



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий
Кафедра Вычислительной техники

Отчет по лабораторной работе №3
по дисциплине

«Проектирование и разработка систем на базе ПЛИС»

Тема работы:
**«Проектирование цифрового узла анализатора
последовательности и его верификация средствами САПР Xilinx
ISE 14.x.»**

Студент группы: ИВБО-02-19

К. Ю. Денисов

Преподаватель: ассистент

А. С. Боронников

Работа выполнена «____» _____ 2021 г.

«Зачтено» «____» _____ 2021 г.

Москва 2021

Постановка задачи

Требуется разработать цифровой узел на основе отладочной платы Digilent Nexys 4, представляющий собой анализатор фиксированной последовательности логических сигналов. Узел должен обеспечивать индикацию ожидаемых и вводимых элементов последовательности посредством входящих в состав отладочной платы семисегментных индикаторов согласно данному заданию.

Узел должен быть реализован в виде синтезируемой модели на языке Verilog HDL.

Интерфейс верхнего уровня иерархии модели должен состоять из набора сигналов, представленного на рис. 1.

Автомат должен иметь интерфейс, представленный на рис ??.

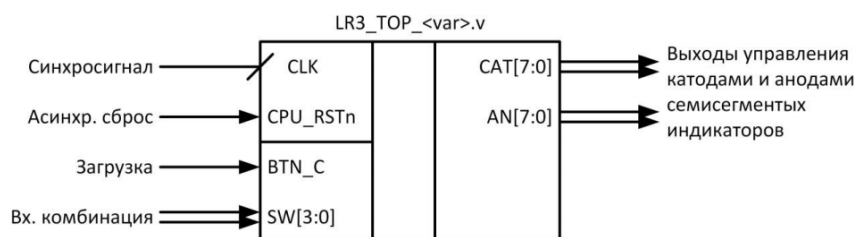


Рисунок 1 — Интерфейс модели цифрового узла

Разрабатываемое устройство является синхронным цифровым узлом, срабатывающим по восходящим фронтам синхросигнала CLK. Исключение составляет асинхронный вход сброса CPU_RSTn, принудительно устанавливающий все регистры узла в исходное состояние. Подача сигнала сброса на вход узла осуществляется посредством соответствующей кнопки (CPU_RSTn) отладочной платы.

Распознавание элементов последовательности осуществляется четверками, т.е. необходимо обеспечить последовательную загрузку в узел элементов Y с номерами 0-3, 4- 7, 8-B, C-F для успешного распознавания последовательности. При осуществлении ввода значения, не соответствующего текущему ожидаемому элементу последовательности, необходимо повторить ввод всей четверки элементов заново

Индикация работы узла посредством двух блоков семисегментных индикаторов для каждого варианта осуществляется аналогично примеру, представленному на рис. 3 и подчиняется следующим правилам:

1. Левый блок семисегментных индикаторов отображает ожидаемый (младший разряд) и введенные (три старших разряда) элементы последовательности в объеме распознаваемой четверки.
2. Правый блок семисегментных индикаторов отображает предысторию ввода комбинаций последовательности. Последняя введенная комбинация отображается в младшем разряде блока.
3. Не задействованные в текущий момент времени семисегментные индикаторы на обоих блоках должны находиться в выключенном состоянии.
4. Обновление отображаемых значений на обоих блоках семисегментных индикаторов рекомендуется выполнять с частотой от 60Гц до 200Гц.

Результат работы

Реализуем устройство, являющегося синхронным цифровым узлом. Его работу организуем по восходящему фронту синхросигнала, асинхронный сброс — по нисходящему фронту сигнала CPU_RSTn.

Моделирование цифрового устройства

Приведем таблицу состояний устройства (см. Таблицу 1). Текущее состояние цифрового устройства зависит от последнего введенного пользователем с помощью движковых переключателей элемента последовательности.

Таблица 1 — Состояния цифрового устройства

Состояние	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Значение	0	4	4	8	3	0	7	2	2	D	7	C	5	2	A	C	–
Разряд	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	–

Приведем граф состояний цифрового устройства (см. Рисунок 2). Переход в следующее состояние происходит только в случае ввода верного элемента последовательности. При неправильном вводе цифровое устройство переходит в ближайшее пройденное состояние, номер которого кратен 4. Номер состояния, в которое перейдет устройство в случае некорректного ввода можно вычислить по формуле

$$S_{i+1} = (S_i \div 4) \cdot 4$$

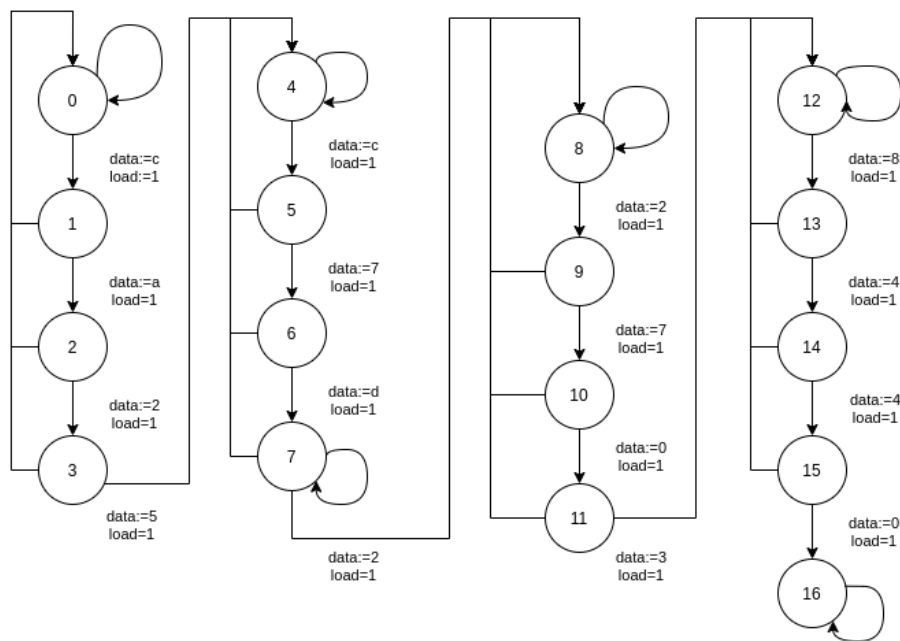


Рисунок 2 — Граф состояний

Приведем блок схему работы устройства (см. Рисунок 3). Значение X представляет собой номер элемента цифровой последовательности, ввод значения которого (Y) ожидается. Значения Y каждого элемента цифровой последовательности определяются вариантом задания.

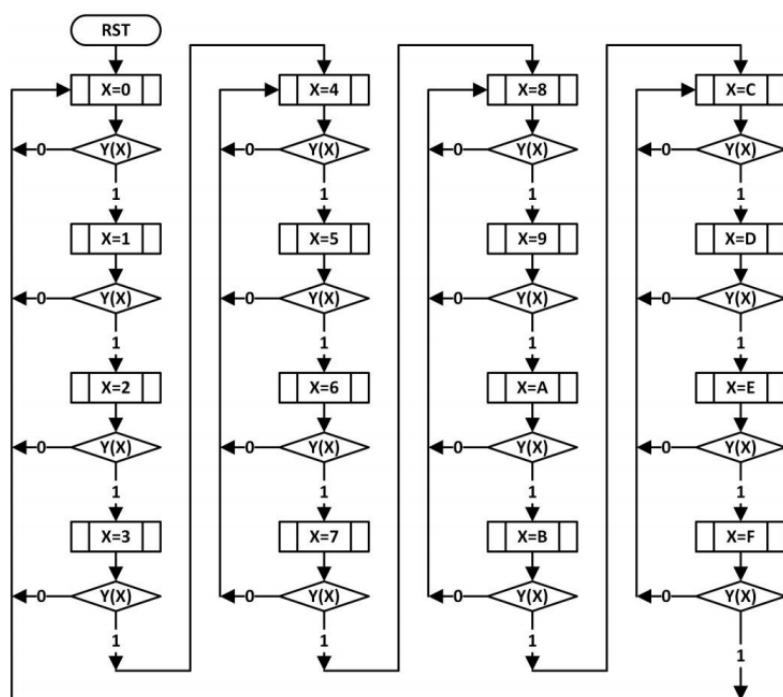


Рисунок 3 — Алгоритм распознавания последовательности

Приведем структурную схему синхронного цифрового узла (см. Рисунок 4).

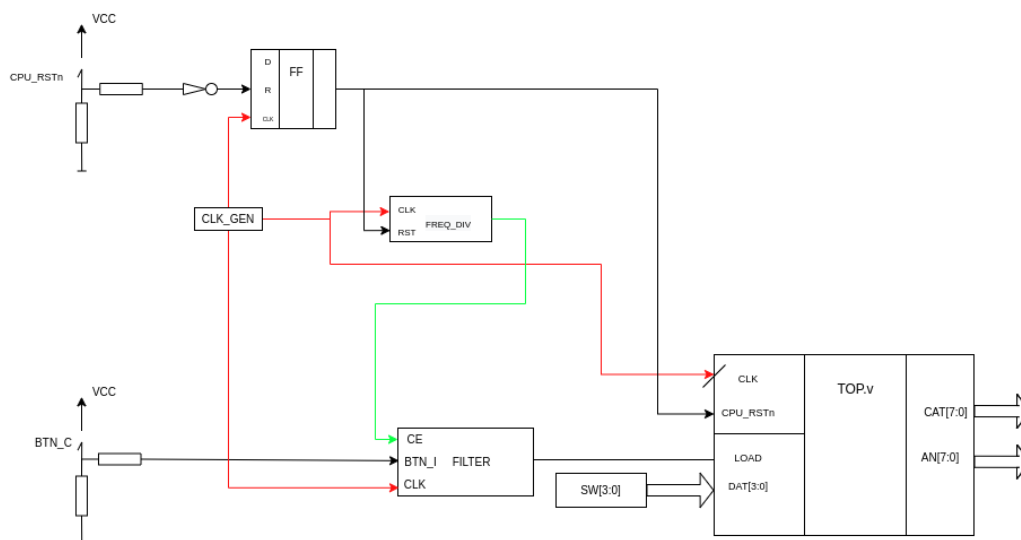


Рисунок 4 — Структурная схема узла

Описание принципа работы

Опишем главный алгоритм работы цифрового устройства — контроль ввода шестнадцатеричных чисел с возможностью повторного задания последовательности.

Данный алгоритм описан в файле `seqAuto.v`. Его содержание приведено в Листинге 1.

Сначала производим первоначальную инициализацию переменных, затем, если есть необходимость обновить изображение на дисплее (переменная `updateDisplay`), начинаем анализировать текущее состояние дисплея и пользовательский ввод.

Если мы находимся в состоянии, номер которого кратен четырем, то отображаем только левый разряд на левом дисплее. Выводим на это место значение соответствующее значению функции, а в остальные 3 разряда левого дисплея выводим нули.

Если же мы находимся в каком-то другом состоянии, то следует перезаписать все разряды на левом дисплее на один разряд влево, записать в освободившийся разряд записываем очередное значение вектор-функции.

В конце устанавливаем переменную `updateDisplay` в нуль.

Если пользователь подал сигнал на загрузку числа, произведем следующие действия:

Перезапишем все разряды на правом дисплее на один разряд влево, в освободившийся разряд запишем заданное пользователем число.

Осуществим проверку введенного числа — сравним его с соответствующим значением вектор–функции. Если пользователь произвел верный ввод, инструментируем переменную состояния автомата (переведем его в следующее состояние). В случае ввода неправильного числа возвращаемся в ближайшее пройденное состояние, номер которого кратен 4.

В конце устанавливаем переменную `updateDisplay` в единицу.

Листинг файлов

Приведем содержание файла `seqAuto.v`, реализующего главный алгоритм управления цифровым устройством, который был описан в предыдущей секции (см. Листинг 1).

Листинг 1 — Описание главного алгоритма

```
timescale 1ns / 1ps
2
module seqAuto(
4   input clk,
   input rst,
6   input load,
   input [3:0] data,
8   output reg [31:0] display,
   output reg [7:0] displayEnable
10  );
   reg [3:0] state;
12  reg updateDisplay;

14  wire[3:0] expected;

16  outFunc CL(
    .in(state),
18    .out(expected)
  );
20  always @(posedge clk, posedge rst)
    begin
22      if (rst)
        begin
24          state <= 0;
          display <= 0;
26          displayEnable <= 0;
          updateDisplay <= 1;
28        end
      else
30        begin
```

```

32     if (updateDisplay)
33         begin
34             // If we are in the first state in the four
35             if (state[1:0] == 2'b00)
36                 begin
37                     // display only the left digit on the left display
38                     displayEnable[7:4] <= 4'h1;
39                     // we output the corresponding value of the function to this
place
40                     display[19:16] <= expected;
41                     // we output zeros to the remaining 3 digits of the left
display
42                     display[31:20] <= 0;
43                 end
44             else
45                 begin
46                     // shift the digits on the left display by 1 digit
47                     displayEnable[7:4] <= {displayEnable[6:4], 1'b1};
48                     display[31:16] <= {display[27:16], expected};
49                 end
50                 updateDisplay <= 0;
51             end
52         // if the download signal is given
53         if (load)
54             begin
55                 // we shift the digits on the right display by 1 digit
56                 displayEnable[3:0] <= {displayEnable[2:0], 1'b1};
57                 display[15:0] <= {display[11:0], data};
58                 // if the correct digit is entered
59                 if (data == expected)
60                     begin
61                         // go to the next state
62                         state <= state + 1'b1;
63                     end
64                 else
65                     // going back to a multiple of the 4th
66                     begin
67                         state <= state & 4'b1100;
68                     end
69                 updateDisplay <= 1;
70             end
71         end
72     endmodule

```

Приведем содержание файла buttonFilter.v, описывающего работу фильтра дребезга, используемого в данной работе для фильтрации цифрового сигнала,

приходящего с кнопок ввода (см. Листинг 2).

Листинг 2 — Описание фильтра дребезга

```
`timescale 1ns / 1ps
2 module buttonFilter(
  input CLK,
4  input CE,
  input BTN_IN,
6  input RST,
  output BTN_OUT,
8  output reg BTN_CEO
);
10  parameter [3:0] CNTR_WIDTH = 4; // Internal Counter Width
  // Internal signals declaration:
12  reg [CNTR_WIDTH - 1:0] FLTR_CNT;
  reg BTN_D, BTN_S1, BTN_S2;
14  //-----// Main Counter:
  always @ (posedge CLK, posedge RST)
16  if(RST) FLTR_CNT <= {CNTR_WIDTH{1'b0}};
  else
18  if(!(BTN_S1 ^ BTN_S2)) // if BTN_S1 = BTN_S2
    FLTR_CNT <= {CNTR_WIDTH{1'b0}}; // Return to Zero
20  else if(CE) // else if Clock Enable
    FLTR_CNT <= FLTR_CNT + 1; // Increment
22  //-----// Input Synchronizer:
  always @ (posedge CLK, posedge RST)
24  if(RST)
    begin
26    BTN_D <= 1'b0;
    BTN_S1 <= 1'b0;
28  end
  else
30  begin
    BTN_D <= BTN_IN;
32    BTN_S1 <= BTN_D;
  end
34  //-----// Output Register:
  always @ (posedge CLK, posedge RST)
36  if(RST) BTN_S2 <= 1'b0;
  else if(&(FLTR_CNT) & CE) BTN_S2 <= BTN_S1;
38  //-----// Output Front Detector Clock Enable
    :
  always @ (posedge CLK, posedge RST)
40  if(RST) BTN_CEO <= 1'b0;
  else BTN_CEO <= &(FLTR_CNT) & CE & BTN_S1;
42
endmodule
```


Приведем содержание файла `freq_div.v`, описывающего работу делителя частоты, применяемого в данном проекте для снижения выходных характеристик частотного генератора для подбора оптимального режима работы с семисегментными индикаторами (см. Листинг 3).

Листинг 3 — Описание делителя частоты

```
'timescale 1ns / 1ps
2 module freq_div(input  rst,
      input  clk,
4
      output reg  co);
6   reg [16:0] counter;
   parameter divisor = 17'd10000;
8
   always @(posedge clk, posedge rst) begin
10    if (rst)
        begin
12        counter <= 0;
        co <= 0;
14    end
    else
16        if (counter >= (divisor - 17'b1))
            begin
18                counter <= 17'b0;
                co <= 1;
20            end
            else
22                begin
                    co <= 0;
24                    counter <= counter + 17'b1;
                end
26    end
endmodule
```

Приведем содержание файла `NexysDisplay.v`, описывающего работу с семисегментными индикаторами (см. Листинг 4).

Листинг 4 — Описание модуля работы с семисегментными индикаторами алгоритма

```
'timescale 1ns / 1ps
2
module NexysDisplay(
4   input CLK,
   input RST,
6   input [31:0] HEX_IN,
   input [7:0] DISP_EN,
8   output CA,
   output CB,
```

```

10 output CC,
   output CD,
12 output CE,
   output CF,
14 output CG,
   output DP,
16 output [7:0] AN
   );
18 // Internal signals declaration:
   //-----
20 reg [9:0] CLK_DIV_H, CLK_DIV_L;
   reg CEO_DIV_H, CEO_DIV_L;
22 reg [2:0] DIGIT_CNT;
   reg [3:0] I_CODE;
24 wire O_SEG_A, O_SEG_B, O_SEG_C, O_SEG_D, O_SEG_E, O_SEG_F, O_SEG_G;
   reg [7:0] ANODE_DC;
26 //-----
   // 1048576 Clock Divider:
28 always @ (posedge CLK, posedge RST)
   if(RST)
30     begin
        CLK_DIV_H <= 10'h000;
32         CLK_DIV_L <= 10'h000;
        CEO_DIV_H <= 1'b0;
34         CEO_DIV_L <= 1'b0;
     end
36 else
     begin
38         if(CLK_DIV_H == 10'h063) // 3FF - :1024 , 063 - :100
             begin
40                 CLK_DIV_H <= 10'h000;
                 CEO_DIV_H <= 1'b1;
42             end
         else
44             begin
                 CLK_DIV_H <= CLK_DIV_H + 1;
46                 CEO_DIV_H <= 1'b0;
             end
         end
48         if(CEO_DIV_H)
             begin
50                 if(&(CLK_DIV_L))
                     CLK_DIV_L <= 10'h000;
52                 else
                     CLK_DIV_L <= CLK_DIV_L + 1;
54             end
         if(&(CLK_DIV_L) & CEO_DIV_H)
56             CEO_DIV_L <= 1'b1;
     else

```

```

58     CEO_DIV_L <= 1'b0;
    end
60 //-----
    // Display Digit Counter:
62 always @ (posedge CLK, posedge RST)
    if(RST) DIGIT_CNT <= 3'd0;
64 else if(CEO_DIV_L) DIGIT_CNT <= DIGIT_CNT + 1;
    //-----
66 // Display Digit Multiplexer:
    always @ (DIGIT_CNT, HEX_IN, DISP_EN)
68 case(DIGIT_CNT)
    3'd0:
70 begin
        I_CODE <= HEX_IN[3:0];
72 ANODE_DC <= DISP_EN[0] ? 8'd1 : 0;
        end
74 3'd1:
        begin
76 I_CODE <= HEX_IN[7:4];
        ANODE_DC <= DISP_EN[1] ? 8'd2 : 0;
78 end
        3'd2:
80 begin
        I_CODE <= HEX_IN[11:8];
82 ANODE_DC <= DISP_EN[2] ? 8'd4 : 0;
        end
84 3'd3:
        begin
86 I_CODE <= HEX_IN[15:12];
        ANODE_DC <= DISP_EN[3] ? 8'd8 : 0;
88 end
        3'd4:
90 begin
        I_CODE <= HEX_IN[19:16];
92 ANODE_DC <= DISP_EN[4] ? 8'd16 : 0;
        end
94 3'd5:
        begin
96 I_CODE <= HEX_IN[23:20];
        ANODE_DC <= DISP_EN[5] ? 8'd32 : 0;
98 end
        3'd6:
100 begin
        I_CODE <= HEX_IN[27:24];
102 ANODE_DC <= DISP_EN[6] ? 8'd64 : 0;
        end
104 default:
        begin

```

```

106 I_CODE <= HEX_IN[31:28];
    ANODE_DC <= DISP_EN[7] ? 8'd128 : 0;
108 end
    endcase
110 //-----
    // 7-Segment Decoder:
112 SevenSegDec DISP_DEC (
    .I_CODE(I_CODE),
114 .O_SEG_A(O_SEG_A),
    .O_SEG_B(O_SEG_B),
116 .O_SEG_C(O_SEG_C),
    .O_SEG_D(O_SEG_D),
118 .O_SEG_E(O_SEG_E),
    .O_SEG_F(O_SEG_F),
120 .O_SEG_G(O_SEG_G));
    //-----
122 assign AN = ~ANODE_DC | {8{RST}};
    assign CA = ~O_SEG_A;
124 assign CB = ~O_SEG_B;
    assign CC = ~O_SEG_C;
126 assign CD = ~O_SEG_D;
    assign CE = ~O_SEG_E;
128 assign CF = ~O_SEG_F;
    assign CG = ~O_SEG_G;
130 assign DP = 0;
    //-----
132 endmodule

```

Приведем содержание файла outFunc.v, описывающего представление бинарной вектор–функции, значения которой сравниваются с пользовательским вводом (см. Листинг 5).

Листинг 5 — Описание вектор–функции

```

'timescale 1ns / 1ps
2 module outFunc(
    input [3:0] in,
4    output reg [3:0] out
    );
6    always @(in)
        case (in)
8        4'h0: out<=4'hc;
        4'h1: out<=4'ha;
10       4'h2: out<=4'h2;
        4'h3: out<=4'h5;
12       4'h4: out<=4'hc;
        4'h5: out<=4'h7;
14       4'h6: out<=4'hd;

```

```

16      4'h7: out<=4'h2;
      4'h8: out<=4'h2;
      4'h9: out<=4'h7;
18      4'ha: out<=4'h0;
      4'hb: out<=4'h3;
20      4'hc: out<=4'h8;
      4'hd: out<=4'h4;
22      4'he: out<=4'h4;
      4'hf: out<=4'h0;
24      default: out<=4'h0;
      endcase
26 endmodule

```

Приведем содержание файла SevenSegDec.v, описывающего работу сигнального дешифратора для работы с семисегментными индикаторами (см. Листинг 6).

Листинг 6 — Описание сигнального дешифратора

```

'timescale 1ns / 1ps
2
module SevenSegDec(
4  input [3:0] I_CODE, // Input HEX-Digit
  output reg O_SEG_A, // Segment-A Active High
6  output reg O_SEG_B, // Segment-B Active High
  output reg O_SEG_C, // Segment-C Active High
8  output reg O_SEG_D, // Segment-D Active High
  output reg O_SEG_E, // Segment-E Active High
10 output reg O_SEG_F, // Segment-F Active High
  output reg O_SEG_G // Segment-G Active High
12 );
  //-----
14 // Decoder Comb. Logic:
  always @ (I_CODE)
16 case(I_CODE)
    4'h0:
18 begin
      O_SEG_A <= 1'b1;
20 O_SEG_B <= 1'b1;
      O_SEG_C <= 1'b1;
22 O_SEG_D <= 1'b1;
      O_SEG_E <= 1'b1;
24 O_SEG_F <= 1'b1;
      O_SEG_G <= 1'b0;
26 end
    4'h1:
28 begin
      O_SEG_A <= 1'b0;

```

```

30 O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b1;
32 O_SEG_D <= 1'b0;
   O_SEG_E <= 1'b0;
34 O_SEG_F <= 1'b0;
   O_SEG_G <= 1'b0;
36 end
   4'h2:
38 begin
   O_SEG_A <= 1'b1;
40 O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b0;
42 O_SEG_D <= 1'b1;
   O_SEG_E <= 1'b1;
44 O_SEG_F <= 1'b0;
   O_SEG_G <= 1'b1;
46 end
   4'h3:
48 begin
   O_SEG_A <= 1'b1;
50 O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b1;
52 O_SEG_D <= 1'b1;
   O_SEG_E <= 1'b0;
54 O_SEG_F <= 1'b0;
   O_SEG_G <= 1'b1;
56 end
   4'h4:
58 begin
   O_SEG_A <= 1'b0;
60 O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b1;
62 O_SEG_D <= 1'b0;
   O_SEG_E <= 1'b0;
64 O_SEG_F <= 1'b1;
   O_SEG_G <= 1'b1;
66 end
   4'h5:
68 begin
   O_SEG_A <= 1'b1;
70 O_SEG_B <= 1'b0;
   O_SEG_C <= 1'b1;
72 O_SEG_D <= 1'b1;
   O_SEG_E <= 1'b0;
74 O_SEG_F <= 1'b1;
   O_SEG_G <= 1'b1;
76 end
   4'h6:

```

```

78 begin
   O_SEG_A <= 1'b1;
80   O_SEG_B <= 1'b0;
   O_SEG_C <= 1'b1;
82   O_SEG_D <= 1'b1;
   O_SEG_E <= 1'b1;
84   O_SEG_F <= 1'b1;
   O_SEG_G <= 1'b1;
86 end
   4'h7:
88 begin
   O_SEG_A <= 1'b1;
90   O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b1;
92   O_SEG_D <= 1'b0;
   O_SEG_E <= 1'b0;
94   O_SEG_F <= 1'b0;
   O_SEG_G <= 1'b0;
96 end
   4'h8:
98 begin
   O_SEG_A <= 1'b1;
100  O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b1;
102  O_SEG_D <= 1'b1;
   O_SEG_E <= 1'b1;
104  O_SEG_F <= 1'b1;
   O_SEG_G <= 1'b1;
106 end
   4'h9:
108 begin
   O_SEG_A <= 1'b1;
110  O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b1;
112  O_SEG_D <= 1'b1;
   O_SEG_E <= 1'b0;
114  O_SEG_F <= 1'b1;
   O_SEG_G <= 1'b1;
116 end
   4'hA:
118 begin
   O_SEG_A <= 1'b1;
120  O_SEG_B <= 1'b1;
   O_SEG_C <= 1'b1;
122  O_SEG_D <= 1'b0;
   O_SEG_E <= 1'b1;
124  O_SEG_F <= 1'b1;
   O_SEG_G <= 1'b1;

```

```

126 end
    4'hB:
128 begin
    O_SEG_A <= 1'b0;
130 O_SEG_B <= 1'b0;
    O_SEG_C <= 1'b1;
132 O_SEG_D <= 1'b1;
    O_SEG_E <= 1'b1;
134 O_SEG_F <= 1'b1;
    O_SEG_G <= 1'b1;
136 end
    4'hC:
138 begin
    O_SEG_A <= 1'b1;
140 O_SEG_B <= 1'b0;
    O_SEG_C <= 1'b0;
142 O_SEG_D <= 1'b1;
    O_SEG_E <= 1'b1;
144 O_SEG_F <= 1'b1;
    O_SEG_G <= 1'b0;
146 end
    4'hD:
148 begin
    O_SEG_A <= 1'b0;
150 O_SEG_B <= 1'b1;
    O_SEG_C <= 1'b1;
152 O_SEG_D <= 1'b1;
    O_SEG_E <= 1'b1;
154 O_SEG_F <= 1'b0;
    O_SEG_G <= 1'b1;
156 end
    4'hE:
158 begin
    O_SEG_A <= 1'b1;
160 O_SEG_B <= 1'b0;
    O_SEG_C <= 1'b0;
162 O_SEG_D <= 1'b1;
    O_SEG_E <= 1'b1;
164 O_SEG_F <= 1'b1;
    O_SEG_G <= 1'b1;
166 end
    default:
168 begin
    O_SEG_A <= 1'b1;
170 O_SEG_B <= 1'b0;
    O_SEG_C <= 1'b0;
172 O_SEG_D <= 1'b0;
    O_SEG_E <= 1'b1;

```



```

174 O_SEG_F <= 1'b1;
    O_SEG_G <= 1'b1;
176 end
    endcase
178 //-----
endmodule

```

Приведем содержание файла top.v, описывающего работу модуля верхнего уровня, объединяющего все файлы и организующего работу с устройством (см. Листинг 7).

Листинг 7 — Описание модуля верхнего уровня

```

`timescale 1ns / 1ps
2 module top(
    input CLK,
    4    input CPU_RSTn,
    input BTN_C,
    6    input  [3:0] SW,
    output [7:0] CAT,
    8    output [7:0] AN
    );

10    reg rst;

12    always @(posedge CLK) begin
14        rst <= ~CPU_RSTn;
    end

16    wire [31:0] hex;
    wire [7:0] disp_en;

18    NexysDisplay Disp(
20        .CLK(CLK),
        .RST(rst),
22        .HEX_IN(hex),
        .DISP_EN(disp_en),
24        .CA(CAT[0]),
        .CB(CAT[1]),
26        .CC(CAT[2]),
        .CD(CAT[3]),
28        .CE(CAT[4]),
        .CF(CAT[5]),
30        .CG(CAT[6]),
        .DP(CAT[7]),
32        .AN(AN)
    );

34    wire divClk;
    freq_div DIV(

```

```

36     .rst(rst),
      .clk(CLK),
38     .co(divClk)
      );
40     wire loadDebounced;
      wire loadDebounced1;
42     M_BTN_FILTER_V10 Filter(
      .CLK(CLK),
44     .CE(divClk),
      .BTN_IN(BTN_C),
46     .RST(rst),
      .BTN_OUT(),
48     .BTN_CEO(loadDebounced)
      );
50     seqAuto Auto(
      .clk(CLK),
52     .rst(rst),
      .load(loadDebounced),
54     .data(SW),
      .display(hex),
56     .displayEnable(disp_en)
      );
58 endmodule

```

Эмуляция работы цифрового устройства

Произведем тестирование работы спроектированного цифрового устройства средствами САПР Xilinx ISE 14.

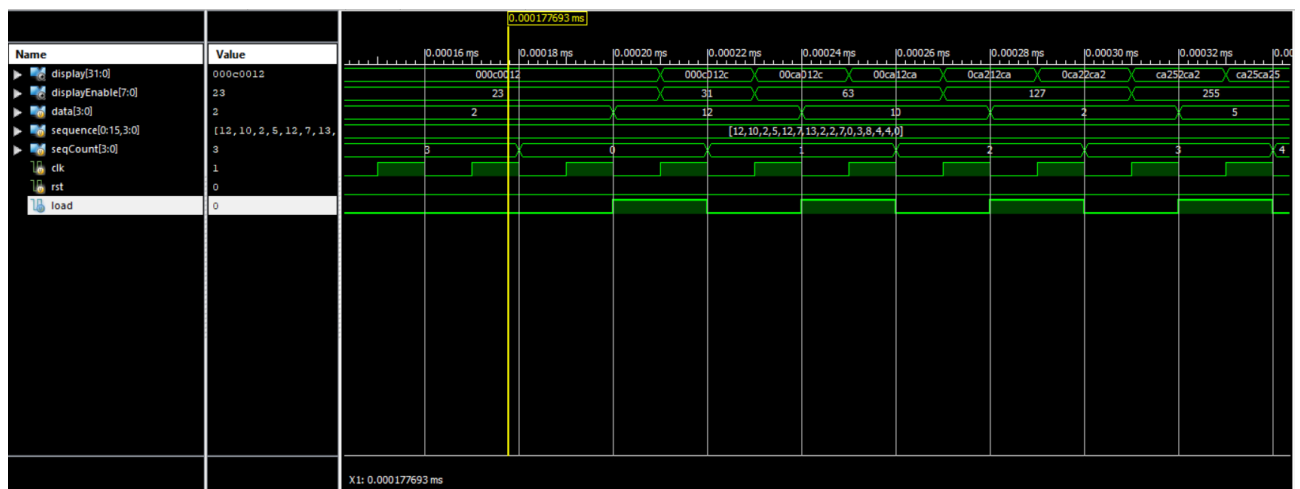


Рисунок 5 — Эмуляция работы. Часть 1

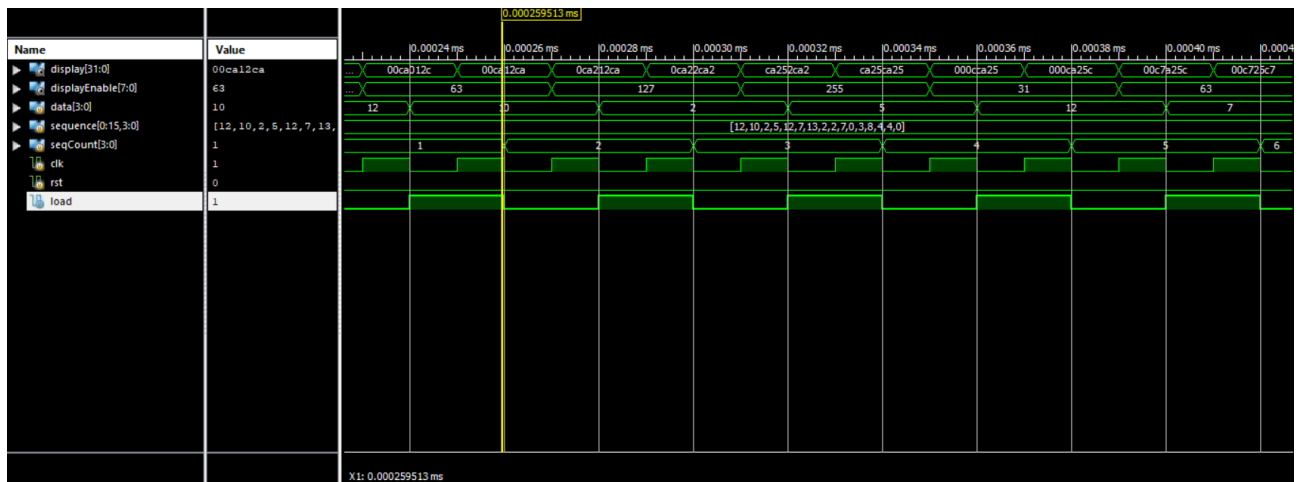


Рисунок 6 — Эмуляция работы. Часть 2

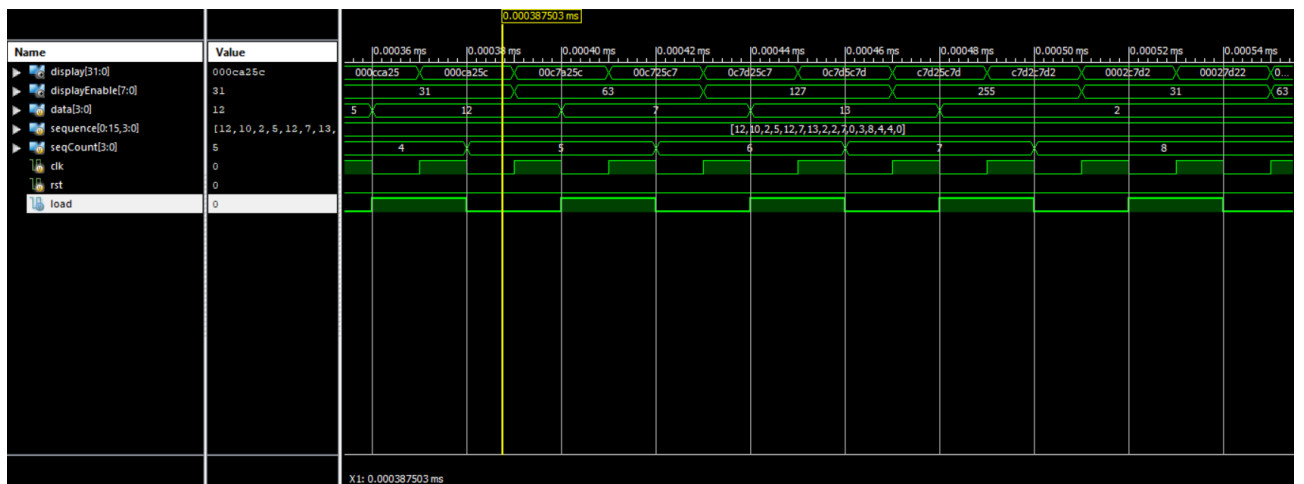


Рисунок 7 — Эмуляция работы. Часть 3

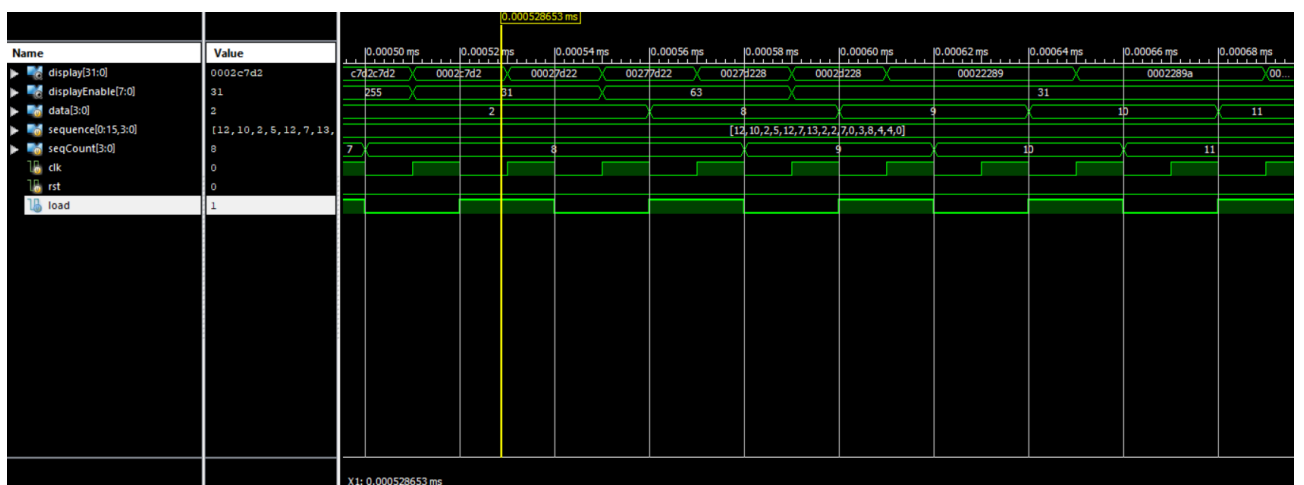


Рисунок 8 — Эмуляция работы. Часть 4

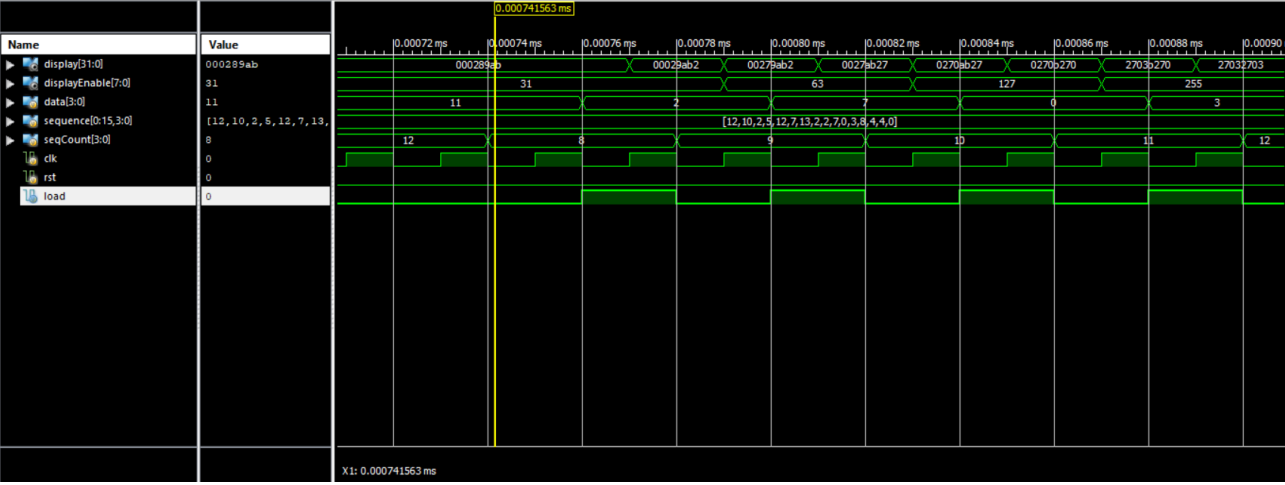


Рисунок 9 — Эмуляция работы. Часть 5

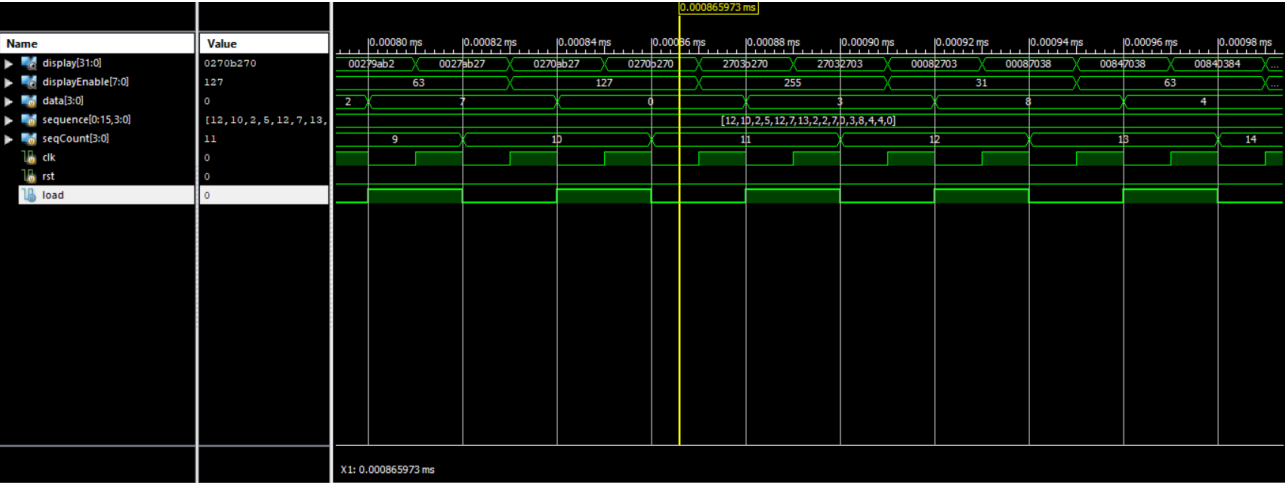


Рисунок 10 — Эмуляция работы. Часть 6

Вывод: В ходе данной практической работы нами были получены общие навыки работы с программным обеспечением Xilinx ISE Design Suite, изучены основы языка Verilog.

С помощью полученных знаний был спроектирован и разработан цифровой узел на основе отладочной платы Digilent Nexys 4, представляющий собой анализатор фиксированной последовательности логических сигналов.