САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчет
по заданию lab6
Дисциплина
«Языки описания аппаратных средств вычислительных систем»

выполнила:
Фам Ба Нам
группа:
3530901/90201
преподаватель:
Федотов А. А.

Санкт-Петербург
2021

Оглавление

Список иллюстраций

1 Задание lab6_1

1.1 Задание

На языке Verilog создайте:
Функцию, обеспечивающую
преобразование двоичного кода в код Грея и обратное преобразование кода Грея в двоичный код:
Особенности:
– Разрядность преобразуемого кода задается
параметром WIDTH
– Направление преобразования задается
параметром DIR
– 0: двоичный код в код Грея
– 1: код Грея в двоичный
– необходимо использовать циклы
Модуль
параметризированный, параметр W -
разрядность преобразуемого кода, содержащий два экземпляра разработанной функции:
– Один экземпляр функции преобразует входной
двоичный код в код Грея (разрядность задается
параметром W)
– Второй экземпляр функции преобразует код Грея в
выходной двоичный код (разрядность задается
параметром W)

1.2 Описание на языке Verilog

```
module lab6_1(
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "49, 46, 25, 24"*) input [W-1:0] sw,

        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "65, 66, 67, 68, 69, 70, 71, 72"*) output [7:0] led);

parameter W = 4;

function [W-1:0] binGray;
        input [W-1:0] x;
        input dir;
        integer i;

        begin
                if (!dir) begin
```

```
                        binGray[W-1] = x[W-1];
                        for (i = W-2; i >= 0; i = i - 1) begin
                                binGray[i] = x[i+1] ^ x[i];
                        end
                end else begin
                        binGray[W-1] = x[W-1];
                        for (i = W-2; i >= 0; i = i - 1) begin
                                binGray[i] = binGray[i+1] ^ x[i];
                        end
                end
        end
endfunction

assign led[7:4] = binGray(sw, 0);
assign led[3:0] = binGray(led[7:4], 1);

endmodule
```

Рис.1.2 Описание на языке SystemVerilog

1.3 Результат синтеза (RTL)

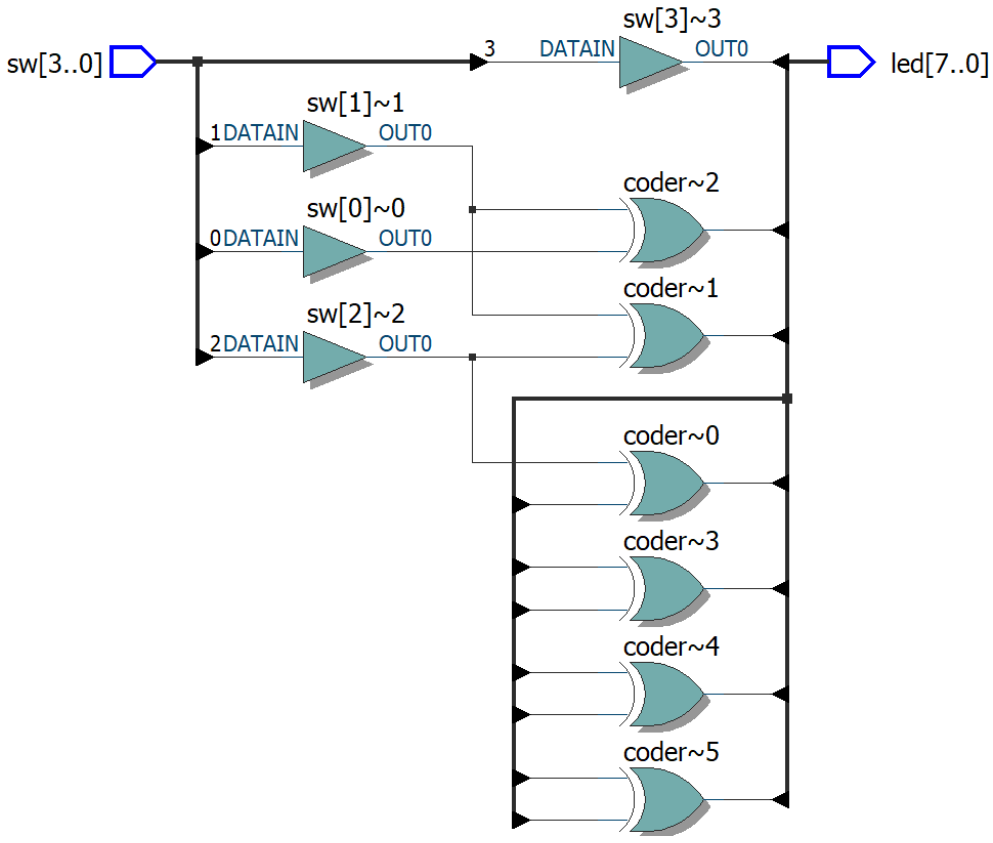


Рис.1.3 Синтезированная схема

1.4 Моделирование

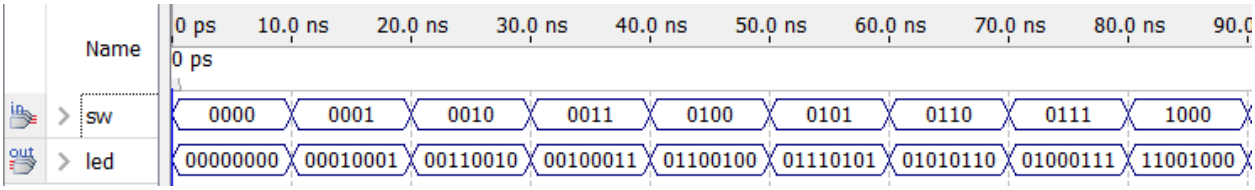


Рис.1.4 Результаты моделирования

1.5 Тестирование на плате miniDiLaB- CIV

Для тестирования проекта на плате использовались тесты, описанные в разделе 2.4

Проведенное на плате miniDiLaB-CIV тестирование разработанного устройства показало: результаты совпадают с ожидаемыми, устройство работает в соответствии с заданием.

1.6 Выводы

В ходе выполнения работы был спроектировано устройство, выполняющее преобразование двоичного кода в код Грея и обратное преобразование кода Грея в двоичный код. Выводы СБИС были назначены с использованием атрибутов.

2 Задание lab6_2

2.1 Задание

Выводы устройства:

Входы:

Переключатель sw[7:6] – операнд A

Переключатель sw[5:4] – операнд B

Переключатель sw[3:2] – операнд C

Переключатель sw[1:0] – операнд D

Clk – тактовый вход

Rst (кнопка pbA) – вход синхронного сброса

Выходы – операнды, отсортированные по убыванию или по возрастанию

светодиоды led[7:6]

светодиоды led[5:4]

светодиоды led[3:2]

светодиоды led[1:0]

Описание на языке Verilog

```
module lab6_2 (
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "25, 24"*) input reg [1:0] a,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "49, 46"*) input reg [1:0] b,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "90, 91"*) input reg [1:0] c,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "88, 89"*) input reg [1:0] d,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "71, 72"*) output reg [1:0] max,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "69, 70"*) output reg [1:0] max_min,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "67, 68"*) output reg [1:0] min_max,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "65, 66"*) output reg [1:0] min,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "23"*) input clk,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "58"*) input pba);

reg [1:0] min1, min2, max1, max2, mid1, mid2;

task sort;
        output reg [1:0] maximum, minimum;
        input reg [1:0] a, b;
        begin
                maximum <= (a > b) ? a : b;
                minimum <= (a > b) ? b : a;
```

```
        end
endtask

always @ (posedge clk, negedge pba) begin
        if (!pba) begin
                max <= 2'b0;
                max_min <= 2'b0;
                min_max <= 2'b0;
                min <= 2'b0;
        end else begin
                sort(max1, min1, a, b);
                sort(max2, min2, c, d);
                sort(max, mid1, max1, max2);
                sort(mid2, min, min1, min2);
                sort(max_min, min_max, mid1, mid2);
        end
end

endmodule
```

Рис.2.2 Описание на языке SystemVerilog

2.2 Результат синтеза (RTL)

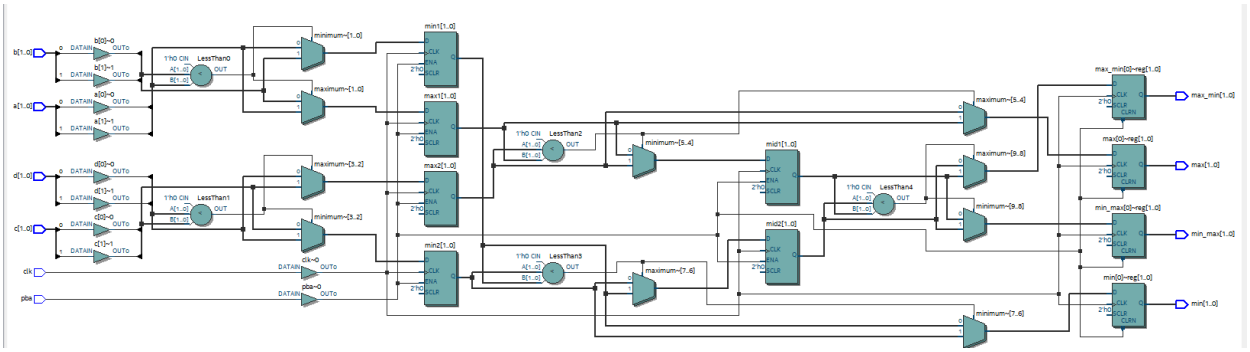


Рис. 2.3 Синтезированная схема

2.3 Моделирование

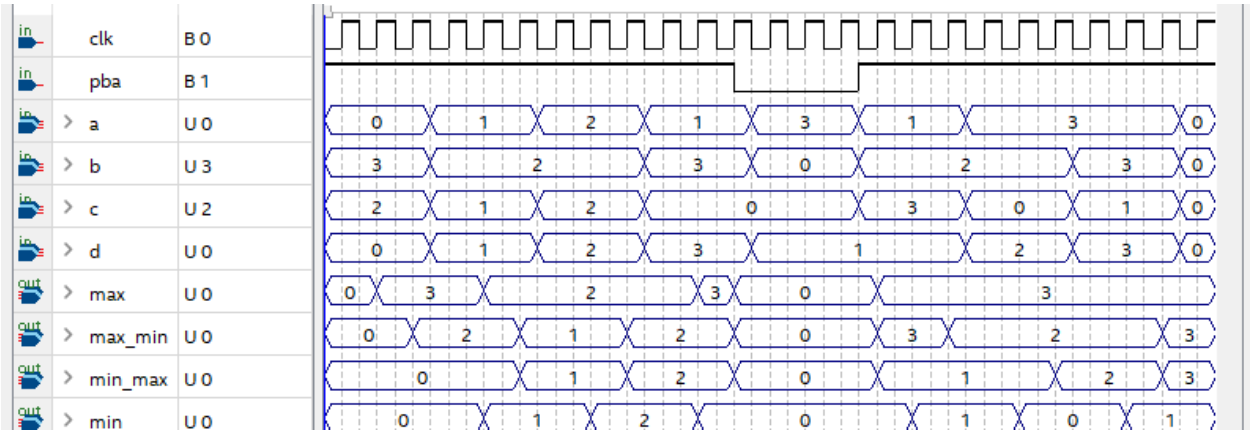


Рис.2.4 Результаты моделирования

2.4 Тестирование на плате miniDiLaB-CIV

Для тестирования проекта на плате использовались тесты, описанные в разделе 2.4

Проведенное на плате miniDiLaB-CIV тестирование разработанного устройства показало: результаты совпадают с ожидаемыми, устройство работает в соответствии с заданием.

2.5 Выводы

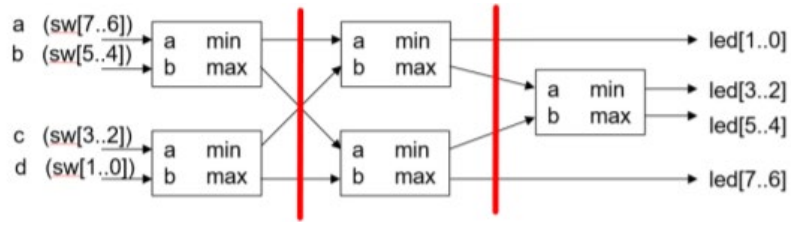
В ходе выполнения работы было спроектировано устройство, сортирующее входные числа по возрастанию. Выводы СБИС были назначены с использованием атрибутов.

3 Задание lab6_3

3.1 Задание

Осуществите конвейеризацию устройства
из части Lab6_2
На рисунке показано место для регистров

конвейеризации.



Проведите исследование:
– Для исходной (lab6_2) реализации зафиксируйте
– с помощью RTL Viewer структуру.
– Аппаратные затраты
– Максимальную тактовую частоту
– Для конвейерной реализации зафиксируйте
результат
– с помощью RTL Viewer структуру.
– Аппаратные затраты
– Максимальную тактовую частоту
– Осуществите сравнение
Для конвейерной реализации осуществите
моделирование.

3.2 Описание на языке Verilog

```
module lab6_3 (
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "25, 24"*) input reg [1:0] a,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "49, 46"*) input reg [1:0] b,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "90, 91"*) input reg [1:0] c,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "88, 89"*) input reg [1:0] d,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "71, 72"*) output reg [1:0] max,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "69, 70"*) output reg [1:0] max_min,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "67, 68"*) output reg [1:0] min_max,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "65, 66"*) output reg [1:0] min,
        (*altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"",
   chip_pin = "23"*) input clk,
        (*altera_attribute = "-name IO_STANDARD \"2.5 V\"",
   chip_pin = "58"*) input pba);

reg [1:0] min1, min2, max1, max2, mid1, mid2, minRg, maxRg;
```

```
task sort;
	output reg [1:0] maximum, minimum;
	input reg [1:0] a, b;
	begin
		maximum <= (a > b) ? a : b;
		minimum <= (a > b) ? b : a;
	end
endtask

always @ (posedge clk, negedge pba) begin
	if (!pba) begin
		max <= 2'b0;
		max_min <= 2'b0;
		min_max <= 2'b0;
		min <= 2'b0;
		min1 <= 2'b0;
		min2 <= 2'b0;
		max1 <= 2'b0;
		max2 <= 2'b0;
		mid1 <= 2'b0;
		mid2 <= 2'b0;
		minRg <= 2'b0;
		maxRg <= 2'b0;
	end else begin
		sort(max1, min1, a, b);
		sort(max2, min2, c, d);
		sort(maxRg, mid1, max1, max2);
		sort(mid2, minRg, min1, min2);
		sort(max_min, min_max, mid1, mid2);
		max <= maxRg;
		min <= minRg;
	end
end

endmodule
```

Рис.2.1 Описание на SystemVerilog

3.3 Результат синтеза (RTL)

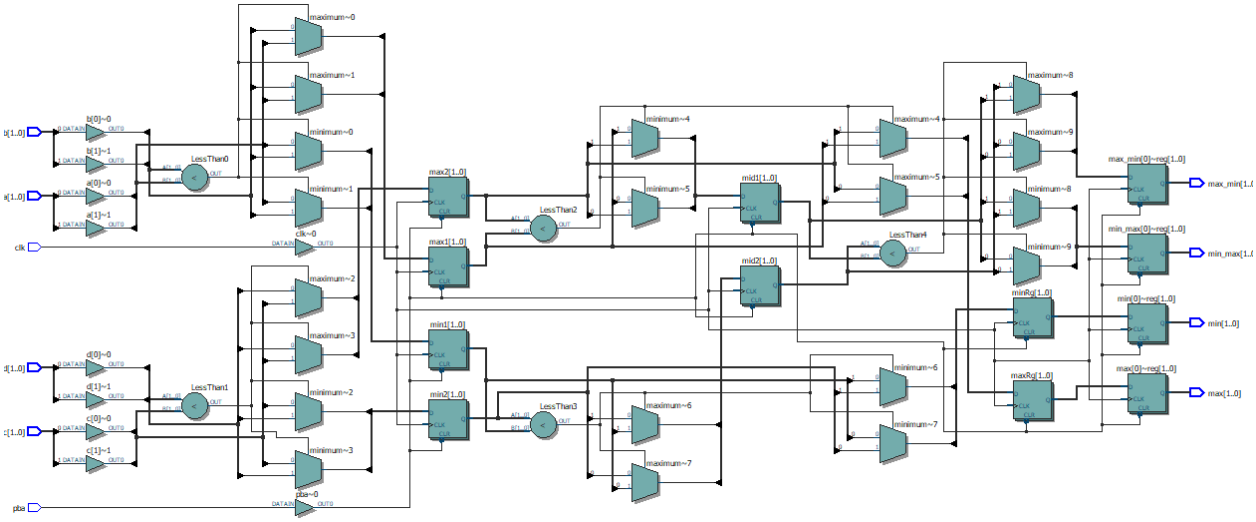


Рис. 3.3 Синтезированная схема

3.4 Моделирование

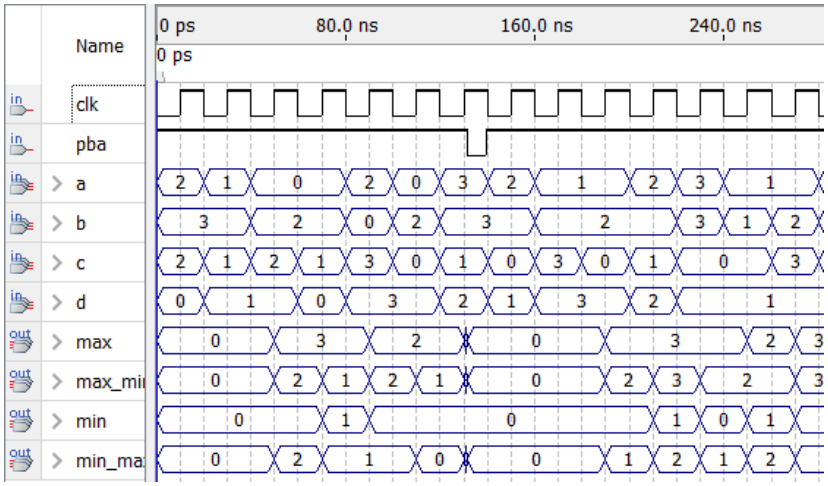


Рис.3.4 Результаты моделирования

3.5 Тестирование на плате miniDiLaB-CIV

Для тестирования проекта на плате использовались тесты, описанные в разделе 3.4

Проведенное на плате miniDiLaB-CIV тестирование разработанного устройства показало: результаты совпадают с ожидаемыми, устройство работает в соответствии с заданием.

3.6 Выводы

В ходе выполнения работы была осуществлена конвейеризация устройства сортировки входных чисел по возрастанию. СБИС были назначены с использованием атрибутов.

4 Задание lab6_4

4.1 Задание

На языке Verilog введите представленный ниже текст

```
module lab7_ (sel,  x, y, z, q1, q2, q3);
input [2:0] sel;
input x, y, z;
output reg q1, q2, q3;
always @*
begin
if (sel == 3'h1) q1 = x;
else if (sel == 3'h2) q2 = y;
     else if ( sel == 3'h3) q3 = z;
      else
         begin
            q1 = 1'b0;
            q2 = 1'b0;
            q3 = 1'b0;
         end
end
endmodule
```

Осуществите анализ и синтез, посмотрите:

- Предупреждения в процессоре сообщений и проверить наличие предупреждений о триггерах защелках (Latch)

-синтезированную пакетом Q схему (RTL Viewer), найти триггеры защелки

- Оцените аппаратные затраты (запомните кол-во использованных логических элементов)

- Исправьте описание так, чтобы не нарушая логику работы исключить возможность появления триггеров-защелок.

- Повторите анализ и синтез, убедитесь, что триггеры-защелки отсутствуют (нет предупреждений в процессоре сообщений; результаты синтеза в RTL Viewer)

- Оцените аппаратные затраты (сравните с результатами, полученными ранее)

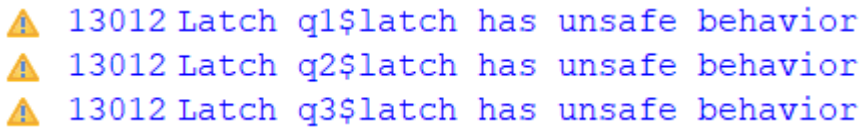


Рис. 3.3 Предупреждения

Total logic elements	9 / 6,272 (< 1 %)
Total combinational functions	9 / 6,272 (< 1 %)
Dedicated logic registers	0 / 6,272 (0 %)
Total registers	0

Рис. 3.3 Аппаратурные затраты

Теперь изменим схему.

4.1 Описание на языке Verilog

```
module lab6_4 (sel, x, y, z, q1, q2, q3);
input [2:0] sel;
input x, y, z;
output reg q1, q2, q3;

always @* begin
q1 = 1'b0;
q2 = 1'b0;
q3 = 1'b0;

if (sel == 3'h1) q1 = x;
if (sel == 3'h2) q2 = y;
if (sel == 3'h3) q3 = z;

end
endmodule
```

Рис.2.1 Описание на SystemVerilog

Приведенные выше предупреждения были устранены.

Total logic elements	3 / 6,272 (< 1 %)
Total combinational functions	3 / 6,272 (< 1 %)
Dedicated logic registers	0 / 6,272 (0 %)
Total registers	0

Рис. 3.3 Аппаратурные затраты

4.1 Результат синтеза (RTL)

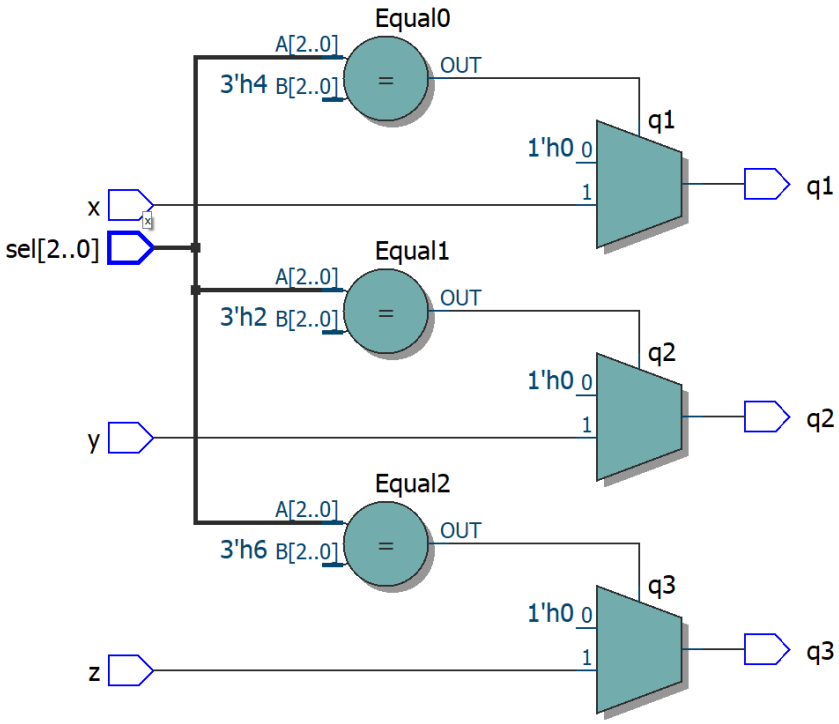


Рис. 3.3 Синтезированная схема

4.2 Выводы

В ходе выполнения работы были проанализированы предупреждения в процессоре сообщений о триггерах защелках (Latch), оценены аппаратные затраты. Схема была изменна таким образом, чтобы уменьшить аппаратные затраты и избавиться от предупреждений.

5 Выводы

В ходе выполнения работы было спроектировано устройство, выполняющее преобразование двоичного кода в код Грея и обратное преобразование кода Грея в двоичный код, устройство сортировки по возрастанию, также была проведена его конвейеризация. Кроме того, был видоизменен исходный код, в результате чего удалось избавиться от предупреждений и уменьшить аппаратные затраты.