



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

«Узлы. Мультиплексор. Дешифратор. Шифратор»

по дисциплине

«Архитектура вычислительных машин и систем»

Выполнил студент группы
ИВБО-11-23

Туктаров Т.А.

Принял ассистент кафедры ВТ

Дуксина И.И.

Практическая работа выполнена

«2» октября 2024 г.

«Зачтено»

«2» октября 2024 г.

Москва 2024

АННОТАЦИЯ

Данная работа включает в себя 1 рисунок, 1 таблицу, 7 листингов.
Количество страниц в работе – 12

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 ХОД РАБОТЫ	5
1.1 Практическое введение.....	5
1.2 Восстановление таблицы истинности.....	5
1.3 Реализация параметрического мультиплексора	6
1.4 Реализация параметрического дешифратора	7
1.5 Реализация параметрического шифратора	7
1.6 Реализация логической функции на мультиплексорах	8
1.7 Реализация логической функции на дешифраторах.....	8
1.8 Реализация логической функции на преобразователе кодов	9
1.9 Верификация.....	9
ЗАКЛЮЧЕНИЕ	11
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	12

ВВЕДЕНИЕ

Мультиплексор - комбинационная схема, имеющая n адресных входов, 2^n информационных входов, 1 выход. На выход поступает значение с того информационного входа, номер которого задаётся при помощи адресных входов. Дешифратор - комбинационная схема, имеющая n входов, 2^n выходов. Сигнал логической единицы будет сформирован на выходе, номер которого в двоичном виде задан на входах.

Шифратор - комбинационная схема, имеющая 2^n информационных входов, и выходов. На выходах формируется номер входа, на который была подана логическая единица. Зачастую сопровождается выходом, который отвечает за индикацию того факта, что на входе нет ни единого сигнала логической единицы. В приоритетных шифраторах больший вес имеет старший вход, на котором сигнал логической единицы.

Преобразователь кодов. Узел может быть реализован как совокупность дешифратора и шифратора. Кодирует входной сигнал согласно внутренним связям.[1][2]

1 ХОД РАБОТЫ

1.1 Практическое введение

Реализовать логическую функцию от 5 переменных на Verilog HDL с использованием дешифраторов, мультиплексоров и преобразователя кодов. Произвести верификацию. Заданная логическая функция: 478E9C16.

1.2 Восстановление таблицы истинности

Имея логическую функцию в векторном виде 478E9C16 воссоздадим таблицу истинности(Таблица 1.1)

Таблица 1.1 – Таблица истинности функции

X1	X2	X3	X4	X5	F
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1

Продолжение таблицы 1.1

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

1.3 Реализация параметрического мультиплексора

Реализуем параметрический мультиплексор в модуле. Реализация представлена в Листинге 2.1

Листинг 2.1 – Модуль реализации мультиплексора

```
`timescale 1ns / 1ps
module mux#(width = 2)(
    input [width - 1:0] a,
    input [0: 2**(width) - 1] in,
    input enable,
    output f
);
    assign f = enable && in[a];
endmodule
```

1.4 Реализация параметрического дешифратора

Реализуем параметрический дешифратор в модуле. Реализация представлена на Листинге 1.2.

Листинг 1.2 – Модуль реализации дешифратора

```
`timescale 1ns / 1ps
module dc#(width = 2) (
    input [width - 1: 0] in,
    input enable,
    output [2**(width) - 1: 0] f
);
    assign f = !enable ? 0 : 1 << in;
endmodule
```

1.5 Реализация параметрического шифратора

Реализуем параметрический шифратор в модуле. Реализация представлена на Листинге 1.3.

Листинг 1.3. – Модуль реализации шифратора

```
`timescale 1ns / 1ps
module cd#(width = 2) (
    input [2**(width) - 1: 0] in,
    output Q,
    output reg [width - 1: 0] f
);
    integer n;
    always@(in)
    begin
        f = 0;
        for (n = 0; n <= 2**(width) - 1; n = n + 1)
            if (in[n])
                begin
                    f = n;
                end
        end
    end
    assign Q = in == 0;
endmodule
```

1.6 Реализация логической функции на мультиплексорах

Реализуем логическую функцию на основе минимального количества мультиплексоров 2-1 и инверторов в модуле. Реализация представлена в Листинге 1.4

Листинг 1.4 – Модуль реализации функции на мультиплексорах

```
`timescale 1ns / 1ps

module func1(
    input [4:0] x,
    output f
);
wire w1_1,w1_2,w1_3,w1_4,w1_5,w1_6,w1_7,w1_8, w2_1,w2_2,w2_3,w2_4, w3_1,w3_2;
// 1st level
mux #(1) mux0(.a(x[1]), .in({x[0], 1'b0}), .enable(1'b1), .f(w1_1));
mux #(1) mux1(.a(x[1]), .in({x[0], 1'b1}), .enable(1'b1), .f(w1_2));
mux #(1) mux2(.a(x[1]), .in({~x[0], 1'b0}), .enable(1'b1), .f(w1_3));
mux #(1) mux3(.a(x[1]), .in({1'b1, ~x[0]}), .enable(1'b1), .f(w1_4));
mux #(1) mux4(.a(x[1]), .in({~x[0], x[0]}), .enable(1'b1), .f(w1_5));
mux #(1) mux5(.a(x[1]), .in({1'b1, 1'b0}), .enable(1'b1), .f(w1_6));
mux #(1) mux6(.a(x[1]), .in({1'b0, x[0]}), .enable(1'b1), .f(w1_7));
mux #(1) mux7(.a(x[1]), .in({x[0], ~x[0]}), .enable(1'b1), .f(w1_8));
// 2nd level
mux #(1) mux8(.a(x[2]), .in({w1_1,w1_2}), .enable(1'b1), .f(w2_1));
mux #(1) mux9(.a(x[2]), .in({w1_3,w1_4}), .enable(1'b1), .f(w2_2));
mux #(1) mux10(.a(x[2]), .in({w1_5,w1_6}), .enable(1'b1), .f(w2_3));
mux #(1) mux11(.a(x[2]), .in({w1_7,w1_8}), .enable(1'b1), .f(w2_4));
//3rd level
mux #(1) mux12(.a(x[3]), .in({w2_1,w2_2}), .enable(1'b1), .f(w3_1));
mux #(1) mux13(.a(x[3]), .in({w2_3,w2_4}), .enable(1'b1), .f(w3_2));
// 4th level
mux #(1) mux14(.a(x[4]), .in({w3_1,w3_2}), .enable(1'b1), .f(f));
endmodule
```

1.7 Реализация логической функции на дешифраторах

Реализуем логическую функцию на основе минимального количества дешифраторов 2-4, инверторов и элементов ИЛИ на дешифраторах в модуле. Реализация представлена в Листинге 1.5.

Листинг 1.5 – Модуль реализации функции на дешифраторах

```
`timescale 1ns / 1ps

module func2(
    input [4:0] x,
    output f
);
wire [3:0] w1_1, w2_1, w2_2, w3_1,w3_2, w3_3, w3_4, w3_5, w3_6,
w3_7,w3_8;
// 1st level
dc#(2) dc0(.in({x[4], x[4]}), .enable(1'b1), .f(w1_1));
```


Продолжение Листинга 1.5

```
// 2nd level
dc#(2) dc1(.in({x[3], x[2]}), .enable(w1_1[0]), .f(w2_1));
dc#(2) dc2(.in({x[3], x[2]}), .enable(w1_1[3]), .f(w2_2));
// 3rd level
dc#(2) dc3(.in({x[1], x[0]}), .enable(w2_1[0]), .f(w3_1));
dc#(2) dc4(.in({x[1], x[0]}), .enable(w2_1[1]), .f(w3_2));
dc#(2) dc5(.in({x[1], x[0]}), .enable(w2_1[2]), .f(w3_3));
dc#(2) dc6(.in({x[1], x[0]}), .enable(w2_1[3]), .f(w3_4));

dc#(2) dc7(.in({x[1], x[0]}), .enable(w2_2[0]), .f(w3_5));
dc#(2) dc8(.in({x[1], x[0]}), .enable(w2_2[1]), .f(w3_6));
dc#(2) dc9(.in({x[1], x[0]}), .enable(w2_2[2]), .f(w3_7));
dc#(2) dc10(.in({x[1], x[0]}), .enable(w2_2[3]), .f(w3_8));

assign f = w3_1[1] | w3_2[1] | w3_2[2] | w3_2[3] | w3_3[0] | w3_4[0] |
w3_4[1] | w3_4[2]
| w3_5[0] | w3_5[3] | w3_6[0] | w3_6[1] | w3_7[3] | w3_8[1] | w3_8[2];
endmodule
```

1.8 Реализация логической функции на преобразователе кодов

Реализация логической функции на основе преобразователя кодов (дешифратор 5-32, шифратор и набор элементов ИЛИ). Реализация представлена в Листинге 1.6.

Листинг 1.6 – Модуль реализации функции на преобразователе кодов

```
`timescale 1ns / 1ps
module func3(
    input [4:0] x,
    output f
);
    wire [31:0] f_dc0;
    wire Q;
    reg [1:0] in_cd;
    dc# (5) dc0(.in(x), .enable(1'b1), .f(f_dc0));
    always @(*)
    begin
        in_cd[1] = f_dc0[1] | f_dc0[5] | f_dc0[6] | f_dc0[7] | f_dc0[8] |
f_dc0[12] | f_dc0[13] | f_dc0[14] | f_dc0[16] | f_dc0[19] | f_dc0[20] |
f_dc0[21] | f_dc0[27] | f_dc0[29] | f_dc0[30];
    end
    cd#(1) cd1(.in(in_cd), .Q(Q), .f(f));
endmodule
```

1.9 Верификация

Произведем верификацию модулей, для это создадим модуль testbench.v(Листинг 1.7).

Листинг 1.7 – Модуль верификации модулей

```

`timescale 1ns / 1ps
module testbench();
    reg [4:0] args;
    reg clk;
    wire f_mux, f_dc, f_cd;
    reg [0:31] reference_reg, error_reg_mux, error_reg_dc, error_reg_cd;
    initial begin
        reference_reg = 32'h478E9C16;
        args = 0;
        clk = 0;
        error_reg_mux = 0;
        error_reg_dc = 0;
        error_reg_cd = 0;
    end
    always #10 clk = ~clk;
    always @(posedge clk) begin
        error_reg_mux[args] <= (f_mux ~^ reference_reg[args]);
        error_reg_dc[args] <= (f_dc ~^ reference_reg[args]);
        error_reg_cd[args] <= (f_cd ~^ reference_reg[args]);
        args <= args + 1;
        if (args == 32'h478E9C16)
            $finish;
    end
    func1 mx_f(
        .x(args),
        .f(f_mux)
    );
    func2 dc_f(
        .x(args),
        .f(f_dc)
    );
    func3 cd_f(
        .x(args),
        .f(f_cd)
    );
endmodule

```

Результат верификации представлен на Рисунке 1.1

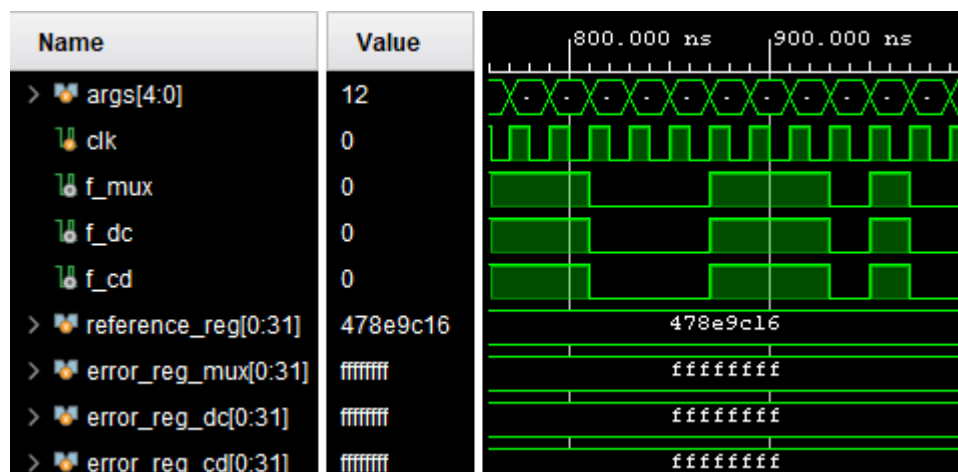


Рисунок 1.1 – Результат верификации.

ЗАКЛЮЧЕНИЕ

В данной работе была реализована логическая функция из 5 переменных с использованием мультиплексоров, дешифраторов и преобразователя кодов при помощи языка описания аппаратуры Verilog средствами САПР Vivado. Произведена верификация полученных модулей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания по ПР № 1 — URL: <https://online-edu.mirea.ru/mod/resource/view.php?id=405132> (Дата обращения: 23.09.2024).
2. Методические указания по ПР № 2 — URL: <https://online-edu.mirea.ru/mod/resource/view.php?id=409130> (Дата обращения: 23.09.2024).
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: 23.09.2024).
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.