6. Тестирование цифровых устройств

Проектирование дискретного устройства начитается с построения его *функциональной модели* (системы булевых функций либо конечного автомата). Этот процесс называется *функциональным уровнем* проектирования устройства. Затем по функциональной модели строится *структурная модель* – логическая сеть, и это *структурный уровень* проектирования. По логической сети из реальных логических элементов изготавливается дискретное устройство, это *физический уровень* проектирования.

Реализованное устройство может отличаться от спроектированного вследствие каких-либо физических причин (обрыва проводника, нарушения изоляционного слоя и т.д.). Подобные *дефекты* вызывают *неисправности* на логическом уровне, т.е. логическая сеть, построенная для устройства с дефектом, будет отличаться от спроектированной. Дефекты проявляются и на логическом уровне в виде *ошибок* – различия в функциональном описании схемы с дефектом и спроектированной схемы.

Мы рассмотрим математические методы обнаружения неисправностей, основная идея которых такова. Определяется класс неисправностей и строится множество *тестов* – наборов значений входных переменных, на которых выходы исправной и неисправной схем отличаются. Тесты могут строиться как при помощи моделирования поведения схем, так и с учетом структуры логической сети, т.е. поиск тестов производится на логическом уровне проектирования. Кроме этого, мы изложим основные принципы синтеза *легкотестируемых схем* – методов синтеза, которые позволяют построить тесты, не используя информацию о конкретной реализации логической сети, т.е. на функциональном уровне проектирования.

6.1. Модели неисправностей

6.1.1. Дефекты, неисправности, ошибки

Определение. *Дефект* – непреднамеренное различие между реализованным и проектируемым оборудованием.

Определение. *Неисправность* – представление дефекта на логическом уровне.

Определение. *Ошибка* – неправильный выходной сигнал, производимый устройством, содержащей дефект.

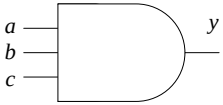
Все сказанное может быть представлено следующей таблицей.

Функциональный уровень	Система булевых функций	Ошибка
Логический уровень	Комбинационная схема	Неисправность
Физический уровень	Дискретное устройство	Дефект

Пример.

Функциональный уровень: $y = abc$.

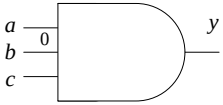
Логический уровень: комбинационная схема.



Физический уровень: дискретное устройство.

Дефект: второй вход вентиля «И» подключен к «земле» вместо входного контакта b .

Неисправность: неисправность типа «константа 0» на входе b (сигнал b постоянно установлен в логический ноль).



Ошибка: при значениях входов $a = b = c = 1$ выход $y = 0$ (а не 1, как в исправном устройстве).

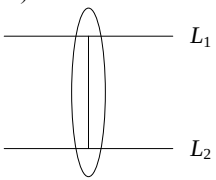
6.1.2. Дефекты

Типичными дефектами являются *обрыв* и *замыкание*.

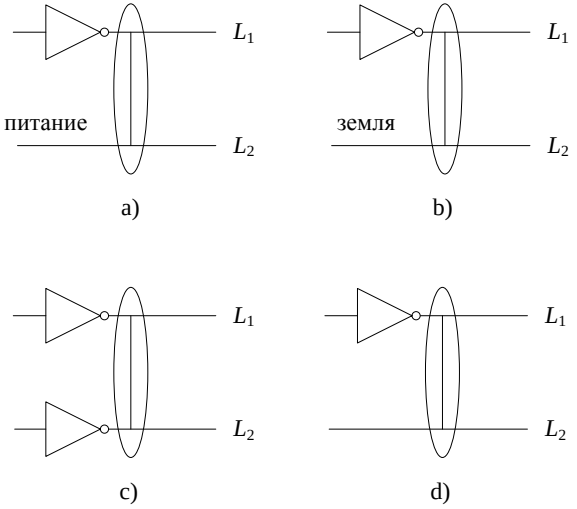
Обрыв соответствует нарушению соединения компонентов устройства. Причиной этого может служить недостаток или отсутствие проводящего материала, например, в металлическом проводнике. С другой стороны, отсутствие соединения может возникнуть вследствие наличия лишних частиц диэлектрика, например, между проводящими слоями.

Замыкание образуется в результате соединения линий устройства, которые в исправной системе должны быть изолированы друг от друга. Оно может быть вызвано наличием лишних проводящих частиц между проводниками, пробоем оксида, и т. д.

Пример. Рассмотрим пример неисправностей, вызванных дефектом «замыкание». Пусть частица замыкает две линии (на рисунке дефект обведен овалом).



На логическом уровне данный дефект может вызвать различные неисправности.



a) Одна из замыкаемых линий соединена с питанием, что проводит к установке выхода инвертора в высокое напряжение. Эта ситуация описывается *константной неисправностью «константа 1»*.

b) Одна из замыкаемых линий соединена с землей, что проводит к установке выхода инвертора в низкое напряжение. Эта ситуация описывается *константной неисправностью «константа 0»*.

c) Замыкание выходов двух различных вентилей приводит к *простой мостиковой неисправности*.

d) Замыкание выхода вентиля со своим входом приводит к *мостиковой неисправности с обратной связью*.

6.1.3. Типовые модели неисправностей

Одиночные константные неисправности (*stack-at fault 0, stack-at fault 1*). Один вход логической схемы принимает постоянное значение 0 или 1.

Кратные константные неисправности (*multiple stack-at faults*). Несколько входов логической схемы принимают постоянные значения сигналов.

Мостиковые неисправности (*bridge faults*). Несколько линий схемы, значения сигналов на которых не зависят друг от друга в исправной схеме, становятся зависимыми в неисправной.

Задержки (*delay faults*). Задержка распространения сигнала по одной или нескольким линиям схемы.

Перемежающиеся неисправности (*transient faults*). Вызываются изменениями внутренних параметров схемы. Ошибки возникают при некоторых (не всех!) состояниях схемы. Как правило, в итоге неисправность проявляется как постоянная.

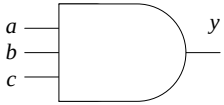
6.2. Построение проверяющих тестов для комбинационных схем

6.2.1. Полный проверяющий тест

Пусть имеется исправная комбинационная схема S, и схема S' – схема S с некоторой неисправностью.

Определение. *Тест* для данной неисправности – это такой набор значений переменных (булев вектор), на котором выход исправной схемы отличается от выхода схемы в присутствии данной неисправности.

Пример. Пусть схема S имеет вид



Рассмотрим константную неисправность $b = 0$. Тестом для нее будет набор 111.

Пусть введено множество неисправностей T для схемы S.

Определение. *Полный проверяющий тест* для множества неисправностей – это множество наборов значений переменных (булев вектор), содержащее по крайней мере по одному тесту для каждой неисправности множества T.

Пример. Для той же схемы S рассмотрим множество T, содержащее все одиночные константные неисправности на входах a, b. c. Легко убедиться, что

- набор 111 обнаруживает неисправности $a=0$, $b=0$, $c=0$;
- набор 011 – неисправность $a=1$;
- набор 101 – неисправность $b=1$;
- набор 110 – неисправность $c=1$.

Остальные наборы не обнаруживают одиночных константных неисправностей. Значит, множество $\{111,011,101,110\}$ является полным проверяющим тестом для множества неисправностей T.

Естественно, в общем случае для одного и того же множества неисправностей T можно построить несколько полных проверяющих тестов. Поэтому возникает вопрос оптимизации полного проверяющего теста по какому-либо критерию. Например, таким критерием может служить мощность теста, и мы приходим к задаче построения *кратчайшего полного проверяющего теста*. Если же каждый тест требует своих временных, финансовых или каких-либо других затрат, то ставится задача построения *минимального полного проверяющего теста* с минимальной суммой этих затрат.

Тестовые наборы (последовательности) могут быть найдены предварительно (вручную или автоматически), затем введены в соответствующее ПЗУ (постоянное запоминающее устройство). Они могут также генерироваться непосредственно в момент реализации тестирования.

Недостатком первого подхода является большой объем ПЗУ. С точки зрения минимизации аппаратурных затрат более выгодным является способ генерации тестов во время тестового эксперимента. Однако, если тест находится предварительно, то можно найти кратчайший либо минимальный полный тест.

Как правило, генерация тестов осуществляется в два этапа. На первом сравнительно быстрыми методами генерируются тесты, обнаруживающие большую часть неисправностей. На втором строятся тесты для неисправностей, не обнаружимых тестами первого этапа, при этом используются методы построения тестов для конкретной неисправности, учитывающие структуру схемы.

6.2.2. Методы генерации тестов

1) *Исчерпывающий и псевдоисчерпывающий тест.* Наиболее простым методом для генерации тестовых наборов является по-

строение исчерпывающего теста. Для схемы с n входами строится множество всевозможных булевых векторов длины n (всего 2^n). Однако, этот метод пригоден только для схем с малым числом входов, так как время тестирования зависит экспоненциально от n. При $n > 20$ исчерпывающее тестирование невозможно.

Возможна генерация псевдоисчерпывающего теста, который требует значительно меньших затрат времени. При псевдоисчерпывающем тесте схема разбивается на подсхемы (не обязательно несвязные), называемые сегментами, и для каждго сегмента строится исчерпывающий тест. Недостатком метода является то, что входы и выходы сегментов не всегда являются входами и выходами схемы. Для обеспечения доступа тестирующего оборудования к входам и выходам сегментов требуется введение дополнительных входов и выходов к схеме в целом, что повышает стоимость аппаратурной реализации.

2) *Псевдослучайный тест.* Псевдослучайный тест один из наиболее часто используемых методов генерации тестовых наборов. Генератор псевдослучайных двоичных чисел генерирует последовательность векторов, которые имеют свойства случайных наборов, однако, имеется возможность их повторения. Псевдослучайные наборы обнаруживают большой процент (90-95%) одиночных константных неисправностей. Оценки полноты теста могут быть получены при помощи моделирования, которое, как правило, требует больших вычислительных затрат.

Если некоторая неисправность обнаруживается тестовым набором, вероятность появления которого очень мала, то вероятность обнаружения такой неисправности псевдослучайным тестом тоже мала. Для устранения таких ситуаций применяются различные модификации псевдослучайного теста.

В простейшем случае очередной псевдослучайный двоичный набор X_i обрабатывается следующим образом. С помощью программы моделирования определяется число неисправностей $n(X_i)$, которые он обнаруживает (без учета тех, которые уже обнаруживаются множеством $X = \{X_1, ..., X_{i-1}\}$). Если $n(X_i)$ превышает некоторый заранее заданный порог, то набор X_i добавляется к множеству X, и генерируется следующий набор. Этот процесс продолжается, пока не будет достигнута заданная пользователем полнота тестовой последовательности, либо пороговая величина длины или времени

построения теста.

Обычно датчик генерирует набор, в котором каждый вход с равной вероятностью принимает значение 0 и 1, но может оказаться так, что сгенерированные таким образом наборы перестают тестировать новые неисправности. В этом случае вероятности генерации 0, 1 для входов «подстраиваются»: либо учитывается структура схемы, либо применяются адаптивные алгоритмы.

3) *Сочетание случайной генерации тестовых наборов и детерминированных тестов*. При этом подходе сравнительно небольшое число предварительно полученных (детерминированных) наборов встраиваются в последовательность, генерируемую датчиком. В этом случае множество хранимых наборов невелико, поэтому стоимость реализации будет ниже, чем при использовании хранимых тестов.

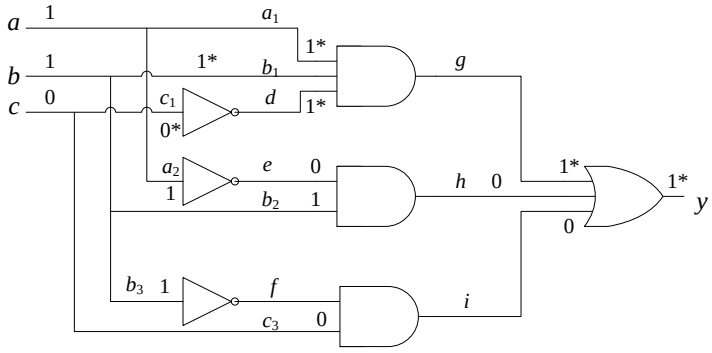
6.2.3. Структурные методы построения тестов

Данные методы, в отличие от предыдущих, используют знания и структуре логической сети.

Метод критических путей. Пусть на входы схемы подается набор α, тогда каждый вход каждого элемента схемы принимает определенное значение.

Определение. Значение на входе называется *критическим*, если его изменение приводит к изменению значения выхода схемы.

Пример. Рассмотрим комбинационную схему с входами a, b, c, на которую подан входной набор 110. Критические значения отмечены символом *.



Понятно, что данный набор является тестом для неисправностей

на входах, принимающих критические значения, причем тест проверяет константные неисправности, инверсные критическим значениям.

Пример. Для данной схемы набор 110 является тестом для константных неисправностей $a_1 = 0$, $b_1 = 0$, $c_1 = 1$, $d = 0$, $g = 0$.

Рассмотрим произвольный путь от входа схемы к ее выходу.

Определение. Путь называется *критическим*, если значения всех лежащих на нем входов элементов являются критическими.

Пример. В предыдущем примере критическим, например, является путь $cdgy$.

Тест проверяет половину всех константных неисправностей на критическом пути. На этом и основан метод критических путей.

Пусть сначала схема состоит из одного вентиля. Выход вентиля может принимать значения 0 и 1. Найдем для каждого значения выхода каждого вентиля наборы, содержащие критические значения, или *критические наборы* (для примера рассмотрим двухвходовые вентили).

Вентиль	«И»	«ИЛИ»	«НЕ»	«И-НЕ»	«ИЛИ-НЕ»
Значение	0	0	0	0	0
Наборы	0*1 10*	0*0*	1*	1*1*	01* 1*0
Значение	1	1	1	1	1
Наборы	1*1*	01* 1*0	0*	0*1 10*	0*0*

Здесь критические значения отмечены символом *.

Алгоритм построения тестов с использованием критических путей

Начало. Дана комбинационная схема, требуется построить для нее критические пути и найти тесты для одиночных константных неисправностей.

Шаг 1. Выход схемы помечается каким-либо критическим значением.

Шаг 2. Рассматриваем очередной вентиль V, выход y которого помечен критическим значением σ^*, или же вход x схемы, помеченный критическим значением σ^*. Если таковых нет, идем на шаг

5. Иначе помечаем значением σ все входы, отождествленные с y (x). Если таких входов нет, переходим к шагу 6. Иначе выбираем x' – отождествленный с y (x) вход вентиля V'.

Шаг 3. Если выход вентиля V' помечен значением δ, выполняем шаг 4, иначе – шаг 5.

Шаг 4. Если при данном значении входа x', учитывая пометки остальных входов, невозможно получить данное значение выхода y', то для рассматриваемого критического значения $y = \sigma^*$ не существует критических путей, идем на шаг 2. Иначе выбираем x' – отождествленный с y (x) вход вентиля V' и идем на шаг 3.

Шаг 5. Если, учитывая пометки остальных входов вентиля V', можно определить значение δ' выхода вентиля V', то приписываем выходу V' и всем отождествленным с ним входам вентилей пометку δ'. Далее выполняем шаг 3 для всех таких входов x'.

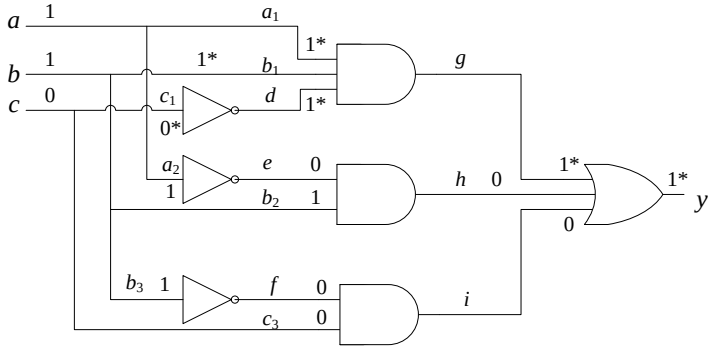
Шаг 6. Если рассматривается выход вентиля V, то находим для этого вентиля все критические наборы (так как их может быть несколько, мы можем получить несколько по разному помеченных схем). Идем на шаг 2.

Шаг 7. Выписываем полученные на всех схемах критические наборы. Если какие-либо входы остались неопределенными, доопределяем их так, чтобы значения на выходах вентилей оставались теми же.

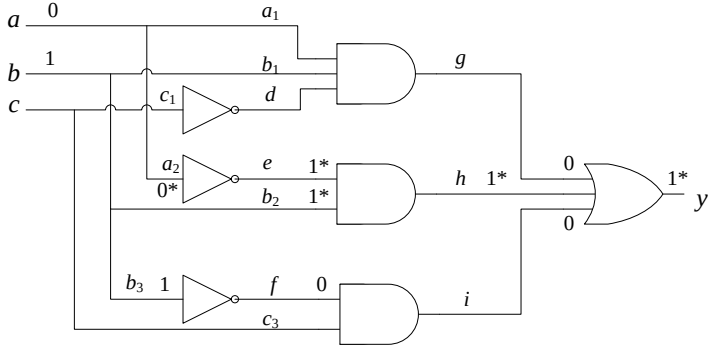
Конец. Получен тест для одиночных константных неисправностей на критических путях.

Тесты строятся «от выходов к входам», но в ходе построения теста полученные значения полюсов вентилей распространяются «от входов к выходам», Если выход элемента помечен критическим значением, то на вход его подается критический набор. Так как таких наборов может быть несколько, мы можем получить несколько по-разному размеченных схем.

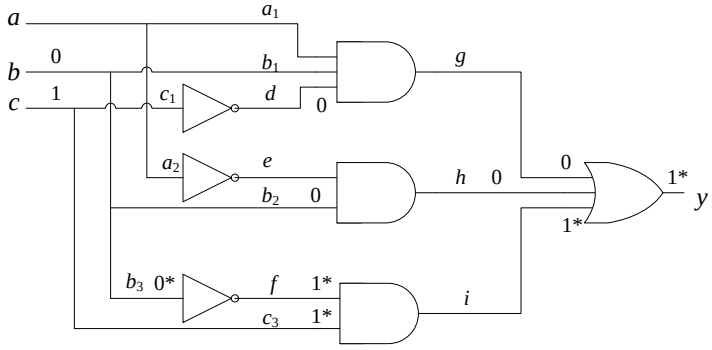
Пример. Рассмотрим комбинационную схему из предыдущего примера, найдем тесты для $y = 1^*$. Так как для этого значения выхода критическими являются три набора: 1^*00, 01^*0, 001^*, в итоге мы получим три по-разному размеченных схемы. Продолжаем размечать схемы от выходов к входам, получаем, что для каждой схемы далее существуют критические наборы.



Для этой схемы критическим является набор $1^*1^*0^*$.



Для этой схемы существуют два критических набора: 0^*1^*0 и 0^*1^*1, эти наборы могут быть объединены в интервал 0^*1^*-.



Для этой схемы существуют два критических набора: 00^*1^* и 10^*1^*, эти наборы могут быть объединены в интервал $–0^*1^*$.

Таким образом, критическими для значения выхода $y=1$ являются наборы: $1^*1^*0^*$, $0^*1^*–$., $–0^*1^*$. Как видно из примера, критические наборы могут быть представлены не булевыми, а троичными векторами. В качестве теста здесь можно взять любой вектор из интервала. В нашем случае множество, например, {110,010,001} проверяет все одиночные константные неисправности для критического значения $y=1$. Добавив к ним тесты для $y=0$, получим тест для одиночных константных неисправностей на критических путях.

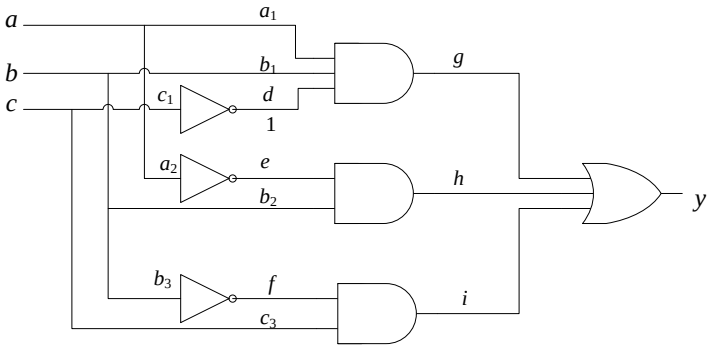
Для сокращения времени построения теста можно применять некоторые эвристические приемы, в частности, в первую очередь рассматривать критические наборы для еще не проверенных константных неисправностей, и т.д. Вообще метод не гарантирует полноты теста, но, как показывает практика, служит хорошим первым приближением к полному тесту. Для непроверенных неисправностей можно затем воспользоваться методами построения теста для конкретной неисправности, рассмотренными ниже.

Метод различающей функции. Пусть S – комбинационная схема без неисправностей с оператором $f(x_1,..,x_n)$, S' – комбинационная схема с неисправностью с оператором $f'(x_1,..,x_n)$. *Различающая функция* имеет вид

$$\varphi(x_1,..,x_n)=f(x_1,..,x_n)\oplus f'(x_1,..,x_n).$$

Любой набор, для которого $\varphi(x_1,..,x_n)=1$, является тестом для рассматриваемой неисправности.

Пример. Рассмотрим ту же схему и неисправность $d=1$.



$$y(a,b,c) = ab\bar{c} \vee \bar{a}b \vee \bar{b}c;$$
$$y'(a,b,c) = ab \vee \bar{a}b \vee \bar{b}c;$$
$$\varphi(a,b,c) = abc.$$

Решая уравнение $abc = 1$, получаем тест 111.

Метод активизации одномерного пути. Основная идея метода состоит в выборе пути от места неисправности до некоторого внешнего выхода и активизации этого пути. *Шаг 1.* Активизация неисправности – на неисправную линию подается сигнал, инверсный неисправности.

Шаг 2. Распространение неисправности до одного из внешних выходов. На этом шаге выбирается произвольный путь от места неисправности до выхода схемы. Затем входам вентилей, вошедших в активизированный путь, присваиваются такие значения, чтобы выход вентиля менялся при изменении вошедшего в активизированный путь входа (то есть чтобы значение этого входа было критическим). Таким образом, входы, не вошедшие в активизированный путь, примут значение 1 для вентилей «И», «И-НЕ», и значение 0 для «ИЛИ», «ИЛИ-НЕ». Активизированный путь прослеживается от входа к выходу, и находятся значения выходов всех принадлежащих ему вентилей.

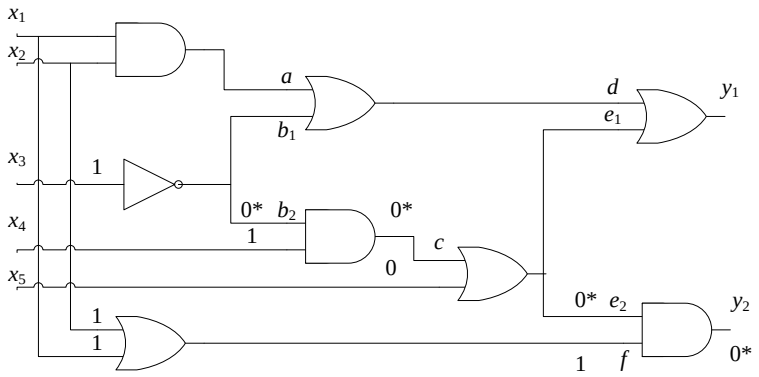
Шаг 3. Доопределение значений сигналов. На этом шаге находятся значения внешних входов схемы, которые обеспечивают требуемые значения на выходах вентилей активизированного пути. При этом путь прослеживается от выходов к входам.

Пример. Рассмотрим комбинационную схему с неисправностью $b_2 = 1$.

Шаг 1. Полагаем $b_2 = 0^*$.

Шаг 2. Выбираем путь $b_2 c e_2 y_2$. Чтобы значение $b_2 = 0^*$ было критическим, полагаем $x_4 = 1$. При этом $c = 0^*$, чтобы это значение было критическим, полагаем $x_5 = 0$. Далее получаем $e_2 = 0^*$, значит, $f = 1$. Окончательно имеем $y_2 = 0^*$.

Шаг 3. Для обеспечения условия $f = 1$ положим, например, $x_1 = 1$, $x_2 = 1$ (возможны и другие значения); для обеспечения условия $b_2 = 0^*$ положим $x_3 = 1$.

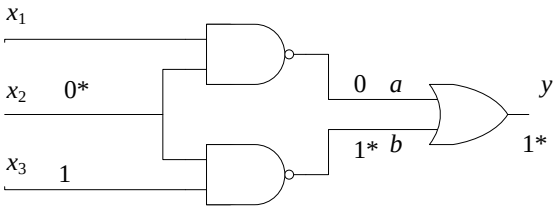


Получен тестовый набор 11110, при наличии неисправности $b_2 = 1$ значение выхода схемы $y_2 = 1$, иначе $y_2 = 0$.

Отметим, что на третьем шаге неудачный выбор значений входных сигналов может привести к несовместимости полученных значений сигналов на некоторых линиях схемы. В этом случае необходимо выбрать другие значения входных сигналов, то есть этот шаг носит переборный характер.

Однако, даже при выполнении полного перебора путей метод не гарантирует построения теста для всех неисправностей, так как схема может быть такой структуры, когда невозможно активизировать только один путь, а активизация нескольких путей одновременно приводит к конфликту.

Пример. Рассмотрим комбинационную схему.



Пусть требуется построить тест для неисправности $x_2 = 1$. Подадим на вход x_2 критическое значение 0^*. Активизация пути x_2by тре-

бует значения $a = 0$, но, поскольку $x_2 = 0$, при любом значении переменной x_1 получаем $a = 1$. В силу симметрии невозможно также активизировать путь $x_2 a y$. Следовательно, тест для данной неисправности не может быть найден методом активизации одномерного пути. Однако, это не значит, что теста не существует – в данном примере им является набор 101. Легко убедиться, что в отсутствие неисправности выход схемы принимает значение единицы, а в присутствии – нуля.

6.3. Синтез легкотестируемых комбинационных схем

Предыдущие методы построения тестов ориентированы на работу с уже готовой логической схемой. Однако, при проектировании устройства целесообразно учесть его контролепригодность, т.е. желательно обеспечивать существование для устройства простой процедуры построения достаточно короткой тестовой последовательности. Это обеспечивает экономию затрат на дополнительное (тестирующее) оборудование и сокращает время тестирования. Обычно проблемы контролепригодности обсуждаются в связи с тестовыми последовательностями, ориентированными на одиночные константные неисправности на полюсах устройства. Кроме того, такой подход позволяет строить тесты, оставаясь на функциональном уровне описания устройства.

Если выход дискретного устройства (ДУ) зависит только от его входа в данный момент времени, и входные и выходные символы суть булевы векторы, то поведение ДУ можно описать системой частичных (не полностью определенных) булевых функций $F(X)$ из m частичных булевых функций

$$F(X) = \{f_1(x_1,...,x_n),...,f_m(x_1,...,x_n)\},$$

зависящих от n переменных $X = \{x_1,...,x_n\}$. Множества определения всех частичных булевых функций назовем *рабочей областью функционирования* (РОФ) ДУ. Для данного входного вектора из РОФ значением системы будем называть вектор значений всех функций на данном входном векторе. Требуется синтезировать комбинационную схему, реализующую систему частичных булевых функций $F(x)$ таким образом, чтобы полный проверяющий тест для схемы был, по возможности, более короткий и существовала простая процедура его построения.

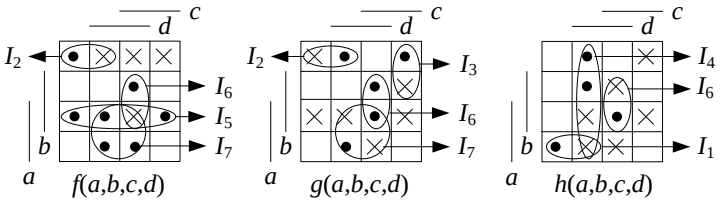
6.3.1. Общая схема построения легкотестируемого устройства

Этап I. Построение для системы частичных функций $F(x)$ безызбыточной системы ДНФ $D=\{D_1,...,D_m\}$.

Определение. Система ДНФ $D=\{D_1,...,D_m\}$ называется безызбыточной для системы функций $F(x)=\ f_1\ X\ ,\ldots,\ f_m\ X\ $, если

- вычеркивание любой конъюнкции из любой ДНФ D_i приводит к тому, что ДНФ D_i перестает задавать функцию $f_i\ \cdot\ ;$
- вычеркивание любой буквы из любой конъюнкции K приводит к тому, что хотя бы одна ДНФ D_i, содержащая конъюнкцию K, перестает задавать функцию $f_i\ \cdot\ .$

Пример. Рассмотрим систему трех булевых функций.



Безызбыточная система ДНФ для нее может иметь, например, такой вид.

$$D=\left\{\begin{array}{l} \text{ДНФ}_f=\underset{K_2}{\bar{a}\bar{b}\bar{c}}\ \vee\ \underset{K_5}{ab}\ \vee\ \underset{K_6}{bcd}\ \vee\ \underset{K_7}{ad}\ ; \\ \text{ДНФ}_g=\underset{K_2}{\bar{a}\bar{b}\bar{c}}\ \vee\ \underset{K_3}{\bar{a}c\bar{d}}\ \vee\ \underset{K_6}{bcd}\ \vee\ \underset{K_7}{ad}\ ; \\ \text{ДНФ}_h=\underset{K_1}{a\bar{b}\bar{c}}\ \vee\ \underset{K_4}{\bar{c}d}\ \vee\ \underset{K_6}{bcd}\ . \end{array}\right.$$

Этап II. Построение схемы факторизационным методом.

Определение. Назовем ДНФ D^* *дизъюнктивным фактором* системы D, если D^* получается из некоторой ДНФ D_i вычеркиванием некоторых конъюнкций. Будем говорить, что D^* *является частью* D_i.

Определение. Назовем конъюнкцию K^* *конъюнктивным фактором* системы D, если K^* получается из некоторой конъюнкции K какой-либо ДНФ D_i вычеркиванием некоторых букв. Будем говорить, что K^* *является частью* K.

Определение. *Порождающим множеством дизъюнктивного фактора* D^* будем называть множество ДНФ системы D, для которых D^* является частью. В порождающее множество могут быть включены не все такие ДНФ.

Определение. *Порождающим множеством конъюнктивного фактора* K^* будем называть множество конъюнкций системы D, для которых K^* является частью. В порождающее множество могут быть включены не все такие конъюнкции.

Факторизационный метод синтеза предполагает выполнение следующих шагов.

Шаг 1. Выбор множества $D^* = \{D_1^*, ..., D_p^*\}$ дизъюнктивных факторов системы и построение для каждого i-го элемента порождающего множества.

Шаг 2. Выбор множества $K^* = \{K_1^*, ..., K_q^*\}$ конъюнктивных факторов системы и построение для каждого i-го элемента порождающего множества.

Шаг 3. Построение на основе выбранных факторов и порождающих множеств схемы C, реализующей систему D.

Шаги 1, 2 выполняются так, чтобы, во-первых, учесть нагрузочные способности и числа входов реальных элементов, из которых строится схема, и во-вторых, сделать схему более простой в том или ином смысле. Конкретизация шагов 1, 2 определяет модификацию факторизационного метода синтеза.

Рассмотрим шаг 3. В схеме C можно выделить два уровня: уровень конъюнкций и уровень дизъюнкций. Входами уровня конъюнкций являются входы схемы, а также инвертированные значения входов. На выходах уровня конъюнкций реализуются конъюнкции системы D. Входы уровня дизъюнкций отождествляются с выходами уровня конъюнкций. Выходы уровня дизъюнкций являются выходами схемы C.

Уровень конъюнкций собирается из древовидных подсхем, реализующих операцию конъюнкции. Древовидная подсхема заменяет элемент «И» с неограниченным числом входов. Подсхемы делятся

по своему назначению на подсхемы конъюнктивных факторов и подсхемы покрытий конъюнкций. Входы подсхемы фактора K_i^* отождествляются с входами схемы C или с их инверсиями. Входы подсхемы покрытия некоторой конъюнкции отождествляются с выходами подсхем факторов этой конъюнкции. Выход подсхемы покрытия конъюнкции является выходом уровня конъюнкций.

Уровень дизъюнкций собирается из древовидных подсхем, реализующих операцию дизъюнкции. Древовидная подсхема заменяет элемент «ИЛИ» с неограниченным числом входов. Подсхемы делятся по своему назначению на подсхемы дизъюнктивных факторов и подсхемы покрытий ДНФ. Входы подсхемы фактора D_i^* отождествляются с теми выходами уровня конъюнкций, на которых реализуются конъюнкции из D_i^*. Входы подсхемы покрытия ДНФ отождествляются с выходами подсхем дизъюнктивных факторов этой ДНФ. Выход подсхемы покрытия ДНФ является выходом схемы C.

Пример. Рассмотрим систему ДНФ из предыдущего примера.

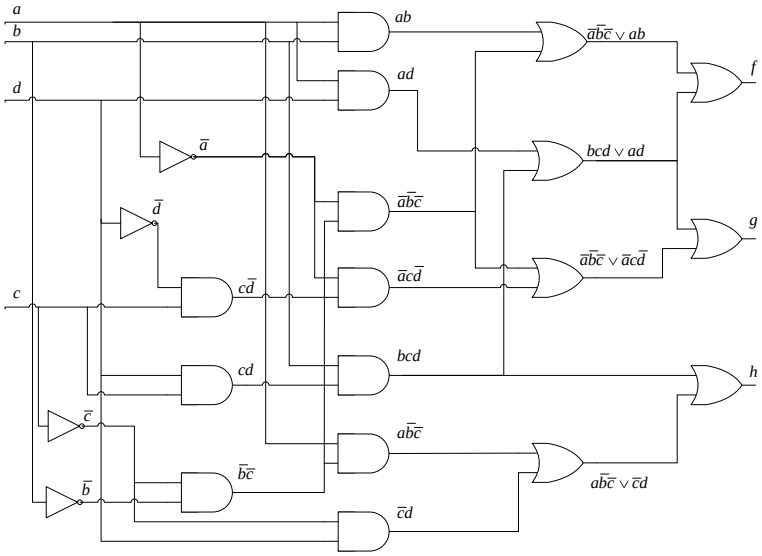
$$D = \left\{ \begin{array}{l} \text{ДНФ}_f = \underset{K_2}{\bar{a}\bar{b}\bar{c}} \vee \underset{K_5}{ab} \vee \underset{K_6}{bcd} \vee \underset{K_7}{ad}\ ; \\ \text{ДНФ}_g = \underset{K_2}{\bar{a}\bar{b}\bar{c}} \vee \underset{K_3}{\bar{a}c\bar{d}} \vee \underset{K_6}{bcd} \vee \underset{K_7}{ad}\ ; \\ \text{ДНФ}_h = \underset{K_1}{a\bar{b}\bar{c}} \vee \underset{K_4}{\bar{c}d} \vee \underset{K_6}{bcd}\ . \end{array} \right.$$

Синтезируем схему на двухвходовых элементах с нагрузочной способностью, равной двум.

Выделим конъюнктивные факторы: это $\bar{b}\bar{c}$ для конъюнкций K_1 и K_2, cd для конъюнкции K_6 и $c\bar{d}$ для конъюнкции K_3 (можно выбрать и другие факторы).

Выделим дизъюнктивные факторы: это $bcd \vee ad$ для ДНФ$_f$ и ДНФ$_g$, $\bar{a}\bar{b}\bar{c} \vee ab$ для ДНФ$_f$, и $\bar{a}\bar{b}\bar{c} \vee \bar{a}c\bar{d}$ для ДНФ$_g$, $a\bar{b}\bar{c} \vee \bar{c}d$ для ДНФ$_h$ (можно выбрать и другие факторы).

Схема, построенная с учетом этих факторов, представлена на следующей странице.



6.3.2. Проявление константных неисправностей на функциональном уровне

Выясним, имея в виду факторизационный синтез по безызбыточной системе ДНФ, какой неисправности безызбыточной системы эквивалентна одиночная или кратная неисправность на полюсах элементов полученной схемы.

Рассмотрим одиночные константные неисправности.

Константа 0 на входе конъюнктора. Эта неисправность приводит к тому, что на выход конъюнктора равен нулю при любом наборе значений входов. Если конъюнктор относится к подсхеме конъюнктивных факторов, то его выход осуществляется с входами конъюнкторов подсхемы покрытий конъюнкций, и в итоге один или несколько выходов этой подсхемы принимают значение 0. Если конъюнктор относится к подсхеме покрытий конъюкций, то его выход отождествляется с одним или несколькими входами уровня дизъюнкций. Значит, данная неисправность приводит к кратной неисправности «константа 0» на уровне дизъюнкций. На функциональном уровне это приводит к исчезновению одной или нескольких конъюнкций из одной или нескольких ДНФ.

Константа 1 на входе конъюнктора. На функциональном уров-

не эта неисправность приводит к тому, что из конъюнкции, реализуемой данным конъюнктором, исчезает буква (если конъюнктор относится к подсхеме конъюнктивных факторов) или несколько букв (если он относится к подсхеме покрытий конъюнкций). В результате исчезает одна или несколько букв из одной или нескольких конъюнкций системы ДНФ.

Константа 0 на входе дизъюнктора. На функциональном уровне эта неисправность приводит к исчезновению конъюнкции из одной или нескольких ДНФ.

Константа 1 на входе дизъюнктора. Эта неисправность приводит к тому, что на выходы одного или нескольких дизъюнкторов становятся равными единице. На функциональном уровне это означает, что вместо некоторых функций реализуется константа 1. Также эту неисправность можно рассматривать как исчезновение всех букв из некоторых конъюнкций системы.

Итак, любая одиночная константная неисправность на входе схемы, синтезированной факторизационным методом, на функциональном уровне проявляется как исчезновение одной или нескольких конъюнкций или исчезновение одной или нескольких букв их конъюнкций системы ДНФ.

Рассмотрим кратные константные неисправности. Они являются комбинациями одиночных неисправностей.

Кратная константная неисправность на входах одного конъюнктора. Если хотя бы на одном входе есть неисправность «константа 0», то кратная неисправность эквивалентна одиночной типа «константа 0», то есть приводит к исчезновению одной или нескольких конъюнкций. Если все неисправности являются неисправностями типа «константа 1», это приводит к исчезновению нескольких букв в одной или нескольких конъюнкциях.

Кратная константная неисправность на входах разных конъюнкторов приводит к исчезновению некоторых конъюнкций, или к исчезновению некоторых букв из конъюнкций, или к сочетанию того и другого.

Кратная константная неисправность на входах одного или нескольких дизъюнкторов. Если хотя бы на одном входе есть неисправность «константа 1», это приведет к исчезновению всех букв в одной или нескольких конъюнкциях. Если все неисправности имеют тип «константа 0», это приведет к исчезновению нескольких конъюнкций. В общем случае кратная константная неисправность такого типа приводит к комбинации исчезновения всех букв в одной или

нескольких конъюнкциях и исчезновения одной или нескольких конъюнкций.

Кратная константная неисправность на входах конъюнкторов и дизъюнкторов в общем случае приводит к комбинации исчезновения всех букв в одной или нескольких конъюнкциях и исчезновения одной или нескольких конъюнкций.

Итак, любая одиночная или кратная константная неисправность схемы, синтезированной по безызбыточной системе ДНФ факторизационным методом, приводит либо к исчезновению конъюнкций, либо к исчезновению букв из конъюнкций, либо к комбинации этих ошибок.

6.3.3. Построение тестов для константных неисправностей

Рассмотрим конъюнкцию K, входящую в подмножество ДНФ $D^K = \{D_{i_1}, ..., D_{i_k}\}$ системы D.

Определение. Набор значений входных переменных называется *a -тестом*, если он обращает в единицу только конъюнкцию K по крайней мере одной ДНФ D_j из множества D^K.

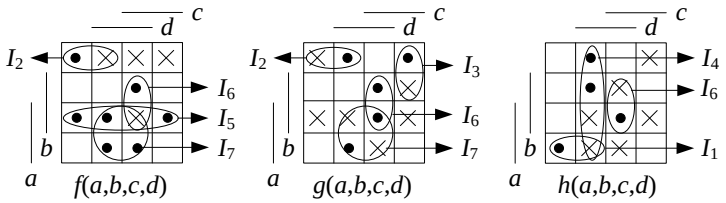
Другими словами, *a* -тест обнаруживает исчезновение конъюнкции из ДНФ.

Пусть конъюнкция $K(x_i)$ получается вычеркиванием переменной x_i из конъюнкции K.

Определение. Набор значений входных переменных называется *b -тестом*, если он обращает в единицу конъюнкцию $K(x_i)$ и обращает в ноль все конъюнкции по крайней мере одной ДНФ D_j из множества D^K.

Итак, *b* -тест обнаруживает исчезновение буквы из конъюнкции.

Пример. Рассмотрим систему булевых функций $F = \{f, g, h\}$, заданную безызбыточной системой ДНФ.



$$D = \left\{ \begin{array}{l} \text{ДНФ}_f = \underset{K_2}{\bar{a}\bar{b}\bar{c}} \vee \underset{K_5}{ab} \vee \underset{K_6}{bcd} \vee \underset{K_7}{ad}\ ; \\ \text{ДНФ}_g = \underset{K_2}{\bar{a}\bar{b}\bar{c}} \vee \underset{K_3}{\bar{a}c\bar{d}} \vee \underset{K_6}{bcd} \vee \underset{K_7}{ad}\ ; \\ \text{ДНФ}_h = \underset{K_1}{a\bar{b}\bar{c}} \vee \underset{K_4}{\bar{c}d} \vee \underset{K_6}{bcd}\ . \end{array} \right.$$

Возьмем конъюнкцию $K_6 = bcd$. Набор 0111 является a-тестом для этой конъюнкции, так как все конъюнкции ДНФ_f, кроме K_6, на этом наборе обращаются в ноль. Этот тест обнаруживает исчезновение конъюнкции K_6 из ДНФ_f.

Возьмем конъюнкцию $K_6(b) = cd$. Набор 0011 является b-тестом для этой конъюнкции, так как все конъюнкции ДНФ_g на этом наборе обращаются в ноль. Этот тест обнаруживает исчезновение буквы b из конъюнкции K_6.

Возьмем конъюнкцию $K_6(c) = bd$. Набор 0101 является b-тестом для этой конъюнкции, так как все конъюнкции ДНФ_f на этом наборе обращаются в ноль. Этот тест обнаруживает исчезновение буквы c из конъюнкции K_6.

Возьмем конъюнкцию $K_6(d) = bc$. Набор 0110 является b-тестом для этой конъюнкции, так как все конъюнкции ДНФ_f на этом наборе обращаются в ноль. Этот тест обнаруживает исчезновение буквы d из конъюнкции K_6.

Выше было показано, что любая константная неисправность схемы, синтезированной по безызбыточной системе ДНФ факторизационным методом, приводит либо к исчезновению конъюнкций, либо к исчезновению букв из конъюнкций, либо к комбинации этих ошибок.

Рассмотрим безызбыточную систему ДНФ $D = \{D_1, ..., D_m\}$ и конъюнкцию K, входящую в подмножество ДНФ $D^K = \{D_{i_1}, ..., D_{i_k}\}$ системы D.

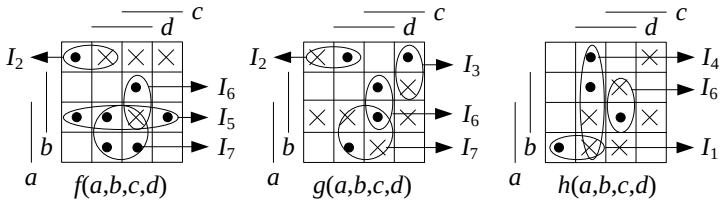
Вычеркивание конъюнкции K из любой D_j приводит к тому, что получившаяся ДНФ $D_j(K)$ перестает задавать частичную

функцию $f_j(x_1,...,x_n)$. Это означает, что существует такой набор $\alpha \in M^1_{f_j}$, на котором D_j принимает значение 1, а $D_j(K)$ – значение 0. Так как $D_j = D_j(K) \vee K$, то набор α обращает в ноль все конъюнкции ДНФ D_j, кроме K, то есть является a-тестом для K. Значит, исчезновение конъюнкции из одной или нескольких ДНФ обнаруживается соответствующим a-тестом.

Вычеркивание любой буквы x_i из конъюнкции K во всех ДНФ множества $D^K = \{D_{i_1},...,D_{i_k}\}$ приводит к тому, что хотя бы одна из получившихся ДНФ $D_j(K,i)$ перестает задавать частичную функцию $f_j(x_1,...,x_n)$. Это означает, что существует такой набор $\beta \in M^0_{f_j}$, на котором D_j принимает значение 0, а $D_j(K,i)$ – значение 1. Так как $D_j(K,i) = D_j \vee K(x_i)$, то набор β обращает в ноль все конъюнкции ДНФ D_j (в том числе K) и обращает в единицу конъюнкцию $K(x_i)$, то есть является b-тестом для K. Значит, исчезновение буквы из одной и той же конъюнкции всех ДНФ обнаруживается соответствующим b-тестом.

Отсюда следует, что любая константная неисправность обнаружится a- или b-тестом, причем все эти тесты лежат в множествах $M^0_{f_j}$, $M^1_{f_j}$, то есть в рабочей области функционирования устройства. Значит, для построения полного теста для константных неисправностей такой схемы требуется построить a- и b-тесты для всех конъюнкций системы ДНФ.

Пример. Рассмотрим ту же систему ДНФ.



$$D = \left\{ \begin{array}{l} \text{ДНФ}_f = \underset{K_2}{\bar{a}\bar{b}\bar{c}} \vee \underset{K_5}{ab} \vee \underset{K_6}{bcd} \vee \underset{K_7}{ad}\ ; \\ \text{ДНФ}_g = \underset{K_2}{\bar{a}\bar{b}\bar{c}} \vee \underset{K_3}{\bar{a}c\bar{d}} \vee \underset{K_6}{bcd} \vee \underset{K_7}{ad}\ ; \\ \text{ДНФ}_h = \underset{K_1}{a\bar{b}\bar{c}} \vee \underset{K_4}{\bar{c}d} \vee \underset{K_6}{bcd}\ . \end{array} \right.$$

a -тесты

K_1	K_2	K_3	K_4	K_5	K_6	K_7
$1000(h)$	$0000(f)$ $0001(g)$	$0010(g)$	$0001(h)$ $0101(h)$	$1100(f)$ $1110(f)$	$0111(f,g)$ $1111(h)$	$1001(f,g)$ $1011(f)$

b -тесты

	K_1	K_2	K_3	K_4	K_5	K_6	K_7
a	$0000(h)$	$1000(f,g)$	$1010(g)$		$0100(f)$ $0101(f)$ $0110(f)$		$0101(f,g)$ $0011(g)$
b	$1100(h)$	$0100(f,g)$ $0101(f,g)$			$1000(f)$ $1010(f)$	$0011(g,h)$	
c	$1010(h)$	$0011(g)$	$0100(g)$	$0011(h)$		$0101(f,g)$	
d			$0011(g)$	$0000(h)$ $0100(h)$ $1100(h)$		$0110(f,h)$	$1000(f,g)$ $1010(f,g)$

Минимальное число всех a -тестов, необходимое для полноты теста, не меньше числа всех конъюнкций, то есть длины системы ДНФ, так как для каждой конъюнкции строится свой тест.

Минимально возможное число b -тестов, необходимое для полноты теста, не превышает суммы букв всех конъюнкций системы ДНФ, то есть ранга системы. Однако, это число может быть уменьшено, так как один и тот же тест может обнаруживать несколько неисправностей. Далее мы рассмотрим проблему минимизации полного теста.

6.4. Минимизация полного теста

Задача минимизации полного теста может быть сведена к задаче поиска кратчайшего покрытия булевой матрицы. Сопоставим столбцам матрицы неисправности, строкам – тесты, элемент с номером i, j положим равным единице, если и только если i-й тест обнаруживает j-ю неисправность. Затем найдем кратчайшее покрытие этой матрицы. Так как задача поиска покрытия является вычислительно сложной, имеет смысл заранее ее упростить, например, выделить неисправности, обнаруживаемые единственным тестом (аналог ядерных столбцов и строк таблицы), и т.д.

Пример. Рассмотрим тесты,из предыдущего примера.

a-тесты

K_1	K_2	K_3	K_4	K_5	K_6	K_7
$1000(h)$	$0000(f)$ $0001(g)$	$0010(g)$	$0001(h)$ $0101(h)$	$1100(f)$ $1110(f)$	$0111(f,g)$ $1111(h)$	$1001(f,g)$ $1011(f)$

b-тесты

	K_1	K_2	K_3	K_4	K_5	K_6	K_7
a	$0000(h)$	$1000(f,g)$	$1010(g)$		$0100(f)$ $0101(f)$ $0110(f)$		$0101(f,g)$ $0011(g)$
b	$1100(h)$	$0100(f,g)$ $0101(f,g)$			$1000(f)$ $1010(f)$	$0011(g,h)$	
c	$1010(h)$	$0011(g)$	$0100(g)$	$0011(h)$		$0101(f,g)$	
d			$0011(g)$	$0000(h)$ $0100(h)$ $1100(h)$		$0110(f,h)$	$1000(f,g)$ $1010(f,g)$

Выделим сначала те неисправности, для которых существует единственный тест.

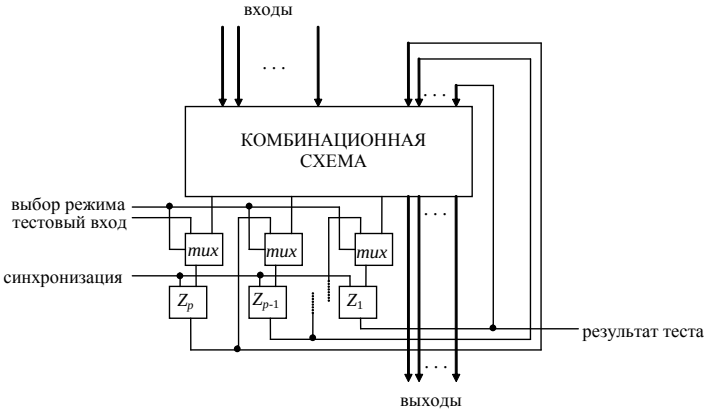
1000	0010	0000	1100	1010	0100	0011	0101	0110
K_1 $K_2(a)$	K_3	$K_1(a)$ K_2 $K_4(d)$	$K_1(b)$ K_5	$K_1(c)$ $K_3(a)$ $K_5(b)$ $K_7(d)$	$K_3(c)$ $K_5(a)$ $K_2(b)$	$K_3(d)$ $K_4(c)$ $K_7(a)$ $K_2(c)$ $K_6(b)$	$K_6(c)$ K_4	$K_6(d)$

Внесем эти тесты в кратчайший полный тест и удалим обнаруживаемые ими неисправности из списка. Не обнаружены неисправности K_6 и K_7, для их обнаружения достаточно добавить в кратчайший тест, например, наборы 0111 и 1001.

Кратчайший полный тест состоит из 11 наборов (вместо 16 в полном тесте).

6.5. Тестирование последовательностных схем

Тестирование последовательностных схем связано с большими вычислительными затратами. Тестовые последовательности для таких схем находятся, как правило, методами моделирования. Регулярные методы синтеза тестовых последовательностей связаны с анализом комбинационных схем (комбинационных эквивалентов длины l, где l – длина тестовой последовательности) и практически редко используется. Для тестирования последовательностных схем можно использовать *метод сканирования*, сводящий задачу тестирования последовательностной схемы к задаче тестирования комбинационной схемы. Общая конфигурация схемы для метода сканирования показана на рисунке.



При методе сканирования элементы памяти (D-триггеры) заменяются на D-триггеры с мультиплексором (*mux*) на входе. Выбор входа данных определяется управляющим сигналом «выбор режима». При подаче на вход «выбор режима» нуля реализуется нормальная работа схемы. В режиме тестирования схемы вход «выбор режима» устанавливается в единицу, и все триггеры образуют сдвиговый регистр с входом «тестовый вход» и выходом «результат тес-

та». Благодаря этому внутренние состояния последовательностной схемы становятся управляемыми и наблюдаемыми. Задача построения теста для сложной последовательностной схемы с произвольной глубиной памяти (числом D-триггеров) превращается в задачу синтеза теста для комбинационной схемы.

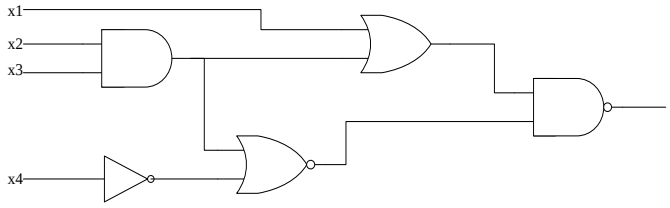
Одним из недостатков метода сканирования является большое время тестирования: тестовые данные должны «вдвигаться» в регистр последовательно. В течение одного такта заносится один бит информации. Альтернативой к методу сканирования является метод частичного сканирования, при котором в сдвиговый регистр включается только подмножество D-триггеров. Если, например, только некоторые неисправности не обнаружимы случайно генерируемой тестовой последовательностью, тогда частичное сканирование может быть направлено на обнаружение только этих неисправностей. Частичное сканирование позволяет уменьшить аппаратурную избыточность и время тестирования, по сравнению с обычным методом сканирования.

Другая модификация метода сканирования – метод параллельного сканирования. При этом методе элементы памяти схемы образуют несколько меньших регистров сдвига, которые имеют свои индивидуальные входы для «тестового входа» и выходы для «результатов теста». Обычно метод сканирования ориентирован на одиночные константные неисправности на полюсах элементов последовательностной схемы. Однако при использовании факторизационных методов синтеза метод может использоваться и для кратных константных неисправностей.

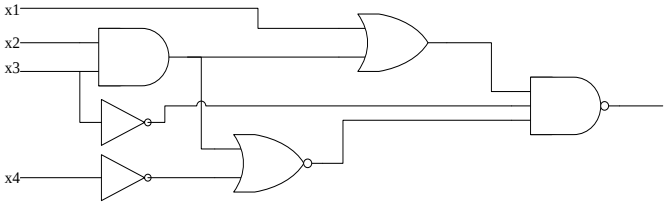
6.6. Задачи

Задача 1. Найти тесты для одиночных константных неисправностей методом критических путей, перечислить неисправности, которые они обнаруживают. Построить кратчайший полный тест.

а)

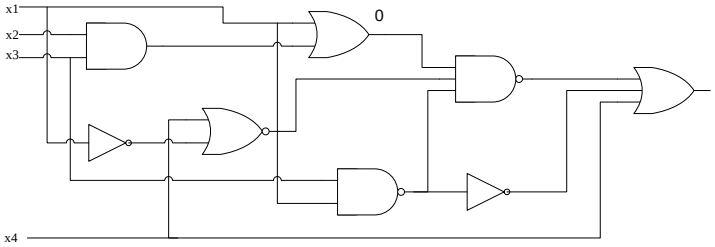


b)

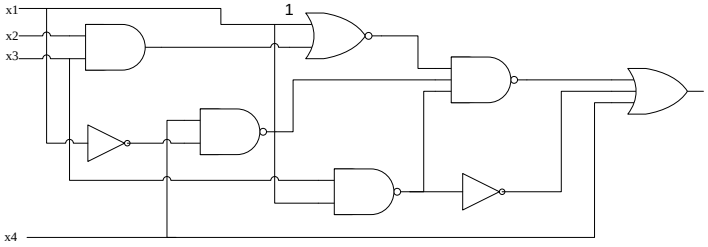


Задача 2. Найти тесты для выделенной неисправности методом активизации одномерного пути или показать, что это невозможно.

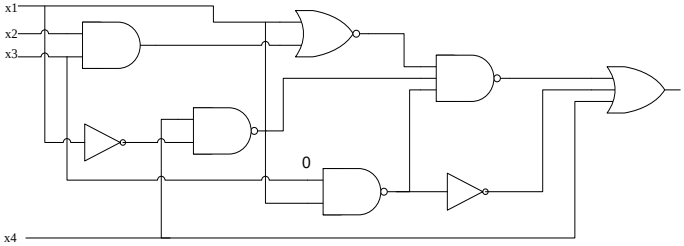
a)



b)



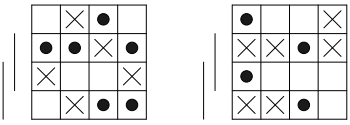
c)



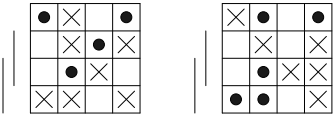
Задача 3. Найти тесты для неисправностей из задачи 2 методом различающей функции или показать, что это невозможно.

Задача 4. Получить для системы булевых функций безызбыточную систему ДНФ и построить кратчайший полный тест.

a)



b)



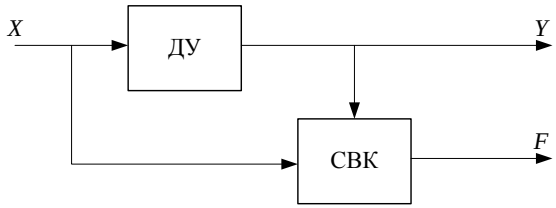
7. Самопроверяемые цифровые устройства

В предыдущей главе рассматривались методы построения тестов для цифровых устройств. При этом предполагалось, что обнаружение неисправностей производится в специальном тестовом режиме. Такой метод носит название *метода тестового диагностирования* (*off-line*). Однако, этот метод не ориентирован на обнаружение перемежающихся и неустойчивых неисправностей.

Метод функционального диагностирования (*on-line*) – метод, при котором наличие в схеме неисправностей некоторого заданного класса оценивается в процессе нормального функционирования схемы. При реализации данного метода обычно состояния выходов схемы представляются специальными кодовыми словами, обнаруживающими проявление неисправностей из рассматриваемого класса. Это достигается введением дополнительных выходов. К выходам схемы подключается дополнительная схема встроенного контроля (детектор кодов), которая выясняет принадлежность производимых проверяемой схемой выходных слов к кодовым словам. Ошибка обнаруживается, если производимое схемой выходное слово не является кодовым словом.

Оба метода требуют дополнительного оборудования (детекторы, генераторы тестовых наборов, анализаторы реакций и т.д.). Схемы, в которых дополнительное оборудование размещено на том же чипе, где располагается проверяемая схема, будем называть *самоконтролируемыми*. Самоконтролируемые схемы, реализованные с использованием методов тестового диагностирования, будем называть *самотестируемыми*, при реализации функционального диагностирования – *самопроверяемыми* схемами.

Самопроверяемая схема имеет следующий вид.



Здесь ДУ – дискретное устройство, СВК – схема встроенного контроля. СВК анализирует входы X и выходы Y схемы и выдает сообщение об ошибке на выходе F.

Будем называть дискретное устройство *полностью самопроверяемым*, если любая его неисправность из заданного класса обнаруживается СВК в момент первого ее проявления на выходах Y устройства. Дискретное устройство будем называть *частично самопроверяемым*, если любая его неисправность из заданного класса обнаруживается схемой встроенного контроля в рабочей области ДУ, но не обязательно в момент первого ее проявления на выходах схемы.

Если СВК обнаруживает неисправности не только ДУ (контролируемого объекта), но и свои собственные, то такие схемы называются *самопроверяемыми СВК* (ССВК). Самопроверяемые СВК устраняют проблему «сторожа над сторожем», то есть позволяют обнаружить неисправности в СВК без использования дополнительного оборудования.

7.1. Синтез схем встроенного контроля для комбинационных устройств

7.1.1. Постановка задачи

Пусть рассматривается комбинационное устройство с входами $X = \{x_1,...,x_n\}$ и выходами $Y = \{y_1,...,y_m\}$, и его таблица истинности T. Задача состоит в том, чтобы получить формальное задание на синтез СВК комбинационного устройства в стандартной форме. СВК должна обнаруживать любые одиночные константные неисправности комбинационного устройства, нарушающие его правильное функционирование.

Тривиальное решение этой задачи может быть следующим. Перечислим все возможные одиночные неисправности $P = \{p_1,...,p_k\}$. Построим таблицы истинности T_i функций выходов устройства Y_i в присутствии неисправности p_i. Объединим таблицы T и $T_1,...,T_k$ в одну таблицу истинности (все столбцы будут теперь столбцами аргументов) функции $f(x_1,...,x_n,y_1,...,y_m)$ по следующему правилу: если на входном наборе $X = \{\alpha_1,...,\alpha_n\}$ выходы устройства $Y = \{\beta_1,...,\beta_m\}$ отличаются от предписанных выходов $\{y_1,...,y_m\}$ комбинационного устройства из таблицы истинности T, то $f(\alpha_1,...,\alpha_n,\beta_1,...,\beta_m) = 0$, иначе $f(\alpha_1,...,\alpha_n,\beta_1,...,\beta_m) = 1$. Ясно, что этот метод является весьма трудоемким. Поэтому далее рассмотрим алгоритм с меньшей вычислительной сложностью.

7.1.2. Построение обобщенного графа

Цель построения обобщенного графа – разбиение комбинационной схемы на максимальные одновыходные подсхемы. Рассмотрение всех неисправностей системы заменяется рассмотрением неисправностей на выходах этих подсхем.

Алгоритм построения обобщенного графа

Начало. Имеется граф комбинационной схемы, заданный матрицей смежности. Добавим к ней еще один столбец, в котором проставим единицы в строках, соответствующих элементам, входы которых отождествлены с выходами схемы. Требуется построить обобщенный граф схемы.

Шаг 1. Отметим в матрице смежности строку, содержащую единицу в последнем столбце.

Шаг 2. Отметим столбцы, имеющие номера отмеченных строк.

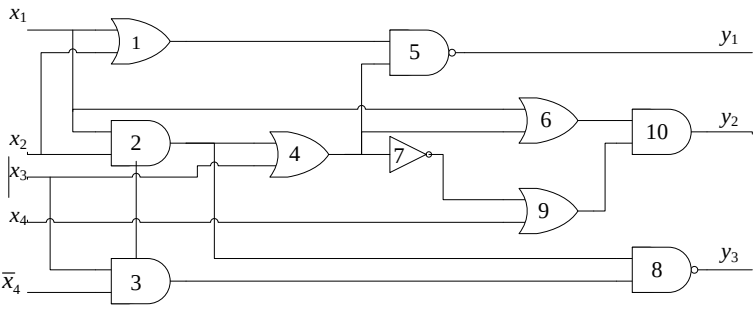
Шаг 3. Отметим строки, содержащие единицы *только* в отмеченных столбцах. Если такие строки были, идем на шаг 2.

Шаг 4. Объединяем номера отмеченных строк в одно подмножество, подчеркнем номер первой выбранной строки. Удаляем вычеркнутые строки и столбцы из матрицы. Если при вычеркивании столбцов вычеркивается хотя бы одна единица в неотмеченной строке, то проставляем в этой строке единицу в последнем столбце. Если матрица не пуста, идем на шаг 1.

Конец. Каждому подмножеству сопоставляем вершину обобщенного графа. Проводим дугу из вершины u в вершину v, если есть хоть одна дуга, соединяющая вершину из подмножества u с вершиной из подмножества v.

Пример.

Рассмотрим комбинационную схему и ее матрицу смежности.



Матрица смежности схемы имеет вид

	1	2	3	4	5	6	7	8	9	10	Выход	
1					1							*
2				1				1				
3								1				
4					1	1	1					
5											1	*
6										1		
7									1			
8											1	
9										1		
10											1	
	*				*							

Шаг 1. Отмечаем строку 5.

Шаг 2. Отмечаем столбец 5.

Шаг 3. Отмечаем строку 1, идем на шаг 2.

Шаг 2. Отмечаем столбец 1.

Шаг 3. Строк, содержащих единицы только в отмеченных столбцах, больше нет.

Шаг 4. Вычеркиваем столбцы и строки с номерами 1 и 5, проставляем единицу в столбце выходов строки 4. Получаем множество $A = \{1,5\}$. Матрица не пуста, идем на шаг 1.

	2	3	4	6	7	8	9	10	Выход	
2			1			1				
3						1				
4				1	1				1	*
6								1		
7							1			
8									1	
9								1		
10									1	
			*							

Шаг 1. Отмечаем строку 4.

Шаг 2. Отмечаем столбец 4.

Шаг 3. Строк, содержащих единицы только в отмеченных столбцах, нет.

Шаг 4. Вычеркиваем столбец и строку с номером 4, проставляем единицу в столбце выходов строки 2. Получаем множество $B = \{4\}$. Матрица не пуста, идем на шаг 1.

	2	3	6	7	8	9	10	Выход	
2					1			1	
3					1				
6							1		*
7						1			*
8								1	
9							1		*
10								1	*
			*	*		*	*		

Шаг 1. Отмечаем строку 10.

Шаг 2. Отмечаем столбец 10.

Шаг 3. Отмечаем строки 6 и 9, идем на шаг 2.

Шаг 2. Отмечаем столбцы 6 и 9.

Шаг 3. Отмечаем строку 7, идем на шаг 2.

Шаг 2. Отмечаем столбец 7.

Шаг 3. Строк, содержащих единицы только в отмеченных столбцах, нет.

Шаг 4. Вычеркиваем столбцы и строки 6, 7, 9, 10. Получаем множество $C = \{6,7,9,10\}$. Матрица не пуста, идем на шаг 1.

	2	3	8	Выход	
2			1	1	*
3			1		
8				1	
	*				

Шаг 1. Отмечаем строку 2.

Шаг 2. Отмечаем столбец 2.

Шаг 3. Строк, содержащих единицы только в отмеченных столбцах, нет.

Шаг 4. Вычеркиваем столбец и строку 2. Получаем множество $D = \{2\}$. Матрица не пуста, идем на шаг 1.

	3	8	Выход	
3		1		*
8			1	*
	*	*		

Шаг 1. Отмечаем строку 8.

Шаг 2. Отмечаем столбец 8.

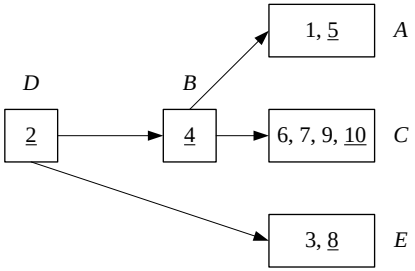
Шаг 3. Отмечаем строку 3, идем на шаг 2.

Шаг 2. Отмечаем столбец 3.

Шаг 3. Строк, содержащих единицы только в отмеченных столбцах, нет.

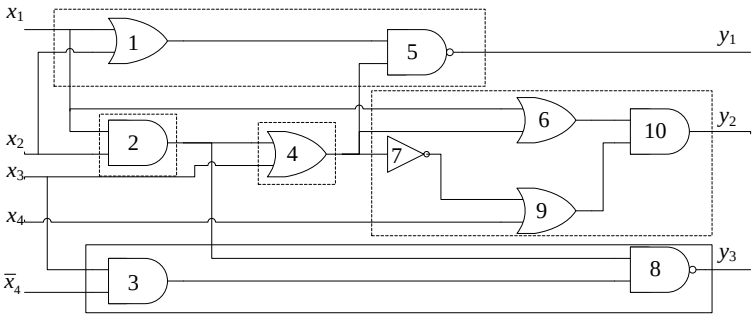
Шаг 4. Вычеркиваем столбцы и строки 3 и 8. Получаем множество $E = \{3,8\}$. Матрица пуста.

Конец. Обобщенный граф имеет вид



Здесь подчеркнуты элементы, выходы которых являются выходами подсхем.

На схеме максимальные одновыходные подсхемы обведены линией.



7.1.3. Построение общей таблицы истинности

Процедура построения обобщенного графа дает разбиение схемы устройства на максимальные одновыходные подсхемы. Любая существенная неисправность схемы (подсхемы) приводит к изменению значения выхода схемы (подсхемы) по крайней мере на одном наборе. С другой стороны, для любого набора найдется логическая неисправность, меняющая на этом наборе значение выхода на инверсное. Значит, можно отказаться от рассмотрения всех неисправностей схемы, и ограничиться рассмотрением одиночных неисправностей на выходах максимальных одновыходных подсхем. Каждая такая неисправность состоит в инвертировании выхода подсхемы.

Таким образом, если число вершин обобщенного графа, а значит, максимальных одновыходных подсхем, равно k, то для задания на синтез СВК необходимо объединить таблицу истинности исправного устройства с k таблицами истинности устройства в присутствии неисправности. Функция выхода СВК $f(x_1,\ldots,x_n,y_1,\ldots,y_m)$ строится, как описано в общем методе.

Построенная по полученной таблице истинности СВК будет обнаруживать все существенные одиночные неисправности всех элементов схемы устройства, а также кратные неисправ ности элементов из одной и той же подсхемы.

Пример. Рассмотрим схему из предыдущего примера. Пусть функционирование устройства задано на шести входных наборах. Символом * отмечены повторяющиеся строки значений аргументов, на этих наборах функцию $f(\cdot)$ определять не нужно.

x_1	x_2	x_3	x_4	y_1	y_2	y_3	f	f_1	f_2	неисправность
0	1	1	0	0	1	1	1	1	1	нет
1	1	1	0	0	1	0	1	1	1	нет
1	0	0	1	1	0	1	1	1	1	нет
0	1	0	1	1	1	1	1	1	1	нет
1	1	0	1	0	0	1	1	1	1	нет
0	0	1	1	1	0	1	1	1	1	нет
0	1	1	0	0	1	0	0	1	0	инверсия на выходе 2
1	1	1	0	0	1	1	0	1	0	инверсия на выходе 2
1	0	0	1	0	0	1	0	0	1	инверсия на выходе 2
0	1	0	1	0	0	1	0	0	1	инверсия на выходе 2
1	1	0	1	1	0	1	0	0	1	инверсия на выходе 2

*	0	0	1	1	1	0	1			инверсия на выходе 2
	0	1	1	0	1	1	1	0	0	1	инверсия на выходе 4
	1	1	1	0	0	0	0	0	0	1	инверсия на выходе 4
*	1	0	0	1	0	0	1			инверсия на выходе 4
*	0	1	0	1	0	0	1			инверсия на выходе 4
*	1	1	0	1	1	0	1			инверсия на выходе 4
	0	0	1	1	1	1	1	0	0	0	инверсия на выходе 4
*	0	1	1	0	1	1	1			инверсия на выходе 5
	1	1	1	0	1	1	0	0	0	1	инверсия на выходе 5
*	1	0	0	1	0	0	1			инверсия на выходе 5
	0	1	0	1	0	1	1	0	0	1	инверсия на выходе 5
*	1	1	0	1	1	0	1			инверсия на выходе 5
	0	0	1	1	0	0	1	0	0	1	инверсия на выходе 5
*	0	1	1	0	0	1	0			инверсия на выходе 8
*	1	1	1	0	0	1	1			инверсия на выходе 8
	1	0	0	1	1	0	0	0	1	0	инверсия на выходе 8
	0	1	0	1	1	1	0	0	1	0	инверсия на выходе 8
	1	1	0	1	0	0	0	0	1	0	инверсия на выходе 8
	0	0	1	1	1	0	0	0	1	0	инверсия на выходе 8
	0	1	1	0	0	0	1	0	0	0	инверсия на выходе 10
	1	1	1	0	0	0	0	0	0	0	инверсия на выходе 10
	1	0	0	1	1	1	1	0	0	0	инверсия на выходе 10
	0	1	0	1	1	0	1	0	0	0	инверсия на выходе 10
	1	1	0	1	0	1	1	0	0	0	инверсия на выходе 10
*	0	0	1	1	1	0	1			инверсия на выходе 10

Если же требуется не только определить, исправно ли устройство, но и осуществить поиск неисправности, можно разбить множество строк таблицы не на два класса: исправное и неисправное устройство (соответственно $f(\cdot)=1$ и $f(\cdot)=0$), а на несколько классов: исправное устройство и устройство с неисправностями в различных подсхемах. Для этого потребуется СВК с несколькими выходами.

Пример. Для схемы из предыдущего примера, в частности, можно построить СВК с выходами $f_1(\cdot)$, $f_2(\cdot)$, где набор значений 11 соответствует исправной схеме, 10 – неисправностям подсхем D или E, 01 – неисправностям подсхем A, B или D, 00 – неисправностям подсхем B или C.

x_1	x_2	x_3	x_4	y_1	y_2	y_3	f_1	f_2	неисправность
0	1	1	0	0	1	1	1	1	нет
1	1	1	0	0	1	0	1	1	нет
1	0	0	1	1	0	1	1	1	нет
0	1	0	1	1	1	1	1	1	нет
1	1	0	1	0	0	1	1	1	нет
0	0	1	1	1	0	1	1	1	нет
0	1	1	0	0	1	0	1	0	инверсия на выходе 2
1	1	1	0	0	1	1	1	0	инверсия на выходе 2
1	0	0	1	0	0	1	0	1	инверсия на выходе 2
0	1	0	1	0	0	1	0	1	инверсия на выходе 2
1	1	0	1	1	0	1	0	1	инверсия на выходе 2
0	1	1	0	1	1	1	0	1	инверсия на выходе 4
1	1	1	0	0	0	0	0	1	инверсия на выходе 4
0	0	1	1	1	1	1	0	0	инверсия на выходе 4
1	1	1	0	1	1	0	0	1	инверсия на выходе 5
0	1	0	1	0	1	1	0	1	инверсия на выходе 5
0	0	1	1	0	0	1	0	1	инверсия на выходе 5
1	0	0	1	1	0	0	1	0	инверсия на выходе 8
0	1	0	1	1	1	0	1	0	инверсия на выходе 8
1	1	0	1	0	0	0	1	0	инверсия на выходе 8
0	0	1	1	1	0	0	1	0	инверсия на выходе 8
0	1	1	0	0	0	1	0	0	инверсия на выходе 10
1	1	1	0	0	0	0	0	0	инверсия на выходе 10
1	0	0	1	1	1	1	0	0	инверсия на выходе 10
0	1	0	1	1	0	1	0	0	инверсия на выходе 10
1	1	0	1	0	1	1	0	0	инверсия на выходе 10

7.1.4. Синтез самопроверяемых СВК

Определение. СВК, которая обнаруживает все одиночные константные неисправности комбинационной схемы, а также все одиночные константные неисправности входов и выходов элементов СВК, называется *самопроверяемой СВК*.

Самопроверяемая СВК имеет два выхода: $f_1(\cdot)$ и $f_2(\cdot)$. При правильном функционировании схемы значения выходов должны быть инверсны (01 или 10), в присутствии неисправности – 00 или 11. При этом каждый набор значений функции должен встречаться хотя бы один раз.

При произвольной реализации СВК, имеющая два выхода, будет гарантированно обнаруживать только неисправности комбинационной схемы. Поэтому одним из условий построения самопроверяемой СВК является раздельная реализация функций $f_1(\cdot)$ и $f_2(\cdot)$. Обоснование этого требования состоит в следующем. Неисправность СВК будет обнаружена при изменении выходов с 01 или 10 на 00 или 11, то есть при инвертировании одного выхода. Раздельная реализация функций $f_1(\cdot)$ и $f_2(\cdot)$ гарантирует, что любая одиночная константная неисправность на входах и выходах элементов СВК не приведет к инвертированию обоих выходов схемы.

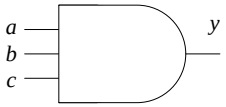
Рассмотрим условия, при выполнении которых возможно построение самопроверяемой СВК по общей таблице истинности.

Как уже отмечалось, все наборы значений функций должны присутствовать в общей таблице истинности. Однако от порядка приписывания выходам различных пар значений зависит как сложность СВК, так и возможность синтеза самопроверяемой СВК.

После заполнения общей таблицы истинности необходимо определить существенные переменые для функций $f_1(\cdot)$ и $f_2(\cdot)$ (заметим, что эти функции обычно не полностью определенные). Чем меньше существенных переменных, тем меньше сложность СВК.

После выявления существенных переменных можно приступать к реализации функций $f_1(\cdot)$ и $f_2(\cdot)$ как функций, зависящих только от существенных переменных. Представим функции их безызбыточными ДНФ и используем факторизационный метод синтеза. Было показано, что в этом случае для любой константной неисправности существует тест из рабочей области устройства. Для обеспечения самопроверяемости к тому же требуется, чтобы все тесты принадлежали подтаблице правильных значений T.

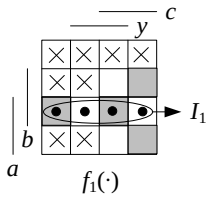
Пример. Построим самопроверяемую СВК для схемы



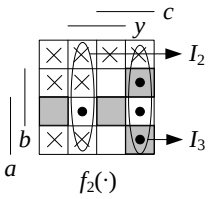
Пусть рабочая область схемы имеет вид $\{111,011,101,110\}$. Как легко убедиться, этот набор является полным тестом для одиночных константных неисправностей схемы. Тогда задание на синтез СВК может иметь, например, следующий вид

a	b	c	y	f_1	f_2	неисправности
0	1	1	0	0	1	нет
1	0	1	0	0	1	нет
1	1	0	0	1	0	нет
1	1	1	1	1	0	нет
0	1	1	1	0	0	$a=1$
1	0	1	1	0	0	$b=1$
1	1	0	1	1	1	$c=1$
1	1	1	0	1	1	$a=0, b=0, c=0$

Построим матрицы Грея функций $f_1(\cdot)$ и $f_2(\cdot)$ и получим для них безызбыточные ДНФ. Серым цветом здесь выделены наборы из подтаблицы правильных значений. Ниже приведена схема дискретного устройства с СВК.

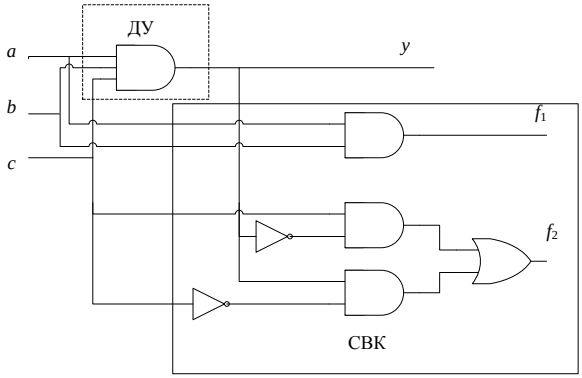


$f_1(\cdot)$



$f_2(\cdot)$

$$D=\begin{cases}\text{ДНФ}_1 = a\,b \\ \qquad\quad K_1 \end{cases};\qquad \begin{cases}\text{ДНФ}_2 = \bar{c}\,y \vee c\,\bar{y} \\ \qquad\quad K_2 \quad K_3 \end{cases}.$$



Теперь проверим СВК на самопроверяемость. Все тесты лежат в рабочей области детектора, но конъюнкция K_2 имеет лишь один a-тест 1101, который не принадлежит подтаблице правильных значений. Таким образом, исчезновение конъюнкции K_2 не будет обнаружено при условии правильной работы схемы. Далее, в случае появления неисправности $c=1$, выходы детектора примут значение 10 вместо предписанного по заданию на синтез 11, и неисправность $c=1$ не будет обнаружена.

Таким образом, СВК не является самопроверяемой для рассмотренной схемы. При этом схема очень проста – состоит из одного конъюнктора, а сложность СВК гораздо больше.

Данный пример демонстрирует трудности, с которыми может столкнуться разработчик при проектировании самопроверяемых схем встроенного контроля для готовых схем. Поэтому в дальнейших разделах мы рассмотрим методы синтеза комбинационных схем, обладающих свойством самопроверяемости.

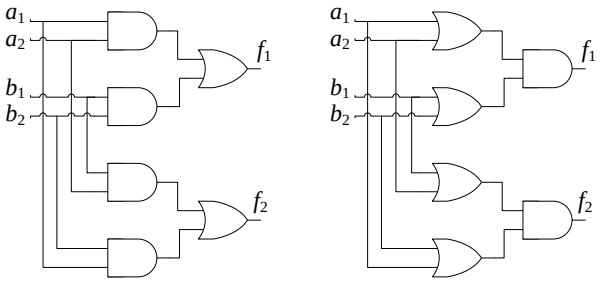
7.2. Синтез самопроверяемых комбинационных схем

7.2.1. Метод дублирования

Известно, что самым простым по реализации и часто самым дорогим по аппаратурным затратам является метод дублирования. В этом случае контролируемое дискретное устройство дополняется аналогичным устройством, выполняющим функции контролирующего. На выходах контролируемого и контролирующего дискретных устройств ставится схема сравнения. Этот метод обнаруживает любые логические неисправности, в том числе и кратные, если они проявляются в одной из двух схем. Недостатком метода является то, что расходы на обнаружение ошибок превышают первоначальные затраты на саму схему более чем в 2 раза. При методе дублирования не обнаруживаются неисправности на входных полюсах схемы. Все модификации приводят лишь к незначительному сокращению оборудования по сравнению с непосредственным дублированием.

Пример. Пусть устройство имеет два выхода a_1, a_2, тогда контролирующее устройство также будет иметь два выхода b_1, b_2. При правильном функционировании обоих устройств выходы контролирующего устройства инверсны выходам основного. Выходы устройств отождествляются с входами самопроверяемой схема сравне-

ния одного из двух типов.

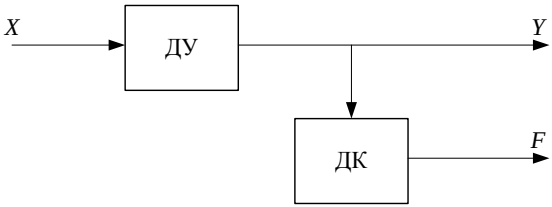


Выходы f_1, f_2 схемы сравнения принимают значения 01 или 10, если и только если $b_1 = \overline{a}_1$, $b_2 = \overline{a}_2$. Все одиночные константные неисправности схемы обнаруживаются на рабочей области схемы – множестве $T = \{1100, 0011, 1001, 0110\}$, которое является кратчайшим полным тестом. Значит, схема сравнения будет самопроверяемой, если на ее входы попадают все возможные наборы из рабочей области, или, что то же самое, выходы устройства принимают все возможные значения.

7.2.2. Неупорядоченные коды

Особый подход к построению самопроверяемых схем основан на кодировании выходов наблюдаемого дискретного устройства специальным кодом. Подключенный к ДУ детектор кодов (ДК) проверяет, принадлежит ли выходное слово ДУ к этому коду.

В этом случае общая схема самопроверяемого устройства имеет следующий вид.



Заметим, что входами детектора являются только выходы устройства, в отличие от схемы, рассмотренной в разделе 7.1, где входами СВК являлись также входы устройства. Необходимость в наблюдении за входами пропадает за счет использования специальных мето-

дов кодирования выходов, которое, в свою очередь, может привести к необходимости введения дополнительных выходов устройства.

Рассмотрим два булевых вектора: $\begin{matrix}\alpha = a_1 \ldots a_n \\ \beta = b_1 \ldots b_n\end{matrix}$.

Определение. Говорят, что вектор α *предшествует* вектору β, или вектор β *следует* за вектором α (обозначается $\alpha \prec \beta$), если $\forall i = \overline{1,n} : a_i \le b_i$.

Определение. Если $\alpha \prec \beta$ или $\beta \prec \alpha$, вектора α и β называются *сравнимыми*, иначе они называются *несравнимыми*.

Пример. Вектора $\begin{matrix}\alpha = 10010 \\ \beta = 11011\end{matrix}$ сравнимы ($\alpha \prec \beta$), вектора $\begin{matrix}\alpha = 10010 \\ \beta = 01011\end{matrix}$ несравнимы.

Определение. Код называется *неупорядоченным*, если любые два слова в нем несравнимы.

Неупорядоченные коды можно разделить на два типа:

- *разделимые* (систематические);
- *неразделимые* (не систематические).

Определение. Код называется *разделимым*, если разряды кодовых слов могут быть разделены на две группы: информационные и контрольные. Если такого разделения сделать нельзя, то код называется *неразделимым*.

Рассмотрим несколько основных типов разделимых кодов.

Код с проверкой на нечетность (*четность*). Данный код обладает минимальной избыточностью, так как для его построения требуется только один дополнительный разряд, который служит для обеспечения нечетности (четности) числа единиц в кодовом слове.

Двухпроводный код. В двухпроводном коде каждый информационный разряд дополняется контрольным разрядом, причем значение в информационном разряде противоположно значению в контрольном разряде. Кодовыми словами являются слова, у которых значения в информационном и контрольном разрядах противоположные.

В *коде Бергера* контрольная часть для кодового слова является двоичным представлением числа нулей в информационной части. Таким образом, если число информационных разрядов равно k, то число контрольных разрядов равно $\log_2(k+1)$.

Пример. Рассмотрим код Бергера для двух информационных разрядов. В этом случае число контрольных разрядов равно двум. Здесь x_1, x_2 – информационные разряды, z_1, z_2 – контрольные.

x_1	x_2	z_1	z_2
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	0

Среди неразделимых кодов рассмотрим один, наиболее используемый класс кодов, *равновесные коды* (или (m,n) -коды). Кодовое слово равновесного кода содержит m единиц в n разрядах ($m < n$).

Пример. Рассмотрим (2,5)-код

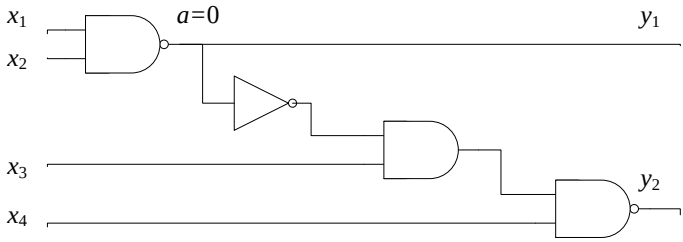
1	1	0	0	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1
0	1	1	0	0
0	1	0	1	0
0	1	0	0	1
0	0	1	1	0
0	0	1	0	1
0	0	0	1	1

7.2.3. Однонаправленное проявление неисправностей

При построении самопроверяемой схемы с использованием кодирования выходов можно обеспечить обнаружение тех неисправностей, которые приводят к ошибке на выходе, меняющей свойство данного кода. Для этого необходимо развивать специальные методы структурного синтеза дискретных устройств.

Определение. *Однонаправленным проявлением неисправностей* называется замена в значениях выходов либо нулей на единицы, либо единиц на нули, но не одновременно.

Пример не однонаправленного проявления неисправностей. На наборе 0011 выходы исправной схемы равны 10, в присутствии неисправности $a = 0$ выходы схемы принимают значения 01, т.е. в присутствии неисправности на одном выходе 1 меняется на 0, на другом – 0 на 1.



Проблемы такого рода возникают из-за ветвления и инверторов, то есть когда на одном пути от неисправности до выхода имеется четное число инверторов, на другом – нечетное.

Теорема. Код обнаруживает все однонаправленные неисправности тогда и только тогда, когда он является неупорядоченным.

Доказательство.

Достаточность. Рассмотрим неупорядоченный код. Пусть в присутствии однонаправленной неисправности кодовое слово α преобразуется в слово β. Поскольку неисправность однонаправленная, при этом либо некоторые единицы в слове α заменятся на нули ($\beta \prec \alpha$), либо нули заменятся на единицы ($\alpha \prec \beta$), т.е. α и β сравнимы, значит, β – не кодовое слово.

Необходимость. Пусть код обнаруживает все однонаправленные неисправности, и существуют два кодовых слова α и β такие, что $\alpha \prec \beta$. Тогда слово β получается из α заменой некоторых нулей на единицы, а значит, код обнаруживает не все однонаправленные неисправности. Следовательно, предположение о сравнимости кодовых слов было неверным, и любые два кодовых слова несравнимы.■

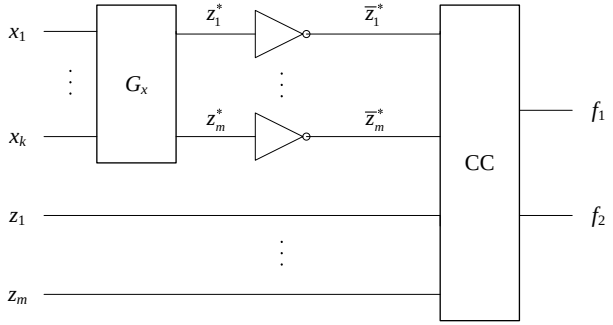
Из теоремы следует общая идея синтеза самопроверяемого устройства: выходы кодируются неупорядоченными кодами, и схема синтезируется таким образом, чтобы обеспечить однонаправленное проявление неисправностей. Кодовые слова на выходах схемы обеспечиваются введением дополнительных выходных переменных.

Для обеспечения однонаправленного проявления одиночных константных неисправностей может использоваться, например, факторизационный метод синтеза схем или использование древовидных схем (без ветвления). Это нужно учитывать при синтезе самопроверяемых СВК по таблице истинности.

7.2.4. Самопроверяемые детекторы кода Бергера

Для начала рассмотрим общую схему детектора разделимого кода. Пусть кодовое слово состоит из двух частей: $I = \{x_1,...,x_k\}$ – информационная часть, $P = \{z_1,...,z_m\}$ – контрольная часть.

Здесь схема G_x осуществляет перекодирование информационных разрядов в контрольные. При условии исправности этой схемы верно $z_1^* = z_1,...,z_m^* = z_m$. Схема СС – это самопроверяемая схема сравнения, выходы которой принимают значения 01 или 10, если ее входы инверсны: $\overline{z}_1^* = \overline{z}_1,...,\overline{z}_m^* = \overline{z}_m$. Схема сравнения с четырьмя входами была рассмотрена в подразделе 7.2.1. Схемы сравнения на большее количество входов синтезируются на основе схем сравнения на 4 входа.



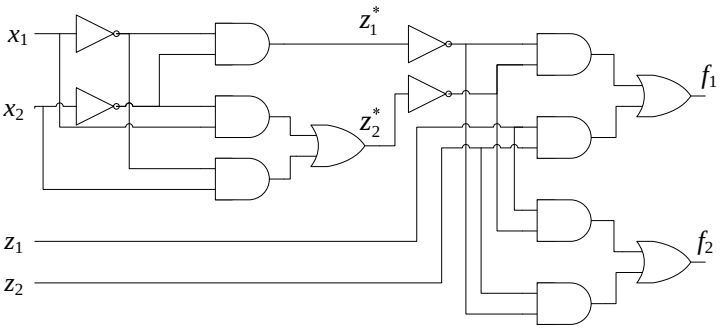
Пример. Рассмотрим детектор кода Бергера для двух информационных разрядов. Код имеет вид

x_1	x_2	z_1	z_2
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	0

Отсюда получаем оператор схемы G_x:

$$\begin{cases} z_1 = \overline{x}_1\overline{x}_2; \\ z_2 = \overline{x}_1 x_2 \vee x_1\overline{x}_2. \end{cases}$$

Детектор имеет вид



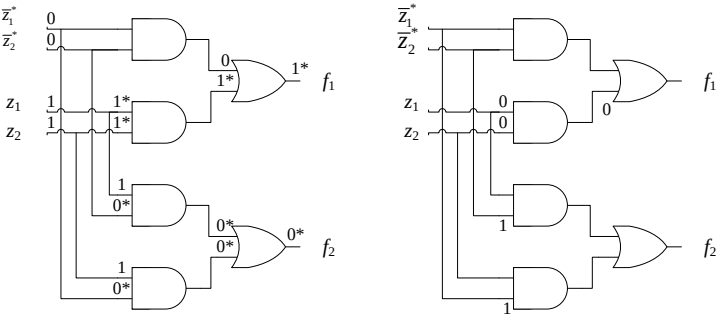
Определение. Разделимый код называется *полным*, если все возможные кодовых слов появляются в его m контрольных разрядах, иначе код называется *неполным*.

Пример. Код Бергера с двумя информационными разрядами является неполным, т.к. в контрольных разрядах не появляется слово 11. Примером полного кода может служить код Бергера с тремя информационными разрядами.

Теорема. Детектор разделимого кода является самопроверяемым, если код является полным.

Доказательство. Если разделимый код является полным, то в контрольных разрядах появляются все возможные наборы значений. Множество этих наборов является кратчайшим полным тестом для схемы сравнения, следовательно, схема сравнения является самопроверяемой. ■

Пример. Рассмотрим схему детектора кода Бергера с двумя информационными разрядами. В него входит схема сравнения на 4 входа.



Число нулей в информационных разрядах кода Бергера меняется от 0 до 2, т.е. в контрольных разрядах не появляется набор 11. Следовательно, на входы схемы сравнения не поступает набор 0011. На схеме слева отмечены критические значения на этом наборе, на схеме справа выделены неисправности, для которых этот набор является единственным тестом из рабочей области.

Определение. Код Бергера называется *кодом Бергера максимальной длины*, если число информационных разрядов равно $k = 2^m - 1$.

Пример. Код Бергера с двумя информационными разрядами не является кодом максимальной длины ($2 \neq 2^2 - 1$), а код Бергера с тремя информационными разрядами – является ($3 = 2^2 - 1$).

Лемма. Код Бергера является полным, если и только если он является кодом Бергера максимальной длины.

Доказательство. Длина кодового слова кода Бергера равна $n = k + \lceil \log_2 \ k+1 \ \rceil$, где $\lceil a \rceil$ – наименьшее целое число, такое что $\lceil a \rceil \geq a$. В k информационных разрядах слова число нулей изменяется от 0 до k. Каждый из $\lceil \log_2 \ k+1 \ \rceil$ наборов появится в контрольных разрядах тогда и только тогда, когда

$$2^{\lceil \log_2 \ k+1 \ \rceil} = k+1.$$

Это возможно если и только если $\lceil \log_2 \ k+1 \ \rceil = m$, где m – целое, т.е. если код Бергера есть код максимальной длины. ■

Отсюда следует, что рассмотренный выше детектор будет самопроверяемым только для кода Бергера максимальной длины. Примером самопроверяемого детектора может служить детектор кода Бергера с тремя информационными разрядами.

7.2.5. Самопроверяемые детекторы равновесного кода

В настоящее время разработаны самопроверяемые детекторы для $(m, 2m)$-кода. Другие равновесные коды преобразуются в $(m, 2m)$-код с помощью каскадного соединения самопроверяемых трансляторов.

Самопроверяемый детектор $(m, 2m)$-кода имеет два выхода, принимающих значения 01 или 10, при наличии ровно m единиц среди $2m$ входов и значения 11 или 00 при числе единиц, большем или меньшем m.

Разобьем множество входов детектора $X=\{x_1,...,x_{2m}\}$ на два подмножества с равным числом элементов в них: $C=\{c_1,...,c_m\}$ и $D=\{d_1,...,d_m\}$. Введем булеву функцию $T(m_d \geq i)$, равную единице в том и только в том случае, когда число единиц m_d на входах подмножества D больше или равно некоторого числа $i \in \{0,...,m\}$. Аналогично функция $T(m_c \geq m-i)$ равна единице в том и только в том случае, когда число единиц m_c на входах подмножества C больше или равно некоторого числа $m-i$. Функции $T(m_d \geq i)$ (и, аналогично, $T(m_c \geq m-i)$) могут быть заданы безызбыточными ДНФ вида

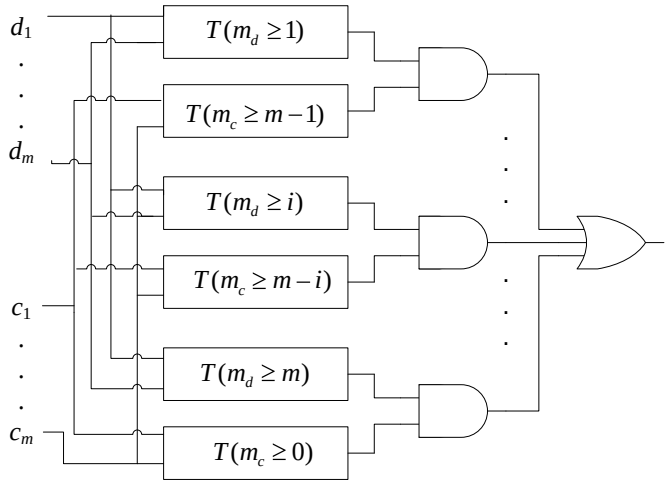
$$T(m_d \geq i) = \bigvee_{k1<k2<\ldots<ki} x_{k1}\ldots x_{ki}, \quad x_{k1},\ldots,x_{ki} \in D.$$

Тогда выходы детектора определяются следующим образом.

$$f_1 = \bigvee_{i \in N_1} T(m_d \geq i)T(m_c \geq m-i), \quad N_1 = 1,3,...,\begin{cases} m, & m-\text{нечетное;} \\ m-1, & m-\text{четное;} \end{cases}$$

$$f_2 = \bigvee_{i \in N_2} T(m_d \geq i)T(m_c \geq m-i), \quad N_2 = 0,2,...,\begin{cases} m, & m-\text{четное;} \\ m-1, & m-\text{нечетное.} \end{cases}$$

Возможная реализация схемы функции f_1 · для нечетного m показана на рисунке.



Функции $T(\cdot)$ представлены в виде соответственно обозначенных прямоугольников. Функция $f_1 \cdot$ для четного m и функция $f_2 \cdot$, могут быть синтезированы аналогично.

Так как любое кодовое слово содержит ровно m единиц, т.е. $m_d + m_c = m$, то $\exists i \in 0,\ldots,m : m_d = i, m_c = m - i$. Тогда при правильной работе детектора значение единицы примут одновременно входы лишь одного конъюнктора, а именно того, который реализует функцию $T(m_d \geq i)T(m_c \geq m - i)$. Для остальных конъюнкторов один из входов примет значение нуля, поскольку $\forall k > i : T(m_d \geq k) = 0$ и $\forall k < i : T(m_c \geq m - k) = 0$. Значит, выход ровно одного конъюнктора примет значение единицы, и, следовательно, либо $f_1 \cdot$, либо $f_2 \cdot$, но не обе вместе, будет равна единице.

Уменьшение числа единиц во входном слове приведет к тому, что выходы всех конъюнкторов примут значение нуля, поскольку $m_d + m_c < m$. Значит, оба выхода детектора примут значение нуля.

Увеличение числа единиц приведет к тому, что выходы как минимум двух конъюнкторов станут равны единице. Пусть $m_d + m_c = m + l$, где $l > 0$. Тогда $\exists i \in l,\ldots,m : m_d = i, m_c = m - i + l$, и выходы по крайней мере двух конъюнкторов примут значение единицы, поскольку

$$T(m_d \geq i)T(m_c \geq m - i) = 1;$$

$$T(m_d \geq i - 1)T(m_c \geq m - i + 1) = 1.$$

Пусть i – нечетное, тогда $i - 1$ четное, значит, выходы конъюнкторов, реализующих эти функции, отождествлены с входами разных дизъюнкторов, и оба выхода детектора примут значение единицы.

Покажем теперь, что детектор является самопроверяемым для одиночных константных неисправностей, если на входах подмножеств C и D появляются все 2^m возможных наборов.

Если неисправность возникает на входе детектора, то это хотя бы на одном наборе приведет к изменению числа единиц входного слова, а это означает, что выходы детектора примут значение 00 или 11, как было показано выше.

Пусть константная неисправность возникает на входе конъюнктора, реализующего функцию $T(m_d \geq i)T(m_c \geq m - i)$. Если это неисправность «константа 0», то выход конъюнктора на любом наборе

примет значение нуля. Тогда для любого входного слова, для которого $m_d = i, m_c = m - i$, выход конъюнктора примет значение нуля вместо единицы, как должно быть в отсутствие неисправности. Поскольку, как было показано выше, в исправном детекторе на любом кодовом слове выход ровно одного конъюнктора равен единице,то в присутствии неисправности выходы детектора примут значение 00, и неисправность обнаружится.

Рассмотрим неисправность «константа 1» на входе конъюнктора. Пусть этот вход отождествлен с выходом элемента, реализующего функцию $T(m_d \geq i)$. Тогда для любого входного слова, для которого $m_d = i-1, m_c = m-i+1$, выход конъюнктора примет значение единицы вместо нуля, и одновременно значение единицы примет конъюнктор, реализующий функцию $T(m_d \geq i-1)T(m_c \geq m-i+1)$. Выходы этих конъюнкторов отождествлены с входами разных дизъюнкторов, значит, выходы детектора примут значение 11.

Для константных неисправностей на входах дизъюнкторов можно провести аналогичные рассуждения.

Пусть константная неисправность возникает на входе элемента, который входит в подсхему, реализующую функцию $T(m_d \geq i)$. Функции заданы безызбыточными ДНФ, значит, при использовании факторизационного метода синтеза константные неисправности проявляются на функциональном уровне как исчезновение конъюнкции или исчезновение буквы из конъюнкции. Поскольку на входы подмножества D поступают все возможные наборы, для каждой такой неисправности есть тест, входящий в рабочую область детектора. Исчезновение конъюнкции означает, что на одном из кодовых слов, для которых $m_d = i, m_c = m - i$, выход конъюнктора примет значение нуля вместо единицы, значит, выходы детектора примут значение 00. Исчезновение буквы означает, что на одном из кодовых слов, для которых $m_d = i-1, m_c = m-i+1$, выход конъюнктора примет значение единицы вместо нуля, значит, выходы детектора примут значение 11.

Таким образом, все одиночные константные неисправности детектора обнаруживаются на множестве из 2^m кодовых слов, что существенно меньше, чем число C_{2m}^m всех возможных кодовых слов равновесного кода.

Пример. Рассмотрим (3,6)-код. Выделим подмножества пере-

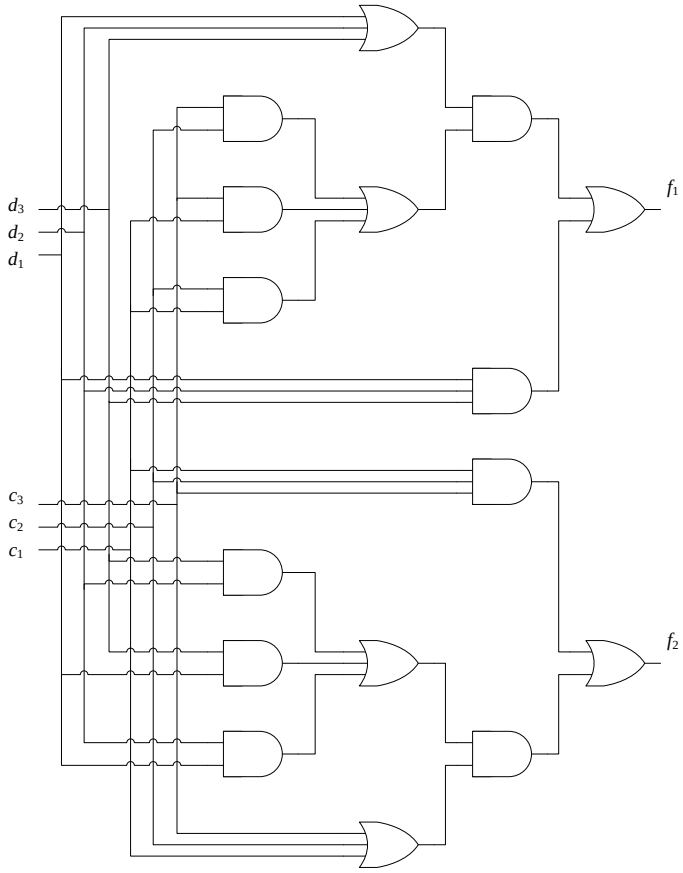
менных: $D = \{d_1 = x_1, d_2 = x_2, d_3 = x_3\}$, $C = \{c_1 = x_4, c_2 = x_5, c_3 = x_6\}$.
Тогда:

$$f_1 = T(m_d \geq 1)T(m_c \geq 2) \vee T(m_d \geq 3)T(m_c \geq 0);$$
$$f_2 = T(m_d \geq 0)T(m_c \geq 3) \vee T(m_d \geq 2)T(m_c \geq 1).$$

Подставляя выражения для функций $T(m_d \geq i)$ и $T(m_c \geq m-i)$, получаем:

$$f_1 = (d_1 \vee d_2 \vee d_3)(c_1c_2 \vee c_2c_3 \vee c_1c_3) \vee d_1d_2d_3;$$
$$f_2 = c_1c_2c_3 \vee (d_1d_2 \vee d_2d_3 \vee d_1d_3)(c_1 \vee c_2 \vee c_3).$$

Детектор имеет вид

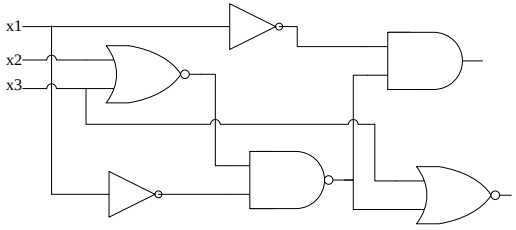


Для самопроверяемости детектора требуется, чтобы в процессе функционирования схемы на входы подмножеств C и D поступили все 8 возможных наборов. Значит, детектор является самопроверяемым для числа кодовых слов от 8 до 20.

Для построения самопроверяемых детекторов (m,n)-код транслируется сначала в $(1,C_n^m)$-код, а затем в подходящий $(k,2k)$-код. В частности, детектор (3,6)-кода можно использовать при $8 \le C_n^m \le 20$; например, (2,5)-код ($C_5^2 = 10$) можно транслировать в (1,10)-код, а затем в (3,6)-код.

7.3. Задачи

Задача 1. Построить обобщенный граф схемы и составить задание на синтез для ее схемы встроенного контроля. Рабочая область схемы {000,011,101,111}. Возможно ли построение самопроверяемой схемы встроенного контроля?



Задача 2. На основе схем сравнения на 4 входа синтезировать схемы сравнения на 6 и 8 входов. При каких условиях они будут самопроверяемыми?

Задача 3. Составить задание на синтез детектора (4,8)-кода. Сформулировать условия самопроверяемости детектора.

Задача 4. Синтезировать детекторы кода Бергера с тремя и четырьмя информационными разрядами. Являются ли они полностью самопроверяемым? Если нет, то какие неисправности детекторов не обнаруживаются?

Литература

1. Агибалов Г.П., Оранов А.М. Лекции по теории конечных автоматов. – Томск: Изд-во Том. Ун-та, 1985. — 120 с.
2. Баранов С.И. Синтез микропрограммных автоматов. – 2-е изд. – Л.: Энергия, 1979. – 232 с.
3. Гилл А. Введение в теорию конечных автоматов. – М., Наука, 1966. – 272 с.
4. Закревский А.Д. Алгоритмы синтеза дискретных автоматов. – М.: Наука, Физматлит, 1971. – 511 с.
5. Карпов Ю.Г. Теория автоматов: учебник для вузов. – СПб.: Питер, 2002. – 224 с.
6. Киносита К., Асада К., Карацу О. Логическое проектирование СБИС. – М., «Мир», 1988. – 309 с.
7. Матросова А.Ю., Останин С.А., Паршина Н.А.. К синтезу контролепригодных комбинационных устройств // Автоматика и телемеханика. – 1999. – №2. – С. 129–137.
8. Пархоменко П.П., Согомонян Е.С. Основы технической диагностики. – М.: Энергоиздат, 1981. – 320 с.
9. Скобцов Ю.А., Скобцов В.Ю. Логическое моделирование и тестирование цифровых устройств. – Донецк: ИПММ НАН Украины, ДонНТУ, 2005. – 436 с.
10. Согомонян Е.С., Слабаков Е.В. Самопроверяемые устройства и отказоустойчивые системы. – М.: Радио и связь, 1989. – 208 с.
11. Хопкрофт Дж. Алгоритм для минимизации конечного автомата // Кибернетический сборник. Новая серия. – М., Мир, 1974. – вып.11. – С. 177–184.