САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА ТЕОРИИ СИСТЕМ УПРАВЛЕНИЯ ЭЛЕКТРОФИЗИЧЕСКОЙ АППАРАТУРОЙ

Мащинский Николай Сергеевич

Магистерская диссертация

Генерация последовательности проверяющих воздействий для цифровых радиоэлектронных устройств в рамках построения тестовых программ

Направление 03.04.01
«Прикладная математика и физика»
Магистерская программа «Математические и информационные технологии»

Научный руководитель,
доктор физ.-мат. наук,
профессор
Овсянников Д. А.

Санкт-Петербург
2018

Содержание

Введение

Спутниковая связь, промышленная автоматика, бытовая электроника, робототехника, медицина, охранные системы, устройства связи и обработки данных — это далеко не полный список областей, в которых широко применяются разнообразные цифровые электронные устройства.

Неотъемлемой частью производства и эксплуатации этих устройств является их тестирование на предмет физических дефектов, влияющих на корректность поведения устройства. На смену неэффективному из-за человеческого фактора ручному тестированию пришло тестирование автоматизированное, позволяющее не только значительно сократить время определения бракованных устройств, но и снизить стоимость исправления дефектов.

В условиях современной действительности наиболее перспективной технологией представляется функциональное тестирование, заключающееся в разработке программной модели, имитирующей поведение объекта контроля, и составлении последовательности тестовых воздействий, должным образом проверяющих исправность внутренних компонентов устройства. Такая последовательность составляется оператором на основе исследования внутренней структуры и алгоритмов работы элементов объекта контроля.

Ввиду многообразия и всевозрастающей сложности современных цифровых устройств, автоматизация процессов, присущих технологии функционального тестирования, является актуальной задачей на сегодняшний день.

Постановка задачи

Цель работы заключается в разработке программы, автоматизирующей составление последовательности тестовых воздействий, и проверке полученных результатов на примере цифрового модуля Субблок 031.

Решение поставленной задачи можно разбить на несколько этапов:

1. разработка и тестирование программных моделей внутренних элементов объекта контроля;
2. составление поведенческой модели всего устройства;
3. разработка алгоритма и реализация программы генерации последовательности тестовых воздействий;
4. проверка качества полученной последовательности.

Итоговая тестовая последовательность должна удовлетворять критерию покрытия, то есть обладать наибольшей возможной степенью покрытия внутренних элементов и сигнальных линий объекта контроля, и критерию полноты — осуществлять по возможности полную проверку каждого внутреннего компонента с учетом присущей ему функциональности, а также изменять состояние выходных сигналов цифрового модуля.

Обзор современных научных исследований

С появлением в середине 20-го века первых микросхем возникла необходимость тестирования электроники. Эта проблема остается актуальной и в настоящее время, поэтому контроль качества сборки электронных изделий и проверка корректности их функционирования являются обязательными этапами современного серийного производства.

В настоящее время существуют различные подходы к тестированию цифровых электронных устройств:

- визуальный автоматизированный контроль;
- внутрисхемное тестирование;
- граничное сканирование;
- функциональное тестирование.

Автоматизированный визуальный контроль основывается на анализе графического образа электронного изделия, полученного с использованием специальной аппаратуры. Системы анализа изображений позволяют с высокой скоростью и точностью определять корректность монтажа элементов на поверхности печатной платы или находить соединения с истонченным припоем. Для расширения возможностей метода автоматизированного визуального контроля в систему может быть добавлен источник рентгеновского излучения, позволяющий, например, контролировать монтаж элементов со скрытыми выводами. Однако для этого требуется сложное дорогостоящее оборудование.

Технология внутрисхемного тестирования заключается в подаче и считывании сигналов непосредственно с разъемов внутренних элементов тестируемого устройства посредством контактных иголок тестера. Эта особенность подхода позволяет определить такие дефекты, как короткие замыкания или разрывы в сигнальных линиях на поверхности печатной

платы, до подачи напряжения на все устройство, что могло бы привести к значительным повреждениям компонентов изделия. Спроектированное для конкретного типа устройств «ложе гвоздей» [1–4] производит процесс диагностики с высокой скоростью, но не может быть использовано при переходе к другому типу изделий. Более приспособляемыми являются системы «летающих щупов» [5–6], однако, в силу особенности своей конструкции, осуществляют тестирование изделий со значительно меньшей производительностью. Кроме того, современные устройства включают в себя микросхемы малого размера и зачастую являются многослойными, что затрудняет использование технологии внутрисхемного тестирования.

Применение граничного сканирования (JTAG — Joint Test Action Group) требует от тестируемых устройств соответствия определенным спецификациям, а именно — реализация в микросхемах стандарта IEEE 1149 [7], выделение групп микросхем в функциональные блоки и соединение JTAG-разъемов с краевыми выводами. Также по границам устройства необходимо размещение сдвигового регистра, ячейки которого располагаются между внешними выводами изделия и функциональным ядром. В режиме тестирования эти ячейки позволяют отключить функциональное ядро от внешних контактов, управлять сигналами на соответствующих группах микросхем и считывать их реакцию, а также сравнивать полученные значения с эталонными [8–16]. Недостаток этого метода состоит в невозможности определения разрывов в сигнальных линиях и проверки аналоговых элементов, так как технология граничного сканирования разрабатывалась для тестирования цифровых электронных устройств.

Подход функционального тестирования основывается на составлении программной модели объекта контроля и разработке учитывающей структурные особенности устройства последовательности входных воздействий [17–31]. Смоделированная на эту последовательность реакция

сравнивается с поведением тестируемого устройства, после чего делается вывод о корректности работы объекта контроля. Данная технология обладает максимальным покрытием внутренних элементов, хотя в некоторых случаях не позволяет точно определить дефектные компоненты.

Среди рассмотренных технологий тестирования, функциональный подход имеет ряд преимуществ:

- по сравнению с системами автоматизированного визуального контроля, требует значительно менее дорогостоящего оборудования и программного обеспечения;
- независимость от физического размера элементов устройства или количества слоев платы;
- от элементов объекта контроля не требуется соответствие стандарту JTAG;
- обладает коротким временем тестирования, что является важным критерием серийного производства цифровых устройств.

В работе используется система автоматизированного проектирования тестов «SimTest», разработанная на базе Санкт-Петербургского государственного университета. Использование САПР «SimTest» позволяет до определенной степени автоматизировать проведение контрольно–диагностических работ [32–36], в частности, уменьшить временные затраты на разработку тест-программы, сократив при этом ее стоимость.

Составленная тест-программа, описывающая входные воздействия и реакцию на них исправного устройства, загружается в установку тестового контроля [37–38] и используется для определения работоспособности цифрового изделия.

Глава 1. Моделирование объекта контроля

1.1. Программное моделирование элементов

Объект контроля Субблок 031 состоит из микросхем 136LA1, 136LA2, 136LA3, 136LA4, 109LI1, выполняющих логические функции 4И–НЕ, 8И–НЕ, 2И–НЕ, 3И–НЕ и 6И соответственно, а также селектора–мультиплексора на 8 входов 133КР7 [39–41] и четырехразрядного двоичного реверсивного счетчика 1533IE7 [42]. На рисунке 1 представлен фрагмент принципиальной схемы объекта контроля, где соединенные логические элементы 2И–НЕ функционируют идентично асинхронному RS–триггеру, что необходимо учитывать при составлении тестовой последовательности.

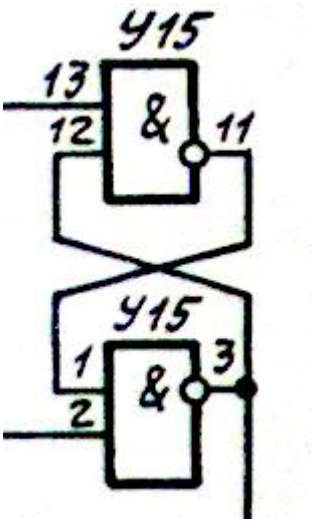

Рисунок 1. Фрагмент принципиальной схемы объекта контроля.

В связи с тем, что САПР «SimTest» предъявляет определенные требования к представлению программных моделей элементов, в качестве языка описания аппаратных средств был выбран Verilog HDL (Hardware Description Language). Помимо удобства использования и наличия достаточного количества справочной информации [43–44], этот HDL язык поддерживается рядом свободно распространяемых систем автоматизированного проектирования электронных устройств, например,

Altera Quartus 2 [45], в котором производится моделирование в данной работе.

Ниже представлены условно–графические отображения (УГО), таблицы истинности и разработанные программные модели для этих элементов.

1) Микросхема 136LA1 содержит два идентичных логических элемента, выполняющих булеву функцию 4И–НЕ (рисунок 2): $out = \overline{in1 \,\&\, in2 \,\&\, in3 \,\&\, in4}$.

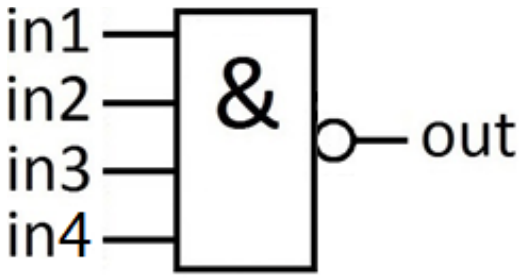

Рисунок 2. УГО логического элемента 4И–НЕ.

Так как логический элемент 4И–НЕ является комбинационным, то есть алгоритм формирования его выходных сигналов зависит только от комбинации значений входных сигналов в данный момент времени, то логика его работы может быть описана при помощи таблицы 1.

in1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
in2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
in3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
in4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
out	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Таблица 1. Таблица истинности элемента 4И–НЕ.

Ниже представлена программная модель этого элемента на языке Verilog HDL:

```
module icp_136la1
(
    input   in1, in2, in3, in4,
    output out
) ;
assign out = ~(in1 & in2 & in3 & in4);
endmodule
```

Ключевое слово «assign» означает присваивание выходной переменной *out* результата выражения правой части на каждом такте моделирования.

УГО и описывающие логику работы таблицы для микросхем 136LA2, 136LA3, и 136LA4, соответственно выполняющих логические функции 8И–НЕ, 2И–НЕ и 3И–НЕ, выглядят аналогично, как и их программные модели на языке Verilog, отличаясь только количеством входных сигналов.

2) Микросхема 109LI1 выполняет логическую функцию 6И: $out = in1 \,\&\, in2 \,\&\, in3 \,\&\, in4 \,\&\, in5 \,\&\, in6$. УГО и программная модель этого элемента незначительно отличаются от таковых для элемента 136LA1, а таблица истинности может быть заменена кратким описанием алгоритма:

ЕСЛИ

на всех входных сигналах $in1 - in6$ имеется логическая «1»;

ТОГДА

значением выходного сигнала out будет логическая «1»;

ИНАЧЕ

значением выходного сигнала out будет логический «0».

3) Как было указано выше, на некоторых фрагментах принципиальной схемы цифрового модуля Субблок 031 (рисунок 1) соединения логических элементов 2И–НЕ обеспечивают функционирование, аналогичное алгоритму работы асинхронного RS–триггера (рисунки 3 и 4).

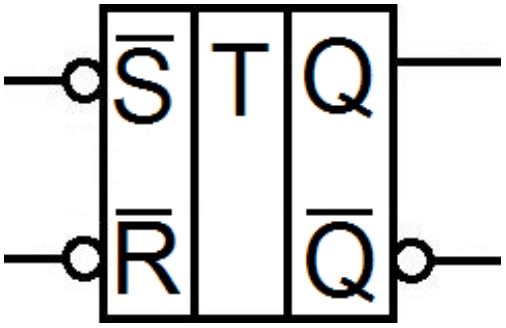

Рисунок 3. УГО асинхронного RS–триггера.

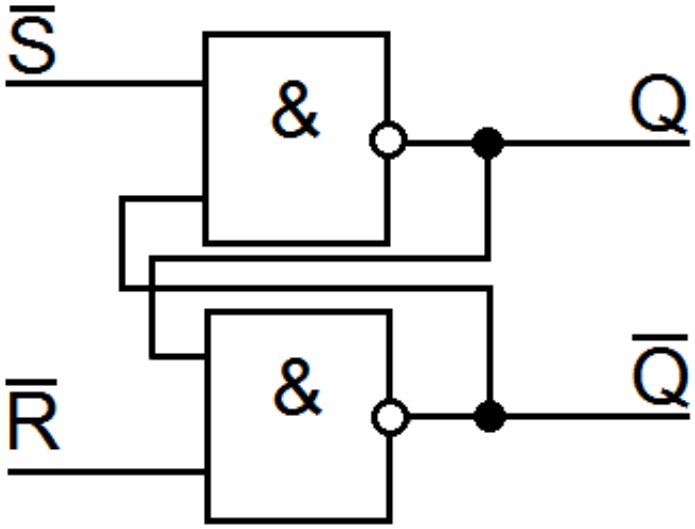

Рисунок 4. Реализация асинхронного RS–триггера на двух элементах 2И–НЕ.

Рассмотрим алгоритм работы такого триггера. В режиме установки, при подаче логического «0» на вход $\overline{\mathrm{S}}\,(set)$ и логической «1» на вход $\overline{\mathrm{R}}\,(reset)$, на прямой и инверсный выходы Q и $\overline{\mathrm{Q}}$ поступят сигналы 1 и 0 соответственно. В режиме сброса, при подаче логического «0» на вход $\bar{R}$ и логической «1» на вход $\bar{S}$, на прямой и инверсный выходы Q и $\overline{\mathrm{Q}}$ поступят сигналы 0 и 1 соответственно. В режиме хранения информации, при подаче логической «1» на оба входа, на прямой выход Q поступит значение, которое было запомнено в режиме установки или сброса. Состояние, когда на оба входа подается логический «0» является запрещенным, так как в этот момент

на прямом и инверсном выходах будет находиться одно и то же значение логической «1».

Наглядно режимы работы RS–триггера представлены в таблице 2. Введенное для удобства обозначение M является значением, запомненным в режиме установки или сброса (1 и 0 соответственно).

Режим	$\bar{S}$	$\bar{R}$	M	Q	$\bar{Q}$
Установка	0	1	1	1	0
Сброс	1	0	0	0	1
Хранение	1	1	1	1	0
	1	1	0	0	1
Запрещенное состояние	0	0	1	1*	1*

Таблица 2. Режимы работы асинхронного RS–триггера.

Программная модель RS–триггера может быть описана следующим образом:

```
module ic_rs_trigger
(
        input s_n, r_n,
        output q0, q0_n
);
reg memory;

always@(s_n or r_n)
begin
        if(s_n == 1'b0)
                memory <= 1'b1;
        else
                if(r_n == 1'b0)
```

```
                memory <= 1'b0;
    end
    assign q0 = memory;
    assign q0_n = (s_n | r_n == 1'b0)? 1'b1: ~memory;
    endmodule
```

Ввиду того, что после ключевого «always» внутри блока *begin/end* возможна запись значений только в регистры, то необходимо ввести запоминающий регистр *memory*, значение которого присваивается выходным сигналам *q0* и *q0_n*.

4) Микросхема 133КР7 представляет собой селектор–мультиплексор данных на 8 каналов со стробированием (рисунок 5).

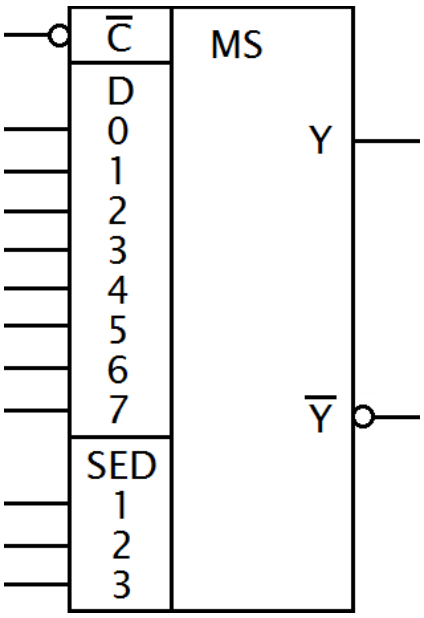

Рисунок 5. УГО селектора–мультиплексора 133КР7.

Алгоритм работы этого элемента заключается в следующем. В зависимости от установленного на входах *SED_1 – SED_3* двоичного кода, разрешается прохождение сигнала только от одного из информационных входов *D0 – D7* на прямой выход *Y*, при этом на входе стробирования $\bar{C}$ должен быть установлен логический «0». При подаче логической «1» на вход стробирования $\bar{C}$ выход *Y* переходит в состояние логического «0».

На инверсный выход $\bar{Y}$ всегда поступает сигнал, отличный от сигнала на прямом выходе *Y*. Таблица истинности этого элемента представлена в таблице 3. Здесь символом *x* обозначается произвольное значение сигнала.

SED_1	SED_2	SED_3	$\bar{C}$	Y	$\bar{Y}$
x	x	x	1	0	1
0	0	0	0	D0	$\overline{D0}$
1	0	0	0	D1	$\overline{D1}$
0	1	0	0	D2	$\overline{D2}$
1	1	0	0	D3	$\overline{D3}$
0	0	1	0	D4	$\overline{D4}$
1	0	1	0	D5	$\overline{D5}$
0	1	1	0	D6	$\overline{D6}$
1	1	1	0	D7	$\overline{D7}$

Таблица 3. Таблица истинности селектора–мультиплексора 133КР7.

Ниже представлена программная модель для селектора–мультиплексора 133КР7:

```
module ic_1533kp7
(
    input c0_n, sed1, sed2, sed3,
    input d7, d6, d5, d4, d3, d2, d1, d0,
    output y0, y0_n
);
assign y0_n = ~y0;

wire [7:0] D1 = {d7, d6, d5, d4, d3, d2, d1, d0};
assign y0 = (c0_n == 1'b0)? D1[{sed3, sed2, sed1}] : 1'b0;
endmodule
```

Здесь выражение «*{d7, d6, d5, d4, d3, d2, d1, d0}*» обозначает конкатенацию входных сигналов в одну восьмиразрядную сигнальную

линию, которая поразрядно присваивается объявленной переменной *D1*. В выражении «*D1[{sed3, sed2, sed1}]*» конкатенация входных сигналов *sed3* – *sed1* интерпретируется как трехразрядное двоичное число, которое определяет порядковый номер одного из информационных входов *d0* – *d7*, значение сигнала которого поступит на выход *y0*.

5) Поясним режимы работы счетчика 1533ИЕ7 (рисунок 6), кратко представленные в таблице 4. Здесь символом ⌜ обозначается положительный импульс сигнала, то есть переходное состояние из логического «0» в логическую «1».

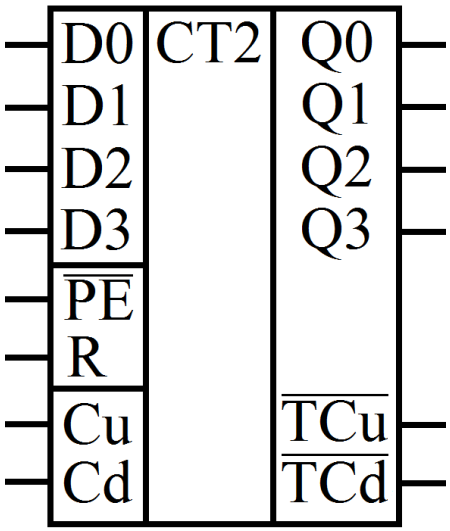

Рисунок 6. УГО счетчика 1533ІЕ7.

В режиме сброса, при подаче логической «1» на вход *R* счетные выходы *Q0* – *Q3* перейдут в состояние логического «0». Для записи в память счетчика определенных сигналов требуется подать на информационные входы *D0* – *D3* соответствующие значения, а на вход параллельной записи $\overline{PE}$ — логический «0». Как следует из названия, режим прямого счета увеличивает хранящееся в памяти счетчика значение. Для этого необходимо подавать на вход прямого счета *Cu* положительные импульсы, а на входе обратного счета *Cd* сохранять значение 1. При этом в случае достижения в памяти счетчика максимального значения (1 1 1 1), выход прямого переноса

TCu сначала перейдет в состояние логического «0», а затем, когда хранящееся в счетчике значение станет (0 0 0 0), — в состояние логической «1». Схожим образом счетчик работает в режиме обратного счета.

Режим	Входы								Выходы					
	R	$\overline{PE}$	Cu	Cd	D0	D1	D2	D3	Q0	Q1	Q2	Q3	$\overline{TCu}$	$\overline{TCd}$
Сброс	1	x	x	0	x	x	x	x	0	0	0	0	1	0
	1	x	x	1	x	x	x	x	0	0	0	0	1	1
Параллельная запись	0	0	x	0	0	0	0	0	0	0	0	0	1	0
	0	0	x	1	0	0	0	0	0	0	0	0	1	1
	0	0	0	x	1	1	1	1	1	1	1	1	0	1
	0	0	1	x	1	1	1	1	1	1	1	1	1	1
Прямой счет	0	1	⌜	1	x	x	x	x	Прямой счет				1	1
Обратный счет	0	1	1	⌜	x	x	x	x	Обратный счет				1	1

Таблица 4. Режимы работы счетчика 1533ИЕ7.

Программная модель этого элемента описывается на языке Verilog следующим образом:

```
module ic_1533ie7
(
        input PE_N, R0, Cu, Cd, D0, D1, D2, D3,
        output Q0, Q1, Q2, Q3, TCu_N, TCd_N
);
reg [3:0] mem;
assign TCu_N = (mem == 4'b1111 && Cu == 1'b0)? 1'b0 : 1'b1;
assign TCd_N = (mem == 4'b0000 && Cd == 1'b0)? 1'b0 : 1'b1;
assign {Q3, Q2, Q1, Q0} = (R0 == 1'b0)? mem: 4'b0000;

wire Tplus, Tminus, Tand;
```

```
assign #1 Tplus = Cu;
assign #1 Tminus = Cd;
assign Tand = Cd & Cu;

always@(negedge PE_N or posedge R0 or posedge Tand)
begin
   if(R0 == 1'b1)    mem <= 4'b0000;
   else    if(PE_N == 1'b0)    mem <= {D3, D2, D1, D0};
      else    if(Tand == 1'b1)
            begin
               if(Tplus == 1'b0 && Tminus == 1'b1)
                  mem = mem + 1'b1;
               else    if(Tplus == 1'b1 && Tminus == 1'b0)
                     mem = mem - 1'b1;
            end
end
endmodule
```

Выражение «*assign #1*» означает непрерывное присвоение сигнала с задержкой в одну единицу модельного времени, значение которой настраивается отдельно. Ввиду того, что в режимах прямого и обратного счета выполнение кода внутри блока «*always*» начинается, когда на обоих входах *Cu* и *Cd* уже находятся значения логической «1» (на одном входе поддерживалось состояние логической «1», а на другом уже завершился переход из логического «0» в логическую «1»), для корректного функционирования программной модели счетчика необходимо введение дополнительных сигналов *Tplus* и *Tminus*. Введение сигнала *Tand* обусловлено обработкой неопределенной ситуацией, когда на оба счетных входа *Cu* и *Cd* одновременно поступили положительные импульсы сигнала.

1.2. Тестирование программных моделей элементов

Для поведенческой модели каждого элемента, входящего в состав объекта контроля, была разработана тестовая последовательность, проверяющая соответствие функционирования программной модели алгоритму работы описываемого компонента.

Ниже представлена составленная последовательность входных воздействий для асинхронного RS–триггера.

//s_n
#GROUP_SIG 0
Val 0 0
Val 1 1
Val 4 0
Val 7 1

//r_n
#GROUP_SIG 1
Val 0 1
Val 2 0
Val 3 1
Val 4 0
Val 5 1
Val 6 0
Val 8 1

На рисунке 7 представлены результаты моделирования воздействия тестовой последовательности на асинхронный RS–триггер в САПР «SimTest». Как можно видеть, реакция программной модели на значения входных сигналов полностью соответствует режимам работы этого элемента (таблица 2). Например, на первых четырех тактах последовательно

сменяются режимы установки, хранения информации, сброса и снова хранения информации. После этого проверяется поведение программной модели в случае запрещенного состояния.

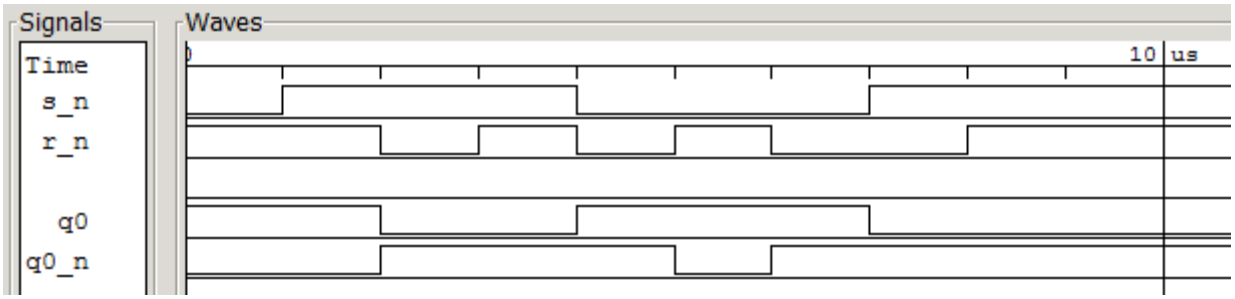

Рисунок 7. Результаты моделирования асинхронного RS–триггера.

На рисунке 8 представлены аналогичные результаты для логического элемента 4И–НЕ, совпадающие с его таблицей истинности (таблица 1).

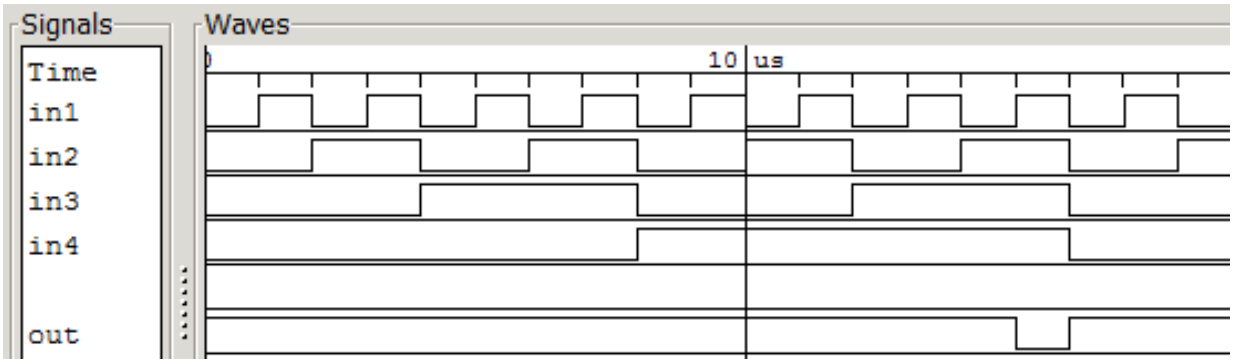

Рисунок 8. Результаты моделирования логического элемента 4И–НЕ.

На рисунке 9 изображены значения сигналов для селектора–мультиплексора 133КР7. Как можно видеть, изменения сигнала на прямом выходе *y0* совпадают с сигналом на одном из информационных выходов *d0–d7*, определяемого комбинацией значений *sed1–sed3*.

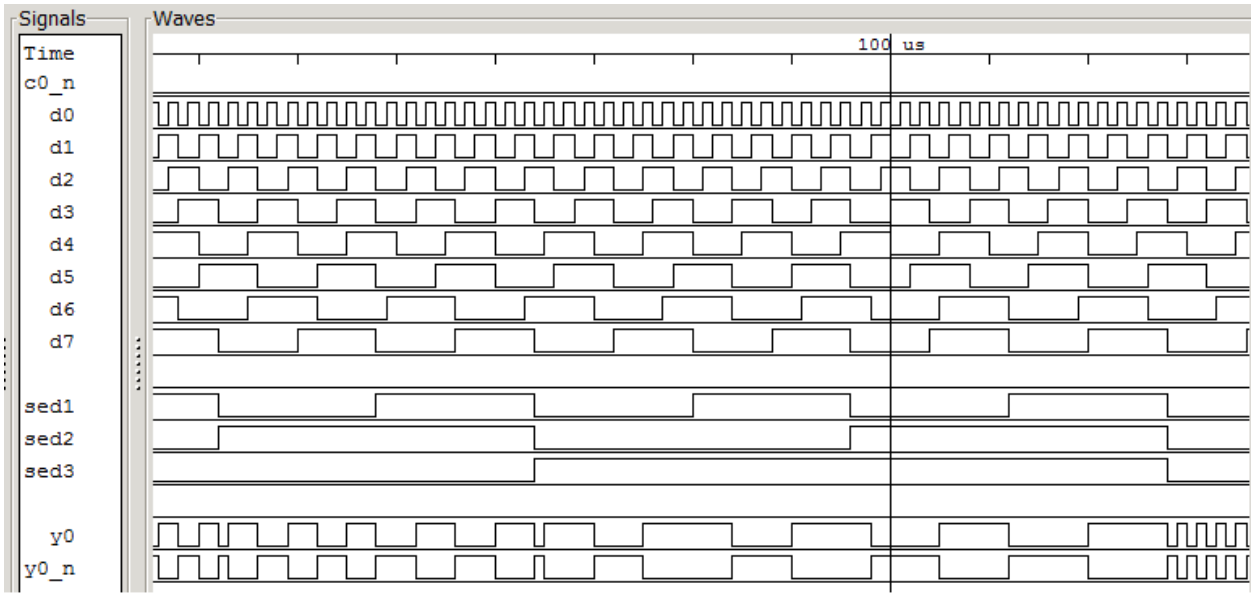

Рисунок 9. Результаты моделирования селектора–мультиплексора 133КР7.

Представленные на рисунке 10 последовательности временных сигналов четырехразрядного двоичного реверсивного счетчика 1533ІЕ7 также согласуются с режимами работы этого элемента (таблица 4).

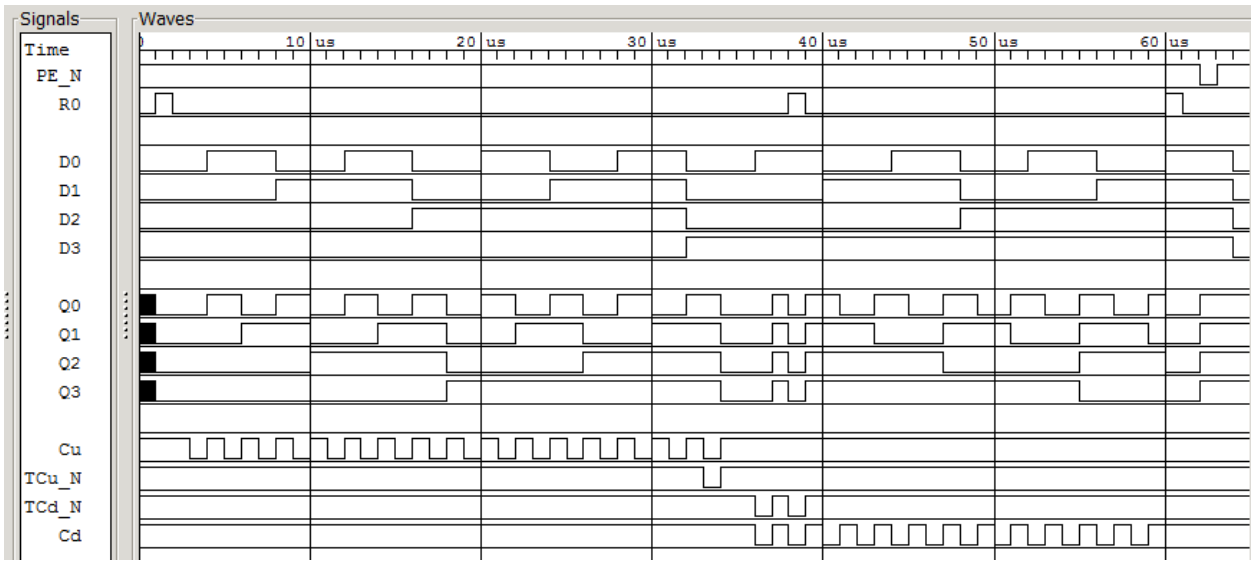

Рисунок 10. Результаты моделирования счетчика 1533ІЕ7.

1.3. Оцифровывание принципиальной схемы цифрового модуля Субблок 031

После составления программных моделей элементов объекта контроля и проверки правильности их функционирования, в САПР Altera Quartus 2 автоматически генерируются редактируемые графические представления для каждого компонента. Эти представления отображают тип элемента, а также количество, расположение и названия его входных и выходных сигналов (рисунок 11).

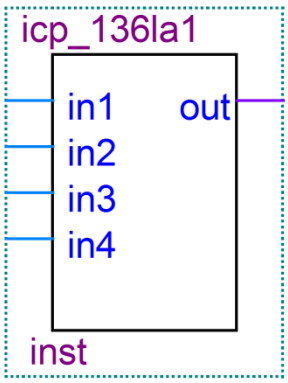

Рисунок 11. Графическое представление элемента 136LA1 в САПР Altera Quartus 2.

Эти представления размещаются в рабочем пространстве системы Quartus, именуются и соединяются сигнальными линиями в соответствии с принципиальной электрической схемой объекта контроля (рисунок 12). Также в систему добавляются краевые входные и выходные сигнальные линии устройства. Цифровой модуль Субблок 031 состоит из 166 элементов (из которых 16 логических элементов образуют соединения с ячейками памяти), 38 входных и 9 выходных сигналов, а также содержит 204 внутренние связи.

По завершении оцифровывания принципиальной электрической схемы объекта контроля, Quartus позволяет произвести компиляцию составленной графической модели в код, описывающий внутреннее представление

цифрового модуля — программные модели элементов устройства, соединенные между собой указанным образом сигнальными линиями.

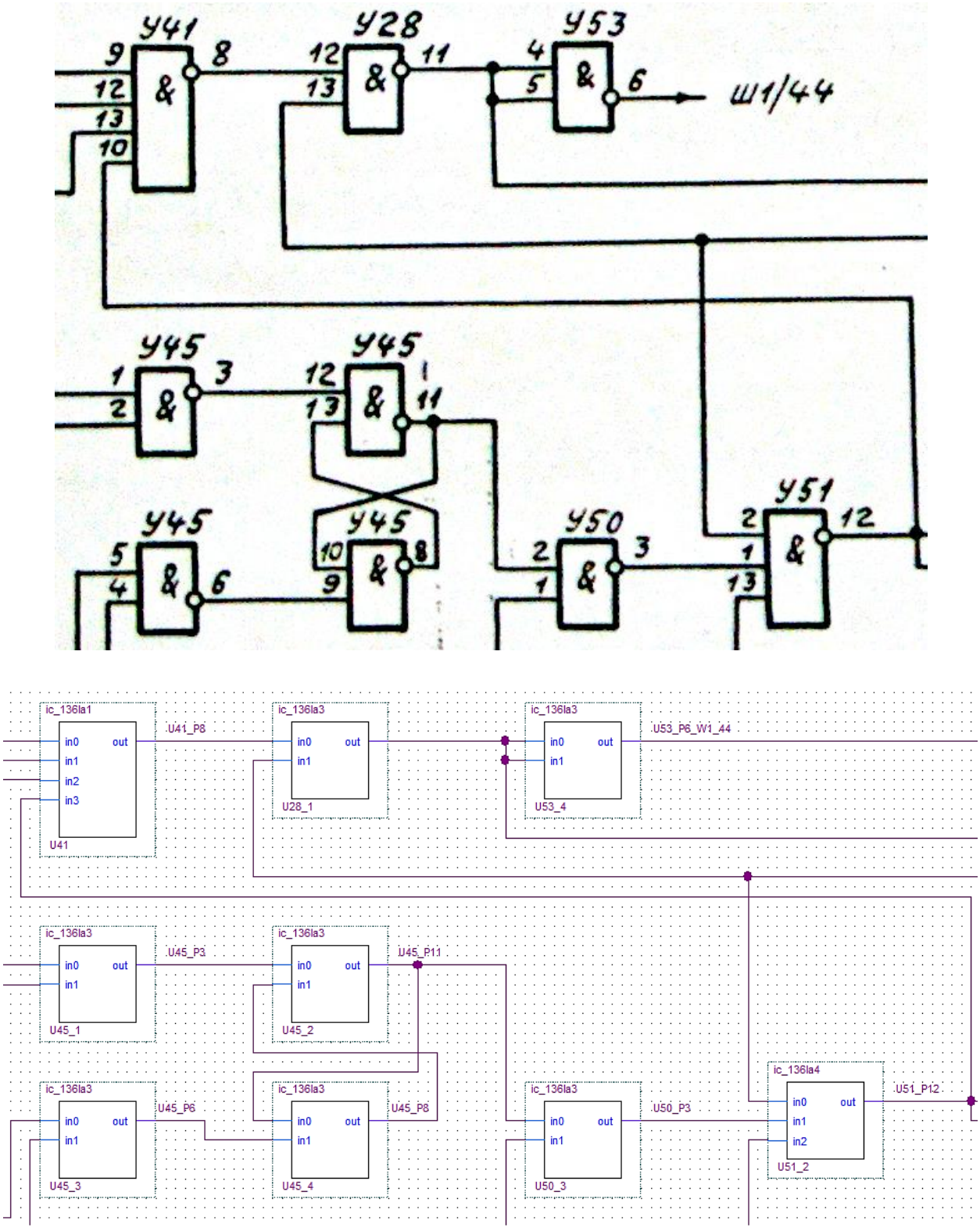

Рисунок 12. Фрагмент принципиальной электрической схемы цифрового модуля Субблок 031и фрагмент оцифрованной программной модели в САПР Altera Quartus 2.

В процессе компиляции Quartus проверяет корректность составления графического представления. Примерами обнаруживаемых ошибок могут быть сигнальные линии, соединенные с выходами нескольких элементов или краевые сигналы, не присоединенные к выводам внутренних элементов.

Ниже представлен фрагмент кода, описывающего внутреннюю структуру цифрового модуля Субблок 031:

```
module Subblock_031(
        WI1_2,
        WI1_3,
        WI1_4,
        ...
        WO1_51,
        WO1_55,
        WO1_59,
);
input  WI1_2;
input  WI1_3;
input  WI1_4;
...
output      WO1_51;
output      WO1_55;
output      WO1_59;

wire   U10_P11;
wire   U10_P6;
wire   U10_P8;
...
wire   W1_7;
wire   W1_70;
```

```
wire   W1_71;

ic_136la3   b2v_U1_1(
        .in0(W1_2),
        .in1(U1_P11),
        .out(U1_P3));
...
ic_136la3   b2v_U53_3(
        .in0(U31_P12),
        .in1(U31_P12),
        .out(U53_P8_W1_40));

assign      W1_2 = WI1_2;
assign      W1_3 = WI1_3;
assign      W1_4 = WI1_4;
...
assign      WO1_51 = U52_P3_W1_51;
assign      WO1_55 = W1_55;
assign      WO1_59 = U56_P8_W1_59;

endmodule
```

Глава 2. Составление тестовой последовательности

2.1. Алгоритм составления тестового портрета

Рассмотрим алгоритм составления тестовых воздействий на примере фрагмента цифровой схемы, представленном на рисунке 13. Для удобства восприятия, на этом рисунке нарушены правила именования сигнальных линий и краевых сигналов.

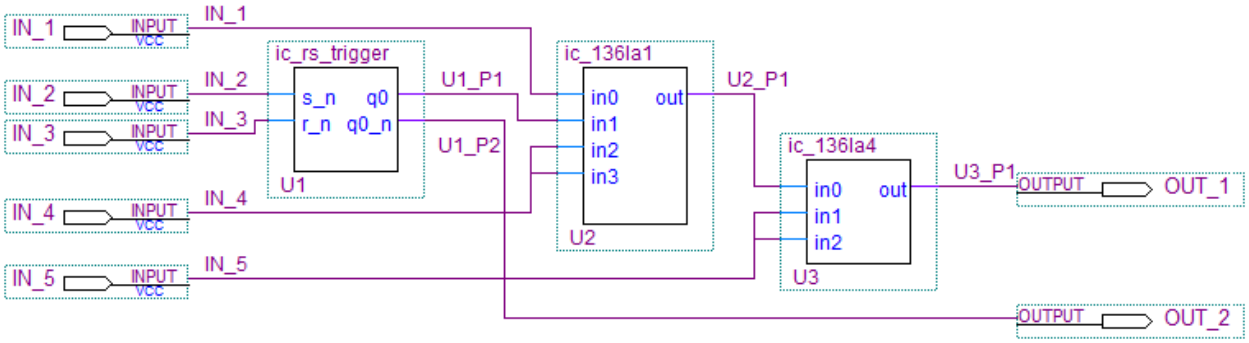

Рисунок 13. Фрагмент цифровой схемы.

Пусть необходимо проверить корректность функционирования элемента *U2*, выполняющего логическую функцию 4И–НЕ. Для этого необходимо подать на его входы комбинации сигналов из таблицы истинности, соответствующей этому элементу (таблица 1), и обеспечить такое состояние остальных элементов, чтобы реакция проверяемого элемента достигла краевых выходов схемы.

Первой комбинацией является последовательность T = (0 0 0 0 | 1). Получение на входах элемента *U2* этой комбинации сигналов требует, чтобы на связанных с его входами источниках сигнала (краевых входах схемы *IN_1*, *IN_4* и выходе *q0* элемента *U1*, асинхронного RS–триггера) были установлены соответствующие значения.

Для того чтобы получить значение 0 на выходе *q0* элемента *U1* необходимо обратиться к алгоритму его работы (таблица 2) и выбрать режим

функционирования, позволяющий получить требуемое значение на выходе элемента. Определив, что подходящим режимом является сброс (режим хранения информации может быть использован при последующем требовании значения 0 от выхода *q0*), необходимо получить на входах *s_n* и *r_n* значения сигналов 1 и 0 соответственно, что в свою очередь требует аналогичных значений от входов схемы *IN_2* и *IN_3*. Получение значения 1 на краевом выходе *OUT_2*, связанном с инверсным выходом *q0_n*, не является условием завершения итерации алгоритма, так как этот выход схемы не связан с реакцией проверяемого элемента *U2* ни напрямую, ни опосредованно, через другие элементы.

Таким образом установлено, что для получения на входах элемента *U2* комбинации сигналов (0 0 0 0) необходимо подать на краевые входы схемы *IN_1 – IN_4* значения (0 1 0 0). Теперь реакцию этого элемента (значение сигнала 1 на выходе *out*) необходимо «провести» до краевого выхода *OUT_1*. Для этого требуется, чтобы на входах *in1, in2* элемента *U3*, которые не связаны с выходом элемента *U2*, была получена «разрешающая» комбинация сигналов. Это такая комбинация, которая позволяет при разных значениях на интересующем входе (в данном случае входе *in0*, поскольку он связан с выходом проверяемого элемента *U2*) получать различную реакцию на выходе рассматриваемого элемента *U3*. В соответствии с логикой работы элемента *U3*, выполняющего логическую функцию 3И–НЕ, разрешающей комбинацией сигналов будут значения (1 1) на входах *in1, in2*, которые может обеспечить краевой вход схемы *IN_5*.

Так как на всех входах элемента *U3* получены сигналы (1 1 1), то в соответствии с алгоритмом работы, на его выходе *out* окажется значение 0, которое поступит на краевой выход схемы *OUT_1*.

После одной итерации алгоритма установлено, что для подачи комбинации (0 0 0 0) на входы проверяемого элемента *U2* и «проведения»

реакции этого элемента до краевых выходов, необходимо установить на входах схемы *IN_1 – IN_5* значения сигналов (0 1 0 0 1). При этом ожидается, что в случае исправности элементов и сигнальных линий, на краевой выход *OUT_1* поступит значение 0, а на выход *OUT_2* — значение 1. Подобным образом подаются остальные комбинации сигналов для элемента *U2*, после чего проверка этого элемента считается законченной. Аналогично проверяются элементы *U1* и *U3*, в соответствии с алгоритмами их функционирования.

В общем случае, рассмотренный алгоритм можно представить в виде блок-схемы, представленной на рисунке 14, где в качестве рассматриваемого элемента *A* выступает элемент *U2*, а списки *Bi* и *Ci* состоят из элементов *U1* и *U3* соответственно. При этом под «проходом вверх» понимается требование определенных значений, распространяющееся от конкретного элемента к краевым входам схемы, а «проход вниз» означает распространение реакции элемента на комбинацию сигналов, установившихся на его входах.

Совокупность полученных временных последовательностей значений на краевых входах и соответствующих им сигналах на краевых выходах схемы называется «тестовым портретом». Подача входных воздействий из тестового портрета на реальное устройство позволяет локализовать область неисправности в том случае, если реакция устройства отличается от ожидаемых «выходных» значений тестового портрета. Например, если на краевом выходе *OUT_2* получено ожидаемое значение, а на выходе *OUT_1* — нет, то это косвенно свидетельствует об исправности элемента *U1* и о том, что неисправность, скорее всего, связана с элементами *U2* и *U3* или связанными с ними сигнальными линиями. Если же оказалось, что на краевом выходе *OUT_1* на время всего тестирования установилось значение 1, то это означает, что хотя бы на один вход элемента *U3* всегда поступал сигнал 0, что так же позволяет локализовать область неисправности в совокупности с анализом сигнала на краевом выходе схемы *OUT_2*.

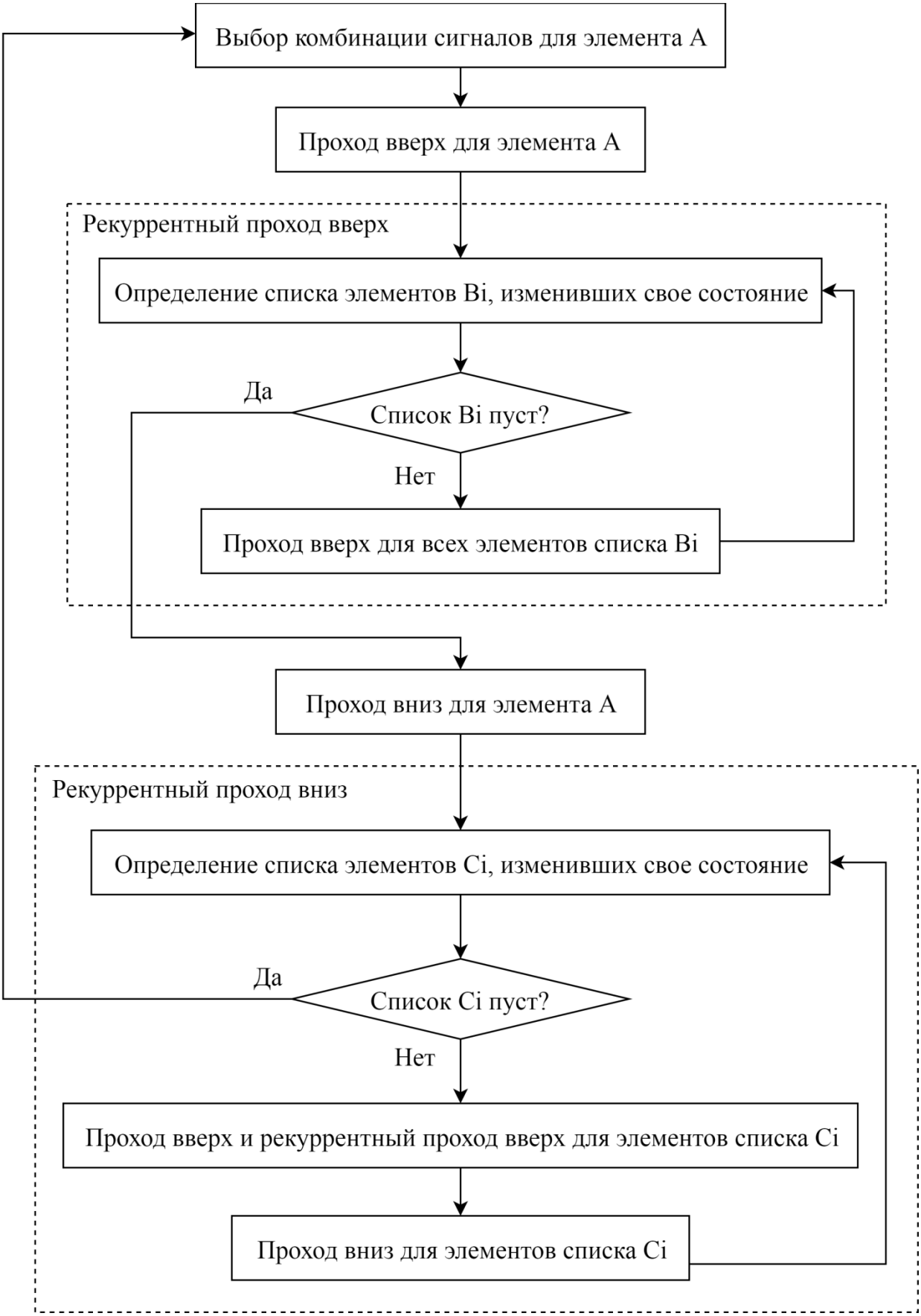

Рисунок 14. Блок-схема алгоритма составления тестового портрета.

Составление тестового портрета аналогичным образом выполняется для более сложных логических элементов и элементов с ячейками памяти, а также для цифровых схем, содержащих большее количество внутренних компонентов [46–48].

Для наглядности также рассмотрим алгоритм составления тестовой последовательности на примере фрагмента схемы с более сложным элементом (рисунок 15), а именно четырехразрядным двоичным реверсивным счетчиком. Здесь элемент *VCC*, подключенный к входам элемента *D1*, является источником постоянного сигнала логической «1», а элемент *D2* выполняет логическую функцию 6И.

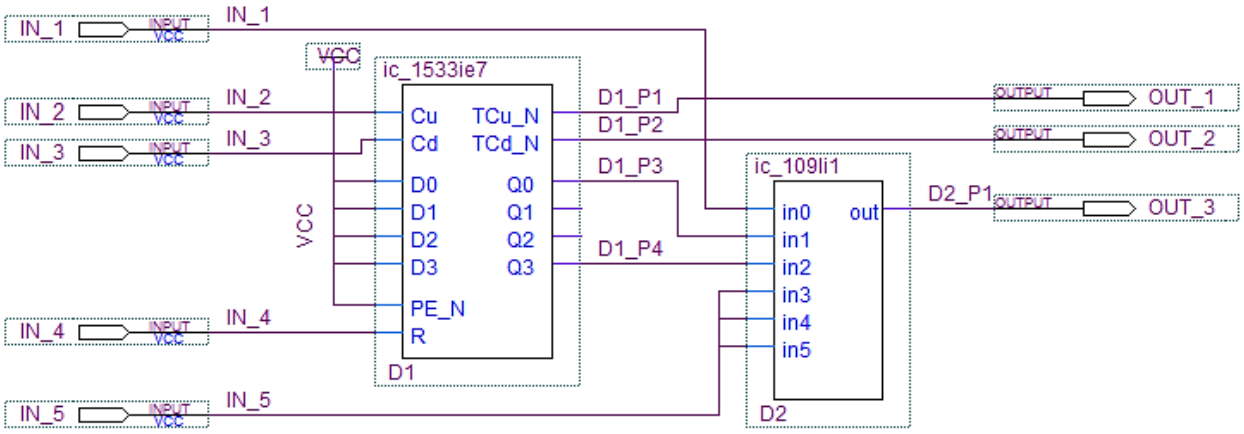

Рисунок 15. Фрагмент цифровой схемы со счетчиком 1533ИЕ7.

Пусть в ходе составления тестовой последовательности на входы элемента *D2* необходимо подать комбинацию сигналов (1 1 1 1 1 1 | 1). Для этого необходимо, чтобы на соответствующих выходах счетчика *D1* и краевых входах *IN_1*, *IN_5* установились требуемые значения. Таким образом, необходимо получить на информационных выходах счетчика *Q0* – *Q3* комбинацию *Res* = (1 *x x* 1). Для определенности рассмотрим ситуацию, когда на текущий момент в счетчике хранится значение *Mem* = (1 1 1 0) , соответствующее выходам *Q0* – *Q3*.

В связи с особенностью фрагмента схемы, а именно подключением входа параллельной записи $\overline{PE}$ к источнику постоянного сигнала, непосредственная запись в память счетчика требуемого значения не представляется возможной. Поэтому необходимо использовать режимы прямого и обратного счета, а также режим сброса, что неизбежно приведет к удлинению тестовой последовательности и повторению некоторых сигналов.

В рассматриваемой ситуации есть два приемлемых способа получения выходной комбинации *Res* из хранящегося в счетчике значения *Mem*. Первый способ заключается в подаче двух положительных импульсов на входе прямого счета *Cu*, что приводит к комбинации *Res_1* = (1 0 0 1) и занимает 4 такта моделирования. Комбинацию *Res_2* = (1 1 1 1) можно получить при использовании режима сброса и последующего обратного счета, что занимает всего 3 такта моделирования. Однако этот способ требует задействования большего числа краевых входов представленного на рисунке 15 фрагмента схемы и может быть нежелательным, если рассматриваемый фрагмент соединен с подобными счетчиками или другими элементами с памятью. В этом случае использование большего числа входов элемента *D1* может привести к еще большему увеличению числа тактов тестовой последовательности.

2.2. Автоматизация составления тестового портрета

Несмотря на то, что выполнение представленного в параграфе 2.1 алгоритма составления тестового портрета не выглядит трудозатратным, выполнение его оператором занимает значительное время. Это может привести к ошибкам, вызванным человеческим фактором, и, как следствие, неполной проверке внутренних элементов объектов контроля. Моделирование в САПР «SimTest» позволяет определить тестовое покрытие $P = (N_{act}/N) * 100\%$, выражающее процентное отношение активированных в ходе выполнения теста сигнальных линий к общему числу сигнальных линий, содержащихся в устройстве, и количество переключений этих линий. Но эта информация никак не сигнализирует о выполнении критерия полноты проверки функциональности элементов. Для выполнения этого критерия необходимо поступление на входы каждого элемента всех комбинаций сигналов, полностью определяющих корректность его работы, и достижение соответствующими реакциями каждого элемента краевых выходов схемы.

В связи с этим является актуальным автоматизация составления тестового портрета для объекта контроля, что позволит не только значительно сократить время на разработку тест-программ, но и повысить качество тестирования цифровых устройств. Для этой цели была разработана программа генерации тестовых портретов «TestGen» на языке C++. Эта программа состоит из 15 классов и содержит более 4000 строк кода. В приведенном ниже описании ее работы под словами «элемент» и «сигнальная линия» следует понимать программные аналоги внутренних элементов и сигнальных линий исходного объекта контроля, существующие только в контексте программы «TestGen».

Чтение данных. Из файла, содержащего описание внутренней структуры объекта контроля (представлен в параграфе 1.3), считываются название цифровой схемы, ее краевые входы и выходы, сигнальные линии,

«assign»–назначения и внутренние элементы — последовательно определяются название и тип элемента, названия выводов и их привязки к сигнальным линиям.

Проверка корректности внутренней структуры. Осуществляется проверка имен краевых выводов схемы на повторяемость, предварительная привязка сигнальных линий к выводам схемы. Создаются временные объекты внутренних элементов, для которых из базы данных программы считываются таблицы истинности и правила определения разрешающих комбинаций.

Проверяется, что все сигнальные линии соединены ровно с одним источником сигнала (краевым входом схемы или выходом внутреннего элемента) и не менее чем с одним приемником сигнала (входом внутреннего элемента или краевым выходом схемы). В ситуации, когда одна сигнальная линия подключена к нескольким входам элемента, выводится соответствующее предупреждение, но выполнение программы не прерывается. Это позволяет отследить возможные ошибки в составленной поведенческой модели.

Далее создаются и соединяются между собой предписанным образом объекты сигнальных линий, внутренних элементов и их выводов, а также объект цифровой схемы, содержащий краевые входы и выходы. После этого список элементов и сигнальных линий передается управляющей генерацией подпрограмме.

Генерация тестового портрета. Составление тестовой последовательности происходит в соответствии с алгоритмом, описанном в параграфе 2.1: обеспечивается возможность подачи определенной комбинации сигналов на входах конкретного элемента, после чего на входах компонентов, находящихся между рассматриваемым элементом и краевыми выходами схемы, устанавливаются «разрешающие» комбинации сигналов.

Значения сигналов, требуемые от краевых входов и поступившие на краевые выходы схемы, запоминаются программой, после чего происходит подача следующей комбинации сигналов из таблицы истинности на входы рассматриваемого элемента.

Запись результатов. После того, все элементы были обработаны, программа «TestGen» сохраняет полученные состояния краевых входов схемы в файл результатов, используемый в САПР «SimTest» как последовательность тестовых входных воздействий. Также программа сохраняет отладочный файл, анализ которого позволяет определить, какие именно элементы изменяли свое состояние на каждом такте «моделирования» в процессе обработки определенного компонента и на какие краевые выходы поступила реакция этого компонента. Эта информация позволяет локализовать область неисправности в случае, если в результате тестирования цифрового модуля были получены сигналы, отличные от ожидаемых.

Результаты моделирования тестовой последовательности для программной модели объекта контроля

На рисунке 16 представлен фрагмент временной последовательности состояний сигнальных линий программной модели цифрового модуля Субблок 031, полученной в результате имитации подачи тестовой последовательности на входы поведенческой модели в САПР «SimTest».

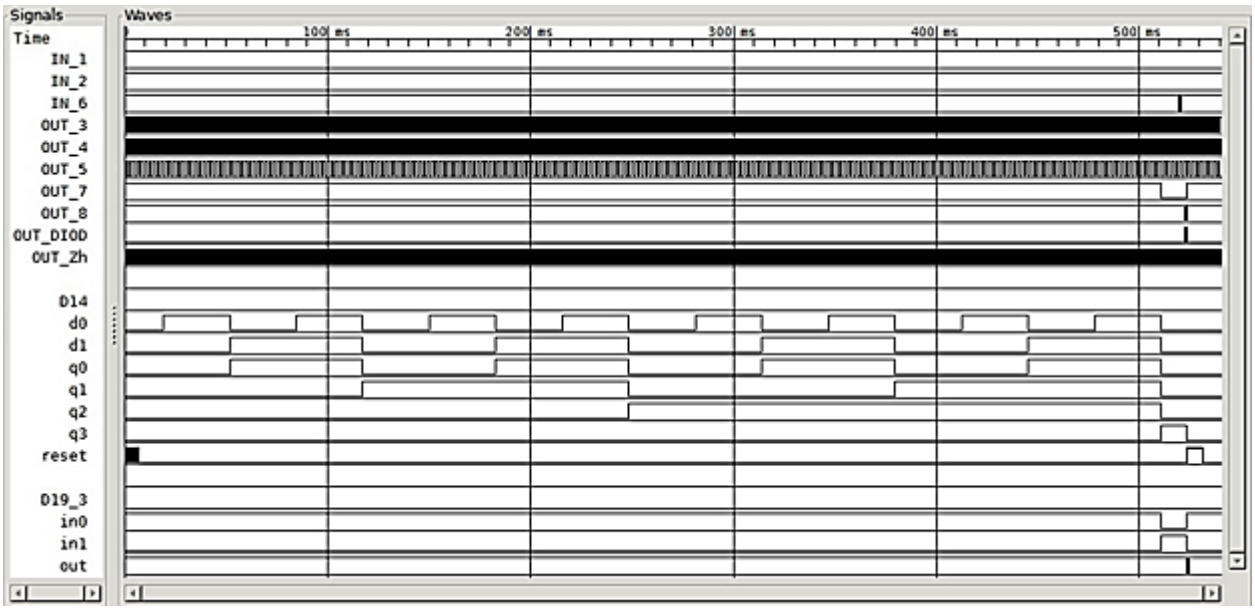

Рисунок 16. Временная последовательность состояний сигнальных линий программной модели объекта контроля.

На рисунках 17 и 18 представлено тестовое покрытие краевых выводов объекта контроля и сигнальных линий. Как можно видеть, достигнуто полное покрытие объекта контроля, что говорит о качестве полученной тестовой последовательности.

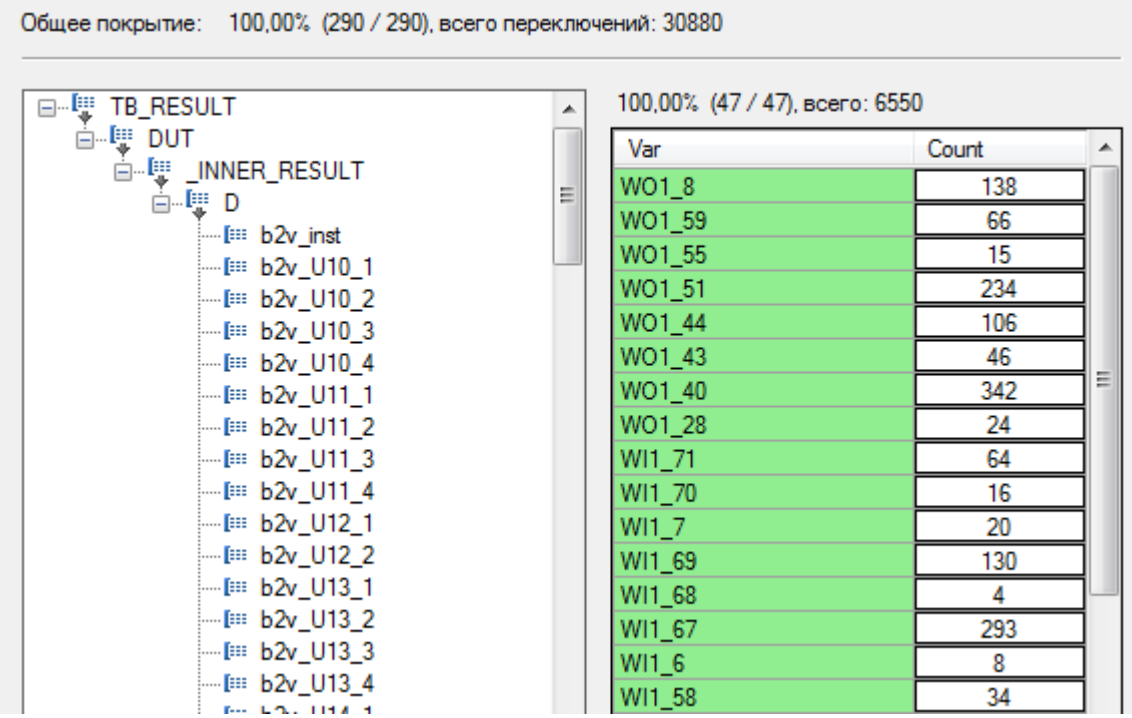

Рисунок 17. Тестовое покрытие краевых выводов.

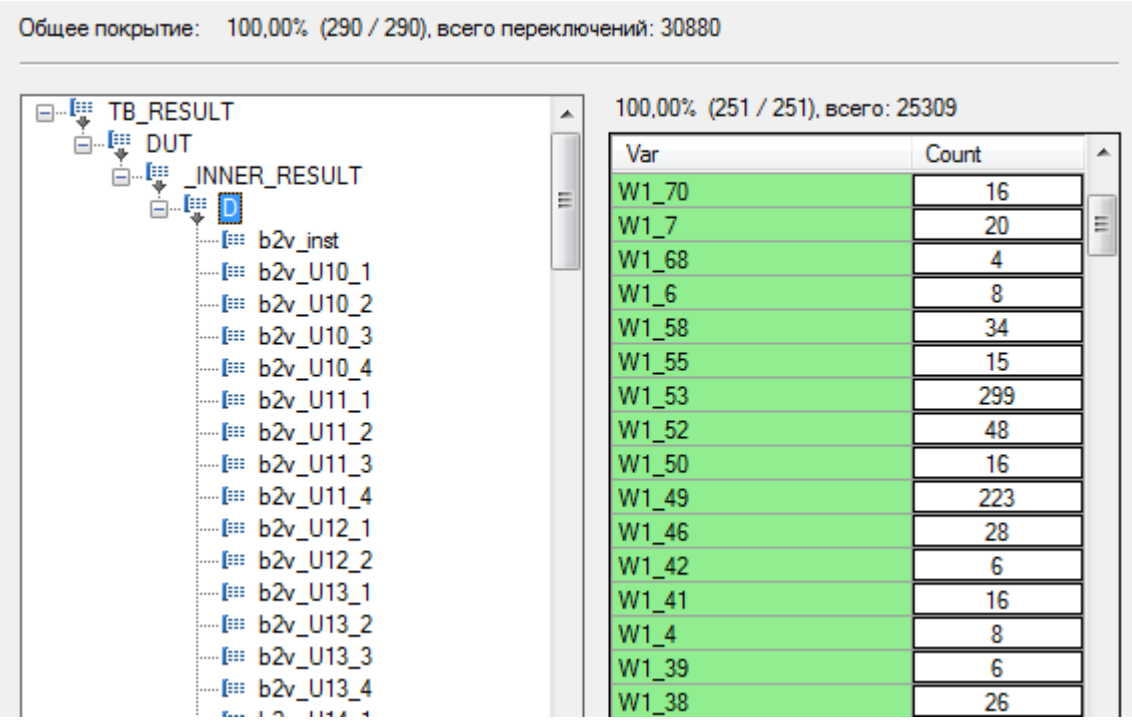

Рисунок 18. Тестовое покрытие внутренних сигнальных линий.

Выводы

1. Произведено программное моделирование входящих в состав объекта контроля микросхем и осуществлена проверка корректности функционирования составленных моделей.

2. Создана программная модель цифрового модуля Субблок 031 в среде автоматизированного проектирования Altera Quartus 2.

3. Предложен алгоритм составления тестовой последовательности.

4. Разработана программа, автоматизирующая составление тестовых воздействий и предоставляющая информацию, которая может быть использована для локализации неисправностей тестируемого устройства.

5. При помощи системы «SimTest» установлено, что полученная вследствие реализации предложенного алгоритма последовательность тестовых воздействий обладает полным покрытием объекта контроля, активирует все внутренние сигнальные линии и удовлетворяет критерию полноты проверки работы компонентов устройства.

Таким образом, все промежуточные этапы были выполнены, и поставленная задача решена в полном объеме.

Заключение

В работе рассмотрены современные методы тестирования цифровых устройств и представлены этапы процесса разработки тест-программы в контексте методики функционального тестирования.

Разработана программа генерации последовательности тестовых воздействий, значительно сокращающая время составления тест-программ и предоставляющая информацию для локализации неисправностей. Полученная последовательность обладает полным тестовым покрытием цифрового модуля и удовлетворяет критерию полноты проверки функциональности элементов устройства, что подтверждается представленными результатами функционального моделирования объекта контроля.

Литература

1. Городецкий А. Снова о внутрисхемном тестировании ICT. Часть 1 // Компоненты и технологии, 2011. № 7. С. 58–59.
2. Городецкий А. Снова о внутрисхемном тестировании ICT. Часть 2 // Компоненты и технологии, 2011. № 8. С. 44–45.
3. Городецкий А. Снова о внутрисхемном тестировании ICT. Часть 3 // Компоненты и технологии, 2011. № 9. С. 6–7.
4. Albee A. J. The evolution of ICT: PCB technologies, test philosophies, and manufacturing business models are driving in-Circuit test evolution and innovations // IPC APEX EXPO Conference and Exhibition 2013, 1. P. 381–401.
5. Holtzer M. In-circuit pin testing: An excellent potential source of value creation // SMT Surface Mount Technology Magazine, 2015, 30 (6). P. 68–71.
6. Nelson R. Systems and software support PCB test // EE: Evaluation Engineering, 2013, 52 (2). P. 14–17.
7. IEEE Std. 1149.1 – Standard Test Access Port and Boundary-Scan Architecture. http://grouper.ieee.org/groups/1149/1.
8. Renbi A., Delsing J. Application of Contactless Testing to PCBs with BGAs and Open Sockets // Journal of Electronic Testing: Theory and Applications, 2015, 31 (4). P. 339–347.
9. Renbi A., Delsing J. Contactless Testing of Circuit Interconnects // Journal of Electronic Testing: Theory and Applications, 2015, 31 (3). P. 229–253.
10. Wang, R., Chakrabarty, K., Bhawmik, S. Interconnect testing and test-path scheduling for interposer-based 2.5-D ICs // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34 (1), art. no. 6936331. P. 136–149.

11. Ren X., Tavares V.G., Blanton R. D. S. Detection of illegitimate access to JTAG via statistical learning in chip // Proceedings – Design, Automation and Test in Europe, 2015, art. no. 7092367. P. 109–114.
12. Nelson R. JTAG and embedded test complement ATE // EE: Evaluation Engineering, 2014, 53 (3). P. 14–17.
13. Shashidhara H. B., Yellampalii S., Goudanavar V. Board level JTAG/boundary scan test solution // Proceedings of International Conference on Circuits, Communication, Control and Computing, 2014, art. no. 7057760. P. 73–76.
14. Yin X.H., Xu C.F. On a method of getting test data for boundary scan interconnection test in multiple scan chains // Advanced Materials Research, 2014, 986–987. P. 1531–1535.
15. Peng K. B., Zhang J. T. Reconfigurable boundary scan tester using cellular-automata register technology // Advanced Materials Research, 2014, 1006–1007. P. 986–989.
16. Wang R., Chakrabarty K., Eklow B. Scan-based testing of post-bond silicon interposer interconnects in 2.5-D ICs // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2014, 33 (9), art. no. 6879596. P. 1410–1423.
17. Deng X., Xu S., Zhang Y. An approach to generating test data sequences of boundary scan test system // Proceedings of 2013 IEEE 11th International Conference on Electronic Measurement and Instruments,2013, 1, art. no. 6743004. P. 264–270.
18. Sangi R., Baranski M., Oltmanns J., Streblow R., Müller D. Modeling and simulation of the heating circuit of a multi-functional building // Energy and Buildings, 2016, 110. P. 13–22.
19. Fujita M., Taguchi N., Iwata K., Mishchenko A. Incremental ATPG methods for multiple faults under multiple fault models // Proceedings – International Symposium on Quality Electronic Design, 2015, art. no. 7085420. P. 177–180.

20. Hobeika C., Thibeault C., Boland J.-F. Functional constraint extraction from register transfer level for ATPG // IEEE Transactions on Very Large Scale Integration 2015, vol. 23, 2, art. no. 6778092. P. 407–412.
21. Kochte M. A., Elm M. b , Wunderlich H.-J. Accurate X-propagation for test applications by SAT-based reasoning // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, vol. 31, 12, art. no. 6349431. P. 1908–1919.
22. Bhowmik B., Deka J. K., Biswas S. Beyond test pattern generation: Coverage analysis // International Conference on Industrial Instrumentation and Control, 2015, art. no. 7151009. P.1620–1625.
23. Thoulath Begam V. M., Baulkani S. Compact test set method for high fault coverage test pattern generation // International Journal of Applied Engineering Research, 2015, vol. 10, 55. P. 453–458.
24. Matinnejad R., Nejati S., Briand L., Bruckmann T., Poull C. Search-based automated testing of continuous controllers: Framework, tool support, and case studies // Information and Software Technology, 2015, vol. 57, 1. P. 705–722.
25. Ghiduk A. S. Automatic generation of basis test paths using variable length genetic algorithm // Information Processing Letters, 2014, vol. 114, 6. P. 304–316.
26. Melnik V. I., Mikhailov A. N., Grishkin V. M., Ovsyannikov D. A., Yelaev Y. V. Methods of modeling of the test inputs for analysis the digital devices // International conference on computer technologies in physical and engineering applications, 2014. P. 112–113.
27. Grishkin V., Yelaev Y., Lopatkin G., Mikhailov A., Ovsyannikov D. Interface method of digital devices testing // Tenth International Vacuum Electron Sources Conference & Second International Conference on Emission Electronics, 2014. P. 107–108.
28. Гришкин В. М., Лопаткин Г. С, Михайлов А. Н., Овсянников Д. А. Интерфейсный метод построения моделей входных воздействий для

тестирования электронных цифровых модулей // Вопросы радиоэлектроники, серия ОТ. 2013. № 1. С. 80–88.

29. Гришкин В. М., Степанов Ю. Л., Лопаткин Г. С., Большаков А. А. Подход к разработке тестов цифровых электронных модулей для автоматического тестового оборудования // Вопросы радиоэлектроники. 2013. Т. 1. № 1. С. 89–99.
30. Melnik V. I., Mikhailov A. N., Grishkin V. M., Ovsyannikov D. A., Yelaev Y. V. Modeling methods of the test inputs for analysis the digital devices // 2nd International Conference on Emission Electronics Selected papers. 2014. P. 48–50.
31. Михайлов А. Н., Мельник В. И., Овсянников Д. А. Тестовый контроль и диагностика радиоэлектронной аппаратуры // Электроника: Наука, технология, бизнес. 2013. № S (128). С. 114–117.
32. Елаев Е. В., Степанов Ю. Л., Ферсенков В. В. Подходы к моделированию микропроцессоров для построения контрольно-диагностических тестов // Процессы управления и устойчивость, 2015. Т. 2. № 1. С. 398–403.
33. Елаев Е. В. Интерфейсный метод автоматизированной генерации тестовых воздействий для цифровых радиоэлектронных объектов контроля // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1: Естественные и технические науки, 2015. № 4. С. 19–24.
34. Лопаткин Г. С. Подход к автоматизации тестирования электронных цифровых устройств // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. 2013. № 4. С. 90–98.
35. Мащинский Н. С., Елаев Е. В., Федюкович П. А. Моделирование сложных цифровых устройств с целью их тестирования // Процессы управления и устойчивость. 2015. Т. 2. № 1. С. 452–457.

36. Мельник В. И., Гришкин В. М., Михайлов А. Н., Овсянников Д. А. Методика разработки тест-программ контроля и диагностики цифровых устройств с использованием САПР «SimTest» // Электроника: Наука, технология, бизнес. 2013. № S (128). С. 118–124.
37. Степанов Ю. Л., Гришкин В. М., Елаев Е. В., Федюкович П. А. Развитие программной среды «Ястек» и ее использование при написании тестовых программ для цифровых модулей // Вопросы радиоэлектроники, 2015. № 2 (2). С. 198–205.
38. Степанов Ю. Л., Гришкин В. М., Большаков А. А., Лопаткин Г. С., Ким М. А. Автоматизированное построение тестов цифровых электронных модулей для комплекса тестового контроля и диагностики УТК-512 // Вопросы радиоэлектроники. 2012. Т. 1. № 1. С. 79–89.
39. Шило В. Л. Популярные цифровые микросхемы. Справочник. М.: Радио и связь, 1987. 352 с.
40. Потехин В. А. Схемотехника цифровых устройств: учебное пособие для вузов. Томск: В-Спектр, 2012. 250 с.
41. Угрюмов Е. П. Цифровая схемотехника: учебное пособие для вузов. СПб: БХВ-Петербург, 2010. 816 с.
42. Справочник по микросхемам КР1533 серии. http://www.datasheet-pdf.ru.
43. Кондратенко Ю. П., Мохор В. В., Сидоренко С. А. Verilog-HDL для моделирования и синтеза цифровых электронных схем. Учебное пособие / Под ред. д.т.н., профессора Ю. П. Кондратенко. Николаев: Изд-во НГГУ им. Петра Могилы, 2002. 206 с.
44. Николай К. Введение в Verilog – язык описания цифровых схем. http://marsohod.org/verilog.
45. Intel Quartus Prime Standard Handbook. https://www.altera.com/.
46. Мащинский Н. С., Елаев Е. В., Нуракунов А., Гусев О. А. Автоматизация генерации тестовых воздействий для комбинационных цифровых схем // Процессы управления и устойчивость. 2016. № 1. С. 389–393.

47. Мащинский Н. С., Пушко Ф. А. Автоматизация генерации тестовых сигналов для цифровых схем с элементами памяти // Процессы управления и устойчивость. 2017. № 1. С. 428–432.
48. Мащинский Н. С., Елаев Е. В. Генерация тестовых воздействий для диагностики цифровых электронных систем // Вестник Санкт-Петербургского университета технологии и дизайна. Серия 1: Естественные и технические науки. 2017. № 4 С. 41–45.