

ГУАП

КАФЕДРА № 44

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

доцент, канд. техн. наук,
доцент

должность, уч. степень, звание

подпись, дата

О. О. Жаринов

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

РАЗРАБОТКА ПРЕОБРАЗОВАТЕЛЕЙ КОДОВ НА ОСНОВЕ ТИПОВЫХ
ФУНКЦИОНАЛЬНЫХ УЗЛОВ КОМБИНАЦИОННОЙ ЛОГИКИ С
ИСПОЛЬЗОВАНИЕМ ЯЗЫКОВ ОПИСАНИЯ АППАРАТУРЫ

по курсу: Схемотехника

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №

4143

подпись, дата

Д. В. Пономарев

инициалы, фамилия

Санкт-Петербург 2024

Цель работы

Изучить принципы работы типовых функциональных узлов комбинационной логики: шифраторов, дешифраторов, мультиплексоров. Разработать проект преобразователя кодов на их основе, с использованием языков описания аппаратуры. Основной целью работы является формирование навыков использования модулей на языке описания аппаратуры.

Таблица истинности

Искомая таблица истинности со значениями согласно варианту №24 показана в таблице 1. Исходя из её содержания, можем сделать выводы о том, какие выходные сигналы нужно получить при каком-либо из входных сигналов.

Таблица 1

Состояния входных сигналов			Состояния выходных сигналов	
X2	X1	X0	Y1	Y0
0	0	0	1	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	0

Листинг программного модуля

В качестве языка описания аппаратуры был выбран Verilog.

Разработаем первый программный модуль на основе пары дешифратор-шифратор. Если повторно обратиться к таблице 1, то можно заметить некоторые особенности.

Так, например, в качестве выходной комбинации больше всего повторяется комбинация из единицы и нуля. То есть, такую комбинацию можно сделать выходной по умолчанию. Можно выделить и другие повторяющиеся группы выводов. Следует отметить, что в записи ниже знак «?» обозначает любой символ (но 0 или 1, естественно), так его содержание неважно.

- 1) Если на входе 0?1, то выход – 01
- 2) Если на входе 010 или 100, то выход – 00
- 3) Если на входе 110, то выход – 11
- 4) Иначе – выход 10

Необходимое использование маски сразу говорит о том, что в коде должен фигурировать блок `casez` – именно он обеспечивает её использование вместо обычного `case`. Очевидно, такой блок будет находиться внутри другого блока `always`, чтобы обеспечить проверку при каждом изменении входной комбинации.

Листинг кода

```
module lab2 (  
    input [2:0] x,  
    output [1:0] y  
);  
reg [1:0] decoder_output;  
always @(x) begin  
    casez (x)  
        3'b0?1 : decoder_output =  
            2'b01;          3'b010      :  
            decoder_output = 2'b00;  
        3'b100 : decoder_output =  
            2'b00;          3'b110      :  
            decoder_output = 2'b11;  
        default : decoder_output =  
            2'b10;  
    endcase  
end
```

```
assign y = decoder_output;
endmodule
```

ПЛИС

Искомый вид ПЛИС продемонстрирован на рисунках 1 – 2.






	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard
1	 x[2]	Input	PIN_A1	2		3.3-V LVTTTL (default)
2	 x[1]	Input	PIN_A2	2		3.3-V LVTTTL (default)
3	 x[0]	Input	PIN_A3	2		3.3-V LVTTTL (default)
4	 y[1]	Output	PIN_A4	2		3.3-V LVTTTL (default)
5	 y[0]	Output	PIN_A5	2		3.3-V LVTTTL (default)

Рисунок 1 – Подключенные входы и выходы на ПЛИС

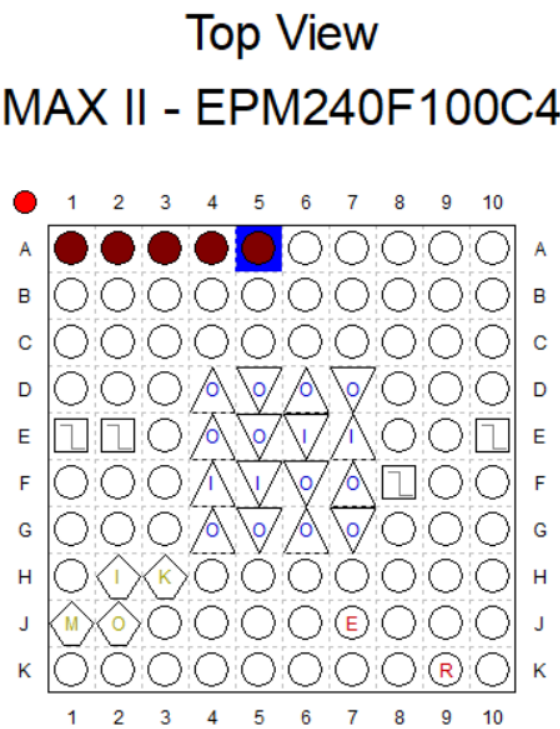


Рисунок 2 – ПЛИС

Временные диаграммы

Результат функциональной и временной симуляций продемонстрированы на рисунках 3 – 4 соответственно.

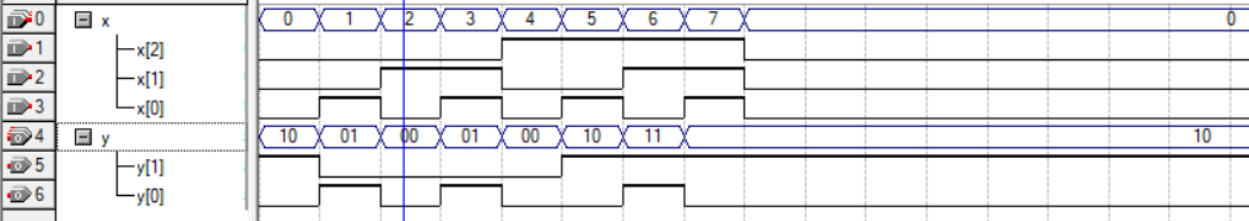


Рисунок 3 – Функциональная симуляция

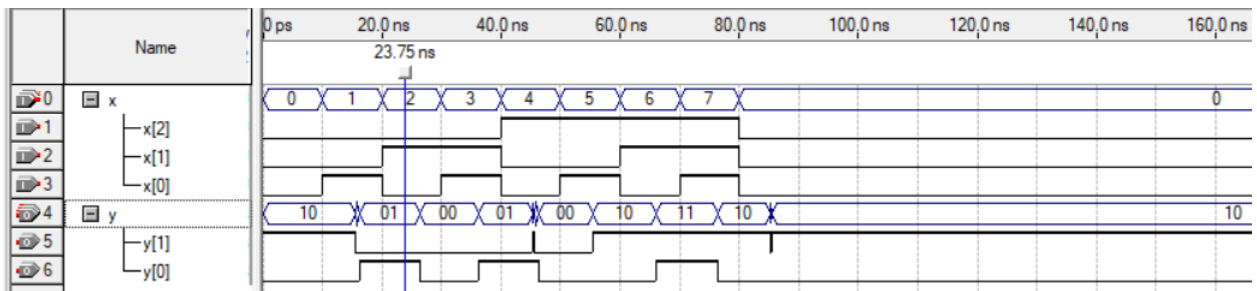


Рисунок 4 – Временная симуляция

Листинг программного модуля

Перейдём к разработке второго программного модуля на основе мультиплексоров. Следует сразу отметить, что это будет многофайловый проект, где в качестве основного файла будет файл lab2.v, а в качестве вспомогательного – lab21.v. В вспомогательном файле будут описаны используемые мультиплексоры. Всего будет использоваться 2 мультиплексора. Первый будет выдавать выходную последовательность y1, а второй – y0.

Чтобы описать поведение каждого из мультиплексоров, использовался тот же подход, что и при разработке первого программного модуля. Всё же подробно рассмотрим его ещё раз.

Опишем, как будет реагировать мультиплексор mux1_to, который будет выдавать выходную последовательность y1:

- 1) Когда на входе 000 → на выходе 1
- 2) Когда на входе 110 → на выходе 1
- 3) Когда на входе 1?1 → на выходе 1
- 4) Иначе → на выходе 0

Опишем также поведение второго мультиплексора mux0_to, который будет выдавать последовательность y0:

- 1) Когда на входе 0?1 → на выходе 1
- 2) Когда на входе 110 → на выходе 1
- 3) Иначе → на выходе 0

Как можно заметить, содержание условного блока каждого из мультиплексоров получился куда меньше, чем в предыдущем модуле в силу

меньшего количества вариантов исхода. Но такую компактность всё равно перекрывает количество мультиплексоров.

Рассмотрим более подробно созданный код. Начнём со вспомогательного файла lab21.v, где описаны используемые мультиплексоры.

Сам файл состоит из двух модулей – mux1_to и mux0_to для первого и второго мультиплексоров соответственно. В качестве входных данных идут трёхбитные сигналы s и a – селектор и массив входов мультиплексора соответственно. В качестве выходного регистра идёт однобитный y. Далее, чтобы мультиплексор выдавал какой-либо результат на каждом такте, необходимо написать все возможные условия входа внутри блока always @*, условия же написаны с помощью уже привычного casez(s). Отличие логики второго мультиплексора по отношению к первому состоит лишь в выходных сигналах и другой используемой переменной a1 (массив входов).

Код рассматриваемого файла продемонстрирован ниже.

Файл lab21.v

```
module mux1_to (  
    input [2:0] s,  
    input [2:0] a,  
    output reg [0:0] y  
);  
    always @* begin  
        casez (s)  
            3'b000: y = 1'b1;  
            3'b110: y = 1'b1;  
            3'b1?1: y = 1'b1;  
            default: y = 1'b0;  
        endcase  
    end  
endmodule  
module mux0_to (  
    input [2:0] s,  
    input [2:0] a1,  
    output reg [0:0] y  
);  
    always @* begin  
        casez (s)
```

```

3'b0?1: y = 1'b1;
3'b110: y = 1'b1;
default: y = 1'b0;
endcase
end
endmodule

```

Теперь же более подробно рассмотрим основной файл lab2.v. Он состоит из одного модуля. В качестве входных данных идёт трёхбитный x, а в качестве выходных – двубитный y.

Далее объявляются провода: двубитные провода mux0_out и mux1_out, которые выводят результирующие значения с мультиплексоров 0 и 1 соответственно, трёхбитный провод select0, которому соответствует вид (x2, x1, x0) – аналог одной последовательности входных значений на мультиплексор, такой же по размерности и назначению провод select1. Причём select1 подаётся на мультиплексор mux1_to, а select0 – на mux0_to соответственно.

Затем идёт описание входов-выходов на каждом из мультиплексоров. Рассмотрим мультиплексор mux1_to mux1. В качестве селектора идёт провод select1, массива – всевозможные варианты входной последовательности, на выходе идёт провод mux1_out. Аналогично обозначен и второй мультиплексор: селектор – select0, массив a1 - те же варианты, выход - mux1_out.

В самом конце идёт обычное присвоение значениям выходного y значений проводов mux0_out и mux1_out. Код рассматриваемого файла продемонстрирован ниже.

Файл lab2.v

```

module lab2 (
  input [2:0] x,
  output [1:0] y
);
  wire [1:0] mux0_out, mux1_out;

  wire [2:0] select0;
  assign select0 = {x[2], x[1], x[0]};

```

```

wire [2:0] select1;
assign select1 = {x[2], x[1], x[0]};
mux0_to mux0 (
.s(select0),
.a1({3'b000, 3'b001, 3'b010, 3'b011, 3'b100, 3'b101, 3'b110, 3'b111}),
.y(mux0_out)
);
mux1_to mux1 (
.s(select1),
.a1({3'b000, 3'b001, 3'b010, 3'b011, 3'b100, 3'b101, 3'b110, 3'b111}),
.y(mux1_out)
);
assign y[0] = mux0_out[0];
assign y[1] = mux1_out[0];
endmodule

```

ПЛИС

Искомый вид ПЛИС продемонстрирован на рисунках 5 – 6.






	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard
1	 x[2]	Input	PIN_A1	2		3.3-V LVTTTL (default)
2	 x[1]	Input	PIN_A2	2		3.3-V LVTTTL (default)
3	 x[0]	Input	PIN_A3	2		3.3-V LVTTTL (default)
4	 y[1]	Output	PIN_A4	2		3.3-V LVTTTL (default)
5	 y[0]	Output	PIN_A5	2		3.3-V LVTTTL (default)

Рисунок 5 – Назначенные выводы ПЛИС

Top View MAX II - EPM240F100C4

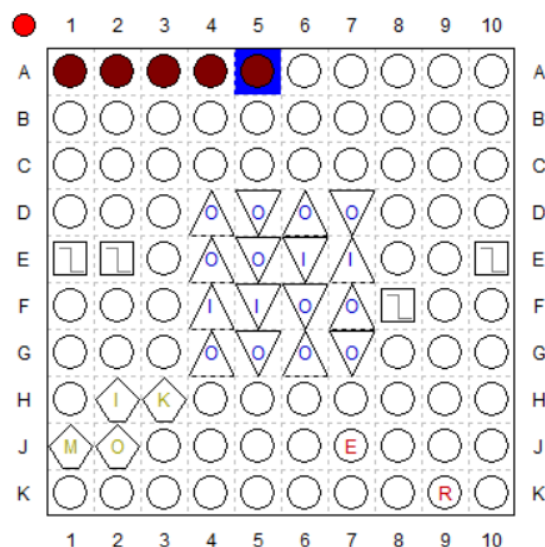


Рисунок 6 – ПЛИС

Временные диаграммы

Результат функциональной и временной симуляций продемонстрированы на рисунках 7 – 8 соответственно.

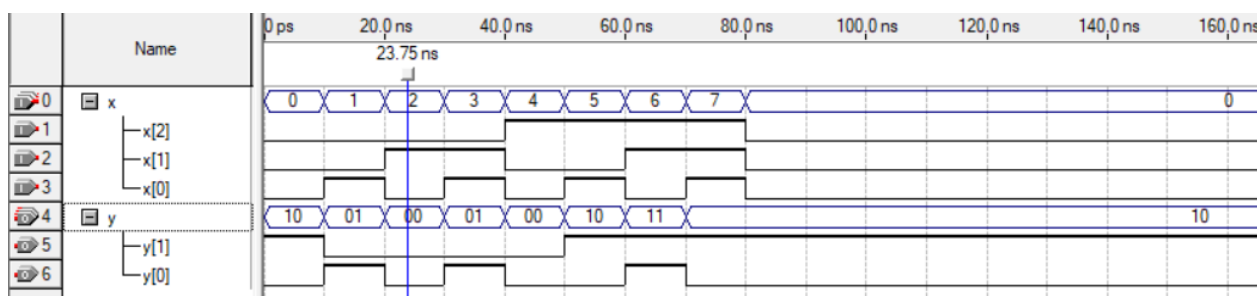


Рисунок 7 – Функциональная симуляция

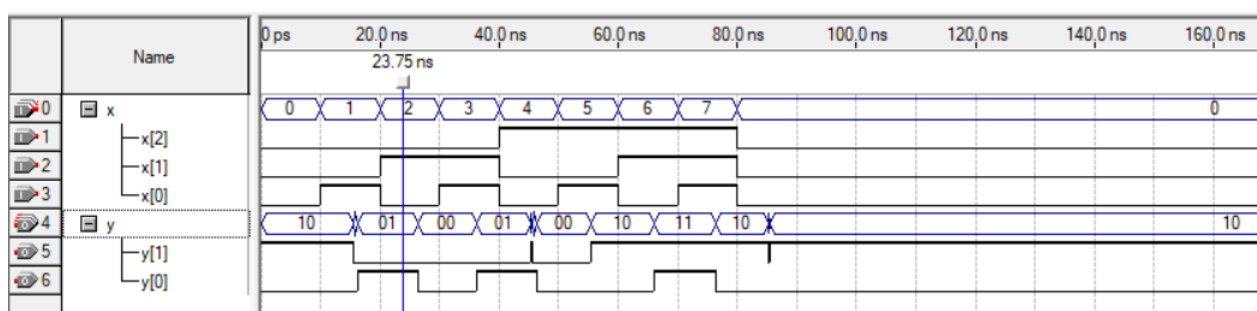


Рисунок 8 – Временная симуляция

Выводы

Изучены принципы работы типовых функциональных узлов комбинационной логики: шифраторов, дешифраторов, мультиплексоров. Разработан проект преобразователя кодов на их основе, с использованием языков описания аппаратуры. Получены навыки использования модулей на языке описания аппаратуры Verilog.

Список используемых источников

1. Проектирование цифровых устройств на базе микросхем программируемой логики: учеб. пособие / А. В. Морозов, В. А. Ненашев. – СПб: ГУАП, 2021. – 78 с.
2. Проектирование цифровых устройств на ПЛИС: учеб. пособие / И.В. Ушенина. - СПб: Лань, 2022. - 408 с.
3. Цифровая схемотехника и архитектура компьютера / Д.М. Харрис, С.Л. Харрис; пер. с англ. ImaginationTechnologies. – М.: ДМК Пресс, 2018. - 792 с

4. Учебно-методические материалы к выполнению лабораторной работы №2 по дисциплине «Схемотехника» (2-й семестр изучения дисциплины) // Жаринов. О.О: [Электронный ресурс] // Санкт-Петербургский государственный университет аэрокосмического приборостроения. URL.: <https://pro.guap.ru/inside/student/tasks/6dfad9405e595153a12329bceb01021c/download>. (Дата обращения: 01.03.24).