Министерство образования и науки Украины
Севастопольский национальный технический университет

Кафедра радиотехники и телекоммуникаций

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

ПРОЕКТИРОВАНИЕ КОМБИНАЦИОННЫХ ЦЕПЕЙ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА *VERILOG*

по дисциплине «Основы проектирования цифровых ИС»
для студентов дневной формы обучения
направления 6.0509 — Радиотехника

Севастополь
2012

УДК 621.374 (075.8)

Методические рекомендации по выполнению лабораторной работы «Проектирование комбинационных цепей с использованием языка *Verilog*» по дисциплине «Основы проектирования цифровых ИС» для студентов дневной формы обучения направления 6.0509 — Радиотехника / Сост. В. В. Вертегел — Севастополь: Изд-во СевНТУ, 2012. — 20 с.

Целью рекомендаций является оказание помощи студентам в выполнении лабораторной работы по дисциплине «Основы проектирования цифровых ИС».

Изложены краткие сведения о комбинационных цепях, методике их проектирования. Даны основные сведения о встроенных примитивах языка *Verilog*. Сформулированы варианты задания к лабораторной работе, требования к отчету, вопросы для самопроверки. Рекомендована дополнительная литература.

Методические рекомендации рассмотрены и утверждены на заседании кафедры радиотехники (протокол № ____ от _____ 2012 г.)

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент:

СОДЕРЖАНИЕ

1. Цель работы и подготовка к работе

Целью работы является приобретение начальных сведений о проектировании комбинационных цепей с использованием языка описания аппаратуры *Verilog* на логическом (вентильном) уровне описания.

Подготовка к работе включает в себя изучение данных методических указаний и рекомендованной в них литературы, подготовку ответов на приводимые вопросы для самопроверки, проработку контрольного примера в соответствии со своим вариантом задания (см. 4.1), составление Verilog-описания проектируемого устройства (см. 4.2, 4.3).

2. Краткие сведения о комбинационных цепях

2.1 Понятие о комбинационных цепях

Узлы цифровых устройств (цифровые узлы) подразделяют на комбинационные и последовательностные. Комбинационные узлы называют также комбинационными цепями (КЦ) или комбинационными схемами, а также автоматами без памяти. Последовательностные узлы называют автоматами с памятью (АП), последовательностными цепями или схемами. КЦ и АП принципиально различаются между собой по функционированию (поведению). Значения выходных сигналов КЦ (после завершения переходных процессов, ограничивающих быстродействие любых цифровых узлов) зависят только от текущих значений входных сигналов. Как говорят, предыстория значения не имеет. В отличие от этого, значения выходных сигналов АП (состояние АП) зависят не только от значений сигналов, поступающих на их входы, но и от того состояния автомата, которое предшествует подаче этих входных сигналов и определяется ранее действовавшими на автомат входными сигналами (предысторией). Указанные функциональные (поведенческие) различия связаны со структурными различиями между КЦ и АП: для АП характерны внутренние обратные связи, т. е. замкнутые петли прохождения сигналов, а в КЦ такие обратные связи отсутствуют.

2.2 Описание цифровых устройств при их проектировании

Проектируемое цифровое устройство (как говорят, объект проектирования или просто объект) прежде всего должно быть формально описано. Известны два основных подхода к описанию таких объектов: **структурный** (структурное описание или описание на структурном уровне) и **поведенческий** (поведенческое описание, описание на уровне поведения или функциональное описание).

Структурное описание — это описание устройства (его структуры) в виде совокупности

компонентов и связей между компонентами. Такое описание может осуществляться в следующих формах:

1. Графическая форма — форма графически представленной схемы.

2. Текстовая форма — текст на одном из специальных HDL языков описания аппаратуры (например, *Verilog*).

Поведенческое описание устройства — это описание зависимостей его состояния (выходных сигналов) от входных сигналов и от предшествующего (подаче упомянутых входных сигналов) состояния. Иначе говоря, поведенческое описание задает функцию или алгоритм, реализуемый устройством. В частности, комбинационные цепи описываются так называемыми переключательными (логическими) функциями.

Переключательной функцией называют логическую функцию, способную принимать одно из двух значений (0 или 1) в зависимости от значений нескольких ее аргументов. При проектировании КЦ с несколькими (n) выходами каждому из возможных наборов входных переменных должно быть поставлено в соответствие не одно двоичное значение функции, а определенное n-разрядное двоичное число (n двоичных значений, где $n > 1$). Тогда получается зависимость, которую можно интерпретировать как совокупность n переключательных функций. Эту совокупность называют системой переключательных функций (заметим, что значение n в общем случае не совпадает с количеством входных переменных).

Поведенческое описание может иметь следующие формы:

1. Описание требуемого логического преобразования таблицей функционирования (таблицей истинности). Например, табл. 2.1 описывает переключательную функцию F трех аргументов ($X2$, $X1$, $X0$) , которая должна быть реализована некоторой проектируемой комбинационной цепью. Для описания автоматов с памятью иногда используется представление таблиц функционирования в формах карт Карно [2].

Таблица 2.1 — Таблица истинности комбинационной схемы

$X2$	$X1$	$X0$	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

2. Аналитическое описание логического преобразования, которое должно осуществляться проектируемым устройством (логические выражения). В частности, переключательная функция может быть представлена в так называемой совершенной дизъюнктивной нормальной форме (СДНФ). Для этого составляют дизъюнкцию тех наборов аргументов, на которых функция принимает единичное значение. Каждый из этих наборов записывается в виде конъюнкции аргументов. Причем нулевому значению какого-либо аргумента в наборе соответствует запись инверсии данного аргумента в конъюнкции (в коньюнктивном терме). Так, для функции, заданной табл. 2.1, СДНФ имеет вид:

$$F = \overline{X}_2\overline{X}_1\overline{X}_0 \vee \overline{X}_2X_1\overline{X}_0 \vee \overline{X}_2X_1X_0 \vee X_2X_1\overline{X}_0.$$

3. Описание требуемого логического преобразования с помощью временных диаграмм сигналов на входах и выходах объекта.

4. Описание объекта так называемой диаграммой состояний (см. с. 106 в [2]).

5. Текстовая форма, которая, как и при структурном описании, может представлять собой текст на одном из специальных *HDL* языков описания аппаратуры (*Verilog*, *VHDL*).

Любой из подходов к описанию объекта при любой форме может быть непосредственно использован для проектирования. Однако для не которых средств современных САПР могут оказаться предпочтительными какой-то определенный из подходов и определенная его форма. Тогда при неподходящем первоначальном описании устройства нужно перейти к предпочтительному описанию. Так, выше приведен пример перехода от табличной формы поведенческого описания КЦ к форме аналитического описания. Заметим, что в данной лабораторной работе изучается текстовая форма описания — *Verilog*-описание комбинационных цепей на вентильном уровне. Для его применения требуется исходное табличное описание объекта преобразовать к аналитическому и далее к текстовому *Verilog*-описанию на логическом (вентильном) уровне абстракции с использованием встроенных примитивов языка *Verilog*.

3. Синтез комбинационных цепей на основе встроенных Verilog примитивов

В состав языка *Verilog* входит набор стандартных примитивов, реализующих основные логические операции, такие как **and**, **or**, **not**, **xor** и прочие, идентификаторы которых совпадают с названиями соответствующих операций. При этом следует обратить внимание, что выходной сигнал в списке интерфейса стандартных логических модулей размещен первым, а входные сигналы, соответственно, — на втором и третьем месте.

Так, например, запись

and (A, B, C);

означает A =B and C, а не C = A and B.

Рассмотрим разработку функциональных моделей устройств на логическом уровне абстракции на примере комбинационной цепи, представленной на рис. 3.1.

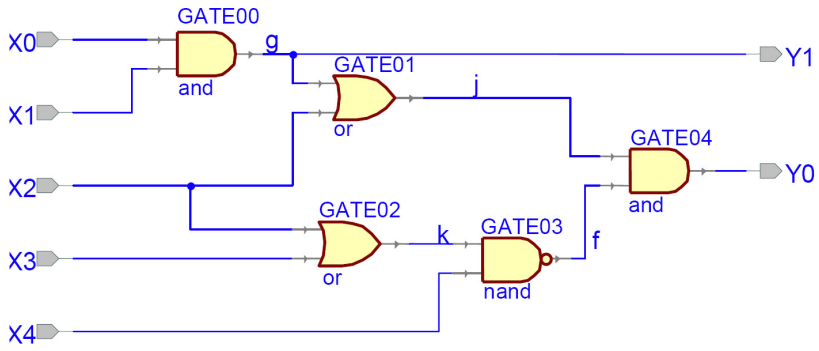
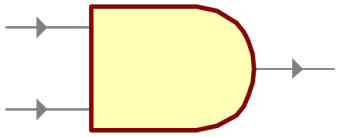


Рис. 3.1. — Схема комбинационной цепи

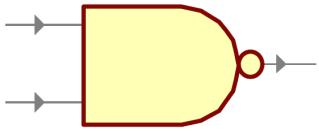
Verilog-описание приведенной комбинационной цепи может быть реализовано на вентильном уровне при помощи встроенных примитивов в виде модуля «comb_net»:

```
module comb_net (X0 ,X1, X2, X3, X4, Y0, Y1 );
 //Описания портов и внутренних сигналов примитива:
    input X0, X1, X2, X3, X4;            // Входные сигналы модуля
    output Y0, Y1;                       // Выходные сигналы модуля
    wire X0, X1, X2, X3, X4, Y0, Y1;
// Определение внутренних сигналов примитива, связывающих
// отдельные структурные компоненты:
    wire k , j, f, g;
//Описание внутренней структуры примитива:
// Включение стандартного примитива and под именем GATE00:
    and GATE00 (Y1, X0, X1);
// Включение прочих стандартных примитивов:
    or GATE01 (j, Y1, X2);
    or GATE02 (k, X2, X3);
    nand GATE03 (f, k, X4);
    and GATE04 (Y0, f, j);
endmodule
```

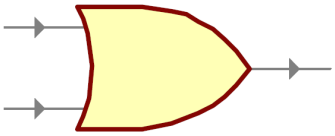
Полный список стандартных логических элементов языка Verilog приведен на рис. 3.2.



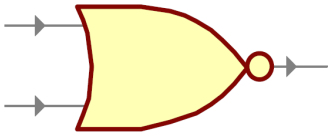
а) элемент И — примитив **and**



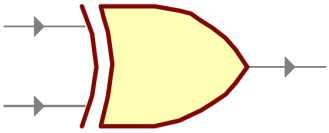
б) элемент И-НЕ — примитив **nand**



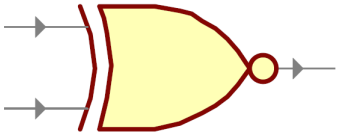
в) элемент ИЛИ — примитив **or**



г) элемент ИЛИ-НЕ — примитив **nor**



д) элемент исключающее ИЛИ — примитив **xor**



е) элемент исключающее ИЛИ-НЕ — примитив **xnor**

Рис. 3.2. — Стандартные логические элементы языка Verilog

В отличие от модулей, создаваемых пользователем, стандартные логические элементы — примитивы не имеют фиксированного количества входных портов. На вход всех элементов, перечисленных на рис. 3.2, можно подавать произвольное количество входных сигналов, но при этом максимальное количество ограничивается используемой программой синтеза.

Например, для реализации цепи, приведенной на рис. 3.3, достаточно использовать следующий оператор включения модуля:

```
wire input0, input2, input3, input3, output0;
and (output0, input0, input1, input2, input3);
```

который осуществит операцию свертки по логической операции **И** всех входных сигналов.

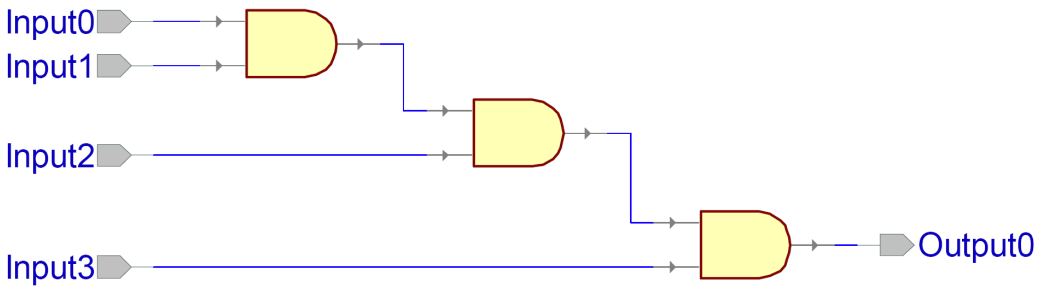
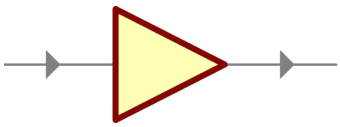


Рис. 3.3. — Логическая цепь, состоящая из 3-х элементов И

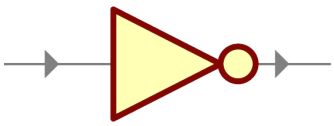
Дополнительно, кроме перечисленных выше примитивов, Verilog содержит шесть базовых логических примитивов: **buf**, **not**, **bufif1**, **bufif0**, **notif1**, **notif0**.

Примитивы **buf** и **not** (рис. 3.4) имеют один входной сигнал и произвольное число выходных сигналов. Входной сигнал всегда должен размещаться последним в списке портов оператора включения примитива. Эти блоки выполняют функцию разветвления сигналов. Кроме того, элемент **not** осуществляет инверсию всех выходных сигналов.



input	output
0	0
1	1
X	X
z	X

а) примитив **buf**

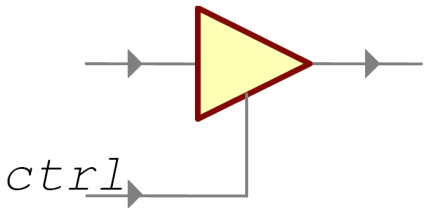


input	output
0	1
1	0
X	X
z	X

б) примитив **not**

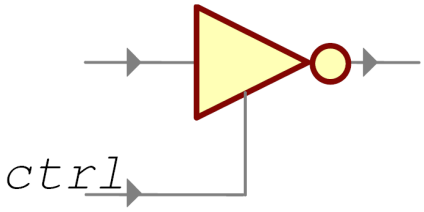
Рис. 3.4. — Стандартные примитивы **buf** и **not**

Примитивы **bufif1**, **bufif0**, **notif1** и **notif0** (рис. 3.5) отличаются от блоков **buf** и **not** наличием управляющего входа ***ctrl***. Если на управляющем входе блока **bufif1** или **notif1** установлен низкий уровень сигнала, то, независимо от состояния второго входа, на выход блока поступает сигнал **z**. Примитивы **bufif0** и **notif0** функционируют аналогичным образом, но с учетом инверсии сигнала ***ctrl***.



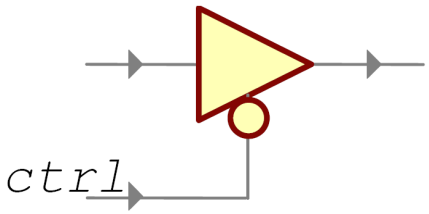
in \ ctrl	0	1	x	z
0	z	0	L	L
1	z	1	H	H
x	z	x	x	x
z	z	x	x	x

а) элемент **bufif1**



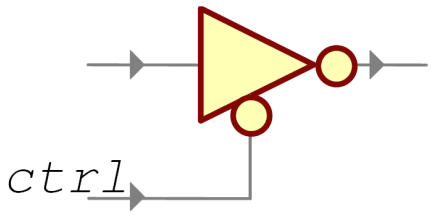
in \ ctrl	0	1	x	z
0	z	1	H	H
1	z	0	L	L
x	z	x	x	x
z	z	x	x	x

б) элемент **notif1**



in \ ctrl	0	1	x	z
0	0	z	L	L
1	1	z	H	H
x	x	z	x	x
z	x	z	x	x

в) элемент **bufif0**



in \ ctrl	0	1	x	z
0	1	z	H	H
1	0	z	L	L
x	x	z	x	x
z	x	z	x	x

г) элемент **notif0**

Рис. 3.5. — Стандартные примитивы **bufif1**, **notif1**, **bufif0** и **notif0**

На примере, приведенном на рис 3.6, рассматривается применение стандартных примитивов **bufif1** для подключения выходов нескольких устройств к общей шине, что и составляет основную задачу их функционирования.

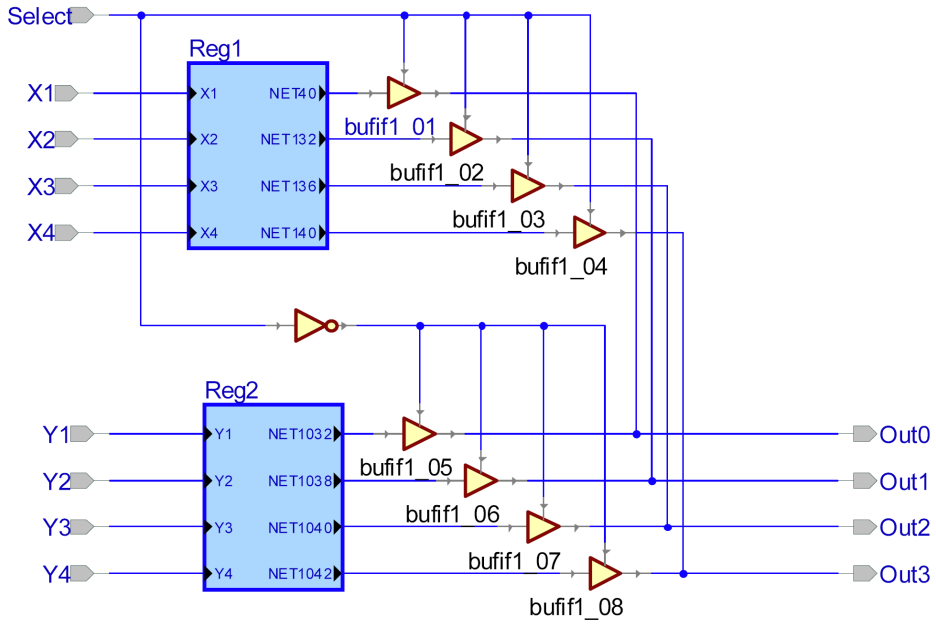


Рис. 3.6 — Схема подключения двух регистров к общей шине

Устройство, показанное на рис. 3.6, содержит два четырехбитных регистра **Reg1** и **Reg2**, подключенных посредством элементов **bufif1** к общему четырехбитному выходу [**Out3** : **Out0**]. Вход **Select** предназначен для выбора того из регистров, выходной сигнал которого будет послан на общий выход устройства.

Если на входной порт **Select** подается высокий уровень сигнала, соответствующий логической 1, то входы **ctrl** блоков **bufif1_01** … **bufif1_04** получают значение 1, а входы **ctrl** блоков **bufif1_05** … **bufif1_08** устанавливаются в 0. Следовательно, в соответствии с таблицей истинности, блоки **bufif1_01** … **bufif1_04** пропускают без изменений значения сигналов, поступающих от регистра **Reg1**, а на выходах блоков **bufif1_01** … **bufif1_04** устанавливается сигнал z (высокий импеданс).

При одновременном присваивании одному и тому же сигналу различных значений в качестве результата присваивания выбирается то из значений, приоритет которого выше. Так как приоритет высокого импеданса z наиболее низкий, то любой сигнал, поступающий от регистра **Reg1**, будет иметь более высокий приоритет и, следовательно, именно этот сигнал устанавливается на общем выходе устройства.

Если же на вход **Select** поступает сигнал логического 0, то, наоборот, компоненты

bufif1_01 … bufif1_04 блокируют прохождение сигналов от регистра **Reg1**, а сигналы регистра **Reg2** беспрепятственно поступают на выходы **Out3 … Out0**.

Представление о логическом синтезе будет неполным без обсуждения вопросов управления временными аспектами моделей.

Известно, что все реальные логические элементы обладают определенным временем задержки. Язык *Verilog* предоставляет средства управления моделированием таких задержек. Для стандартных логических элементов существует возможность управления временем нарастания, временем спада и временем отключения выходного сигнала.

Под временем нарастания понимается интервал модельного времени, в течение которого выходной сигнал устройства переключается от любого значения к 1. Задержка спада и задержка отключения представляют собой время перехода выхода устройства к состояниям 0 или z, соответственно.

Синтаксис установки параметров задержек в операторах включения стандартных логических модулей можно представить следующим образом:

тип_элемента #(t_rise, t_fall, t_off) ***element_name*** (выход, список входов, сигнал управления);

где **тип_элемента** — идентификатор включаемого стандартного модуля (рис. 3.3, 3.4);

element_name — уникальный идентификатор данного элемента;

t_rise — задержка возрастания;

t_fall — задержка спада;

t_off — задержка отключения (только для элементов со входом управления).

Все задержки измеряются в шагах модельного времени.

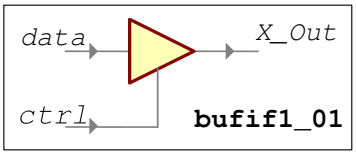
Для иллюстрации управления задержками рассмотрим работу следующего модуля:

```
`timescale 10 ns/1ps       // Шаг моделирования равен 10 ns

module Logic_Delay_Demo (data, ctrl, X_Out);
     output X_Out ;
     wire X_Out ;
     input data, ctrl;
     wire data, ctrl;
// Под уникальным именем bufif1_01 включается стандартный модуль bufif1
// со следующими параметрами задержек:
// t rise = 1*10 ns = 10 ns
// t_fall = 2*10 ns = 20 ns
// t_off  = 3*10 ns = 30 ns

     bufif1 #(1, 2, 3) bufif1_01 (X_Out, data, ctrl);

endmodule
```



В результате моделирования работы данного модуля получена временная диаграмма, представленная на рис. 3.7.

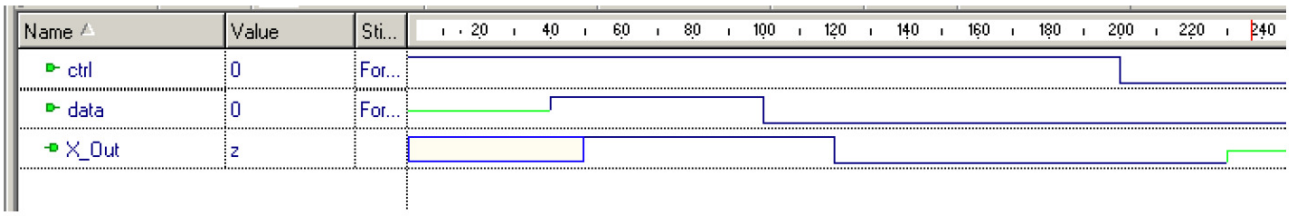


Рис. 3.7 — Временная диаграмма, илюстрирующая задержки логичесого элемента

Из диаграммы (рис. 3.7) видно, что событие, приводящее к установке на выходе модуля **bufif1_01** логической единицы наступает в момент времени $T_{mod} = 40\ ns$.

Соответственно, изменение выходного сигнала должно произойти в момент времени:

$$T_{rise} = 40\ ns + t\_rise = 50\ ns,$$

что и подтверждается результатами моделирования. Аналогично совпадают времена переключения выходного сигнала в 0:

$$T_{fall} = 100\ ns + t\_fall = 120\ ns$$

и отключения устройства (установки выхода в z):

$$T_{off} = 200\ ns + t\_off = 230\ ns.$$

4. Задание на проектирование

4.1. Общая формулировка задания и его варианты

В качестве задания к данной лабораторной работе предлагается проектирование генератора сигнала псевдослучайной последовательности (ПСП). ПСП представляет собой последовательность элементов d_i (i = 0, 1, 2, 3, … ∞ — номер элемента последовательности), повторяющихся с периодом N = 8, т. е. $d_{i+8} = d_i$.

Нулевой элемент последовательности $d_0 = 0$, а каждый последующий, начиная с d_1 по d_7, может принимать одно из двух значений: +1 или –1, в зависимости от варианта задания. Чтобы записать указанный отрезок ПСП, необходимо выбрать первые четыре элемента последовательности (d_0, d_1, d_2, d_3) согласно номеру варианта задания (табл. 4.1), а последующие четыре элемента определить по следующим рекуррентным соотношениям:

$d_i = -d_{i-3} \times d_{i-2}$ — для вариантов 1…7;

$d_i = -d_{i-3} \times d_{i-1}$ — для вариантов 8…14.

Таблица 4.1 — Варианты индивидуальных заданий

№ варианта задания	d_0, d_1, d_2, d_3	№ варианта задания	d_0, d_1, d_2, d_3
1	0, –1, –1, +1	8	0, –1, –1, +1
2	0, –1, +1, –1	9	0, –1, +1, +1
3	0, +1, –1, +1	10	0, +1, +1, +1
4	0, –1, +1, +1	11	0, +1, +1, –1
5	0, +1, +1, +1	12	0, +1, –1, +1
6	0, +1, +1, –1	13	0, –1, +1, –1
7	0, +1, –1, –1	14	0, +1, –1, –1

Следует отметить, что указанный отрезок представляет только первый период последовательности, а каждый последующий период ПСП должен быть его копией, при этом всю последовательность элементов достаточно ограничить четырьмя периодами.

Кроме того, каждый элемент последовательности (0, +1, –1) нужно сформировать проектируемым генератором в 4-разрядном дополнительном коде (элементу 0 соответствует код 0000, элементу +1 соответствует код 0001, а элементу –1 соответствует код 1111).

Требуется разработать указанный генератор ПСП в виде модуля на языке Verilog с использованием логического уровня описания и провести его моделирование.

4.2. Структурное описание проектируемого устройства и его компонентов в текстовой форме

Рассмотрим пример составления *Verilog*-описания проектируемого устройства (перехода к текстовой форме его структурного описания) с последующим использованием программы **ModelSim** в качестве средства ввода описания и моделирования проектируемого устройства.

В *Verilog* генератор ПСП можно иерархически представить как модуль, содержащий совокупность субмодулей (модулей более низкого уровня иерархии). В качестве примера разобьем его на три модуля: генератор элементов отрезка; счетчик элементов отрезка; кодер:

Генератор элементов отрезка можно построить на основе комбинационной схемы (рис. 4.1) . Эта схема будет формировать последовательность элементов d_i в пределах одного периода. Для формирования восьми элементов необходимо иметь три адресных входа (a0, a1, a2), на которые от модуля счетчика элементов отрезка будем подавать двоичные коды номера элемента ($x2$, $x1$, $x0$).

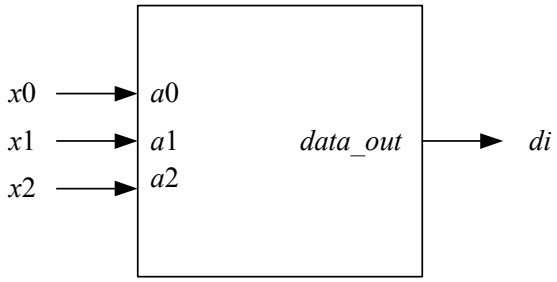


Рис.4.1— Модуль генератора элементов отрезка ПСП

Адресные входы модуля генератора элементов отрезка предназначены для декодирования трехразрядных двоичных порядковых номеров от 000 до 111 в параллельном коде, что соответствует десятичным значениям номеров от 0 до 7.

Сигналы $\boldsymbol{d_i}$ на выходе ***data_out*** генератора элементов отрезка следует задать по следующему принципу: элементу +1 будет соответствовать логическая единица, для формирования элементов последовательности со значениями –1 и 0 будет использовать логический нуль. Поскольку элемент со значением 0 для всех вариантов расположен в последовательности под нулевым порядковым номером (000), то для его кодирования не требуется использовать дополнительный разряд сигнала d_i.

В результате на выходе генератора элементов отрезка образуется последовательность логических единиц и нулей, которая в дальнейшем должна быть преобразована в последовательность дополнительных кодов элементов ПСП (чисел +1, –1 и 0 соответственно).

Пусть в качестве примера задана последовательность d_i = 0, +1, –1, +1, +1, –1, –1, +1.

В соответствии с вышеуказанным принципом получаем двоичную последовательность d_i = 01011001. Приведенный алгоритм формирования элементов ПСП можно представить в виде таблицы истинности комбинационной схемы (табл. 4.2)

Таблица 4.2 — Таблица истинности генератора элементов отрезка ПСП

$x2$	$x1$	$x0$	d_i
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Составим СДНФ в соответствии с правилами, приведенными в параграфе 2.2.

$$F = \overline{x2} \cdot \overline{x1} \cdot x0 + \overline{x2} \cdot x1 \cdot x0 + x2 \cdot \overline{x1} \cdot \overline{x0} + x2 \cdot x1 \cdot x0 . \qquad (4.1)$$

Полученная функция может быть упрощена, однако для составления *Verilog*-описания модуля, реализующего указанную функцию в виде комбинационой цепи на логическом уровне, делать это не будем. Из (4.1) видно, что для синтеза этой комбинационой цепи, необходимо использовать логические элементы: три элемента **НЕ** для получения сигналов $\overline{x2},\ \overline{x1},\ \overline{x0}$; четыре трёхвходовых **И** и один четырехвходовый **ИЛИ**.

В качестве примера для составления *Verilog*-описания комбинационной цепи можно воспользоваться структурой модуля, приведенного на стр. 7.

Для формирования порядковых номеров генерируемых элементов ПСП на адресных входах генератора элементов отрезка можно использовать счетчик (модуль **counter** из лабораторной работы № 1). У этого модуля (рис 4.2) должен быть модуль счета N = 8 (добавлена цепь сброса или удален один Т-триггер). Это позволит получить 3-битный счетчик тактовых импульсов (синхроимпульсов), поступающих на счетный вход **clk**, который изменяет своё состояние по передним фронтам этих импульсов при условии высого уровня сигнала на входе асинхронного сброса **reset**.

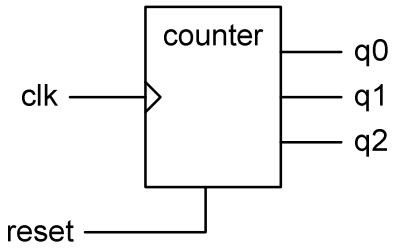


Рис. 4.2 — Модуль синхронного счетчика

Исходным является нулевое состояние счетчика (000 на выходной 3-разрядной шине). При этом счетчик готов к реакции на сигнал **reset** (в исходном состоянии **reset** = 1). Для начала счета и последовательного изменения выходных сигналов счетчика на вход **reset** подается уровень логического нуля, который должен сохраняться до завершения формирования четырех периодов ПСП. Результат счета (количество импульсов, поступивших на вход **clk**) отражается двоичным кодом на выходах **q0**, **q1**, **q2.** Выход **q0** соответствует младшему разряду выходного числа, а выход **q2** — старшему. После 7-го тактового импульса с момента начала счета на выходах появится число 7 в двоичном коде

111. Следующим (8-м) тактовым импульсом счетчик сбрасывается в исходное состояние 000. Если в это время на входе **reset** действует уровень логического нуля, то счет повторяется. В противном случае счет прекращается, т. е. счетчик остается в нулевом состоянии.

В результате формирования сигнала **reset** = 0 необходимой длительности (например, 8 периодов повторения импульсов **clk**) позволяет реализовать однократную последовательную смену восьми состояний счетчика от 000 до 111 с последующим возвратом в состояние 000. Как отмечалось ранее, состояния счетчика являются порядковыми номерами элементов отрезка ПСП (элементов сигнала) и в соответствии с заданием должны обеспечить периодическое формирование этих элементов, в количестве четырех периодов.

Генерацию сигналов **clk** и **reset** необъодимо выполнить в модуле **test_bench**: эти сигналы должны задаваться с помощью соответствующих конструкций *Verilog*, как это сделано, например, в модуле **test_bench** лабораторной работы № 1.

Для преобразования последовательности логических единиц и нулей в дополнительные двоичные коды чисел 0, +1 и –1 сигнал с выхода модуля генератора элементов отрезка необходимо подать на вход кодера. Одним из вариантов реализации кодера является шинный мультиплексор (рис. 4.3).

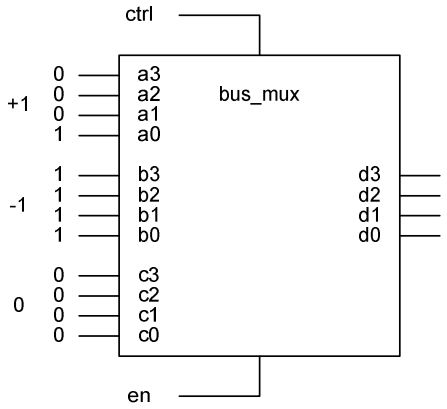


Рис. 4.3 — Модуль кодера (шинный мультиплексор)

Шинный мультиплексор размерности 3 : 1, можно выполнить на стандартных примитивах **bufif1**, **bufif0** или **notif0**, **notif1**.

На рис. 4.3 входы (**a0**, **a1**, **a2**, **a3**), (**b0**, **b1**, **b2**, **b3**) и (**c0**, **c1**, **c2**, **c3**) образуют три входные информационные шины мультиплексора, (**d0**, **d1**, **d2** , **d3**) — его выходная шина. Выходная шина мультиплексора будет выходной шиной всего генератора ПСП.

Поскольку входных шин три, то для выбора одной из них необходимо два адресных входа. Одним адресным входом является вход **ctrl**: при логическом нуле на этом входе выбирается первая из входных шин (**a0**, **a1**, **a2**, **a3**), а при логической единице — вторая (**b0**, **b1**, **b2**, **b3**).

Вторым адресным входом кодера является вход **en** (enable): при **en** = 0 к выходу шинного мультиплексора подключается одна из входных шин, определяемых сигналом **ctrl**, а при **en** = 1 выходная шина подключается к щине **c0**, **c1**, **c2**, **c3**.

На вход кодера **ctrl** (рис. 4.3) нужно подавать сигнал d_i с выхода **data_out** модуля генератора элементов отрезка сигнала, который в ПСП определяет +1 и –1. Если на этот вход поступает логический нуль (соответствует элементу –1), то для появления на выходной 4-разрядной шине параллельного кода 1111 нужно на все линии входной шины (**b0**, **b1**, **b2**, **b3**) подать логическую единицу. Если же на упомянутый вход поступает логическая единица (соответствует элементу +1), то для его преобразования в комбинацию 0001 такая комбинация подается на шину (**a0**, **a1**, **a2**, **a3**).

Поскольку в нулевой временной интервал генерации указанных элементов требуется формировать 0 в дополнительном коде (0000), то три выхода модуля **counter**, на которых в это время формируется код 000, необходимо соединить с тремя входами элемента **3НЕ-И** (созданного из примитивов *Verilog*), подключая выход последнего ко входу **en** кодера.

В результате на выходной шине генератора ПСП (после запуска счетчика **counter**) синхронно с тактовыми импульсами должны появляться нужные значения всех восьми элементов генерируемого отрезка ПСП.

4.3 Составление *Verilo*-описания схемы

1. Используя примитивы языка *Verilog* составить структурное описание всех модулей заданной схемы на логическом уровне: генератор элементов отрезка сигнала, счетчик, кодер.

2. Создать модуль генератора ПСП, который будет включать в себя экземпляры ранее указанных модулей

3. Создать модуль **test_bench**, в котором создать экземпляр модуля генератора ПСП, герератор тактовых импульсов **clk** и сигнал сброса **reset**. В качестве образца генераторов этих сигналов можно использовать соответствующие **initial** и **always** блоки из модуля **test_bench** лабораторной работы № 1. Предусмотреть в модуле **test_bench** системные функции Verilog, позволяющие выводить значения сигналов ПСП в различные моменты времени в окне **Transcript** симулятора **ModelSim**.

4.4 Компиляция проекта

После того как средствами текстового редактора будет составлено *Verilog*-описание всех модулей проектируемого устройства необходимо осуществить компиляцию проекта.

В качестве модуля верхнего уровня (top модуля) необходимо выбрать модуль **test_bench**.

4.5 Верификация проекта

Следующим этапом после компиляции является верификация проекта, т.е. моделирование и проверка правильности функционирования спроектированного устройства.

Проверка правильности работы устройства осуществляется следующим образом. Вначале составляется так называемый контрольный пример. Он представляет собой выбранный набор временных диаграмм входных сигналов и временные диаграммы требуемой (например, рассчитанной вручную по таблице истинности, заданной для проектирования) реакции спроектированного устройства на эти входные сигналы. На следующем этапе проверки временные диаграммы выбранных входных сигналов задаются в сигнальном редакторе. И, наконец, последним этапом проверки является запуск моделирования (симуляции) работы устройства при заданных входных сигналах, сравнение результатов моделирования с контрольным примером и корректировка введенного описания (например, схемы) устройства в случае недопустимого различия результатов моделирования и ожидаемых (в соответствии с контрольным примером) результатов.

Контрольный пример для данной лабораторной работы должен быть создан при самостоятельной подготовке (после изучения принципов построения проектируемого устройства).

Как уже отмечалось, входными сигналами проектируемого генератора являются**:** последовательность тактовых импульсов **clk** и сигнал сброса **reset**. Их временные диаграммы задаются следующим образом. Длительность одного периода импульсов (меандра) **clk** не рекомендуется выбирать менее 10 нс. Рекомендуемые параметры сигнала **reset** указаны в 4.2.

После того, как будут заданы все необходимые сигналы, можно запустить симуляцию (моделирование) функционирования устройства. Для начала моделирования проектируемого устройства необходимо в строке меню основного окна **ModelSim** выбираем пункт **Simulate → Start Simulation**. В результате откроется окно **Start Simulation**.

В этом окне необходимо раскрыть библиотеку **work**, нажав «+» слева от символа этой библиотеки и выбрать модуль верхнего иерархического уровня. В нашем примере этим модулем является **test_bench**. Нажать кнопку **ОК**.

В результате этого в окно **Transcript** будет выведена информация об успешном подключении объектов из библиотеки **work,** в закладке **Sim** окна **Workspace** будут отображены все экземпляры модулей, а в основном окне **ModelSim** появится окно **Objects,** содержащее сигналы **clk, reset, d.**

Для запуска моделирования выберите в основном окне **ModelSim** выберите пункт меню **Simulate → Run → Run 350 ns** или нажмите кнопку  (Run All). Симуляция будет выполнена для значения по умолчанию (350 ns). Результаты моделирования будут

отображены в окне **Wave** в виде графиков выходных сигналов. Для изменения масштаба временных диаграмм, воспользуйтесь кнопками  панели инструментов.

5. Требования к отчету по работе

В качестве отчета по данной работе представляются в бумажном виде и сопровождаются пояснениями следующие результаты: *Verilog*-описание проектируемого устройства; временные диаграммы, полученные при верификации проекта, выводы по работе.

6. Вопросы для самопроверки

1. В соответствии с какими соображениями выбирается комбинационная схема генератора элементов отрезка сигнала? Сколько адресных входов должен иметь такой модуль?
2. С помощью каких примитивов может быть составлено описание комбинационной схемы генератора элементов отрезка сигнала? Каковы их таблици истинности?
3. Чем определяется требуемая разрядность (количество выходов) счетчика?
4. Какие сигналы и почему должны поступать на входы шинного мультиплексора в заданном проекте?
5. Какие примитивы используются для построения шинного мультиплексора?
6. Каким образом формируется код нулевого элемента в проектируемом генераторе ПСП?
7. Возможна ли верификация проекта без предварительной его компиляции?

Библиографический список

1. Фролкин В.Т., Попов Л.Н. Импульсные и цифровые устройства: Учеб. пособие для вузов.— М.: Радио и связь, 1992.— 336 с.
2. Угрюмов Е.П. Цифровая схемотехника.— СПб.**:** БХВ, 2001.– 528 с.
3. Стешенко В.Б. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов.— М.: ДОДЭКА, 2000.– 128 с.