МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий
Кафедра Вычислительной Техники (ВТ)

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4

«Схемотехнические узлы комбинационного типа»

по дисциплине

«Архитектура вычислительных машин и систем»

Выполнил студент группы Зырянов М.А.

ИВБО-01-22

Принял ассистент кафедры ВТ Дуксина И.И.

Лабораторная работа выполнена « __ » _____ 2023 г.

«Зачтено» « _ » _____ 2023 г.

Москва 2023

АННОТАЦИЯ

Данная работа включает в себя 3 рисунка, 1 таблицу, 1 формулу и 7 листингов. Количество страниц в работе — 18.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной практической работе задана логическая функция от пяти переменных в 16-теричной векторной форме. Требуется создать проект САПР Vivado. Создать модули мультиплексора, дешифратора, шифратора, модули реализации заданной функции на каждой комбинационной схеме и тестовый на языке Verilog HDL. Протестировать работу модулей и убедиться в их правильности. Затем подготовить отчёт о проделанной работе и защитить её.

1 ПРЕОБРАЗОВАНИЕ ЛОГИЧЕСКОЙ ФУНКЦИИ

Исходя из персонального варианта практической работы №4 имеется Логическая функция в векторном виде: 2297320E

Используя простейший алгоритм перевода шестнадцатиричного числа в его двоичное представление было получено число: 00100010100101110011001000001110

По данному числу была воссоздана Таблица истинности 1.1:

Таблица 1.1 — Таблица истинности

x_1	x_2	x_3	x_4	x_5	F
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0

Продолжение Таблицы 1.1

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

2 МУЛЬТИПЛЕКСОР

В данной практической работе нужно реализовать функцию на мультиплексора 2–1. У мультиплексора 2–1 есть один адресный вход a_0, два информационных x_0 и x_1, а также один выход f. Если посмотреть реализацию мультиплексора 2–1, то видно, что он работает по Формуле 2.1.

$$f = \overline{a_0}\ \&\ x_0 + a_0\ \&\ x_1 \tag{2.1}$$

В модуле нужно создать три входных переменных и одну выходную, в которую будет записан результат работы мультиплексора Листинг 2.1.

Листинг 2.1 — Модуль мультиплексора 2–1 на языке Verilog HDL

```
`timescale 1ns / 1ps

module mux(

    input a0, x0, x1,

    output f

    );

assign f = ~a0 & x0 | a0 & x1;

endmodule
```

Для того, чтобы реализовать функцию на мультиплексорах 2–1 нужно разделить таблицу истинности на маленькие мультиплексоры так, чтобы каждая переменная, кроме младшей, были хоть раз поданы на адресный вход и чтобы не было одинаковых мультиплексоров. Код для готовой схемы представлен в Листинге 2.2:

Листинг 2.2 — Модуль реализации функции f(a, b, c, d, e) на мультиплексорах 2–1 на языке Verilog HDL

```
`timescale 1ns / 1ps

module MuxFunc(
input a, b, c, d, e,
    output f
    );
wire w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11;

mux mux1(.a0(d), .x0(d), .x1(!e), .f(w1));
mux mux2(.a0(d), .x0(!e), .x1(e), .f(w2));
mux mux3(.a0(d), .x0(e), .x1(d), .f(w3));
mux mux4(.a0(c), .x0(w2), .x1(w3), .f(w4));
mux mux5(.a0(b), .x0(w1), .x1(w4), .f(w5));

//

mux mux6(.a0(d), .x0(d), .x1(!e), .f(w6));
mux mux7(.a0(c), .x0(d), .x1(w6), .f(w7));
mux mux8(.a0(d), .x0(c), .x1(!e), .f(w8));
mux mux9(.a0(c), .x0(c), .x1(w8), .f(w9));
mux mux10(.a0(b), .x0(w7), .x1(w9), .f(w10));

//

mux mux11(.a0(a), .x0(w5), .x1(w10), .f(f));
endmodule
```

3 ДЕШИФРАТОРЫ

В данной практической нужно реализовать заданную функцию, на дешифраторах 2–4. Дешифратор состоит из одного разрешающего входа Ena, n входов a_0-a_n и 2^n выходов Y_0-Y_2^n.

Из-за того, что для реализации понадобится также дешифратор 1–2, модуль будет реализован с параметром width, который по умолчанию будет иметь значение 2. Реализован модуль будет при помощи регистра r размерностью 2^n, который будет совершать операцию r = 1 >> a каждый раз, когда будет изменяться хотя бы одна переменная модуля и переменная Ena будет ненулевой. В конце модуля значения регистра r будет присвоено выходной переменной Y Листинг 3.1.

Листинг 3.1 — Модуль параметризованного дешифратора на языке Verilog HDL

```
`timescale 1ns / 1ps

module dc #(width = 2)(
    input Ena,
    input [width - 1:0] a,
    output [2**(width) - 1:0] Y
    );

reg [2**(width) - 1:0] r;

always @*
begin
    if(!Ena)
        r = 0;
    else
        r = 1 << a;
end

assign Y = r;

endmodule
```

Для реализации на дешифраторах нужно также части таблицы истинности реализовать на отдельных дешифраторах и в дальнейшем соединены между собой. Код для готовой схемы представлен в Листинге 3.2:

Листинг 3.2 — Модуль реализации функции f(a, b, c, d, e) на дешифраторах на языке Verilog HDL

```
wire [1:0] w0;
wire [3:0] w1, w2, w3, w4, w5, w6, w7, w8, w9, w10;

dc #(1) dc0(.Ena(1), .a(a), .Y(w0));

dc #(2) dc1(.Ena(w0[0]), .a({b, c}), .Y(w1));
dc #(2) dc2(.Ena(w0[1]), .a({b, c}), .Y(w2));

dc #(2) dc3(.Ena(w1[0]), .a({d, e}), .Y(w3));
dc #(2) dc4(.Ena(w1[1]), .a({d, e}), .Y(w4));
dc #(2) dc5(.Ena(w1[2]), .a({d, e}), .Y(w5));
dc #(2) dc6(.Ena(w1[3]), .a({d, e}), .Y(w6));
dc #(2) dc7(.Ena(w2[0]), .a({d, e}), .Y(w7));
dc #(2) dc8(.Ena(w2[1]), .a({d, e}), .Y(w8));
dc #(2) dc9(.Ena(w2[2]), .a({d, e}), .Y(w9));
dc #(2) dc10(.Ena(w2[3]), .a({d, e}), .Y(w10));

assign f = w4[3] | w5[2] | w5[3] | w6[0] | w6[2] | w6[3] | w7[0] | w7[2]
| w8[1] | w8[2] | w8[2] | w9[0] | w9[1] | w9[2] | w10[0] | w10[1] |
w10[3];

endmodule
```

4 ШИФРАТОРЫ

Шифратор — это комбинационная схема, которая, что понятно из названия, является полной противоположностью дешифратора как по функционалу, так и по входам-выходам. Имеет 2^n входов и n выходов. Суть его работы в том, что на какой его вход поступает единица, такое значение получается на выходе Листинг 4.1.

Листинг 4.1 — Модуль параметризованного шифратора на языке Verilog HDL

```
`timescale 1ps / 1ps

module cd# (width = 2)
(
    input [2**(width)-1:0] a,
    output reg [ width - 1:0] Y,
    output Q
);

integer n;

always @(a)
begin
    Y = 0;
    for(n = 0; n <= 2**(width)-1; n = n + 1)
        if(a[n])
        begin
        Y = n;
        end
    end

assign Q = a == 0;

endmodule
```

Для преобразователя кодов понадобится один дешифратор 5–32, который даст нам поток для каждого состояния нашей функции, и шифратор 2–1, во входы которого будут поданы суммы состояний, когда функция даёт 1 и 0. Код для готовой схемы представлен в Листинге 4.2:

Листинг 4.2 — Модуль реализации функции f(a, b, c, d, e) на преобразователе кодов на языке Verilog HDL

```
`timescale 1ns / 1ps

module CdFunc(
    input a, b, c, d, e,
    output f
    );

wire [31:0] w0;
wire Q;
dc #(5) dc0(.Ena(1), .a({a, b, c, d, e}), .Y(w0));
cd #(1) cd0(.a({(w0[0] | w0[2] | w0[3] | w0[4] | w0[7] | w0[9]
               | w0[13] | w0[15] | w0[16] | w0[18] | w0[20] | w0[21]
               | w0[22] | w0[24] | w0[27] | w0[28] | w0[29] | w0[30]),
               (w0[1] | w0[5] | w0[6] | w0[8] | w0[10] | w0[11] | w0[12]
               | w0[14] | w0[17] | w0[19] | w0[23] | w0[25] | w0[26]
               | w0[31])}), .Y(f), .Q(Q));
     endmodule
```

5 ВЕРИФИКАЦИЯ

Для запуска и проверки работоспособности модулей нам нужен симулирующий тестовый модуль. (Листинг 5.1).

Листинг 5.1 — Тестовый модуль testbench.v

```
`timescale 1ps / 1ps
module testbench();

reg [5:0] args;
reg clk;
wire res;
reg [0:31] reference_reg, error_reg;

initial
begin
    reference_reg = 32'013ba6fd;
    args = 5'b00000;
    clk = 0;
    error_reg = 0;
end
always #10 clk = ~clk;
always@(posedge clk)
begin
    error_reg[args] = res ~^ reference_reg[args];
    args = args + 1;
    if(args == 6'd33)
        $finish;
end
MuxFunc mux (.a(args[4]), .b(args[3]), .c(args[2]), .d(args[1]),
.e(args[0]), .f(res));
//DcFunc dc (.a(args[4]), .b(args[3]), .c(args[2]), .d(args[1]),
.e(args[0]), .f(res));
//CdFunc cd (.a(args[4]), .b(args[3]), .c(args[2]), .d(args[1]),
.e(args[0]), .f(res));
```

Вне зависимости от того, подключим к тестовому модулю testbench модуль реализации функции на мультиплексорах, дешифраторах или преобразователе кодов, временная диаграмма во всех случаях будет одинаковой - Рисунок 5.1, Рисункок 5.2, Рисунок 5.3

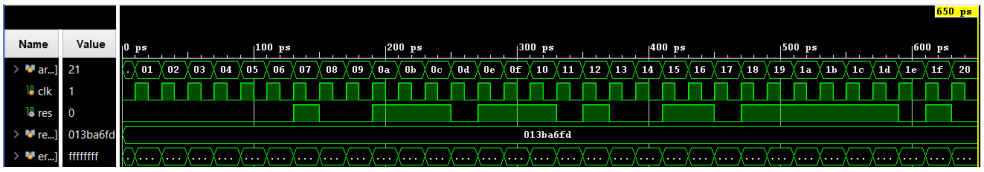

Рисунок 5.1 – Временная диаграмма реализации функции на плексорах (Mux)

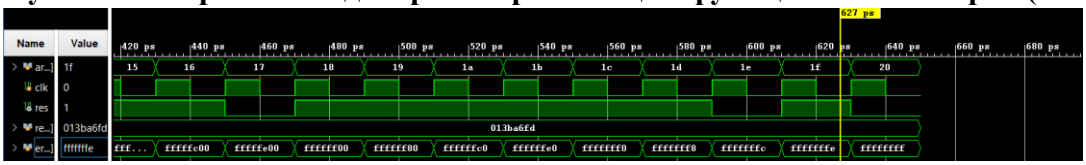

Рисунок 5.2 – Временная диаграмма реализации функции на плексорах (Dc)

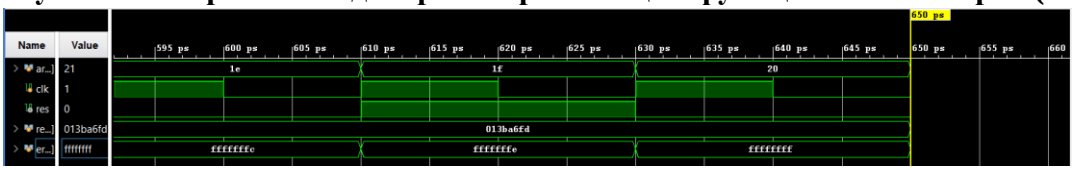

Рисунок 5.3 – Временная диаграмма реализации функции на плексорах (Cd)

На временной диаграмме мы можем увидеть, что одноразрядная цепь res принимает значение 1 при тех же наборах значений, что и в таблице истинности Таблица 1.1. Это означает, что результат работы модулей корректен

ЗАКЛЮЧЕНИЕ

Для заданной логической функции была построена таблица истинности, в САПР Vivado по ней были созданы модули мультиплексора 2–1, параметрического дешифратора, параметрического шифратора, реализаций функции на плексорах и тестовый testbench. Для этого были использованы следующие операции: инверсия, конъюнкция, дизъюнкция. Осуществлена симуляция работы данных модулей, а также проведена успешная верификация работы заданных схем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания по ПР № 1 — URL: https://online-edu.mirea.ru/mod/resource/view.php?id=405132 (Дата обращения: 23.09.2022).

2. Методические указания по ПР № 2 — URL: https://online-edu.mirea.ru/mod/resource/view.php?id=409130 (Дата обращения: 23.09.2022).

3. Смирнов С.С. Информатика [Электронный ресурс]: Методические указания по выполнению практических и лабораторных работ / С.С. Смирнов — М., МИРЭА — Российский технологический университет, 2018. — 1 электрон. опт. диск (CD-ROM).

4. Тарасов И.Е. ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. — М.: Горячая линия — Телеком, 2021. — 538 с.: ил.

5. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие / Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).

6. Антик М.И. Математическая логика и программирование в логике [Электронный ресурс]: Учебное пособие / Антик М.И., Бражникова Е.В.— М.: МИРЭА – Российский технологический университет, 2018. — 1 электрон. опт. диск (CD-ROM).

7. Жемчужникова Т.Н. Конспект лекций по дисциплине «Архитектура вычислительных машин и систем» — URL: https://drive.google.com/file/d/12OAi2_axJ6mRr4hCbXs-mYs8Kfp4YEfj/view?usp=sharing (Дата обращения: 23.09.2022).

8. Антик М.И. Теория автоматов в проектировании цифровых схем [Электронный ресурс]: Учебное пособие / Антик М.И., Казанцева Л.В. — М.: МИРЭА – Российский технологический университет, 2020. — 1 электрон. опт. диск (CD-ROM).

9. Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. 4-е изд. — СПб.: Питер, 2018. — 688 с.: ил.

10. Шустов М.А. Цифровая схемотехника. Основы построения. — СПб.: Наука и Техника, 2018. — 320 с.: ил.

11. Рафиков Р. А. Электронные сигналы и цепи. Цифровые сигналы и устройства: Учебное пособие. — СПб.: Издательство «Лань», 2016. — 320 с., ил. — (Учебники для вузов. Специальная литература).

12. Угрюмов Е. П. Цифровая схемотехника: учеб. пособие для вузов. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 816 с.: ил.