**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «мэи»**

**Институт Радиотехники и электроники им. В.А. Котельникова**

**Кафедра Электроники и наноэлектроники**

**Типовой расчет**

Дисциплина: Основы цифрового синтеза

Тема: Проектирование HDL-описания компонента интегральной схемы

|  |  |  |
| --- | --- | --- |
| Студент гр. ЭР-05-20 |  | Волчков Д.Н. |
| Преподаватель, к.т.н., доцент | (подпись) | Баринов А.Д. |
|  | (оценка/зачёт, подпись) |  |

Москва

2023

Содержание

[Задание: 3](#_Toc153318183)

[1. Функциональная схема устройства 4](#_Toc153318184)

[2. Временная диаграмма 6](#_Toc153318185)

[3. HDL – описание модулей 7](#_Toc153318186)

[4. Модули тестирования HDL – описаний 7](#_Toc153318187)

[5. Моделирование полной схемы 8](#_Toc153318188)

[6. RTL – представление 8](#_Toc153318189)

[7. Gate – level – моделирование 8](#_Toc153318190)

[8. Ресурсы ПЛИС 9](#_Toc153318191)

[Приложение А 10](#_Toc153318192)

[Приложение Б 11](#_Toc153318193)

[Приложение В 26](#_Toc153318194)

[Приложение Г 46](#_Toc153318195)

[Приложение Д 57](#_Toc153318196)

[Приложение Е 69](#_Toc153318197)

### Задание:

1. Для разрабатываемого устройства приведите *функциональную* или *структурную* схему устройства и *описание* принципа его работы. Здесь же приведите *таблицу*, описывающую входные и выходные сигналы, а также их активных уровней на основе примера, приведённого ниже (пример разработан для суммирующего счётчика с предустановкой его начального значения).
2. Приведите пример временной диаграммы, демонстрирующей работоспособность устройства во всех (или типичных) режимах его работы из предыдущего семестра. Прокомментируйте диаграмму.
3. Приведите поведенческое HDL-описание модулей, составляющих устройство.
4. Для каждого модуля приведите его модуль тестирования (testbench) с демонстрацией работоспособности модуля и описанием того, на что читатель должен обратить внимание при просмотре временной диаграммы.
5. Для полной схемы приведите её модуль тестирования и также продемонстрируйте её работоспособность с описанием того, на что читатель должен обратить внимание при просмотре временной диаграммы.
6. Для каждого модуля в пп. 4-5 приведите его RTL-представление и технологическое отображение для ПЛИС семейства Cyclone IV EP4CE6E22C8.
7. Приведите результат моделирования для RTL-моделирования и Gate-level-моделирования. Покажите, что оно совпадает с моделированием RTL-уровня.
8. Укажите, сколько комбинационной логики и регистров использовано в качестве ресурсов ПЛИС.

### Функциональная схема устройства

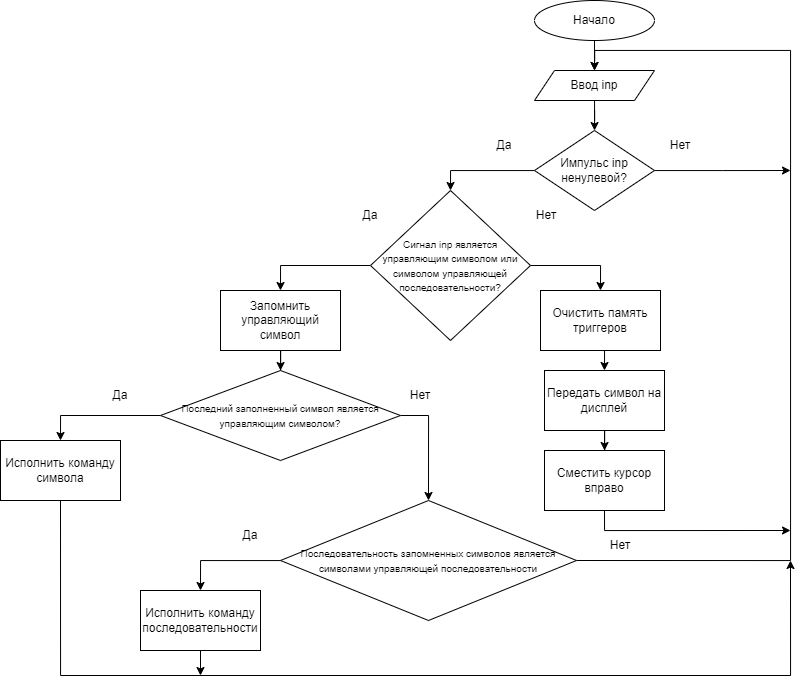


Рисунок 1 – Функциональная схема устройства

Таблица 1. Входные и выходные сигналы

|  |  |  |  |
| --- | --- | --- | --- |
| Сигнал | Направление | Функционал | Примечание |
| gen\_in | Вход | Тактируемый сигнал | Работает по переднему фронту |
| Priem\_in | Вход | Сигнал приема inp<1:8> | Передний фронт передает сигнал на дальнейшие блоки |
| inp<1:8> | Вход | Сигнал для записи в регистр | Сигнал с клавиши в кодировке ASCII |
| Disp<1:4> | Выход | Сигнал на дисплей | Сигнал BCD кодировки на семисегментный индикатор |
| Reset\_<1:4> | Вход | Сброс значения регистров | Работает по переднему фронту |
| clear\_disp | Вход | Сброс дисплея | Работает по переднему фронту |
| VSS | Вход | Контакт «земли» для ядра микросхемы | Напряжение 2.5 В |
| VDD | Вход | Контакт «питания» для ядра микросхемы | Напряжение 0 В |

### Временная диаграмма

Промоделируем поведение схемы из курсовой работы (Приложение А). Тактовый генератор является входом gen\_in к которому подключается тактирующий сигнал. Четыре последующих входа являются сигналами Reset\_<1:4> для сброса сигналов в регистре в режиме отладки схемы. Восьмибитный сигнал является сигналом inp со входа схемы. Последующие четыре четырехбитных сигнала являются сигналами Disp<1:4>, которые подаются на дисплей и через преобразователь BCD сигнала идут на семисегментный индикатор. Предпоследний сигнал является сигналом Priem\_in, с помощью которого подается сигнал передачи входного сигнала inp на вход схемы и его обработки. Последний сигнал является сигналом очистки дисплея clear\_disp и используется для отладки схемы.

Рассмотрим диаграмму моделирования. Сначала вводится символ «3», который является как обычным символом, так и символом управляющей последовательности. В данном случае схема распознает его как символ обычный символ, поскольку в памяти регистров нет предыдущих символов управляющей последовательности.

Затем подается череда сигналов управляющей последовательности «курсор вправо», что сдвигает курсор на одну ячейку вперед, после чего подается управляющий сигнал «пробел», что также сдвигает курсор на ячейку вперед.

Затем вводится символ «7», который выводится на 4-ой ячейке дисплея.

Тем самым мы подтвердили работоспособность схемы согласно функциональной схеме в полной мере.

### HDL – описание модулей

Разделим устройство на отдельные модули, согласно разделению в курсовой работе и зададим для каждого HDL – описание. Всего модулей будет 12. Из них 10 являются модулями из курсовой работы, один является модулем старшего уровня, а последний является регистром на 8 бит. Описания модулей приведены в приложении Б.

### Модули тестирования HDL – описаний

Составим модули тестирования для каждого модуля HDL – описания, чтобы протестировать их соответствие модулям из курсовой работы. Описания модулей тестирования приведены в приложении В. Отдельно разберем временные диаграммы тестирований (Приложение Г):

Рассмотрим первый рисунок временной диаграммы модуля del\_par. Как можно заметить, блок выдает единицу на выходе при получении пятого сигнала согласно параметру Delay блока. Сигнал priem сбрасывает счетчик сигналов, при этом сигнал с выхода не попадает на выход (Рисунок 1).

Следующий рисунок временной диаграммы gate. Сигнал попадает на выход только при положительных сигналах sig и clk. Сигнал right\_cursor подает сигнал на перемещение курсора через 3 тактирующих импульса (Рисунок 2).

В временной диаграмме to\_save сигналы YP и YC меняются в зависимости от того является символ символом управляющей последовательности, управляющим символом или обычным символом.

В временной диаграмме Cont\_seq в зависимости от значений запомненных символов переключаются значения сигналов ESC (Рисунок 3).

Временная диаграмма модуля register не приведена, поскольку модуль является обычным 8-ми битным регистром (Рисунок 4).

В временной диаграмме Comands выходные сигналы зависят от того, какая последовательность управляющих символов запомнена (Рисунок 5).

В временной диаграмме obrabotka в зависимости от входных сигналов запоминаются управляющие символы и на основе запомненных символов выводятся определённые сигналы команд (Рисунок 6).

В временной диаграмме generator выводятся шесть тактирующих сигналов и далее ожидается очистка символов сигналом priem\_in (Рисунок 7).

В временной диаграмме regist в зависимости от поданного сигнала переключаются сигналы SET, при этом сигналы не выходят дальше положений SET1 и SET4 (Рисунок 8).

В временной диаграмме Display в зависимости от положения курсора (сигналы SET) записывается значение для дисплея, при этом присутствует защита от перезаписи в одну ячейку значения, значение можно только удалить и записать новое (Рисунок 9).

В предпоследней временной диаграмме ascii\_to\_bcd сигнал на выходе выводится только при правильной ASCII – кодировке символа числа. Иначе выводятся все единицы что воспринимается дешифратором сигнала BCD в сигналы семисегментного индикатора как отсутствие сигнала (Рисунок 10).

Последняя временная диаграмма main повторяет временную диаграмму из приложения А, что подтверждает что RTL – моделирование задания курсового проекта является верным (Рисунок 11).

### Моделирование полной схемы

Моделирование полной схемы также представлено в приложениях Г и Д (Рисунок 11 и 12).

### RTL – представление

RTL – представления модулей представлено в приложении Д.

### Gate – level – моделирование

RTL – моделирования и Gate – level – моделирования совпадают, их временные диаграммы представлены в приложении Е.

### Ресурсы ПЛИС

Рассмотрим какие ресурсы требуются от ПЛИС для моделирования поведения данной схемы (Рисунок 2):

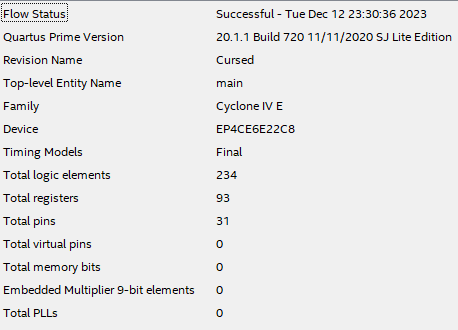
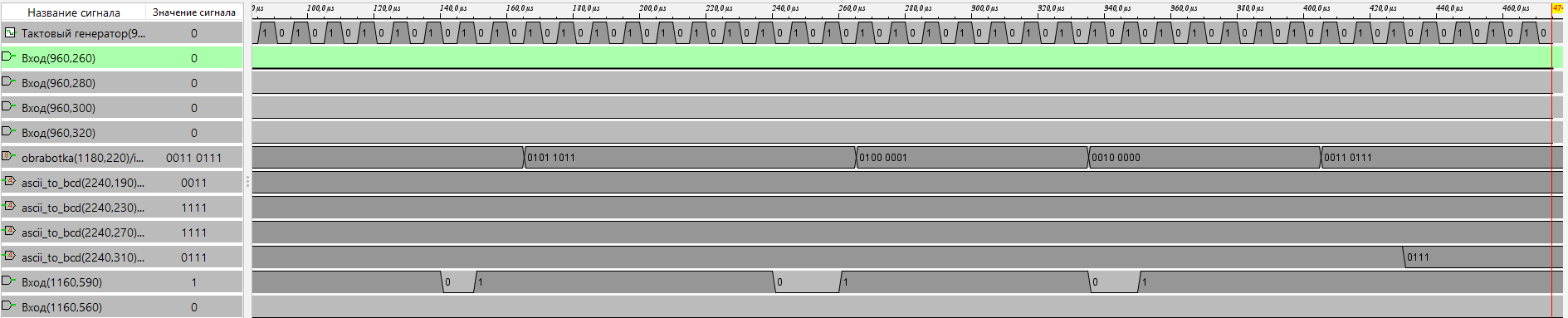


Рисунок 2 – Отчет моделирования

Согласно отчету моделирования, проект потребует 234 логических элемента и 93 регистра.

### Приложение А



### Приложение Б

HDL – описание модулей

Описание модуля del\_par

module delay\_par #(parameter delay = 1) (clk, priem, del);

input logic clk, priem;

output logic del;

logic clear;

logic [2:0] Q = 0;

always\_ff @(posedge clk, posedge clear)

if (clear) Q <= 0;

else if (clk && Q < delay) Q <= Q + 1;

else Q <= Q;

assign del = (Q < delay) ? 0 : 1;

assign clear = (Q == delay && priem) ? 1 : 0;

endmodule

Описание модуля gate

module gate (clk, sig, inp, priem, res, right\_cursor);

input logic clk, sig, priem;

input logic [7:0] inp;

output logic [7:0] res;

output logic right\_cursor;

logic w;

logic [3:0] s;

assign w = (clk & sig) ? 1 : 0;

assign res[7] = (w) ? inp[7] : 0;

assign res[6] = (w) ? inp[6] : 0;

assign res[5] = (w) ? inp[5] : 0;

assign res[4] = (w) ? inp[4] : 0;

assign res[3] = (w) ? inp[3] : 0;

assign res[2] = (w) ? inp[2] : 0;

assign res[1] = (w) ? inp[1] : 0;

assign res[0] = (w) ? inp[0] : 0;

delay\_par #(.delay(3)) U1 (.clk(clk), .priem(priem), .del(s[2]));

delay\_par #(.delay(4)) U2 (.clk(clk), .priem(priem), .del(s[1]));

assign s[3] = ~s[1];

and(right\_cursor, s[2], s[3], sig);

endmodule

Описание модуля to\_save

module to\_save(inp, ESC1, ESC2, ESC3, YP, YC);

input logic ESC1, ESC2, ESC3;

input logic [7:0] inp;

output logic YP, YC;

logic ESC, CSI, cur\_left, cur\_right, Delete1, Delete2, Space, BackSpace, Enter;

or(YP, ~YC, ESC, CSI, cur\_left, cur\_right, Delete1, Delete2);

nor(YC, Space, BackSpace, Enter);

assign ESC = (inp == 8'b00011011) ? 1 : 0;

assign CSI = (inp == 8'b01011011 & ESC1) ? 1 : 0;

assign cur\_left = (inp == 8'b01000100 & ESC2) ? 1 : 0;

assign cur\_right = (inp == 8'b01000001 & ESC2) ? 1 : 0;

assign Delete1 = (inp == 8'b00110011 & ESC2) ? 1 : 0;

assign Delete2 = (inp == 8'b01111110 & ESC3) ? 1 : 0;

assign Space = (inp == 8'b00100000) ? 1 : 0;

assign BackSpace = (inp == 8'b00001000) ? 1 : 0;

assign Enter = (inp == 8'b00001101) ? 1 : 0;

endmodule

Описание модуля Cont\_seq

module Cont\_seq (Trig\_1, Trig\_2, Trig\_3, clk, priem, ESC1, ESC2, ESC3);

input logic clk, priem;

input logic [7:0] Trig\_1, Trig\_2, Trig\_3;

output logic ESC1, ESC2, ESC3;

// enum logic [7:0] {ESC=8'b00011011, CSI=8'b01011011, left\_cursor=8'b01000100, right\_cursor=8'b01000001, Delete1=8'd00110011, Delete2=8'b01111110, Space=8'b00100000, BackSpace=8'b00001000, Enter=8'b00001101} help;

logic s;

logic [2:0] set, reset;

delay\_par #(.delay(5)) U1 (.clk(clk), .priem(priem), .del(s));

assign set[0] = (s & Trig\_1 == 8'b00011011) ? 1 : 0;

assign set[1] = (s & Trig\_1 == 8'b01011011 & Trig\_2 == 8'b00011011) ? 1 : 0;

assign set[2] = (s & Trig\_1 == 8'b00110011 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011) ? 1 : 0;

assign reset[0] = (s & ((Trig\_1 == 8'b01011011 & Trig\_2 == 8'b00011011) | (Trig\_1 == 8'b00110011 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011) | !(Trig\_1 == 8'b00011011 | (Trig\_1 == 8'b01011011 & Trig\_2 == 8'b00011011) | (Trig\_1 == 8'b00110011 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011) ))) ? 1 : 0;

assign reset[1] = (s & (Trig\_1 == 8'b00011011 | (Trig\_1 == 8'b00110011 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011) | !(Trig\_1 == 8'b00011011 | (Trig\_1 == 8'b01011011 & Trig\_2 == 8'b00011011) | (Trig\_1 == 8'b00110011 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011) ))) ? 1 : 0;

assign reset[2] = (s & (Trig\_1 == 8'b00011011 | (Trig\_1 == 8'b01011011 & Trig\_2 == 8'b00011011) | !(Trig\_1 == 8'b00011011 | (Trig\_1 == 8'b01011011 & Trig\_2 == 8'b00011011) | (Trig\_1 == 8'b00110011 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011) ))) ? 1 : 0;

always\_ff @(posedge set[0], posedge reset[0])

if (set[0]) ESC1 <= 1;

else if (reset[0]) ESC1 <= 0;

else ESC1 <= ESC1;

always\_ff @(posedge set[1], posedge reset[1])

if (set[1]) ESC2 <= 1;

else if (reset[1]) ESC2 <= 0;

else ESC2 <= ESC2;

always\_ff @(posedge set[2], posedge reset[2])

if (set[2]) ESC3 <= 1;

else if (reset[2]) ESC3 <= 0;

else ESC3 <= ESC3;

endmodule

Описание модуля Comands

module Comands (Trig\_1, Trig\_2, Trig\_3, Trig\_4, right\_from\_gate, clk, priem, left\_cursor, right\_cursor, Delete, Enter);

input logic [7:0] Trig\_1, Trig\_2, Trig\_3, Trig\_4;

input logic right\_from\_gate, clk, priem;

output logic left\_cursor, right\_cursor, Delete, Enter;

logic s[2:0];

logic delete;

and(s[2], s[1], !s[0]);

delay\_par #(.delay(3)) U1 (.clk(clk), .priem(priem), .del(s[1]));

delay\_par #(.delay(4)) U2 (.clk(clk), .priem(priem), .del(s[0]));

assign Enter = (s[1] & Trig\_1 == 8'b00001101) ? 1 : 0;

assign Delete = (s[0] & (Trig\_1 == 8'b00001000 | (Trig\_1 == 8'b01111110 & Trig\_2 == 8'd00110011 & Trig\_3 == 8'b01011011 & Trig\_4 == 8'b00011011))) ? 1 : 0;

assign right\_cursor = (s[2] & (right\_from\_gate | Trig\_1 == 8'b00100000 | (Trig\_1 == 8'b01000001 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011))) ? 1 : 0;

assign left\_cursor = (s[2] & (Trig\_1 == 8'b00001000 | (Trig\_1 == 8'b01000100 & Trig\_2 == 8'b01011011 & Trig\_3 == 8'b00011011))) ? 1 : 0;

endmodule

Описание модуля register

module register (Q, D, clk, rst);

input logic clk, rst;

input logic [7:0] D;

output logic [7:0] Q = 0;

always\_ff @(posedge rst, posedge clk)

if (rst) Q <= 0;

else if (clk) Q <= D;

else Q <= Q;

endmodule

Описание модуля obrabotka

module obrabotka (inp, clk, priem, Reset\_1, Reset\_2, Reset\_3, Reset\_4, res, left\_cursor, right\_cursor, Delete, Enter);

input logic clk, priem, Reset\_1, Reset\_2, Reset\_3, Reset\_4;

input logic [7:0] inp;

output logic [7:0] res;

output logic left\_cursor, right\_cursor, Delete, Enter;

logic ESC1, ESC2, ESC3;

logic nclk, res0, res1 ,res2, res3, res4, right\_from\_gate, sig, YC, YP, zero;

logic [2:0] s;

logic [7:0] Trig\_1, Trig\_2, Trig\_3, Trig\_4;

assign zero = (inp == 8'b00000000) ? 0 : 1;

register reg\_1 (.Q(Trig\_1), .D(inp), .clk(nclk), .rst(res1));

register reg\_2 (.Q(Trig\_2), .D(Trig\_1), .clk(nclk), .rst(res2));

register reg\_3 (.Q(Trig\_3), .D(Trig\_2), .clk(nclk), .rst(res3));

register reg\_4 (.Q(Trig\_4), .D(Trig\_3), .clk(nclk), .rst(res4));

delay\_par #(.delay(1)) U1 (.clk(clk), .priem(priem), .del(s[0]));

delay\_par #(.delay(2)) U2 (.clk(clk), .priem(priem), .del(s[1]));

assign s[2]= (~s[1] & s[0]) ? 1 : 0;

assign sig = (YC & ~YP & zero) ? 1 : 0;

gate gate\_1 (.clk(clk), .sig(sig), .inp(inp), .priem(priem), .res(res), .right\_cursor(right\_from\_gate));

Comands comands\_1 (.Trig\_1(Trig\_1), .Trig\_2(Trig\_2), .Trig\_3(Trig\_3), .Trig\_4(Trig\_4), .right\_from\_gate(right\_from\_gate), .clk(clk), .priem(priem), .left\_cursor(left\_cursor), .right\_cursor(right\_cursor), .Delete(Delete), .Enter(Enter));

Cont\_seq Cont\_seq\_1 (.Trig\_1(Trig\_1), .Trig\_2(Trig\_2), .Trig\_3(Trig\_3), .clk(clk), .priem(priem), .ESC1(ESC1), .ESC2(ESC2), .ESC3(ESC3));

to\_save to\_save\_1 (.inp(inp), .ESC1(ESC1), .ESC2(ESC2), .ESC3(ESC3), .YP(YP), .YC(YC));

assign res0 = (!YP & s[2] & clk);

assign res1 = (res0 | Reset\_1) ? 1 : 0;

assign res2 = (res0 | Reset\_2) ? 1 : 0;

assign res3 = (res0 | Reset\_3) ? 1 : 0;

assign res4 = (res0 | Reset\_4) ? 1 : 0;

assign nclk = (s[2] & zero & YP) ? 1 : 0;

endmodule

Описание модуля generator

module generator (gen\_in, priem\_in, clk, priem\_out);

input logic gen\_in, priem\_in;

output logic clk, priem\_out;

logic [2:0] Q;

logic nclk;

always\_ff @(posedge priem\_out, posedge nclk)

if(priem\_out) Q <= 0;

else if (nclk) Q <= Q + 1;

else Q <= Q;

assign priem\_out = ~priem\_in;

assign nclk = (Q <= 3'd6 & gen\_in ) ? 1 : 0;

assign clk = (gen\_in & priem\_in & Q <= 3'd6) ? 1 : 0;

endmodule

Описание модуля regist

module regist (left, right, first\_pos, SET1, SET2, SET3, SET4);

input logic left, right, first\_pos;

output logic SET1, SET2, SET3, SET4;

logic [2:0] Q = 1;

logic left\_, right\_;

always\_ff @(posedge first\_pos, posedge left\_, posedge right\_)

if (first\_pos) Q <= 1;

else if (left\_) Q <= Q - 1;

else if (right\_) Q <= Q + 1;

else Q <= Q;

assign left\_ = (left & Q > 1) ? 1 : 0;

assign right\_ = (right & Q < 4) ? 1 : 0;

assign SET1 = (Q == 3'd1) ? 1 : 0;

assign SET2 = (Q == 3'd2) ? 1 : 0;

assign SET3 = (Q == 3'd3) ? 1 : 0;

assign SET4 = (Q == 3'd4) ? 1 : 0;

endmodule

Описание модуля Display

module Display (inp, clk, priem, SET1, SET2, SET3, SET4, Reset\_1, Reset\_2, Reset\_3, Reset\_4, Disp1, Disp2, Disp3, Disp4);

input logic clk, priem, SET1, SET2, SET3, SET4, Reset\_1, Reset\_2, Reset\_3, Reset\_4;

output logic [7:0] Disp1=8'd0, Disp2=8'd0, Disp3=8'd0, Disp4=8'd0;

input logic [7:0] inp;

logic [1:0] s;

logic zero, nclk, clk1, clk2, clk3, clk4;

register reg\_1 ( .Q(Disp1), .D(inp), .clk(clk1), .rst(Reset\_1));

register reg\_2 ( .Q(Disp2), .D(inp), .clk(clk2), .rst(Reset\_2));

register reg\_3 ( .Q(Disp3), .D(inp), .clk(clk3), .rst(Reset\_3));

register reg\_4 ( .Q(Disp4), .D(inp), .clk(clk4), .rst(Reset\_4));

delay\_par #(.delay(2)) U1 (.clk(clk), .priem(priem), .del(s[0]));

delay\_par #(.delay(3)) U2 (.clk(clk), .priem(priem), .del(s[1]));

assign zero = (inp == 8'd0) ? 0 : 1;

assign nclk = (zero & s[0] & ~s[1]) ? 1 : 0;

assign clk1 = (nclk & SET1 & Disp1 == 8'd0) ? 1 : 0;

assign clk2 = (nclk & SET2 & Disp2 == 8'd0) ? 1 : 0;

assign clk3 = (nclk & SET3 & Disp3 == 8'd0) ? 1 : 0;

assign clk4 = (nclk & SET4 & Disp4 == 8'd0) ? 1 : 0;

endmodule

Описание модуля ascii\_to\_bcd

module ascii\_to\_bcd (ASCII, BCD);

input logic [7:0] ASCII;

output logic [3:0] BCD;

assign BCD = (ASCII[7:4] == 4'b0011) ? ASCII[3:0] : 4'b1111;

endmodule

Описание модуля main

module main (inp, Reset\_1, Reset\_2, Reset\_3, Reset\_4, gen\_in, priem\_in, clear\_disp, Disp1, Disp2, Disp3, Disp4);

input logic Reset\_1, Reset\_2, Reset\_3, Reset\_4, gen\_in, priem\_in, clear\_disp;

input logic [7:0] inp;

output logic [3:0] Disp1, Disp2, Disp3, Disp4;

logic nclk, priem\_out, left\_cursor, right\_cursor, Delete, Enter, first\_pos, res1 , res2, res3, res4, SET1, SET2, SET3, SET4;

logic [7:0] res, dis1, dis2, dis3, dis4;

obrabotka obrabotka\_1 (.inp(inp), .clk(nclk), .priem(priem\_out), .Reset\_1(Reset\_1), .Reset\_2(Reset\_2), .Reset\_3(Reset\_3), .Reset\_4(Reset\_4), .res(res), .left\_cursor(left\_cursor), .right\_cursor(right\_cursor), .Delete(Delete), .Enter(Enter));

generator generator\_1 (.gen\_in(gen\_in), .priem\_in(priem\_in), .clk(nclk), .priem\_out(priem\_out));

regist regist\_1 (.left(left\_cursor), .right(right\_cursor), .first\_pos(first\_pos), .SET1(SET1), .SET2(SET2), .SET3(SET3), .SET4(SET4));

Display Display\_1 (.inp(res), .clk(nclk), .priem(priem\_out), .SET1(SET1), .SET2(SET2), .SET3(SET3), .SET4(SET4), .Reset\_1(res1), .Reset\_2(res2), .Reset\_3(res3), .Reset\_4(res4), .Disp1(dis1), .Disp2(dis2), .Disp3(dis3), .Disp4(dis4));

ascii\_to\_bcd ascii\_to\_bcd\_1 (.ASCII(dis1), .BCD(Disp1));

ascii\_to\_bcd ascii\_to\_bcd\_2 (.ASCII(dis2), .BCD(Disp2));

ascii\_to\_bcd ascii\_to\_bcd\_3 (.ASCII(dis3), .BCD(Disp3));

ascii\_to\_bcd ascii\_to\_bcd\_4 (.ASCII(dis4), .BCD(Disp4));

assign first\_pos = (clear\_disp | Enter) ? 1 : 0;

assign res1 = ((Delete & SET1) | (clear\_disp | Enter)) ? 1 : 0;

assign res2 = ((Delete & SET2) | (clear\_disp | Enter)) ? 1 : 0;

assign res3 = ((Delete & SET3) | (clear\_disp | Enter)) ? 1 : 0;

assign res4 = ((Delete & SET4) | (clear\_disp | Enter)) ? 1 : 0;

endmodule

### Приложение В

Модули тестирования HDL – описаний

Модуль тестирования delay\_par

`timescale 1ns/1ns

module tb1;

logic clk, priem, s;

delay\_par #(.delay(5)) U1 (.clk(clk), .priem(priem), .del(s));

initial begin

clk = 0;

priem =0;

#20 priem = 1;

#20 priem = 0;

#20 priem = 1;

end

always #1 clk = !clk;

initial #100 $stop;

endmodule

Модуль тестирования gate

`timescale 1ns/1ns

module tb2;

logic clk, sig, priem, right\_cursor;

logic [7:0] inp, res;

gate gate\_1(.clk(clk), .sig(sig), .inp(inp), .priem(priem), .res(res), .right\_cursor(right\_cursor));

initial begin

clk = 0;

sig = 0;

priem =0;

inp = 8'b11101111;

#20 priem = 1;

#20 priem = 0;

#20 sig = 1;

#20 priem = 1;

end

always #1 clk = !clk;

initial #100 $stop;

endmodule

Модуль тестирования to\_save

`timescale 1ns/1ns

module tb3;

logic ESC1, ESC2, ESC3, YP, YC;

logic [7:0] inp;

to\_save save(.inp(inp), .ESC1(ESC1), .ESC2(ESC2), .ESC3(ESC3), .YP(YP), .YC(YC));

initial begin

ESC1=0;

ESC2=0;

ESC3=0;

inp = 8'b00001000;

#5 ESC1 = 1;

#5 ESC2 = 1;

#5 ESC3 = 1;

#5 ESC1 = 0;

#5 ESC2 = 0;

#5 ESC3 = 0;

#1 inp = 8'b01111110;

#5 ESC3 = 1;

#5 ESC3 = 0;

#1 inp = 8'b01000001;

#5 ESC2 = 1;

#5 ESC2 = 0;

#1 inp = 8'b01011011;

#5 ESC1 = 1;

#5 ESC1 = 0;

#1 inp = 8'b00011011;

#5 ESC1 = 1;

#5 ESC2 = 1;

#5 ESC3 = 1;

#5 ESC1 = 0;

#5 ESC2 = 0;

#5 ESC3 = 0;

#1 inp = 8'b11111111;

end

//always #1 clk = !clk;

initial #300 $stop;

endmodule

Модуль тестирования Cont\_seq

`timescale 1ns/1ns

module tb4;

logic clk, priem, ESC1, ESC2, ESC3;

logic [7:0] Trig\_1, Trig\_2, Trig\_3;

Cont\_seq seq(.Trig\_1(Trig\_1), .Trig\_2(Trig\_2), .Trig\_3(Trig\_3), .clk(clk), .priem(priem), .ESC1(ESC1), .ESC2(ESC2), .ESC3(ESC3));

initial begin

Trig\_1 = 8'b00011011;

Trig\_2 = 0;

Trig\_3 = 0;

clk = 0;

priem = 0;

#20 priem = 1;

Trig\_1 = 8'b01011011;

Trig\_2 = 8'b00011011;

Trig\_3 = 0;

#1 priem = 0;

#20 priem = 1;

Trig\_1 = 8'b00110011;

Trig\_2 = 8'b01011011;

Trig\_3 = 8'b00011011;

#1 priem = 0;

end

always #1 clk = !clk;

initial #300 $stop;

endmodule

Модуль тестирования Comands

`timescale 1ns/1ns

module tb5;

logic clk, priem, right\_from\_gate, left\_cursor, right\_cursor, Delete, Enter;

logic [7:0] Trig\_1, Trig\_2, Trig\_3, Trig\_4;

Comands comands(.Trig\_1(Trig\_1), .Trig\_2(Trig\_2), .Trig\_3(Trig\_3), .Trig\_4(Trig\_4), .right\_from\_gate(right\_from\_gate), .clk(clk), .priem(priem), .left\_cursor(left\_cursor), .right\_cursor(right\_cursor), .Delete(Delete), .Enter(Enter));

initial begin

Trig\_1 = 8'b00001000;

Trig\_2 = 8'd0;

Trig\_3 = 8'd0;

Trig\_4 = 8'd0;

right\_from\_gate = 0;

clk = 0;

priem = 1;

#10 priem = 0;

#40 priem = 1;

#5 Trig\_1 = 8'b01111110;

Trig\_2 = 8'd00110011;

Trig\_3 = 8'b01011011;

Trig\_4 = 8'b00011011;

#10 priem = 0;

#10 priem = 1;

#5 Trig\_1 = 8'b01000100;

Trig\_2 = 8'b01011011;

Trig\_3 = 8'b00011011;

Trig\_4 = 8'd0;

#1 priem = 0;

#10 priem = 1;

#5 Trig\_1 = 8'b01000001;

Trig\_2 = 8'b01011011;

Trig\_3 = 8'b00011011;

Trig\_4 = 8'd0;

#1 priem = 0;

end

always #1 clk = !clk;

initial #300 $stop;

endmodule

Модуль тестирования register

`timescale 1ns/1ns

module tb6;

logic clk, rst;

logic [7:0] Q, D;

register register(.Q, .D, .clk, .rst);

initial begin

clk = 0;

rst = 1;

D = 8'd100;

#10 rst = 0;

#15 D = 8'd45;

end

always #1 clk = !clk;

initial #300 $stop;

endmodule

Модуль тестирования obrabotka

`timescale 1ns/1ns

module tb7;

logic clk, priem, Reset\_1, Reset\_2, Reset\_3, Reset\_4, left\_cursor, right\_cursor, Delete, Enter;

logic [7:0] inp, res;

obrabotka obrabotka\_1(.inp(inp), .clk(clk), .priem(priem), .Reset\_1(Reset\_1), .Reset\_2(Reset\_2), .Reset\_3(Reset\_3), .Reset\_4(Reset\_4), .res(res), .left\_cursor(left\_cursor), .right\_cursor(right\_cursor), .Delete(Delete), .Enter(Enter));

initial begin

clk = 0;

priem = 1;

#50 priem = 0;

Reset\_1 = 0;

Reset\_2 = 0;

Reset\_3 = 0;

Reset\_4 = 0;

inp = 8'b11111111;

#50 priem = 1;

#4 priem = 0;

inp = 8'b00011011;

#50 priem = 1;

#4 priem = 0;

inp = 8'b01011011;

#50 priem = 1;

#4 priem = 0;

inp = 8'd00110011;

#50 priem = 1;

#4 priem = 0;

inp = 8'b01111110;

#50 priem = 1;

#4 priem = 0;

inp = 8'b11111111;

#50 priem = 1;

#4 priem = 0;

inp = 8'd0;

#50 priem = 1;

#4 priem = 0;

inp = 8'b00001000;

#50 priem = 1;

#4 priem = 0;

inp = 8'b01111111;

#50 priem = 1;

#4 priem = 0;

inp = 8'd0;

#50 priem = 1;

#4 priem = 0;

inp = 8'b00001000;

#50 priem = 1;

#4 priem = 0;

inp = 8'b01111111;

#50 priem = 1;

#4 priem = 0;

inp = 8'b00000000;

end

always #1 clk = !clk;

initial #1000 $stop;

endmodule

Модуль тестирования generator

`timescale 1ns/1ns

module tb8;

logic gen\_in, priem\_in, clk, priem\_out;

generator gen(.gen\_in(gen\_in), .priem\_in(priem\_in), .clk(clk), .priem\_out(priem\_out));

initial begin

gen\_in = 0;

priem\_in = 0;

#20 priem\_in = 1;

#6 priem\_in = 0;

#6 priem\_in = 1;

end

always #1 gen\_in = !gen\_in;

initial #100 $stop;

endmodule

Модуль тестирования regist

`timescale 1ns/1ns

module tb9;

logic left, right, first\_pos, SET1, SET2, SET3, SET4;

regist regist(.left(left), .right(right), .first\_pos(first\_pos), .SET1(SET1), .SET2(SET2), .SET3(SET3), .SET4(SET4));

initial begin

left = 0;

right = 0;

first\_pos = 0;

#1 right = 1;

#1 right = 0;

#1 right = 1;

#1 right = 0;

#1 right = 1;

#1 right = 0;

#1 right = 1;

#1 right = 0;

#1 first\_pos = 1;

#1 first\_pos = 0;

#1 right = 1;

#1 right = 0;

#1 right = 1;

#1 right = 0;

#1 right = 1;

#1 right = 0;

#1 right = 1;

#1 right = 0;

#1 left = 1;

#1 left = 0;

#1 left = 1;

#1 left = 0;

#1 left = 1;

#1 left = 0;

#1 left = 1;

#1 left = 0;

#1 left = 1;

#1 left = 0;

end

//always #1 gen\_in = !gen\_in;

initial #30 $stop;

endmodule

Модуль тестирования Display

`timescale 1ns/1ns

module tb10;

logic clk, priem, SET1, SET2, SET3, SET4, Reset\_1, Reset\_2, Reset\_3, Reset\_4;

logic [7:0] inp, Disp1, Disp2, Disp3, Disp4;

Display display(.inp(inp), .clk(clk), .priem(priem), .SET1(SET1), .SET2(SET2), .SET3(SET3), .SET4(SET4), .Reset\_1(Reset\_1), .Reset\_2(Reset\_2), .Reset\_3(Reset\_3), .Reset\_4(Reset\_4), .Disp1(Disp1), .Disp2(Disp2), .Disp3(Disp3), .Disp4(Disp4));

initial begin

inp = 8'd46;

clk = 0;

priem = 0;

SET1 = 0;

SET2 = 0;

SET3 = 0;

SET4 = 0;

Reset\_1 = 0;

Reset\_2 = 0;

Reset\_3 = 0;

Reset\_4 = 0;

#2 SET1 = 1;

SET2 = 1;

priem = 1;

#20 priem = 0;

#30 Reset\_1 = 1;

#1 Reset\_1 = 0;

end

always #1 clk = !clk;

initial #100 $stop;

endmodule

Модуль тестирования ascii\_to\_bcd

`timescale 1ns/1ns

module tb11;

logic [7:0] ASCII;

logic [3:0] BCD;

ascii\_to\_bcd ascii(.ASCII(ASCII), .BCD(BCD));

initial begin

ASCII = 8'd0;

#5 ASCII = 8'b00001000;

#5 ASCII = 8'b00111000;

#5 ASCII = 8'b00111001;

#5 ASCII = 8'b00111010;

end

//always #1 clk = !clk;

initial #100 $stop;

endmodule

Модуль тестирования main

`timescale 1ns/1ns

module tb12;

logic [7:0] inp;

logic [3:0] Disp1, Disp2, Disp3, Disp4;

logic Reset\_1, Reset\_2, Reset\_3, Reset\_4, gen\_in, priem\_in, clear\_disp;

main main(inp, Reset\_1, Reset\_2, Reset\_3, Reset\_4, gen\_in, priem\_in, clear\_disp, Disp1, Disp2, Disp3, Disp4);

initial begin

inp = 0;

Reset\_1 = 0;

Reset\_2 = 0;

Reset\_3 = 0;

Reset\_4 = 0;

gen\_in = 0;

priem\_in = 0;

clear\_disp = 0;

#10 priem\_in = 1;

#20 priem\_in = 0;

#10 inp = 8'b00110011;

priem\_in = 1;

#20 priem\_in = 0;

#10 inp = 8'b00011011;

priem\_in = 1;

#20 priem\_in = 0;

#10 inp = 8'b01011011;

priem\_in = 1;

#20 priem\_in = 0;

#10 inp = 8'b01000001;

priem\_in = 1;

#20 priem\_in = 0;

#10 inp = 8'b00100000;

priem\_in = 1;

#20 priem\_in = 0;

#10 inp = 8'b00110111;

priem\_in = 1;

#20 priem\_in = 0;

end

always #1 gen\_in = !gen\_in;

initial #1000 $stop;

endmodule

### Приложение Г

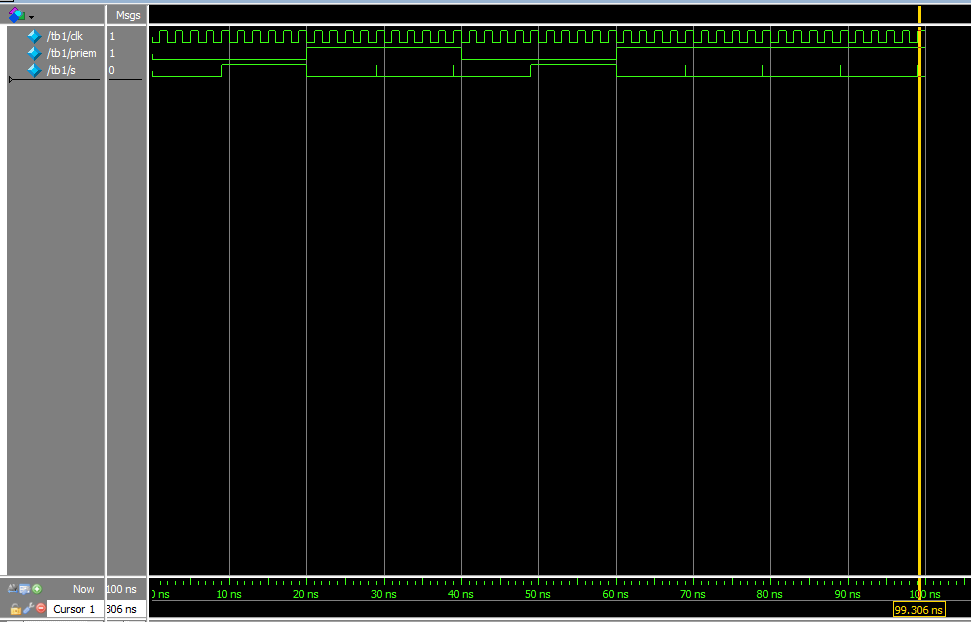


Рисунок 1 – Временная диаграмма модуля del\_par

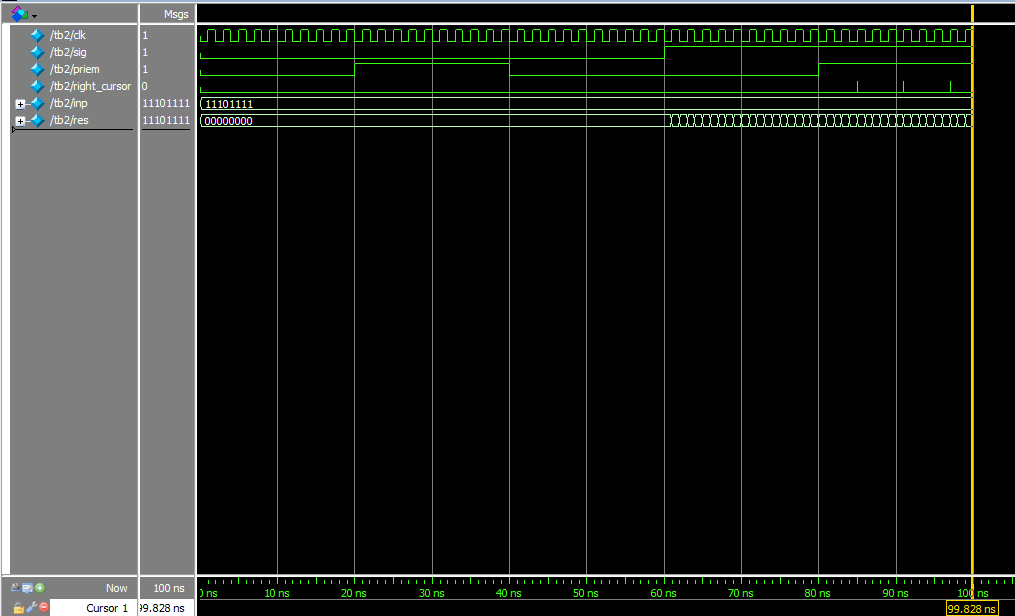


Рисунок 2 – Временная диаграмма модуля gate

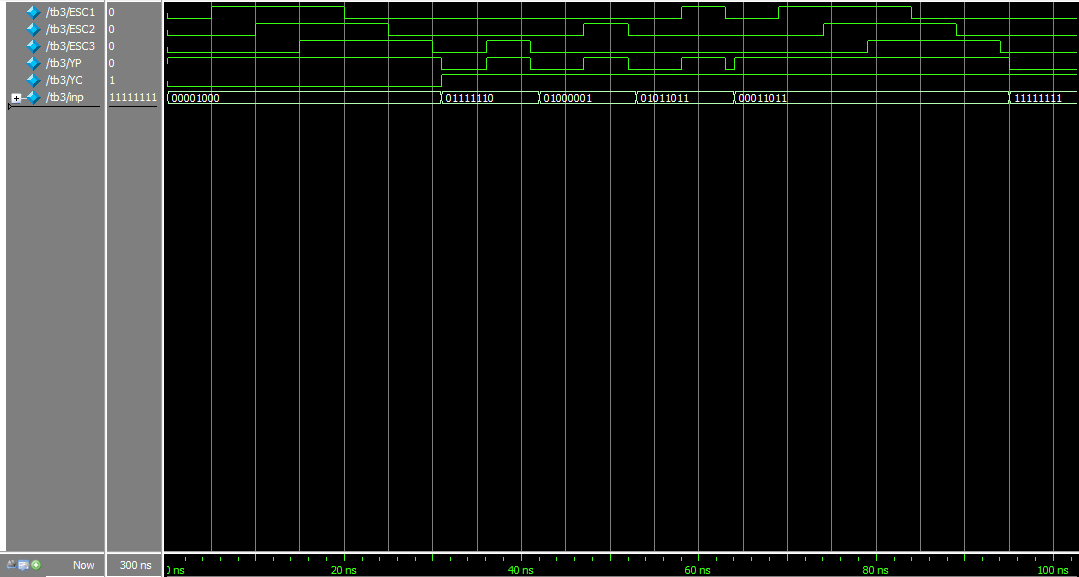


Рисунок 3 – Временная диаграмма модуля to\_save

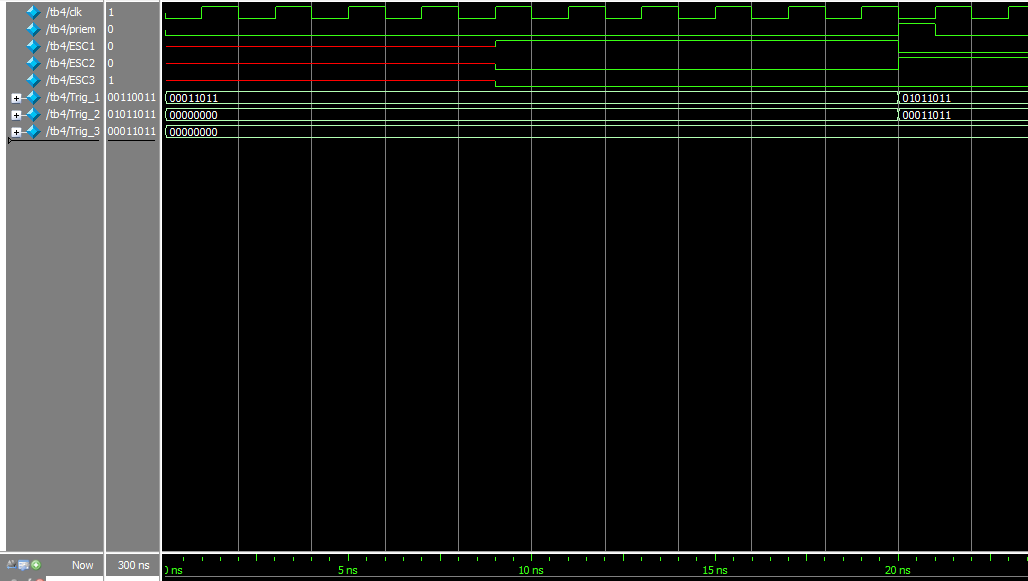


Рисунок 4 – Временная диаграмма модуля Cont\_seq

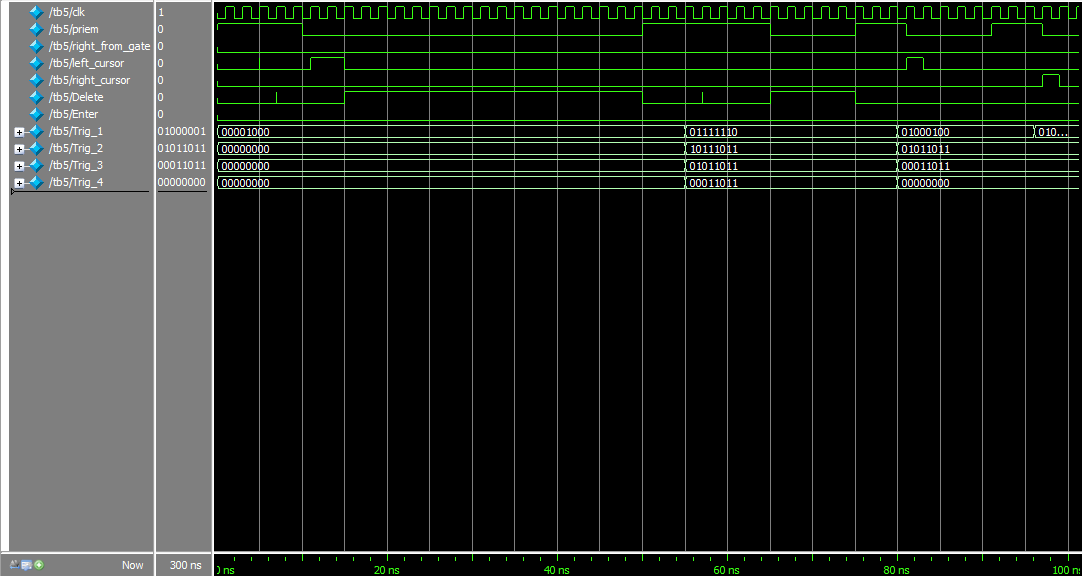


Рисунок 5 – Временная диаграмма модуля Comands

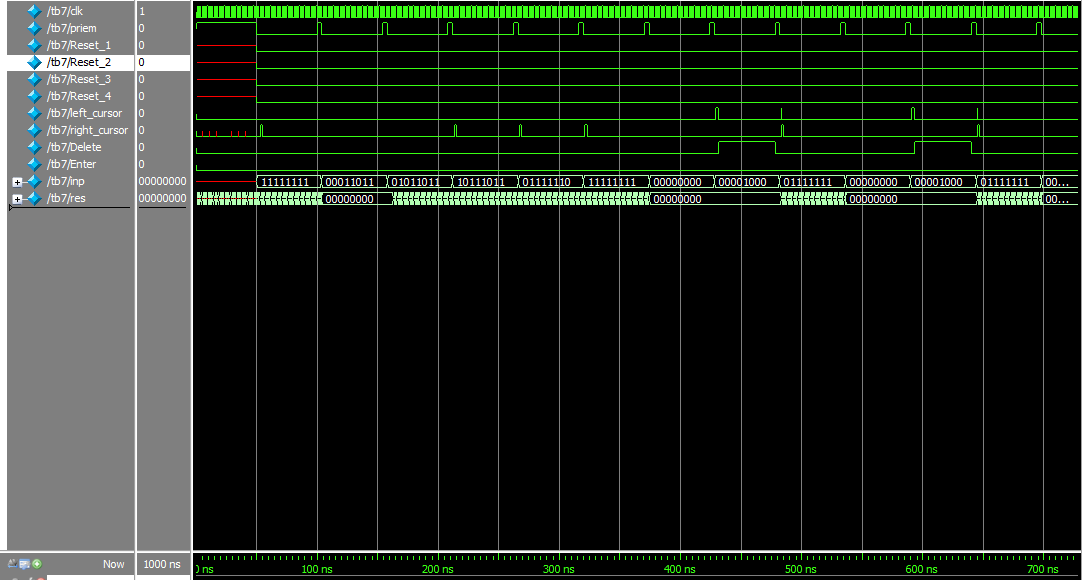


Рисунок 6 – Временная диаграмма модуля obrabotka

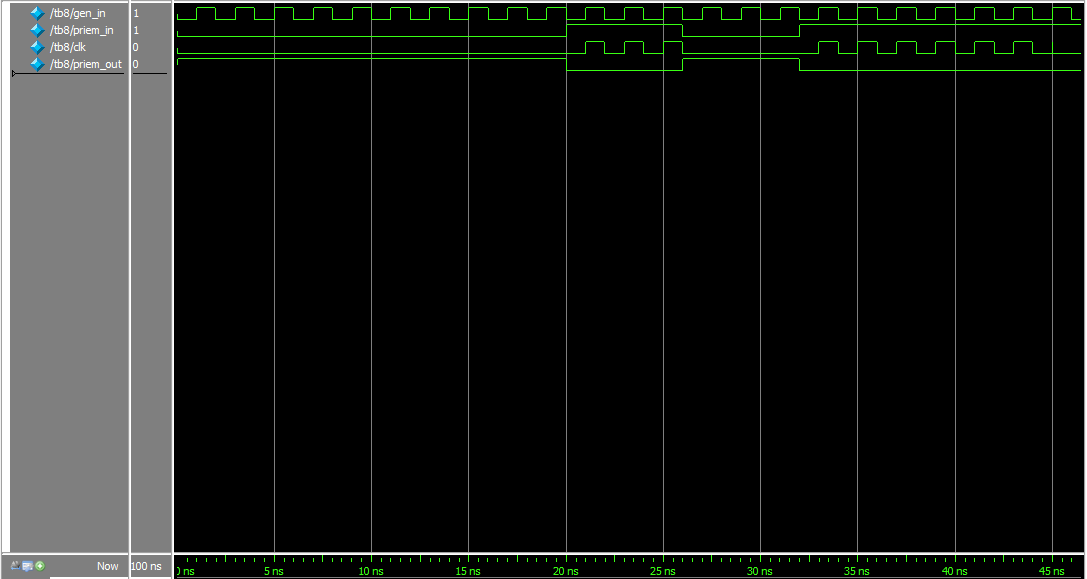


Рисунок 7 – Временная диаграмма модуля generator

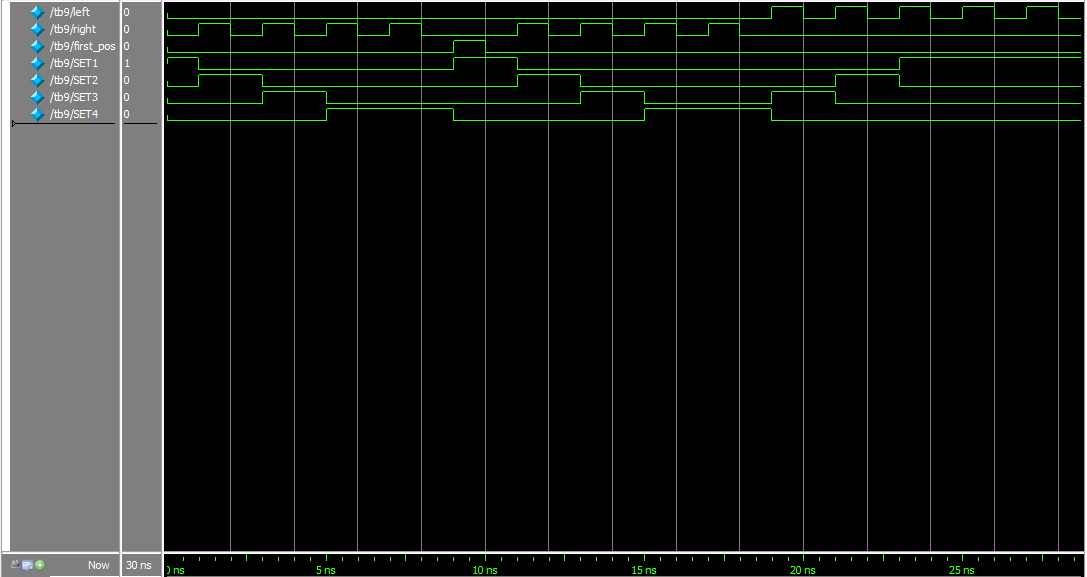


Рисунок 8 – Временная диаграмма модуля regist

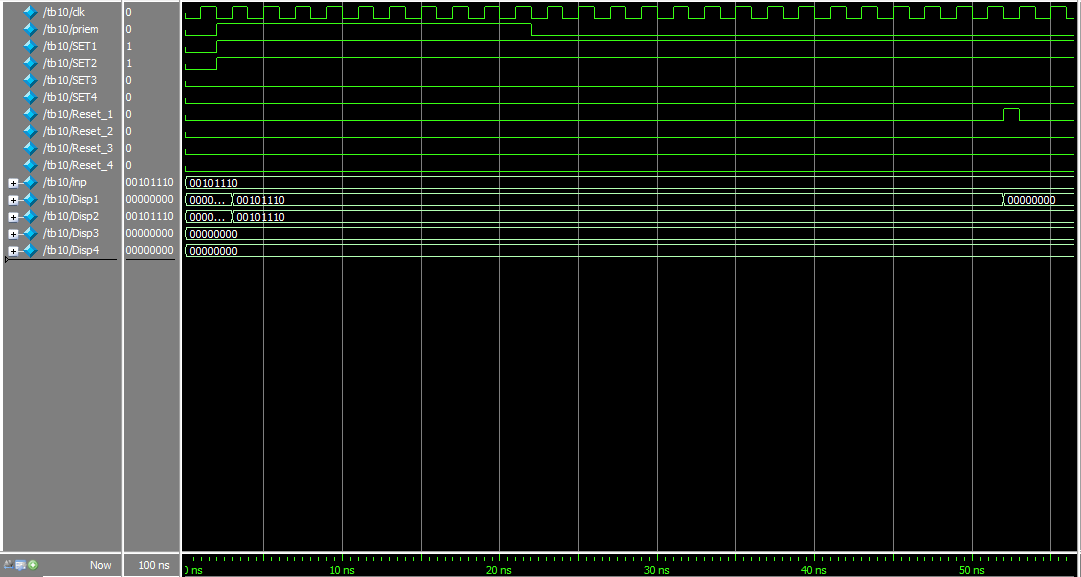


Рисунок 9 – Временная диаграмма модуля Display

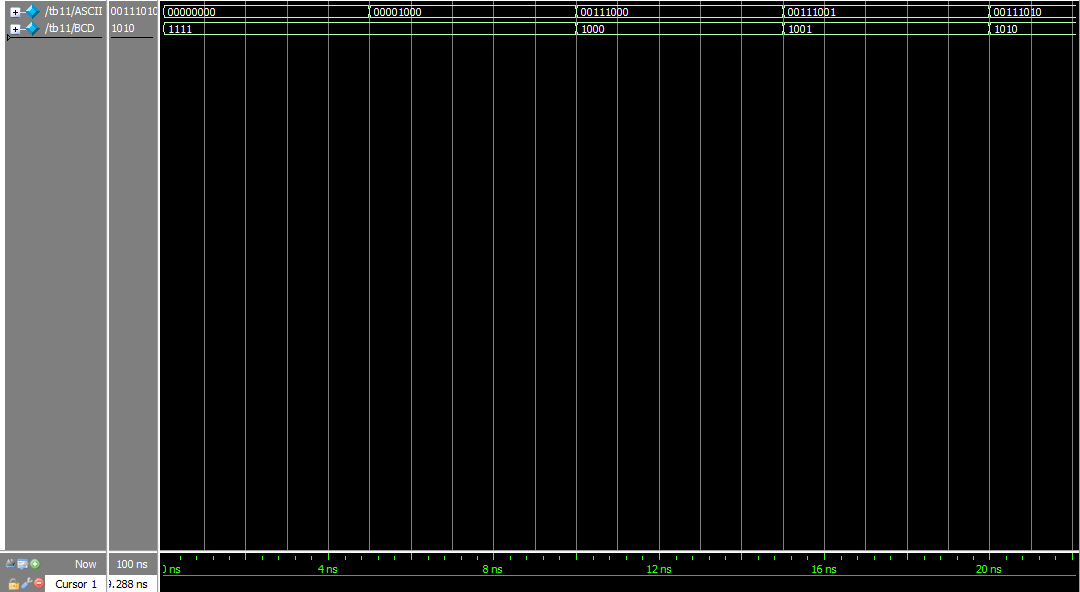


Рисунок 10 – Временная диаграмма модуля ascii\_to\_bcd

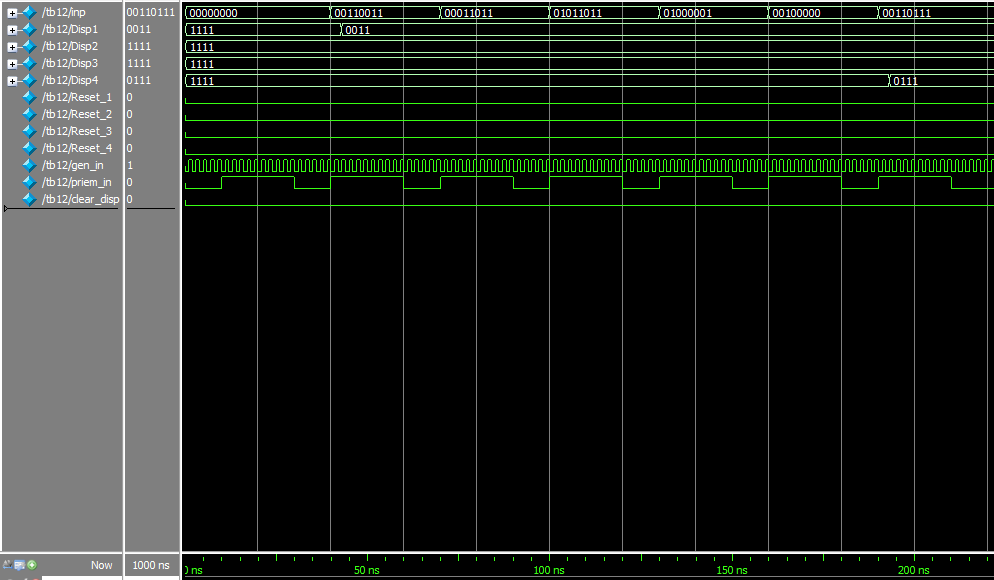


Рисунок 11 – Временная диаграмма модуля main

### Приложение Д

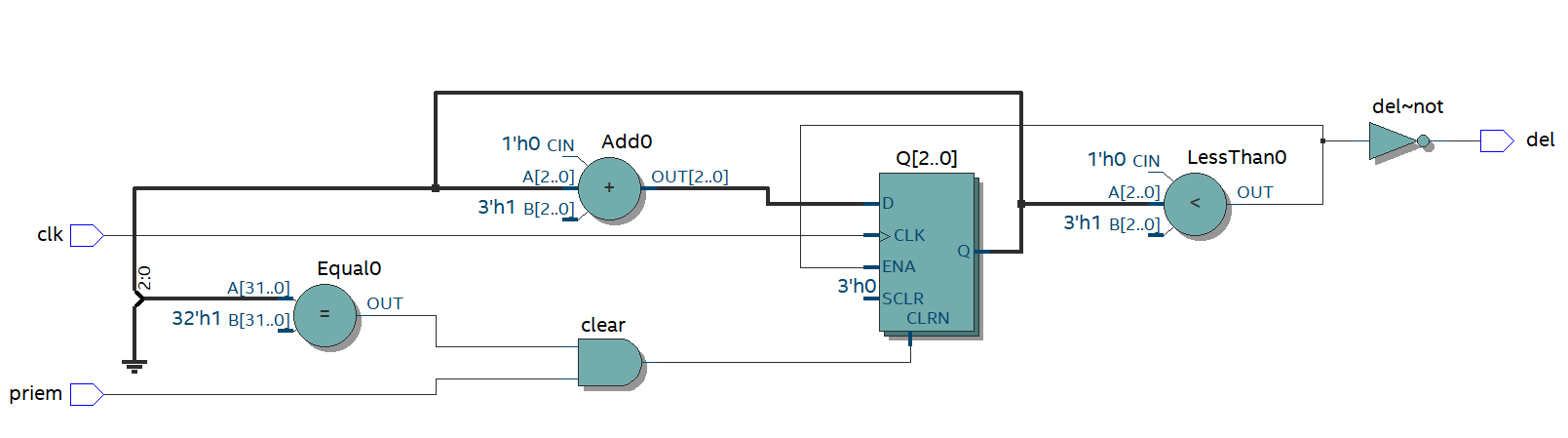


Рисунок 1 – RTL – представление модуля del\_par

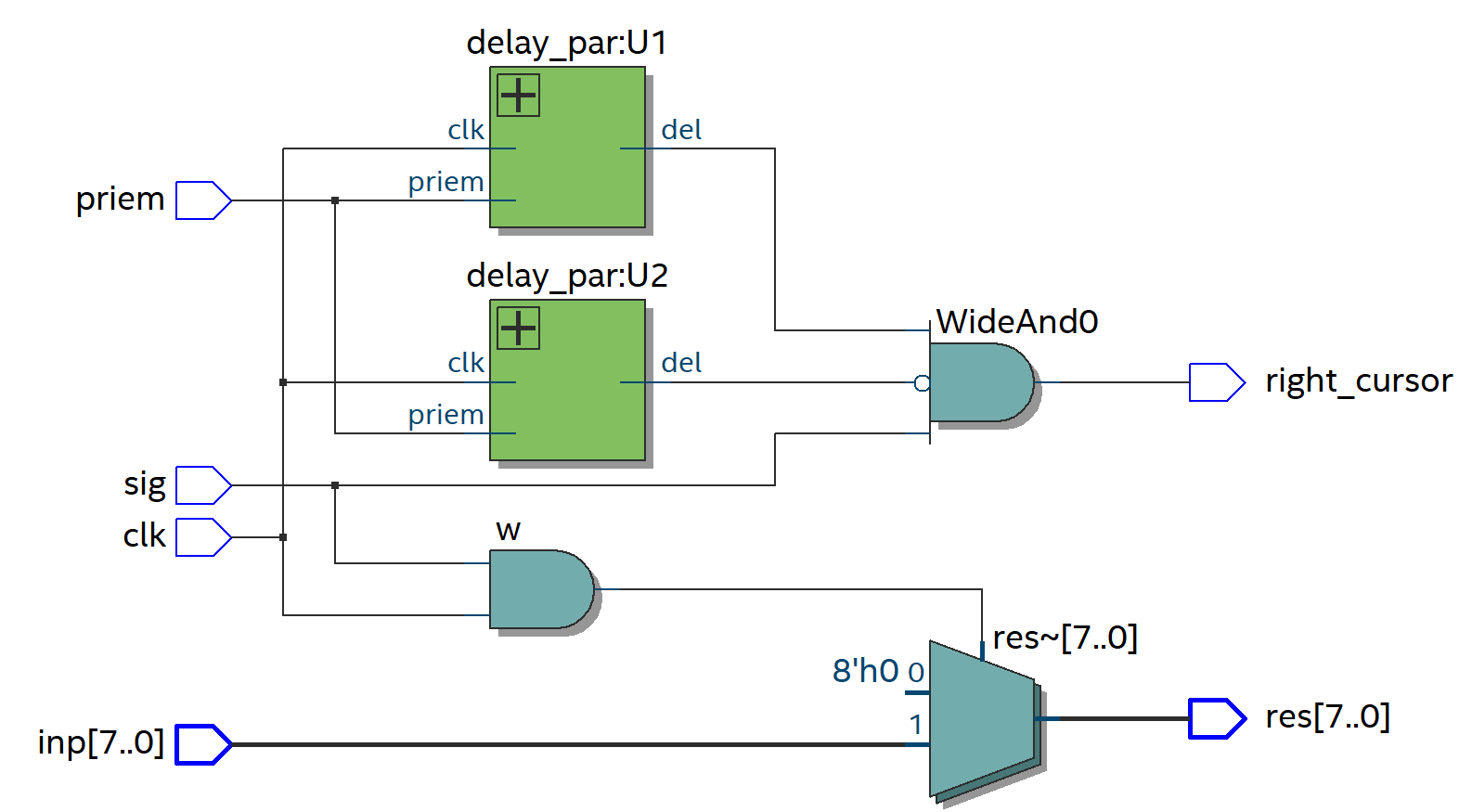


Рисунок 2 – RTL – представление модуля gate

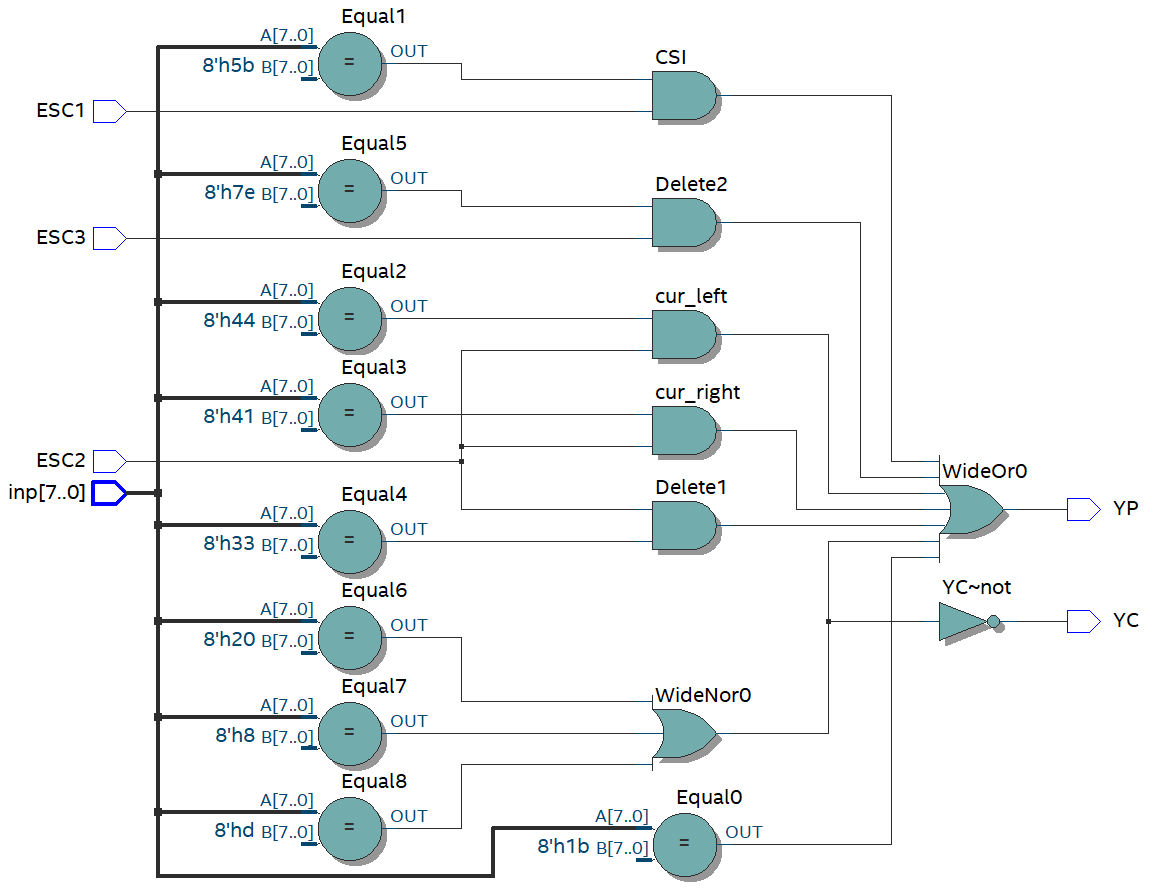


Рисунок 3 – RTL – представление модуля to\_save

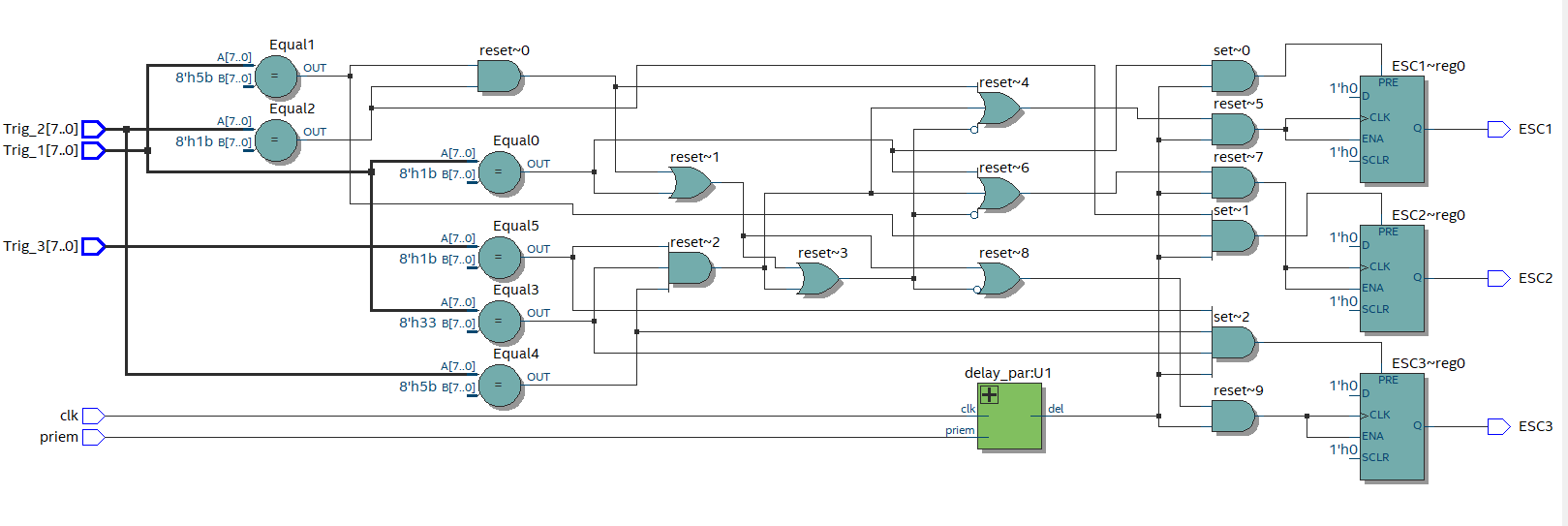


Рисунок 4 – RTL – представление модуля Cont\_seq

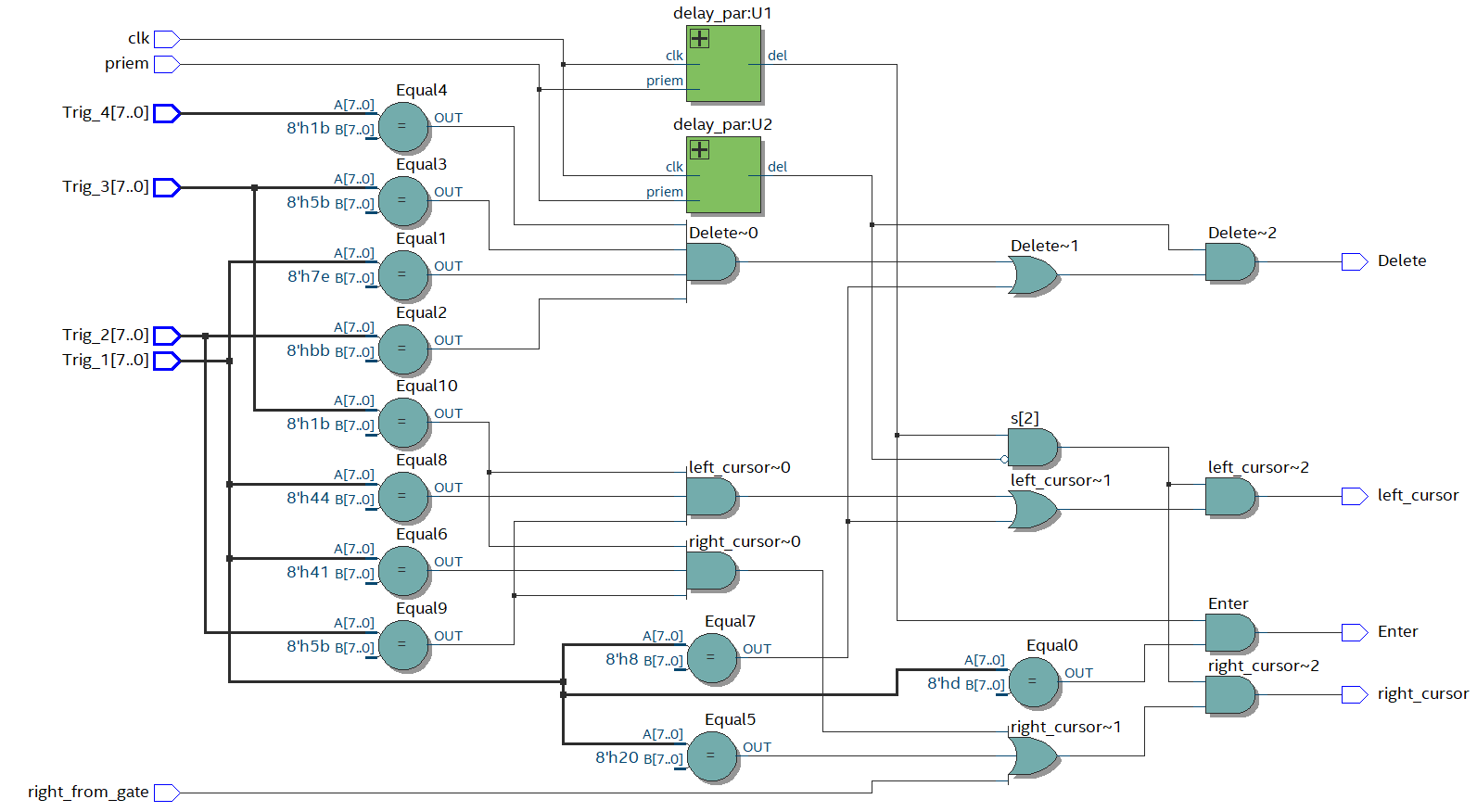


Рисунок 5 – RTL – представление модуля Comands

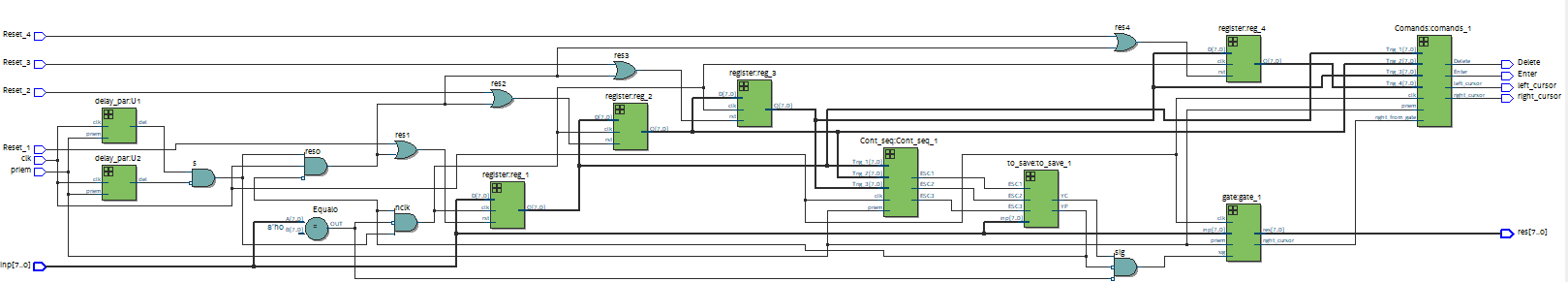


Рисунок 6 – RTL – представление модуля obrabotka

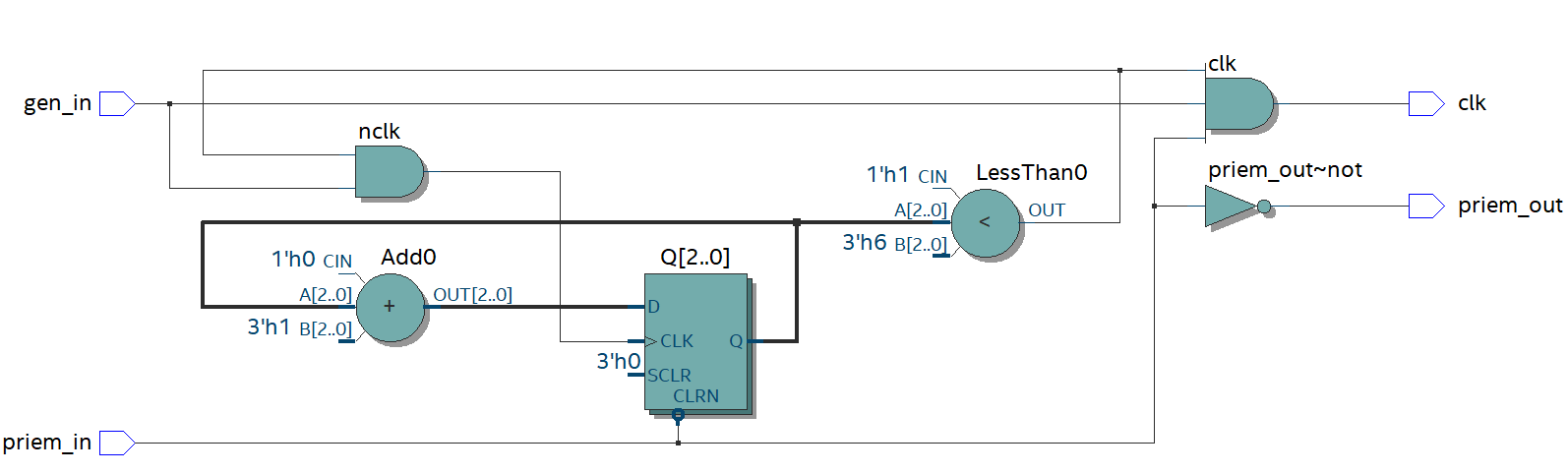


Рисунок 7 – RTL – представление модуля generator

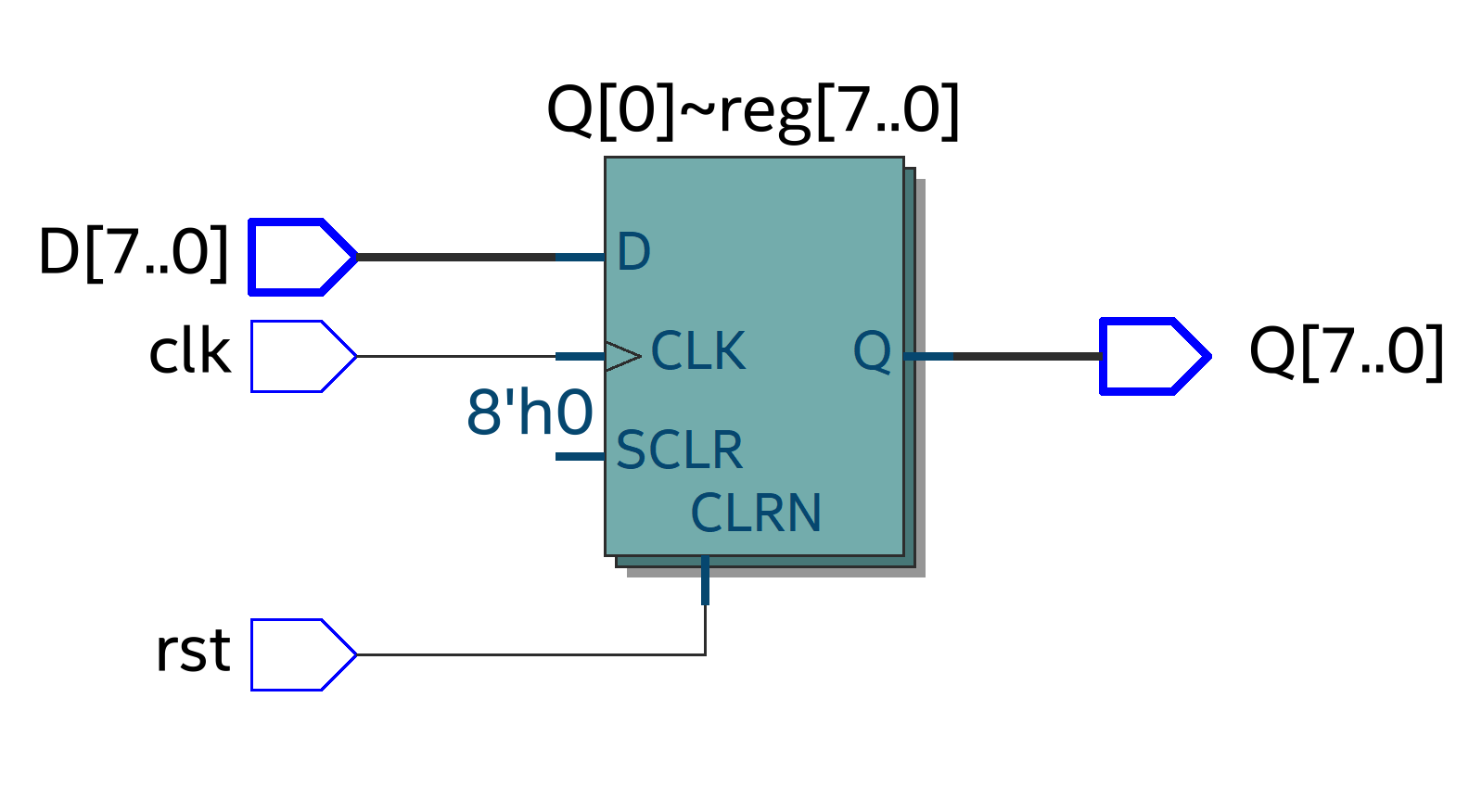


Рисунок 8 – RTL – представление модуля register

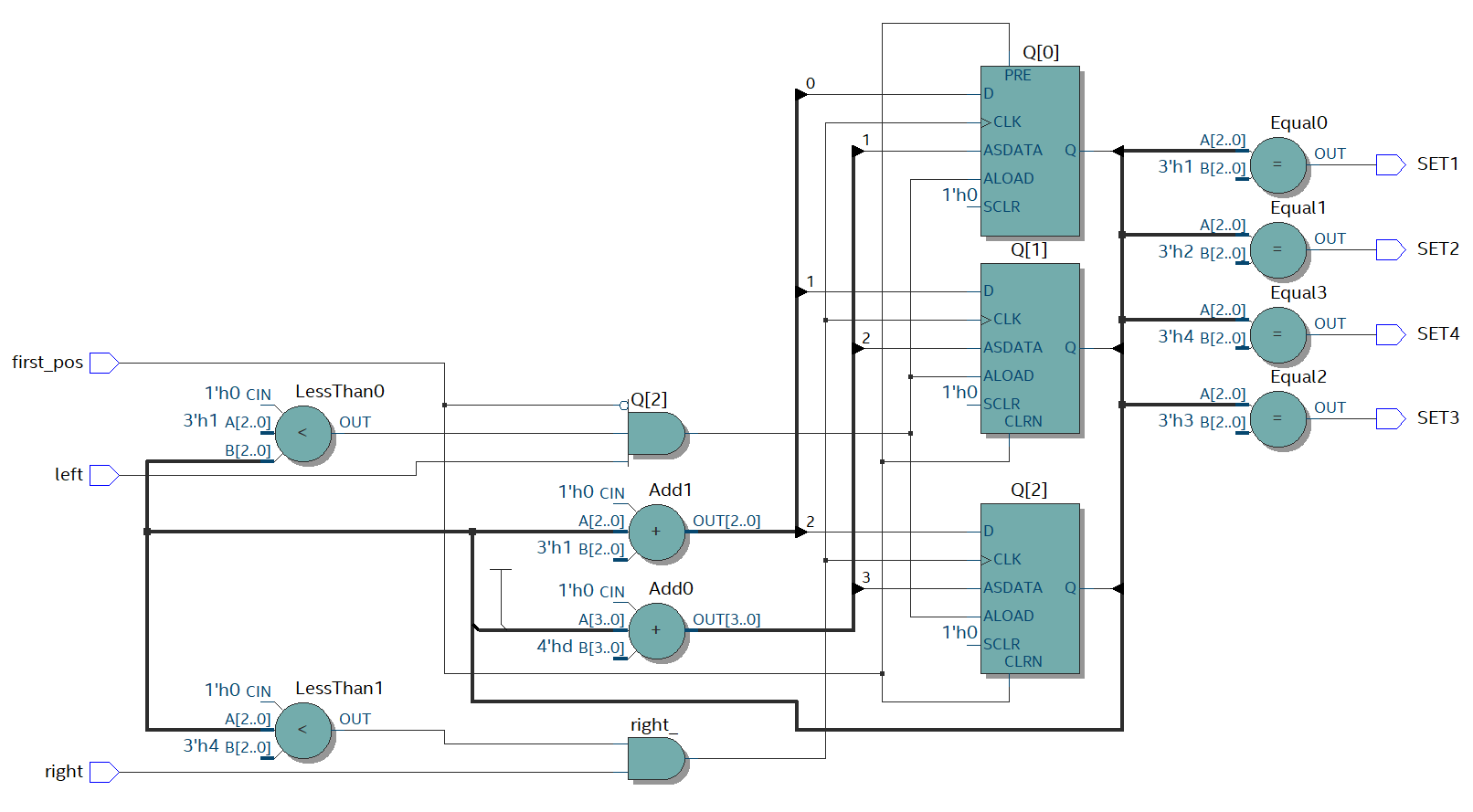


Рисунок 9 – RTL – представление модуля regist

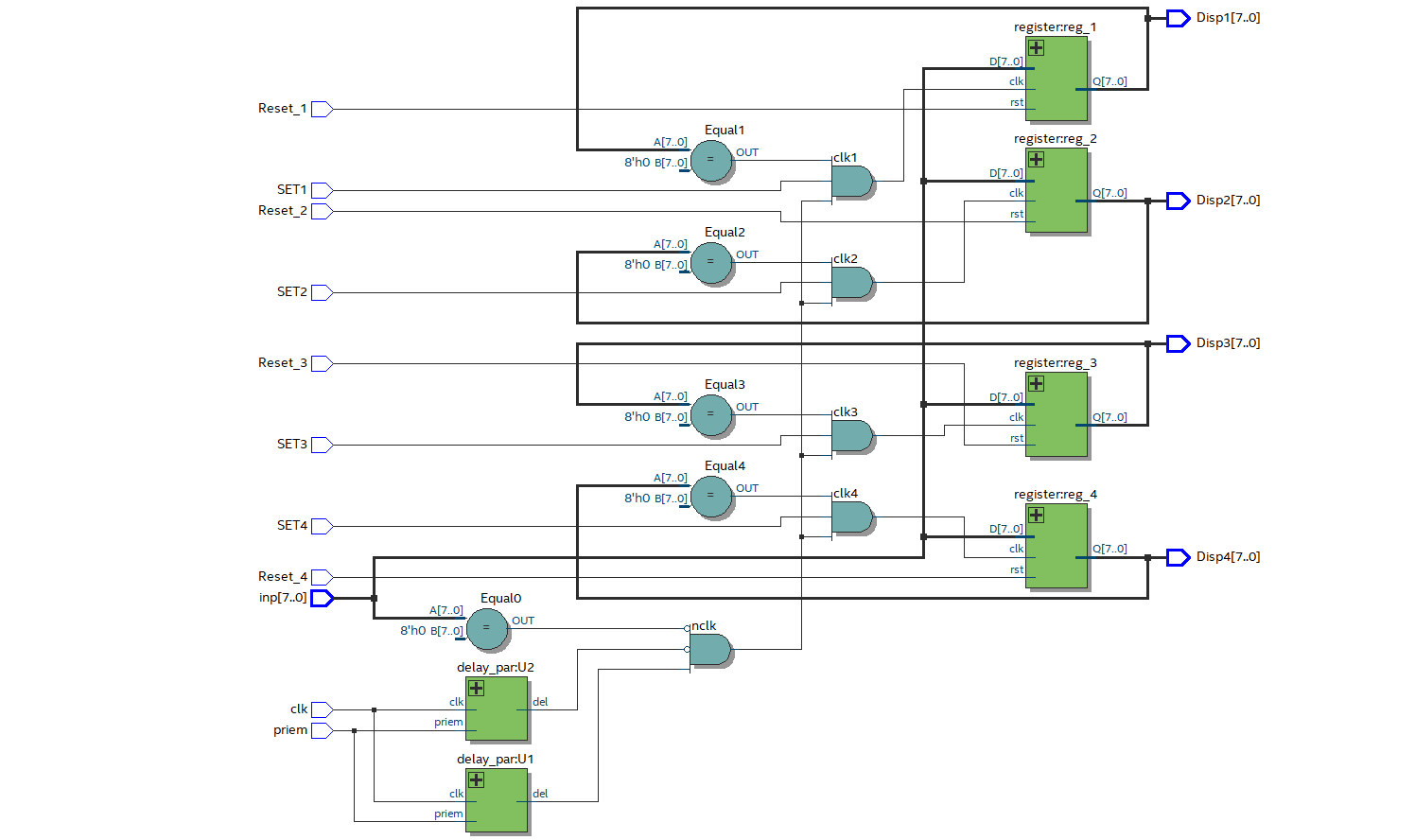


Рисунок 10 – RTL – представление модуля Display

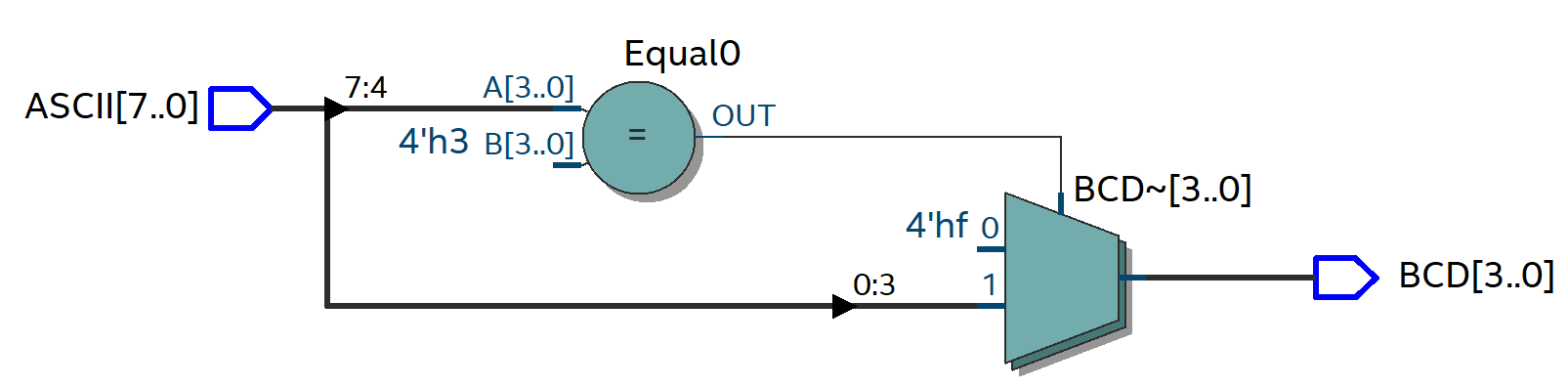


Рисунок 11 – RTL – представление модуля ascii\_to\_bcd

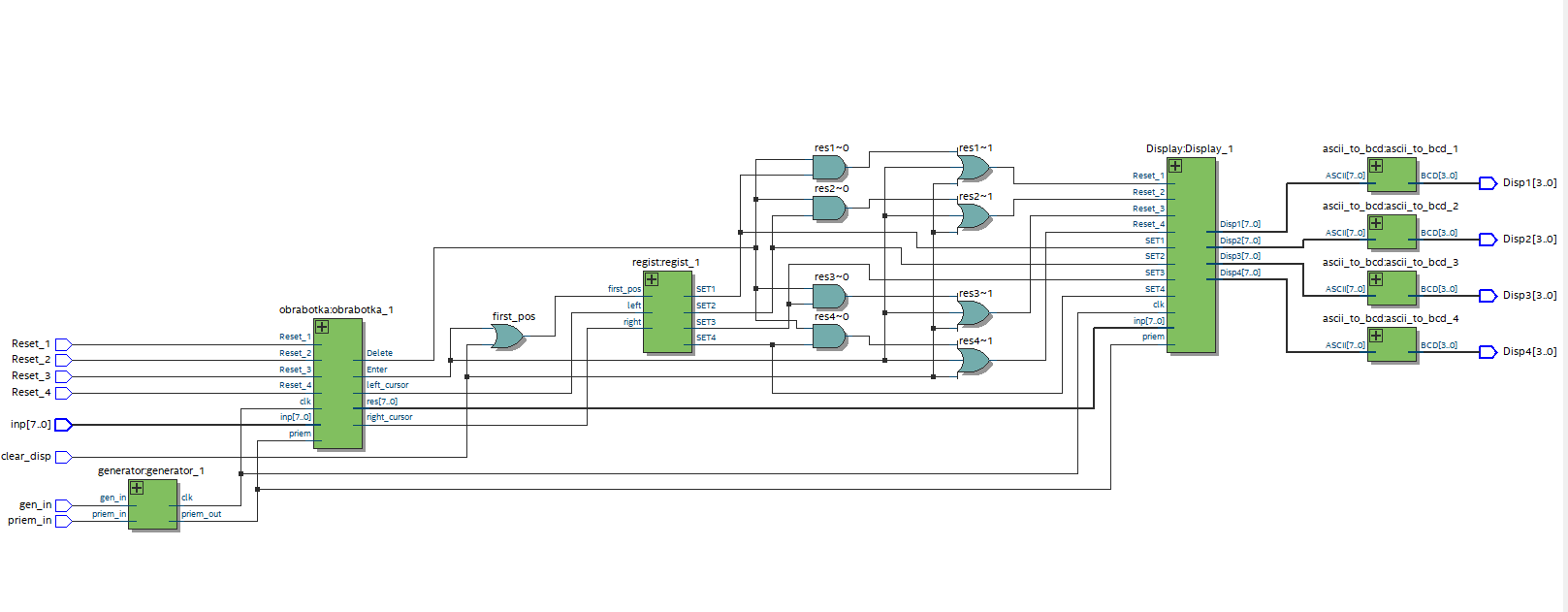


Рисунок 12 – RTL – представление модуля main

### Приложение Е

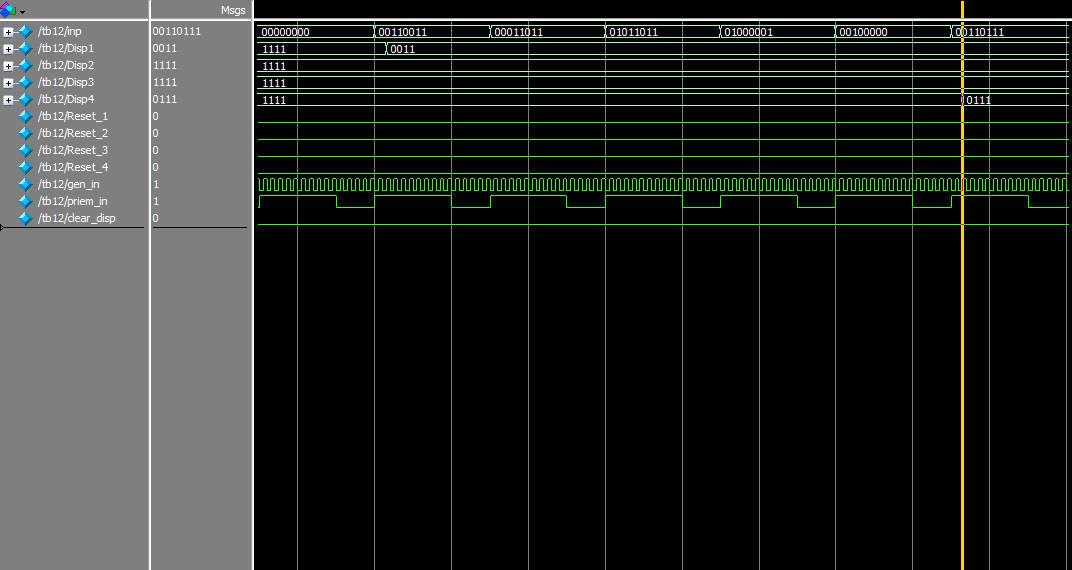


Рисунок 1 – RTL – моделирование

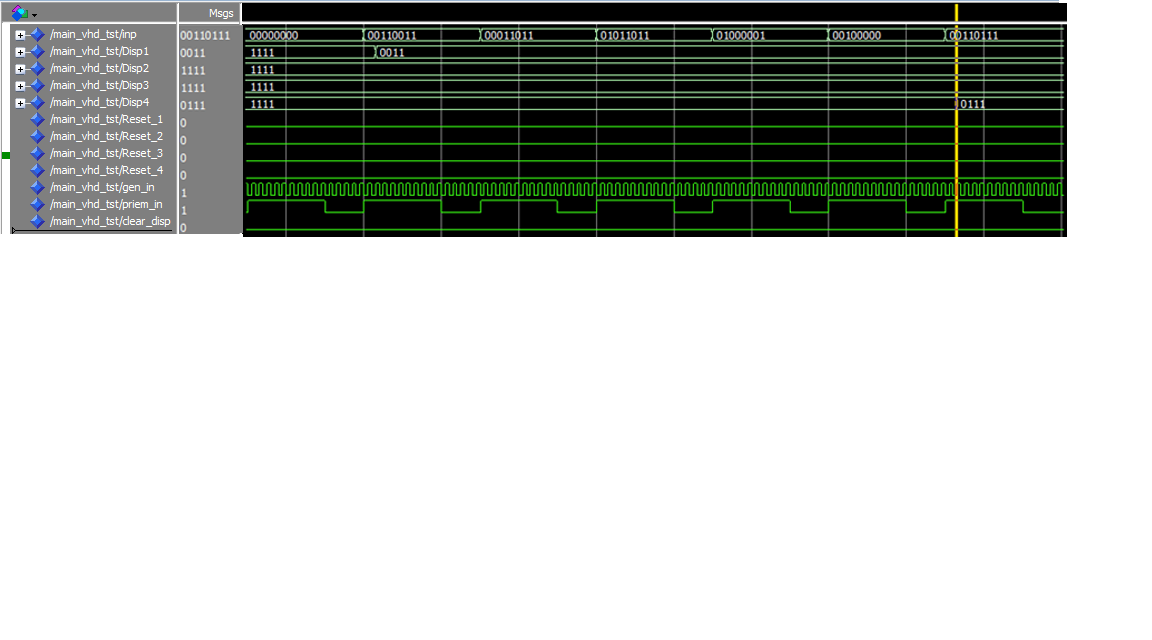


Рисунок 2 – Gate – level – моделирование