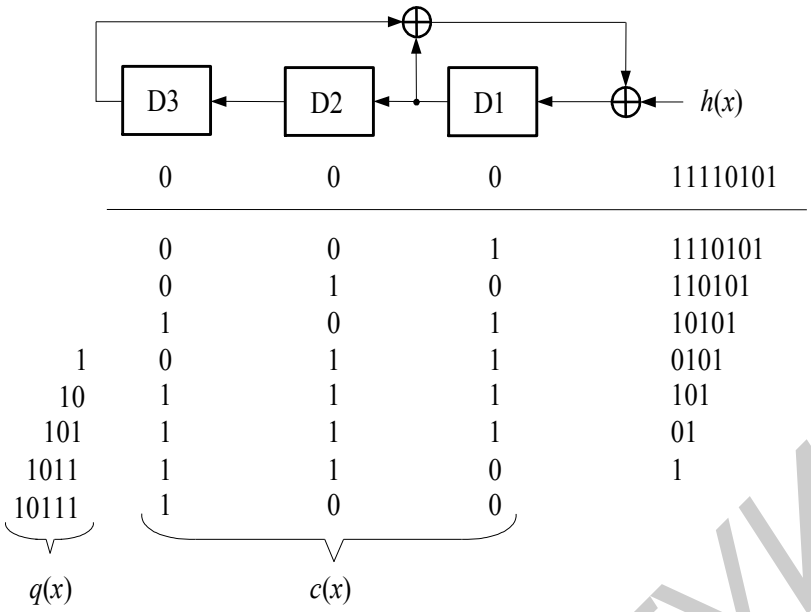


Рис. 4.2. Сигнатурный анализатор
с внешними сумматорами по модулю два

Сигнатура $c(x)$, полученная для случая анализатора с внешними сумматорами по модулю два, связана линейным соотношением с сигнатурой $s(x)$ таким образом, что между значениями данных сигнатур существует однозначное соответствие.

В общем случае полнота обнаружения неисправной схемы зависит от качества тестовых воздействий, т. е. если неисправность не активизирована тестом и не проявляется в выходных последовательностях, то она не будет обнаружена с помощью СА. Поэтому под достоверностью сигнатурного анализа понимают его эффективность обнаружения ошибок в потоке сжимаемых выходных данных. Для ее оценки используется вероятностный подход.

В качестве меры эффективности применяется вероятность необнаружения ошибок P_n в анализируемой последовательности данных. При этом предполагается, что эталонная последовательность данных может принимать равновероятно произвольное значение, а любая конфигурация ошибочных бит также равновероятна. Пусть эталонная последовательность состоит из l бит, тогда при сжатии на СА формируется $(l - m)$–разрядное частное $q(x)$ и m-разрядная сигнатура $s(x)$ (остаток), что следует из алгоритма деления, приведенного на рис. 4.1. При этом соответствие реальной последовательности $h(x)$* ее эталонной версии $h(x)$ оценивается только по равенству их m-разрядных сигнатур.

Для 2^{l-m} $(l–m)$-разрядных частных будет формироваться одинаковая сигнатура. Среди этих 2^{l-m} частных только одно частное соответствует эталонной последовательности, а $2^{l-m}-1$ – последовательностям данных содержащих ошибки. Таким образом, в среднем с учетом принятых допущений $2^{l-m}-1$ ошибочных последовательностей будут иметь такую же сигнатуру, что и эталонная последовательность. Общее число ошибочных последовательностей равняется

Библиотека БГУИР

$2^l - 1$. Тогда вероятность P_n необнаружения ошибок в анализируемой последовательности данных будет определяться как

$$P_n = \frac{2^{l-m} - 1}{2^l - 1}. \tag{4.1}$$

Для больших значений l с учетом неравенства $l >> m$ получим, что для ошибок, кратность которых больше трех, выполняется равенство

$$P_n \approx \frac{1}{2^m}. \tag{4.2}$$

Кроме того, сигнатурный анализ позволяет обнаружить все одиночные ошибки для $\forall l$ и двойные при выполнении неравенства $l \le 2^m - 1$.

Реальная тестируемая схема содержит не один, а много выходов. Поэтому при тестировании стоит проблема одновременного анализа большого количества тестовых реакций, получаемых на различных выходах тестируемой схемы. Одним из первых был предложен подход, основанный на мультиплексировании выходных последовательностей тестируемой схемы, приведенный на рис. 4.3.

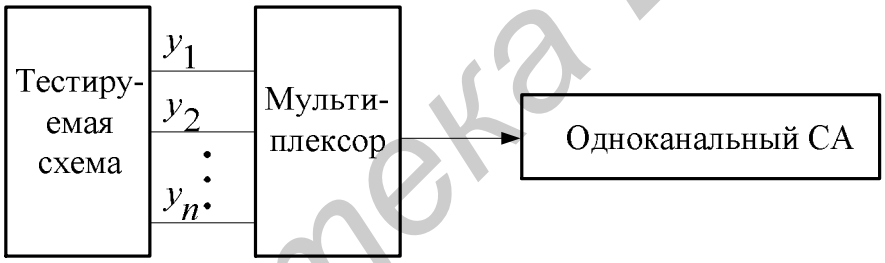


Рис. 4.3. Многоканальный сигнатурный анализатор на основе мультиплексора

Данный подход основан на применении одноканального сигнатурного анализатора (ОСА), что значительно увеличивает время тестирования.

Следующий подход предполагает двухступенчатое преобразование сжимаемых данных (рис. 4.4). Сначала выходная реакция сжимается в пространстве при помощи многовходовых сумматоров по модулю два, затем во времени при помощи ОСА.

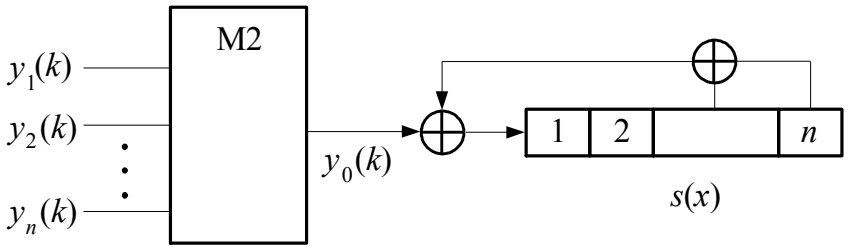


Рис. 4.4. Многоканальный сигнатурный анализатор на основе двухступенчатого сжатия выходных реакций

Библиотека БГУИР

Применение многоканальных сигнатурных анализаторов (МСА) является основным методом сжатия многоразрядных выходных реакций при тестировании цифровых устройств.

Рассмотрим пример проектирования МСА.

Функционирование одноканального СА описывается выражением

$$\begin{vmatrix} a_1(k+1) \\ a_2(k+1) \\ a_3(k+1) \\ \vdots \\ a_m(k+1) \end{vmatrix} = \begin{vmatrix} a_1 & a_2 & a_3 & \dots & a_{m-1} & a_m \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{vmatrix} \times \begin{vmatrix} a_1(k) \\ a_2(k) \\ a_3(k) \\ \vdots \\ a_m(k) \end{vmatrix} \oplus \begin{vmatrix} y(k) \\ 0 \\ 0 \\ \vdots \\ 0 \end{vmatrix}, \tag{4.3}$$

где $a_i(k)$ – состояние i-го элемента памяти в k-й момент времени, $i = \overline{1, m}$, $m = \deg \varphi(x)$.

В матричной форме получим, что

$$\begin{aligned} &A(k+1) = V \cdot A(k) \oplus Y(k), \\ &A(k+2) = V \cdot A(k+1) + Y(k+1) = V^2 A(k) \oplus V \cdot Y(k) \oplus Y(k+1), \\ &\vdots \\ &A(k+n) = V^n \cdot A(k) \oplus \sum_{i=1}^{n} V^{n-i} Y(k-1+i). \end{aligned} \tag{4.4}$$

Например, для $n = 2$ имеем

$$A(k+2) = V^2 A(k) \oplus V \cdot Y(k) + Y(k+1). \tag{4.5}$$

Пусть $\varphi(x) = x^4 \oplus x^3 \oplus 1$, $m = \deg \varphi(x) = 4$, тогда

$$V = \begin{vmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}, \quad V^2 = \begin{vmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{vmatrix}, \quad A(k) = \begin{vmatrix} a_1(k) \\ a_2(k) \\ a_3(k) \\ a_4(k) \end{vmatrix}, \quad Y(k) = \begin{vmatrix} y(k) \\ 0 \\ 0 \\ 0 \end{vmatrix},$$

$$Y(k+1) = \begin{vmatrix} y(k+1) \\ 0 \\ 0 \\ 0 \end{vmatrix}.$$

Библиотека БГУИР

Для $n = 2$ получим

$$A(k+2) = \begin{vmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{vmatrix} \cdot \begin{vmatrix} a_1(k) \\ a_2(k) \\ a_3(k) \\ a_4(k) \end{vmatrix} \oplus \begin{vmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \cdot \begin{vmatrix} y(k) \\ 0 \\ 0 \\ 0 \end{vmatrix} \oplus \begin{vmatrix} y(k+1) \\ 0 \\ 0 \\ 0 \end{vmatrix} = \tag{4.6}$$

$$= \begin{vmatrix} a_2(k) \oplus a_3(k) \\ a_3(k) \oplus a_4(k) \\ a_1(k) \\ a_2(k) \end{vmatrix} \oplus \begin{vmatrix} 0 \\ y(k) \\ 0 \\ 0 \end{vmatrix} \oplus \begin{vmatrix} y(k+1) \\ 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} a_2(k) \oplus a_3(k) \oplus y(k+1) \\ a_3(k) \oplus a_4(k) \oplus y(k) \\ a_1(k) \\ a_2(k) \end{vmatrix}.$$

Данное выражение описывает СА, который за один такт сжимает сразу два символа тестовых реакции – $y(k)$ и $y(k + 1)$. Заменим $y(k + 1)$ на другую последовательность $z(k)$ и запишем уравнения состояний для системы элементов памяти двухканального МСА, сжимающего две последовательности $y(k)$ и $z(k)$:

$$\begin{aligned} a_1(k+1) &= a_2(k) \oplus a_3(k) \oplus z(k); \\ a_2(k+1) &= a_3(k) \oplus a_4(k) \oplus y(k); \\ a_3(k+1) &= a_1(k); \\ a_4(k+1) &= a_2(k). \end{aligned} \tag{4.7}$$

Функциональная схема двухканального сигнатурного анализатора, построенная на основании (4.7), приведена на рис. 4.5.

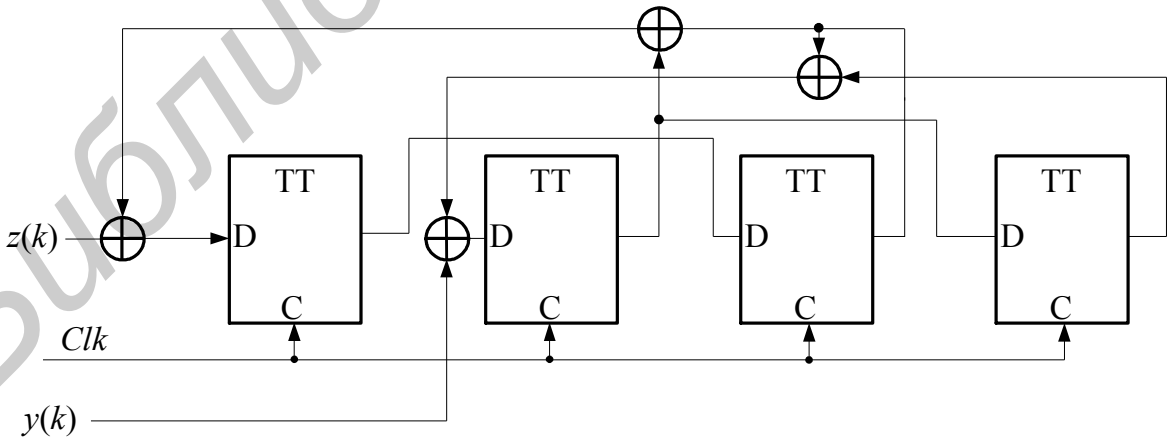


Рис. 4.5. Двухканальный сигнатурный анализатор

Рассмотрим пример синтеза четырехканального ($n = 4$) МСА для порождающего полинома $\varphi(x) = x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$.

Библиотека БГУИР

1. Первоначально строится матрица V:

$$V = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}.$$

2. С помощью V находится матрица V^4:

$$V^4 = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{vmatrix}.$$

3. На основании V^4 строится функциональная схема МСА для $n = 4$ (рис. 4.6).

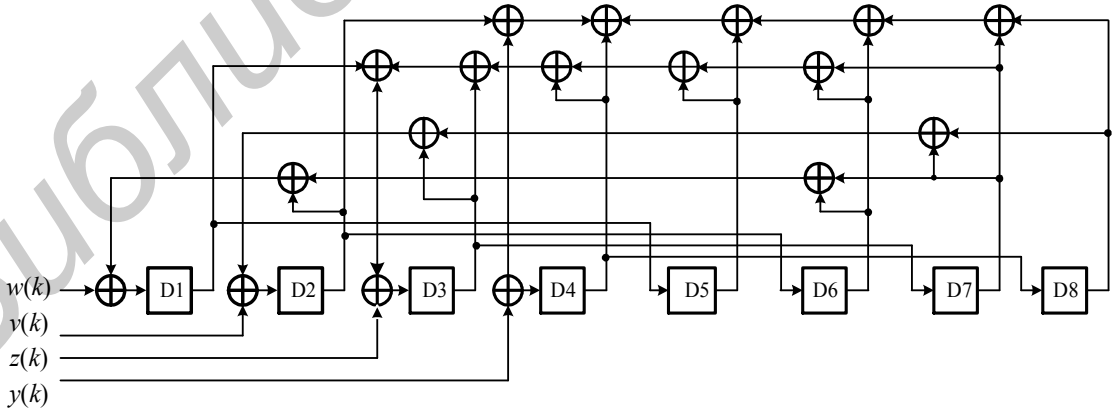


Рис. 4.6. Четырехканальный сигнатурный анализатор

Анализ приведенной схемы показывает возможность ее оптимизации с целью уменьшения количества повторяющихся двухвходовых сумматоров по модулю два.

Библиотека БГУИР

4.1. Задание для выполнения лабораторной работы

Задание для лабораторной работы №5

1. Разработать программу, которая моделирует процесс компактного тестирования с применением генератора тестовых воздействий и сигнатурного анализатора по схеме «генератора тестовых воздействий – тестируемая схема – сигнатурный анализатор» (рис. 4.7).

В качестве генератора тестовых воздействий использовать генератор псевдослучайных тестовых наборов на основе LFSR, порождающий полином, для которого задается по результатам лабораторной работы №3 (см. табл. 3.3). В качестве тестируемой схемы необходимо использовать вариант схемы из лабораторной работы №1. Моделирование работы осуществляется до достижения исчерпывающего покрытия всевозможных одиночных константных неисправностей, при этом количество тактов моделирования должно быть не менее 127.

2. Подобрать порождающий полином сигнатурного анализатора с внешними сумматорами по модулю два (см. рис. 4.2) для степеней m = 7, 8 и 9 таким образом, чтобы аппаратные затраты на реализацию были минимальны, при этом необходимо обеспечить 100 %-ное обнаружение одиночных константных неисправностей.

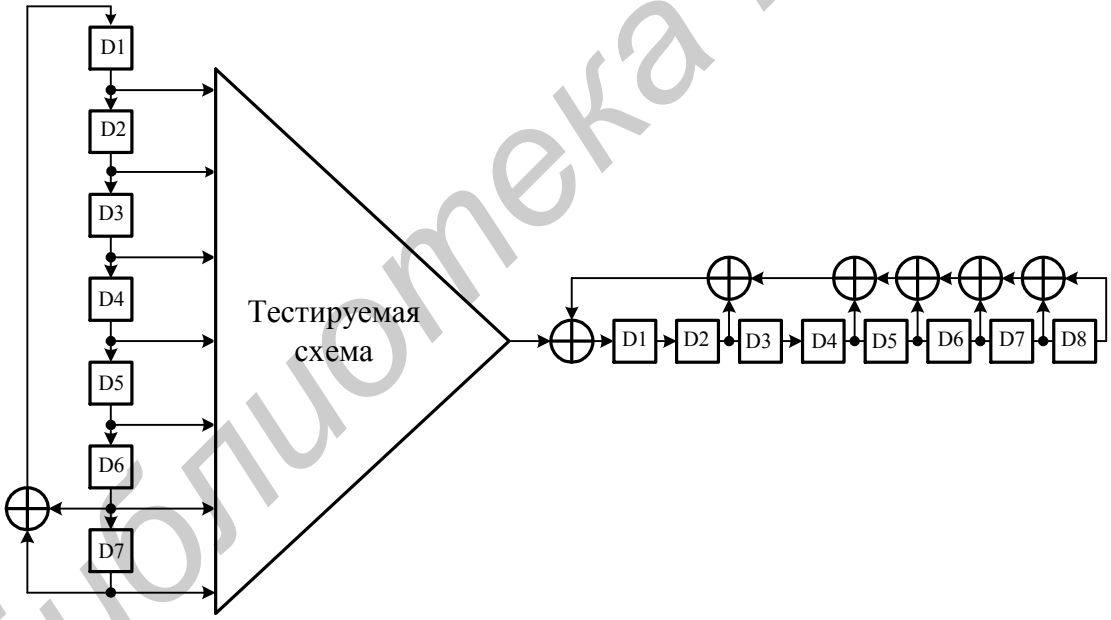


Рис. 4.7. Схема проведения тестового эксперимента

Для этого следует уменьшить число сумматоров по модулю два в цепи обратной связи анализатора и путем перебора порождающих полиномов той же степени проверить условие 100 %-ного обнаружения одиночных константных неисправностей. Так, для m = 8 начальный полином сигнатурного анализатора – $\varphi(x) = x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$, а конечный полином $\varphi(x) = x^3 \oplus x \oplus 1$.

Библиотека БГУИР

Результаты работы программы оформить в виде таблицы, в которой для каждого полинома привести аппаратурную сложность в виде количества двухвходовых сумматоров по модулю два и процентную полноту покрытия.

Задание для лабораторной работы №6

1. Разработать программу, которая моделирует работу одноканального сигнатурного анализатора (см. рис. 4.2), работа которого описывается полиномом восьмой степени в соответствие с заданным вариантом (табл. 4.1). Произвольным образом сформировать последовательность сжимаемых данных из l символов. Исследовать обнаруживающую способность ОСА с точки зрения обнаружения однократных, двукратных, трехкратных и четырехкратных ошибок в сжимаемой последовательности. Вывести все комбинации необнаруживаемых ошибок.

Таблица 4.1

№ по разрядам	Полином	l
1	$x^8 \oplus x^6 \oplus x^5 \oplus x \oplus 1$	40
2	$x^8 \oplus x^5 \oplus x^3 \oplus x \oplus 1$	50
3	$x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$	48
4	$x^8 \oplus x^5 \oplus x^3 \oplus x^2 \oplus 1$	40
5	$x^8 \oplus x^6 \oplus x^3 \oplus x^2 \oplus 1$	52
6	$x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$	50
7	$x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$	44
8	$x^8 \oplus x^6 \oplus x^5 \oplus x \oplus 1$	56
9	$x^8 \oplus x^5 \oplus x^3 \oplus x \oplus 1$	44
10	$x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$	52
11	$x^8 \oplus x^5 \oplus x^3 \oplus x^2 \oplus 1$	56
12	$x^8 \oplus x^6 \oplus x^3 \oplus x^2 \oplus 1$	40
13	$x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$	44
14	$x^8 \oplus x^7 \oplus x^6 \oplus x^3 \oplus x^2 \oplus x \oplus 1$	58
15	$x^8 \oplus x^7 \oplus x^6 \oplus x^3 \oplus x^2 \oplus x \oplus 1$	60
16	$x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$	40

2. Для заданного варианта порождающего полинома синтезировать двухканальный сигнатурный анализатор. Оценить его эффективность обнаружения ошибок и сравнить с результатами, полученными при выполнении п. 1.

Библиотека БГУИР

5. ТЕСТИРОВАНИЕ ОПЕРАТИВНЫХ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

5.1. Классификация запоминающих устройств

В настоящее время существует много разнообразных устройств, которые позволяют хранить информацию и выполнять с ней такие операции, как чтение и запись. Это такие устройства, как регистры, оперативная память, кэш-память, дисковые накопители, флэш-память, постоянные запоминающие устройства, магнитные карточки и другие разновидности памяти. Далее будем рассматривать только полупроводниковую память, представленную в виде отдельной микросхемы (кристалла) или являющуюся частью СБИС, так называемую встроенную память (*embedded memory*).

К основным характеристикам памяти можно отнести:

1) информационную емкость – максимально возможный объем хранимой информации;

2) организацию – число хранимых слов и их разрядность, иногда указывают количество банков памяти;

3) число портов ввода-вывода данных (одно-, двух- или многопортовая память) и их реализация (двунаправленные или раздельные линии на ввод и вывод данных);

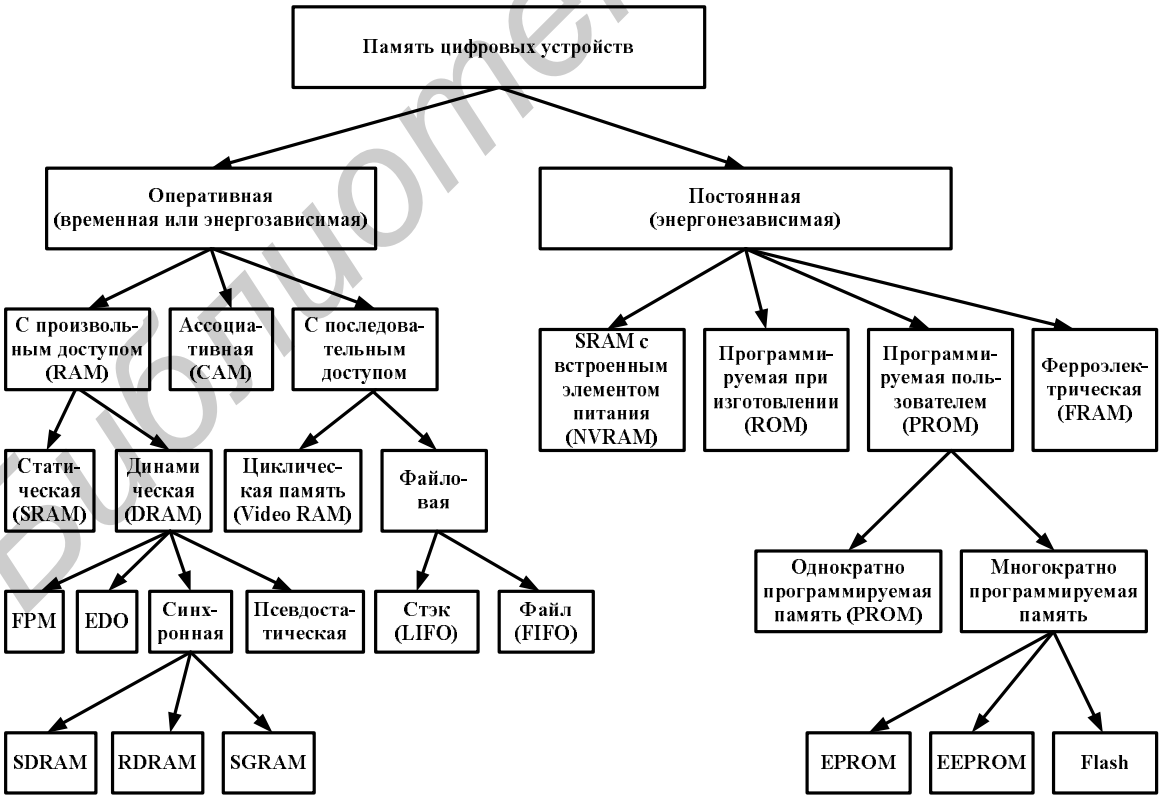


Рис. 5.1. Классификация полупроводниковой памяти

Библиотека БГУИР

4) быстродействие – оценивается временем чтения, записи или длительностью цикла чтения-записи.

На рис. 5.1 приведена классификация полупроводниковой памяти, используемой в современных компьютерах. В данной работе исследуются вопросы тестирования оперативных запоминающих устройств (ОЗУ).

5.2. Классические модели неисправностей запоминающих устройств

На рис. 5.2 представлена общая функциональная модель динамического ОЗУ. Для статического ОЗУ схема будет аналогичной, только в ней будет отсутствовать блок управления регенерацией.

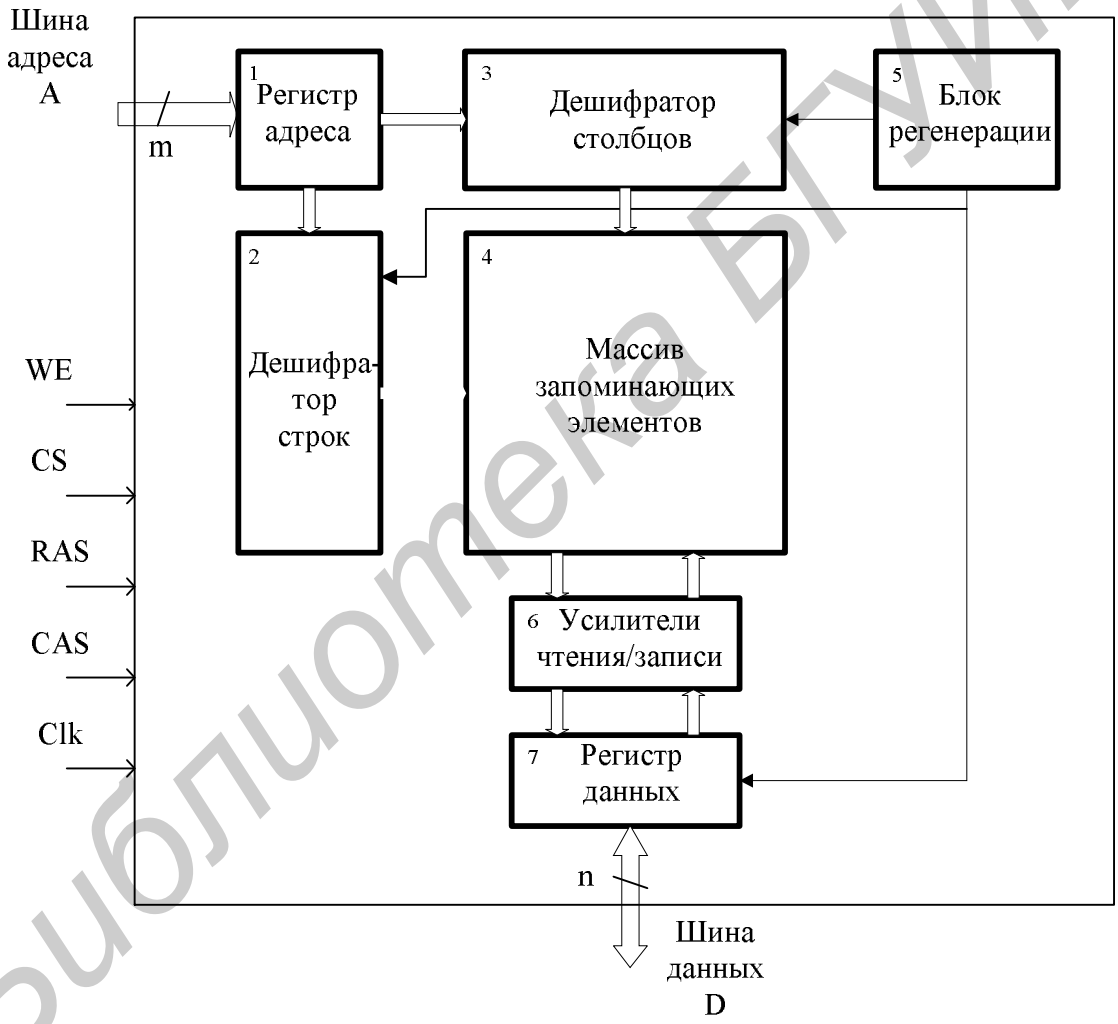


Рис. 5.2. Функциональная модель ОЗУ

В соответствии с функциональной моделью ОЗУ, которая приведена на рис. 5.2, возможные неисправности микросхем ЗУ можно представить двумя основными группами:

1) неисправности схем обрамления;

2) неисправности массива запоминающих элементов памяти.

Рассмотрим более подробно данную классификацию неисправностей.

Библиотека БГУИР

5.2.1. Неисправности схем обрамления

К неисправностям схем обрамления относят следующие модели неисправностей

1. Модели адресных неисправности (неисправности дешифраторов адреса). Традиционно адресные неисправности (*Address Decoder Faults – AF*) включают в себя следующие модели:

– ни одна из всех ячеек памяти недоступна по заданному адресу;

– заданная ячейка памяти недоступна;

– по заданному адресу памяти осуществляется доступ сразу к нескольким ячейкам;

– доступ к заданной ячейке памяти осуществляется по нескольким адресам.

2. Модели неисправностей логики чтения/записи.

3. Модель неисправности типа обрыв (*Stuck Open Fault – SOF*).

4. Модель неисправности «разрушающая операция чтения» (*Read Disturb Fault Model – RDF*).

5. Модель неисправности «слабый запоминающий элемент» (*Weak Cell Fault Model – WCF*).

6. Модель неисправности «ошибочная запись» (*False Write Through Fault Model – FWTF*).

Все приведенные неисправности в электронном обрамлении запоминающего устройства могут быть обнаружены классическими тестами, которые выявляют неисправности матрицы запоминающих элементов, либо путем внесения временных задержек в данные тесты памяти.

5.2.2. Неисправности массива ячеек запоминающих устройств

Все множество неисправностей матрицы запоминающих ячеек ЗУ делится на подмножества в зависимости от количества ячеек, участвующих в описании конкретной неисправности. Выделяют неисправности, в которых участвуют:

1) одна ячейка ЗУ;

2) две ячейки ЗУ;

3) более двух ячеек ЗУ (кодочувствительные неисправности).

Данная классификация характерна как для статических, так и для динамических ОЗУ.

К неисправностям, затрагивающим одну ячейку ЗУ, относят:

1. *Константные неисправности* (*Stuck-At Faults – SAF*). Данные неисправности характеризуются тем, что состояние конкретной ячейки ЗУ (неисправной) постоянно находится в одном из возможных состояний: нуля (0) или единицы (1) независимо от выполненных ранее операций записи в данную ячейку противоположного значения. Соответственно различают неисправность

Библиотека БГУИР

константного нуля (*Stuck-At 0 – SAF*) и неисправность константной единицы (*Stuck-At 1 – SAF1*). Также выделяют однократные и многократные константные неисправности.

2. *Переходные неисправности* (*Transition Faults – TF*). Подобные неисправности характеризуются невозможностью логического перехода состояния неисправной ячейки ЗУ из 0 в 1 (*transition up*) $\langle\uparrow\rangle$ или из 1 в 0 (*transition down*) $\langle\downarrow\rangle$ при выполнении соответствующих операций записи. Данный тип неисправности достаточно близок по своей сути к константным неисправностям. Действительно, если ячейка, имеющая переходную неисправность, оказывается в состоянии, из которого она не может перейти в другое состояние, ее поведение повторяет поведение ячейки, содержащей константную неисправность. Это обстоятельство делает возможным использование приемов, позволяющих обнаруживать одновременно оба типа неисправностей.

Среди неисправностей, в которых участвуют две ячейки ЗУ, выделяют:

1. *Инверсные неисправности взаимного влияния* (*Inversion Coupling Faults – CFin*). В данной неисправности участвует две ячейки ЗУ: a_i и a_j , где $i \neq j$, одна из которых a_i называется агрессором (*aggressor*), а вторая a_j – жертвой (*victim*). Расположение агрессора и жертвы в адресном пространстве ЗУ произвольно. При наличии данной неисправности логический переход из 1 в 0 или из 0 в 1 в агрессоре a_i приводит к инверсии логического значения в жертве a_j. Таким образом, различают два вида инверсных неисправностей: $\langle\uparrow, a_j\rangle$ и $\langle\downarrow, a_j\rangle$. Для представления соотношения адресов агрессора и жертвы используют символы $\wedge$ и $\vee$, причем символ $\wedge$ означает факт того, что адрес агрессора меньше адреса жертвы ($i < j$), символ $\vee$ обозначает, что адрес агрессора наоборот больше адреса жертвы ($i < j$). Тогда имеем четыре различных инверсных неисправности $\wedge\langle\uparrow, a_j\rangle$, $\wedge\langle\downarrow, a_j\rangle$, $\vee\langle\uparrow, a_j\rangle$ и $\vee\langle\downarrow, a_j\rangle$.

2. *Неисправности прямого действия* (*Idempotent Coupling Faults – CFid*). При данной неисправности во время логического перехода из 1 в 0 или из 0 в 1 во влияющей a_i (агрессоре) ячейке происходит принудительная установка определенного логического значения 0 или 1 в ячейке жертве a_j, на которую оказывается влияние агрессора. Различают восемь неисправностей прямого действия $\wedge\langle\uparrow,0\rangle$, $\wedge\langle\uparrow,1\rangle$, $\wedge\langle\downarrow,0\rangle$, $\wedge\langle\downarrow,1\rangle$, $\vee\langle\uparrow,0\rangle$, $\vee\langle\uparrow,1\rangle$, $\vee\langle\downarrow,0\rangle$ и $\vee\langle\downarrow,1\rangle$. При анализе эффективности тестов ЗУ анализируется их покрывающая способность для всех 12 неисправностей, в которых участвуют две ячейки (*2-coupling faults*).

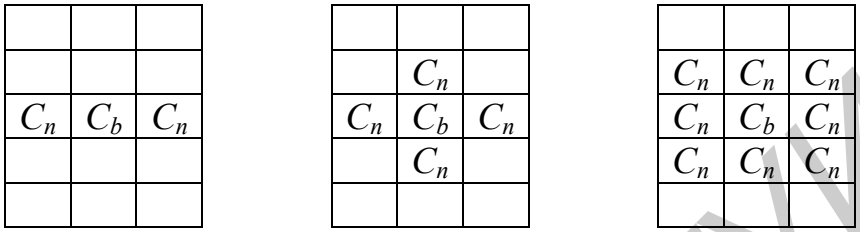
3. *Статические неисправности взаимного влияния* (*State Coupling Faults – CFst*). Переход в ячейке жертве a_j в другое состояние b_j возможен при определенном значении b_i влияющей a_i (агрессоре) ячейке. Возможно восемь неисправностей CFst: $\wedge(0,0)$, $\wedge(0,1)$, $\wedge(1,0)$, $\wedge(1,1)$, $\vee(0,0)$, $\vee(0,1)$, $\vee(1,0)$ и $\vee(1,1)$.

Неисправности, затрагивающие несколько ячеек ЗУ, называются *кодочувствительными неисправностями* (*Pattern Sensitive Faults – PSF*). Для подобных неисправностей логическое состояние одной ячейки ЗУ, которую называют *базовой ячейкой* (*base cell*), зависит от содержимого (0 или 1) или от логических переходов из 1 в 0 или из 0 в 1 *соседних ячейках* (*neighborhood cells*) ЗУ. Разли-

Библиотека БГУИР

чают два вида кодочувствительных неисправностей: неограниченные (*unrestricted*) и ограниченные (*restricted*) (или граничные (*neighborhood*). При тестировании массива ячеек ЗУ обычно придерживаются последней более реальной модели кодочувствительных неисправностей.

На рис. 5.3 представлено схематическое изображение моделей граничной кодочувствительной неисправности, в которых участвуют 3 (рис. 5.3, а), 5 (см. рис. 5.3, б) и 9 (см. рис. 5.3, в) ячеек, из которых одна (C_b) является базовой, остальные (C_n) – соседними.



*NPSF*3 *NPSF*5 *NPSF*9

а б в

Рис. 5.3. Модели кодочувствительной неисправности

В зависимости от эффекта влияния на базовую ячейку различают несколько видов кодочувствительных неисправностей:

1. *Пассивные кодочувствительные неисправности* (*Passive NPSF – PNPSF*), при которых состояние базовой ячейки не может быть изменено для определенного кода в $k - 1$ соседних ячейках ЗУ.

2. *Активные кодочувствительные неисправности* (*Active NPSF – ANPSF*), в которых базовая ячейка изменяет свое состояние из-за изменения кода в соседних. Изменение кода для подобных неисправностей происходит в результате изменения состояния на противоположное только в одном соседнем элементе, в то время как остальные ячейки сохраняют предыдущее состояние.

3. *Статические кодочувствительные неисправности* (*Static NPSF – SNPSF*) характеризуются тем, что для определенной комбинации значений в соседних ячейках состояние базовой принудительно устанавливается в состояние 0 или состояние 1. Главным отличием статических неисправностей от активных кодочувствительных неисправностей является длительность процесса установления неверного значения в базовой ячейке. Для статических неисправностей это время существенно больше.

5.3. Разрушающие тесты оперативных запоминающих устройств

К данному моменту времени разработано большое количество разнообразных функциональных тестов ОЗУ классов N, N^2 и $N^{3/2}$, соответствующих зависимости числа циклов в последовательности тестирования от емкости N запоминающего устройства. Ниже приведены примеры наиболее распространенных тестов.

Библиотека БГУИР

Memory scan, или MSCAN. Это наиболее простой тест, который состоит из следующей последовательности действий. Во все ячейки памяти записываются 0, значение ячеек проверяется на соответствие 0. Потом во все ячейки памяти записывается 1, значение ячеек проверяется на равенство 1. Процедура проверки памяти является очень быстрой, однако имеет низкую покрывающую способность. Данный тест имеет сложность $4N$ и гарантирует 100 %-ное обнаружение только *SAF*.

GALPAT и *Walking* 0/1. Эти два теста состоят из похожего набора операций. Сначала тест записывает 0 или 1 во все ячейки, исключая базовую ячейку, которая содержит 1 или 0 соответственно. В течение теста базовая ячейка последовательно перемещается через всю память. Различие между GALPAT и *Walking* 0/1 состоит в способе чтения значения базовой ячейки. Walking 0/1 после каждого шага выполнения читает значения всех ячеек памяти и только после этого читает базовую ячейку. GALPAT также читает значения всех ячеек, однако чтение базовой ячейки производится после чтения каждой ячейки памяти. Данные тесты имеет сложность $O(N^2)$. Например, GALPAT имеет сложность $2N^2 + 6N$, при этом гарантирует 100 %-ное обнаружение *SAF*, *AF*, *TF*. Для более сложных неисправностей (*CF* и *PSF*) он не дает гарантии 100 %-ного обнаружения.

В настоящее время объем оперативной памяти в персональных компьютерах составляет более 1 Гб . Поэтому применение тестов со сложностью N^2 и $N^{3/2}$ неэффективно вследствие большого времени тестирования. Рассмотрим пример проверки памяти объемом 256 Мб, которая работает на частоте 100 МГц (цикл обращения равен 10 нс). Запись в каждую ячейку одного фиксированного значения (0 или 1) и чтение этого значения займет 5368709120 нс ≈ ≈ 5,37 с (2^{28} = 268435456). Тест сложности $N^{3/2}$ будет выполняться $2^{42}\cdot 10$ нс = = 43 980 с ≈ 733 мин или 12,2 ч. Соответственно нереальным представляется применение таких тестов.

Поэтому широкое распространение нашли только тесты сложности $O(N)$, получившие название *маршевых тестов* (табл. 5.1). Эти тесты обладают достаточной для большинства приложений покрывающей способностью и простотой реализации, что очень важно при проектировании встроенных систем самотестирования ОЗУ.

Любой маршевый тест состоит из конечного числа маршевых элементов. Маршевый элемент представляет собой конечную последовательность операций, применяемых к каждой ячейке памяти перед переходом к тестированию следующей. Направление перехода может осуществляться в одном из двух направлений: в направлении увеличения адреса (обозначается символом ⇑), т. е. адреса ячеек изменяются от 0 до $N - 1$, либо в направлении уменьшения адреса (обозначается символом ⇓), т. е. адреса ячеек изменяются в обратном порядке. Когда направление изменения адреса не имеет значения, это указывается символом ⇕ (при реализации конкретного маршевого теста необходимо выбрать какое-то определенное направление). Увеличение адресов необязательно озна-

Библиотека БГУИР

чает изменение от 0 до $N-1$. В общем случае это может быть произвольная последовательность адресов, например, 5, 7, 2, 0, 3, 4, 1, 6. Необходимым условием является то, чтобы направления, обозначаемые символами ⇑ и ⇓, были бы противоположны друг другу. Подобные последовательности адресов (в которых не соблюдается принцип постоянного увеличения или уменьшения адреса) используются для обнаружения неисправностей в адресной логике систем памяти. Набор операций, применяемых в рамках маршевого элемента к каждой ячейке памяти, содержит следующие операции:

– запись значения логического 0 в ячейку памяти (обозначается $w0$);

– запись значения логической 1 в ячейку памяти (обозначается $w1$);

– считывание хранимого значения из ячейки памяти и сравнение его с логическим нулем (обозначается символом $r0$);

– считывание хранимого значения из ячейки памяти и сравнение его с логической единицей (обозначается символом $r1$).

Таблица 5. 1

№	Тест	Алгоритм	N
1	2	3	4
1	MATS	⇕($w0$); ⇑($r0$,$w1$); ⇕($r1$);	$4N$
2	MATS+	⇕($w0$); ⇑($r0$,$w1$); ⇓($r1$,$w0$);	$5N$
3	MATS++	⇕($w0$); ⇑($r0$,$w1$); ⇓($r1$,$w0$,r0);	$6N$
4	Marching 1/0	⇕($w0$); ⇑($r0$,$w1$,$r1$); ⇓($r1$,$w0$,$r0$); ⇑($w1$); ⇑($r1$,$w0$,$r0$); ⇓($r0$,$w1$,$r1$);	$14N$
5	March X	⇕($w0$); ⇑($r0$,$w1$); ⇓($r1$,$w0$); ⇕($r0$);	$6N$
6	March Y	⇕($w0$); ⇑($r0$,$w1$,$r1$); ⇓($r1$,$w0$,$r0$); ⇕($r0$);	$8N$
7	March C	⇕($w0$); ⇑($r0$,$w1$); ⇑($r1$,$w0$); ⇕($r0$); ⇓($r0$,$w1$); ⇓($r1$,$w0$); ⇕($r0$)	$11N$
8	March C-	⇕($w0$); ⇑($r0$,$w1$); ⇑($r1$,$w0$); ⇓($r0$,$w1$); ⇓($r1$,$w0$); ⇕($r0$);	$10N$
9	March A	⇕($w0$); ⇑($r0$,$w1$,$w0$,$w1$); ⇑($r1$,$w0$,$w1$); ⇓($r1$,$w0$,$w1$,$w0$); ⇓($r0$,$w1$,$w0$);	$15N$
10	March B	⇕($w0$); ⇑($r0$,$w1$,$r1$,$w0$,$r0$,$w1$); ⇑($r1$,$w0$,$w1$); ⇓(r1,w0,w1,w0);⇓(r0,w1,w0);	$17N$
11	Algorithm B	⇕($w0$); ⇑($r0$,$w1$,$w0$,$w1$); ⇑($r1$,$w0$,$r0$,$w1$); ⇓($r1$,$w0$,$w1$,$w0$); ⇓($r0$,$w1$,$r1$,$w0$);	$17N$
13	March C-R	⇕($w0$); ⇑($r0$,$r0$,$w1$); ⇑($r1$,$r1$,$w0$); ⇓($r0$,$r0$,$w1$); ⇓($r1$,$r1$,$w0$); ⇕ ($r0$,$r0$);	$15N$

Библиотека БГУИР

Окончание табл. 5. 1

1	2	3	4
14	PMOVI	⇳(*w*0); ⇑(*r*0,*w*1,*r*1); ⇑(*r*1,*w*0,*r*0); ⇓(*r*0,*w*1,*r*1); ⇓(*r*1,*w*0,*r*0);	13 *N*
15	PMOVI-R	⇳(*w*0); ⇑(*r*0,*w*1,*r*1,*r*1); ⇑(*r*1,*w*0,*r*0,*r*0); ⇓(*r*0,*w*1, *r*1,*r*1); ⇓ (*r*1,*w*0,*r*0,*r*0);	17 *N*
17	March U	⇳(*w*0); ⇑(*r*0,*w*1,*r*1,*w*0); ⇑(*r*0,*w*1); ⇓(*r*1,*w*0,*r*0,*w*1); ⇓(*r*1,*w*0);	13 *N*
19	March U-R	⇳(*w*0); ⇑(*r*0,*w*1,*r*1,*r*1,*w*0); ⇑(*r*0,*w*1); ⇓(*r*1,*w*0,*r*0, *r*0,*w*1); ⇓(*r*1,*w*0);	15 *N*
20	March LR	⇳(*w*0); ⇓(*r*0,*w*1); ⇑(*r*1,*w*0,*r*0,*w*1); ⇑(*r*1,*w*0); ⇑ (*r*0,*w*1,*r*1,*w*0); ⇓(*r*0);	14 *N*
21	March LA	⇳(*w*0); ⇑(*r*0,*w*1,*w*0,*w*1,*r*1); ⇑(*r*1,*w*0,*w*1,*w*0,*r*0); ⇓(*r*0,*w*1,*w*0,*w*1,*r*1); ⇓(*r*1,*w*0,*w*1,*w*0,*r*0); ⇓(*r*0);	22 *N*
22	March M	⇳(*w*0); ⇑(*r*0,*w*1,*r*1,*w*0); ⇳(*r*0); ⇑(*r*0,*w*1); ⇳(*r*1); ⇓(*r*1,*w*0,*r*0,*w*1);	16 *N*
23	March PS	⇳(*w*0); ⇑(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1); ⇑(*r*1,*w*0,*r*0,*w*1,*r*1); ⇑(*r*1,*w*0,*r*0,*w*1,*r*1,*w*0);⇑(*r*0,*w*1,*r*1,*w*0,*r*0);	23 *N*

Например, маршевый тест March C выглядит следующим образом:
⇳(*w*0); ⇑(*r*0,*w*1); ⇑(*r*1,*w*0); ⇳(*r*0); ⇓(*r*0,*w*1); ⇓(*r*1,*w*0); ⇳(*r*0);

Если в результате операции сравнения оказывается, что считанное из памяти значение не совпадает с ожидаемым, считается, что данная ячейка содержит ошибочное значение, вызванное наличием неисправности.

5.4. Задание для выполнения лабораторной работы

Задание для лабораторной работы №7

Разработать программу, которая моделирует работу ОЗУ с заданной согласно вариантам его организацией, и исследовать покрывающую способность маршевых тестов для различных классов неисправностей (табл. 5.2).

Таблица 5.2

Вариант	Тест 1	Тест 2	Размер ОЗУ, Мбит	Моделируемые неисправности
1	2	3	4	5
1	GALPAT	MATS	4	*AF*, *SAF*, *CFin*, *CFid*
2	Walking 0/1	MATS+	2	*AF*, *SAF*, *CFin*, *CFid*

Библиотека БГУИР

Продолжение табл. 5. 2

1	2	3	4	5
3	WALKING 0/1	MATS++	1	*AF*, *SAF*, *CFin*, *CFid*
4	GALPAT	Marching 1/0	8	*AF*, *SAF*, *CFin*, *CFid*
5	Walking 0/1	March X	6	*AF*, *SAF*, *CFin*, *CFid*
6	GALPAT	March Y	4	*AF*, *SAF*, *CFin*, *CFid*
7	GALPAT	March C	2	*AF*, *SAF*, *CFin*, *CFid*
8	Walking 0/1	March C-	1	*AF*, *SAF*, *CFin*, *CFid*
9	GALPAT	March A	8	*AF*, *SAF, CFin*, *CFid*
10	GALPAT	March B	6	*AF*, *SAF*, *CFin*, *CFid*
11	Walking 0/1	Algorithm B	4	*AF*, *SAF*, *CFin*, *CFid*
12	GALPAT	March C-	2	*AF*, *SAF*, *CFin*, *CFid*
13	GALPAT	March A	1	*AF*, *SAF*, *CFin*, *CFid*
14	Walking 0/1	March B	8	*AF*, *SAF*, *CFin*, *CFid*
15	WALKING 0/1	March X	6	*AF*, *SAF*, *CFin*, *CFid*
16	GALPAT	March Y	4	*AF*, *SAF*, *CFin*, *CFid*
17	Walking 0/1	March C	2	*AF*, *SAF*, *CFin*, *CFid*
18	WALKING 0/1	March C-	1	*AF*, *SAF*, *CFin*, *CFid*
19	GALPAT	March A	8	*AF*, *SAF*, *CFin*, *CFid*
20	Walking 0/1	MATS+	6	*AF*, *SAF*, *CFin*, *CFid*
21	WALKING 0/1	MATS++	4	*AF*, *SAF*, *CFin*, *CFid*
22	GALPAT	Marching 1/0	2	*AF*, *SAF*, *CFin*, *CFid*

Библиотека БГУИР

Окончание табл. 5. 2

1	2	3	4	5
23	Walking 0/1	March X	1	*AF*, *SAF*, *CFin*, *CFid*
24	WALKING 0/1	March Y	8	*AF*, *SAF*, *CFin*, *CFid*
25	GALPAT	March C	6	*AF*, *SAF*, *CFin*, *CFid*
26	Walking 0/1	March C-	4	*AF*, *SAF*, *CFin*, *CFid*
27	WALKING 0/1	March A	2	*AF*, *SAF*, *CFin*, *CFid*
28	GALPAT	March C-	1	*AF*, *SAF*, *CFin*, *CFid*
29	Walking 0/1	March A	8	*AF*, *SAF*, *CFin*, *CFid*
30	WALKING 0/1	March B	6	*AF*, *SAF*, *CFin*, *CFid*

Оценить временную сложность тестов для заданного ОЗУ.

Отчет по работе должен содержать следующие сведения о маршевом тесте, например, March C: $\Updownarrow(w0)$; $\Uparrow(r0,w1)$; $\Uparrow(r1,w0)$; $\Updownarrow(r0)$; $\Downarrow(r0,w1)$; $\Downarrow(r1,w0)$; $\Updownarrow(r0)$ и сведения об его покрывающей способности.

Задание для лабораторной работы №8

Разработать программу, которая моделирует работу ОЗУ с заданной согласно приведенным в табл. 5.3 вариантам его организации, и исследовать покрывающую способность маршевых тестов для кодочувствительных неисправностей, в которых участвует различное число ячеек.

Таблица 5.3

Вариант	Тест 1	Тест 2	Размер ОЗУ, Мбит	Моделируемые неисправности
1	2	3	4	5
1	MATS	March PS	4	*PNPSFK*3, *ANPSFK*5
2	MATS+	March PS	2	*PNPSFK*5, *ANPSFK*9
3	MATS++	March PS	1	*PNPSFK*9, *ANPSFK*3
4	Marching 1/0	March PS	8	*PNPSFK*3, *ANPSFK*5
5	March X	March PS	6	*PNPSFK*5, *ANPSFK*9
6	March Y	March PS	2	*PNPSFK*9, *ANPSFK*3
7	March C	March PS	1	*PNPSFK*3, *ANPSFK*5

Библиотека БГУИР

Окончание табл. 5.3

1	2	3	4	5
8	March C-	March PS	8	*PNPSFK*5, *ANPSFK*9
9	March A	March PS	6	*PNPSFK*9, *ANPSFK*3
10	March B	March PS	1	*PNPSFK*3, *ANPSFK*5
11	Algorithm B	March PS	8	*PNPSFK*5, *ANPSFK*9
12	March C-R	March PS	6	*PNPSFK*9, *ANPSFK*3
13	PMOVI	March PS	8	*PNPSFK*3, *ANPSFK*5
14	PMOVI-R	March PS	6	*PNPSFK*5, *ANPSFK*9
15	March U	March PS	4	*PNPSFK*9, *ANPSFK*3
16	March U-R	March PS	2	*PNPSFK*3, *ANPSFK*5
17	March LR	March PS	1	*PNPSFK*5, *ANPSFK*9
18	March LA	March PS	8	*PNPSFK*9, *ANPSFK*3
19	March M	March PS	6	*PNPSFK*3, *ANPSFK*5
20	MATS	March PS	4	*PNPSFK*5, *ANPSFK*9
21	MATS+	March PS	2	*PNPSFK*9, *ANPSFK*3
22	MATS++	March PS	1	*PNPSFK*3, *ANPSFK*5
23	Marching 1/0	March PS	8	*PNPSFK*5, *ANPSFK*9
24	March X	March PS	2	*PNPSFK*9, *ANPSFK*3
25	March Y	March PS	4	*PNPSFK*3, *ANPSFK*5
26	March C	March PS	1	*PNPSFK*5, *ANPSFK*9
27	March C-	March PS	4	*PNPSFK*9, *ANPSFK*3
28	March A	March PS	2	*PNPSFK*3, *ANPSFK*5
29	March B	March PS	4	*PNPSFK*5, *ANPSFK*9
30	Algorithm B	March PS	2	*PNPSFK*9, *ANPSFK*3

Библиотека БГУИР

ЛИТЕРАТУРА

Основная

1. Каган, Б. М. Основы эксплуатации ЭВМ / Б. М. Каган, И. Б. Мкртумян. – М. : Энергоатомиздат, 1988. – 218 с.

2. Горяшко, А. П. Синтез диагностируемых схем вычислительных устройств / А. П. Горяшко. – М. : Мир, 1987. – 244 с.

3. Согомонян, Е. С. Самопроверяемые устройства и отказоустойчивые системы / Е. С. Согомонян, Е. В. Слабаков. – М. : Радио и связь, 1989. – 208 с.

4. Садыхов, Р. Х. Технический сервис однородных вычислительных устройств / Р. Х. Садыхов, М. М. Татур. – Минск : Университетское, 2001. – 279 с.

5. Иванюк, А. А. Проектирование контролепригодных цифровых устройств / А. А. Иванюк, В. Н. Ярмолик. – Минск : Бестпринт, 2006. – 296 с.

6. Неразрушающее тестирование запоминающих устройств / В. Н. Ярмолик [и др.]. – Минск : Бестпринт, 2005. – 230 с.

7. Проектирование самотестируемых СБИС : монография. В 2 т. / В. Н. Ярмолик [и др.]. – Минск : БГУИР, 2001.

8. Ярмолик, В. Н. Контроль и диагностика цифровых узлов ЭВМ / В. Н. Ярмолик. – Минск : Наука и техника, 1988. – 240 с.

9. Ярмолик, С. В. Маршевые тесты для самотестирования ОЗУ / С. В. Ярмолик, А. П. Занкович, А. А. Иванюк. – Минск : Изд. центр БГУ, 2009. – 270 с.

Дополнительная

10. Бибило, П. Н. Основы языка VHDL / П. Н. Бибило. – М. : Солон-Р, 2000. – 200 с.

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

12. Adams, R. D. High Performance Memory Testing: Design Principles, Fault Modeling and Self – Test / R. D. Adams. – NY. : Kluwer Academic Publishers, 2003. – 247 p.

13. Bardell, P. H. Built – In self – test for VLSI: pseudorandom techniques / P. H. Bardell, W. McAnney, J. Savir. – NY. : John Wiley&Sons, 1987. – 576 p.

14. Goor, A. J. Testing Semiconductor Memories, Theory and Practice / A. J. Goor. – UK, Chichester : John Wiley & Sons, 1991. – 536 p.

15. Prince, B. High–performance memories: new architecture DRAM's and SRAM's, evolution and function / B. Prince – UK, Chichester : John Wiley & Sons Ltd., 1996. – 338 p.

Библиотека БГУИР

Св. план 2010, поз. 49

Учебное издание

Ярмолик Вячеслав Николаевич
Мурашко Игорь Александрович
Ярмолик Светлана Вячеславовна

КОНТРОЛЬ И ДИАГНОСТИКА СРЕДСТВ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Методическое пособие по выполнению лабораторных работ
для студентов специальности
«Вычислительные машины, системы и сети»
дневной формы обучения

Редактор Е. Н. Батурчик
Корректор Л. А. Шичко
Компьютерная верстка М. В. Чечетко

Подписано в печать	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Отпечатано на ризографе.	Усл. печ. л.
Уч.-изд. л.	Тираж 100 экз.	Заказ 745

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6

Библиотека БГУИР