



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 №2
по дисциплине
«Проектирование и разработка систем на базе ПЛИС»

Тема работы:
«Проектирование синтезируемой модели конечного автомата и её
верификация средствами САПР Xilinx ISE 14.x.»

Выполнил: студент группы ИВБО-02-19

К. Ю. Денисов

Принял: ассистент

А. С. Боронников

Работа выполнена «_____» _____ 202__

«Зачтено» «_____» _____ 202__

Москва 2021

Содержание

1	Ход работы	3
1.1	Постановка задачи	3
1.2	Индивидуальный вариант 149	4
1.3	Структурная схема автомат	4
1.4	Кодировка состояний автомата в двоичной и шестнадцатиричной системах	4
1.5	Граф состояний	6
1.6	Создание проекта САПР Xilinx ISE	6
1.7	Тестирование и отладка средствами симулятора iSim	13
2	Вывод	15

1 Ход работы

1.1 Постановка задачи

Требуется описать конечный автомат, представляющий собой генератор фиксированной последовательности логических сигналов, в виде синтезируемой модели на языке Verilog HDL.

Автомат должен иметь интерфейс, представленный на рис 1.

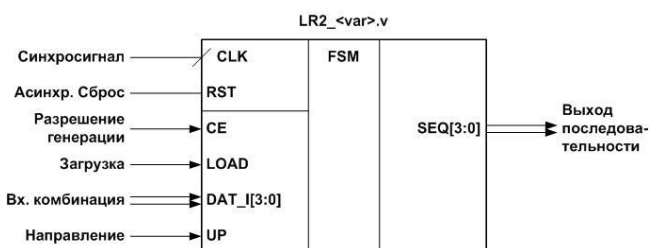


Рисунок 1 — Интерфейс цифрового автомата

Автомат является синхронным цифровым узлом, срабатывающим по восходящим фронтам синхросигнала *CLK*. Исключение составляет асинхронный вход сброса *RST*, принудительно устанавливающий регистр автомата в исходное состояние (определяется вариантом). Автомат должен реагировать на входные воздействия согласно таблице 1.

RST	CLK	LOAD	CE	UP	Действие
1	X	X	X	X	Асинхронный сброс $SEQ \leq Func(4'h0)$
0	posedge	1	X	X	Загрузка $SEQ \leq Func(DAT_I)$
0	posedge	0	1	0	Обратная генерация $SEQ \leq Func(i-1)$
0	posedge	0	1	1	Прямая генерация $SEQ \leq Func(i+1)$
0	posedge	0		X	Хранение $SEQ \leq SEQ$

Таблица 1 — Таблица функционирования автомата

Последовательность генерируемых сигналов определяется функцией $Func(i)$, где i — 4-разрядный двоичный индекс, представляющий собой номер элемента последовательности.

Инкремент индекса соответствует прямой генерации последовательности. Декремент индекса соответствует обратной генерации последовательности.

Последовательность для каждого варианта выполнения работы определяется из таблицы вариантов следующим образом: индекс i задан

входными комбинациями от F до 0 в верхней строке таблицы, а выходные комбинации $Func(i)$, формируемые на выходах $SEQ[3:0]$, заданы строкой таблицы, соответствующей выбранному варианту. Допускается использовать различные варианты кодировки состояний автомата. Автомат может иметь организацию согласно абстрактным моделям Мили или Мура.

1.2 Индивидуальный вариант 149

Требуется описать конечный автомат, представляющий собой генератор фиксированной последовательности логических сигналов, в виде синтезируемой модели на языке Verilog HDL согласно данной таблице истинности и вектор-функции (см. таблицу 2).

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0	4	4	8	3	0	7	2	2	D	7	C	5	2	A	C

Таблица 2 — Вектор-функция

1.3 Структурная схема автомат

Построим структурную схему цифрового устройства. Используем делитель частоты для снижения частоты тактового генератора, фильтр дребезга для использования кнопок в качестве устройств ввода. См. рис. 2.

1.4 Кодировка состояний автомата в двоичной и шестнадцатиричной системах

Опишем модуль `behaviour.v`, указав в нем состояния автомата, приведенные в шестнадцатеричной системе.

`/home/denilai/Documents/repos/latex/scripts/behaviour.v`

```

1 `timescale 1ns / 1ps
2
3 module behaviour(
4     input[3:0] X,
5     output reg [3:0] Y
6 );

```

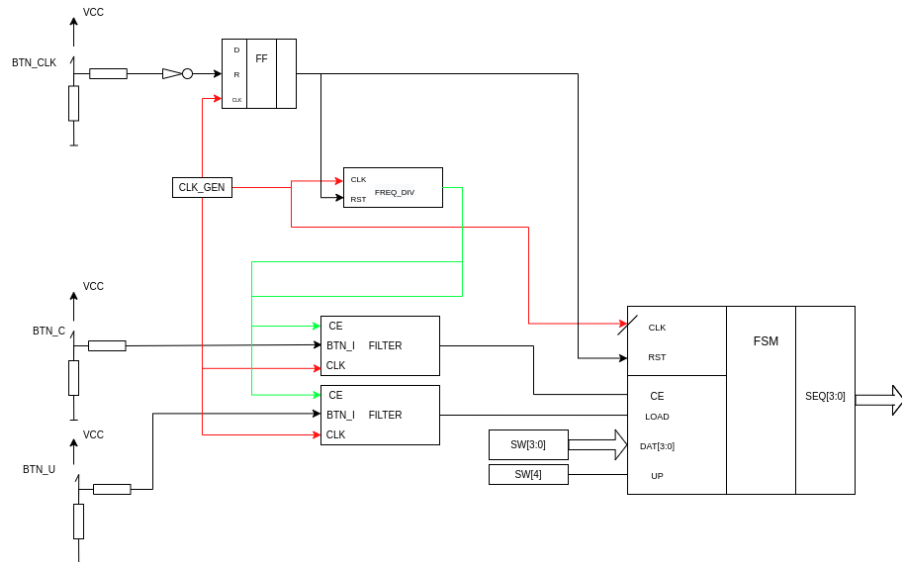


Рисунок 2 — Структурная схема устройства

```

always@(X)
8   case(X)
    4'h0: Y<=4'hc;
    10  4'h1: Y<=4'ha;
    4'h2: Y<=4'h2;
    12  4'h3: Y<=4'h5;
    4'h4: Y<=4'hc;
    14  4'h5: Y<=4'h7;
    4'h6: Y<=4'hd;
    16  4'h7: Y<=4'h2;
    4'h8: Y<=4'h2;
    18  4'h9: Y<=4'h7;
    4'ha: Y<=4'h0;
    20  4'hb: Y<=4'h3;
    4'hc: Y<=4'h8;
    22  4'hd: Y<=4'h4;
    4'he: Y<=4'h4;
    24  4'hf: Y<=4'h0;
    default: Y<=4'h0;
26
endcase
28 endmodule

```

1.5 Граф состояний

Опишем граф перехода цифрового автомата согласно указанным режимам работы (переход в следующее или предыдущее состояние, загрузка состояния, хранение, сброс). См рис. 3.

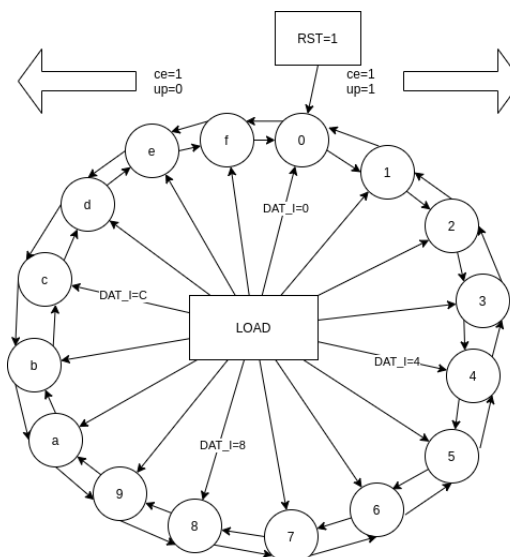


Рисунок 3 — Граф переходов

1.6 Создание проекта САПР Xilinx ISE

Приведем содержание verilog-модуля, описывающего цифровой автомат.

/home/denilai/Documents/repos/latex/scripts/fsm.v

```
'timescale 1ns / 1ps
2 //
module fsm(input      rst,
4         input      clk,
         input      ce,
6         input      load,
         input      up,
8         input  [3:0] data,

10        output [3:0] seq);

12 reg [3:0] state;

14 behaviour beh (.X(state),
                .Y(seq));

16 always @(posedge clk, posedge rst)
```

```

18 begin
    if (rst)
20         state <= 4'h0;
    else
22         begin
            if (load)
24                 state <= data;
            if (ce && up)
26                 state <= state + 4'h1;
            if (ce && !up)
28                 state <= state - 4'h1;
            if (!ce && !load)
30                 state <= state;
        end
    end
32 end
endmodule

```

Приведем содержание verilog-модуля, описывающего делитель частоты.

/home/denilai/Documents/repos/latex/scripts/freq_div.v

```

`timescale 1ns / 1ps
2
// freq_div instantiation example: freq_div #(10000) (.....)
4 module freq_div(input      rst,
    input      clk,
6
    output reg  co);
8     reg [16:0] counter;
    parameter divisor = 17'd10000;
10
    always @(posedge clk, posedge rst) begin
12         if (rst)
            begin
14                 counter <= 0;
                co <= 0;
16             end
        else
18             if (counter >= (divisor - 17'b1))
                begin
20                     counter <= 17'b0;
                    co <= 1;
22                 end
            else
24                 begin
                    co <= 0;
26                     counter <= counter + 17'b1;
                end
28     end
end

```

endmodule

Приведем содержание verilog-модуля, описывающего фильтр дребезга.

/home/denilai/Documents/repos/latex/scripts/m_btn_filter.v

```
`timescale 1ns / 1ps
2 module M_BTN_FILTER_V10(
4     input        CLK,
5     input        CE,
6     input        BTN_IN,
7     input        RST,
8
9     //output      BTN_OUT,
10    output reg    BTN_CEO
11    );
12    parameter [3:0] CNTR_WIDTH = 4; // Internal Counter Width
13    // Internal signals declaration:
14
15    reg [CNTR_WIDTH - 1:0] FLTR_CNT;
16    reg BTN_D, BTN_S1, BTN_S2;
17
18    //-----// Main Counter:
19    always @ (posedge CLK, posedge RST)
20    if(RST)
21        FLTR_CNT <= {CNTR_WIDTH{1'b0}};
22    else
23        if(!(BTN_S1 ^ BTN_S2)) // if BTN_S1 = BTN_S2
24            FLTR_CNT <= {CNTR_WIDTH{1'b0}}; // Return to Zero
25        else if(CE) // else if Clock Enable
26            FLTR_CNT <= FLTR_CNT + 1; // Increment
27    //-----// Input Synchronizer:
28    always @ (posedge CLK, posedge RST)
29    if(RST)
30        begin
31            BTN_D <= 1'b0;
32            BTN_S1 <= 1'b0;
33        end
34    else
35        begin
36            BTN_D <= BTN_IN;
37            BTN_S1 <= BTN_D;
38        end
39    end
40    //-----// Output Register:
41    always @ (posedge CLK, posedge RST)
42    if(RST)
43        BTN_S2 <= 1'b0;
44    else if(&(FLTR_CNT) & CE)
45        BTN_S2 <= BTN_S1;
```

```

46 //-----// Output Front Detector Clock
    Enable:
    always @ (posedge CLK, posedge RST)
48     if(RST)
        BTN_CEO <= 1'b0;
50     else
        BTN_CEO <= &(FLTR_CNT) & CE & BTN_S1;
52 endmodule
//-----assign BTN_OUT = BTN_S2;
54 //-----

```

Приведем содержание verilog-модуля, описывающего тестовое окружение, описывающее входные воздействия для данной модели.

/home/denilai/Documents/repos/latex/scripts/test-fsm.v

```

'timescale 1ns / 1ps
2
module test_fsm;
4
    // Inputs
6     reg     rst;
    reg     clk;
8     reg     ce;
    reg     load;
10    reg     up;
    reg [3:0] data;
12
    // Outputs
14    wire [3:0] seq;

16    // Instantiate the Unit Under Test (UUT)
    fsm uut (
18        .rst(rst),
        .clk(clk),
20        .ce(ce),
        .load(load),
22        .up(up),
        .data(data),
24        .seq(seq)
    );
26    always
        #5 clk=~clk;
28
    initial begin
30        // Initialize Inputs
        rst = 1;
32        clk = 0;
        ce = 0;

```

```

34     load = 0;
      up = 0;
36     data = 0;

38     // Wait 100 ns for global reset to finish
      #100;

40     // count forward

42     rst=0;
44     ce=1;
      up=1;
46     #165;

48     //count bashward

50     rst=0;
      ce=1;
52     up=0;
      #180;

54     //one state (store mode)

56     rst=0;
58     ce=0;
      up=0;
60     #100;

62     rst=0;
64     ce=0;
      up=0;
66     load=1;

68     //load mode
      data=4'h0;
70     #20;
      data=4'h1;
72     #20;
      data=4'h2;
74     #20;
      data=4'h3;
76     #20;
      data=4'h4;
78     #20;
      data=4'h5;
80     #20;
      data=4'h6;

```

```

82     #20;
      data=4'h7;
84     #20;
      data=4'h8;
86     #20;
      data=4'h9;
88     #20;
      data=4'ha;
90     #20;
      data=4'hb;
92     #20;
      data=4'hc;
94     #20;
      data=4'hd;
96     #20;
      data=4'he;
98     #20;
      data=4'hf;
100    // #20;

102    $stop;

104    // Add stimulus here

106    end
endmodule

```

Приведем содержание verilog-модуля верхнего уровня

/home/denilai/Documents/repos/latex/scripts/top2.v

```

'timescale 1ns / 1ps
2
module top(
4   input      clk,
   input [4:0] SW,
6   input      CPU_RESET,
   input      BTNC,
8   input      BTNU,
   output [3:0] LED
10  );

12
   reg          RST_I;
14  wire         RST;
   wire         CO;
16  wire         BTNC_CEO;
   wire         BTNU_CEO;
18

```

```

always @ (posedge clk, negedge CPU_RESET)
20   begin
        if(~CPU_RESET)
22           RST_I <= 1'b1;
        else
24           RST_I <= 1'b0;
        end
26   assign RST=RST_I;
   freq_div #(10000) FREQ_CO (
28       .rst (RST),
       .clk (clk),
30       .co  (CO)
   );

   M_BTN_FILTER_V10 b_f_u(
32       .CLK      (clk),
       .CE        (CO),
34       .BTN_IN   (BTNU),
       .RST       (RST_I),
36       .BTN_CEO  (BTNU_CEO)
   );

   M_BTN_FILTER_V10 b_f_c(
40       .CLK      (clk),
       .CE        (co),
42       .BTN_IN   (BTNC),
       .RST       (RST_I),
44       .BTN_CEO  (BTNC_CEO)
   );

   fsm FSM1(
50       .rst      (RST_I),
       .clk       (clk),
52       .ce       (BTNC_CEO),
       .load      (BTNU_CEO),
54       .up       (SW[4]),
       .data      (SW[3:0]),
56       .seq      (LED)
   );
58   endmodule

```

1.7 Тестирование и отладка средствами симулятора iSim

После компоновки проекта, подключения модуля верхнего уровня, проведем верификацию спроектированных моделей с помощью симулятора

iSim из состава САПР Xilinx ISE Design Suite. Результаты тестирования можно видеть на рис. 4 и 5.

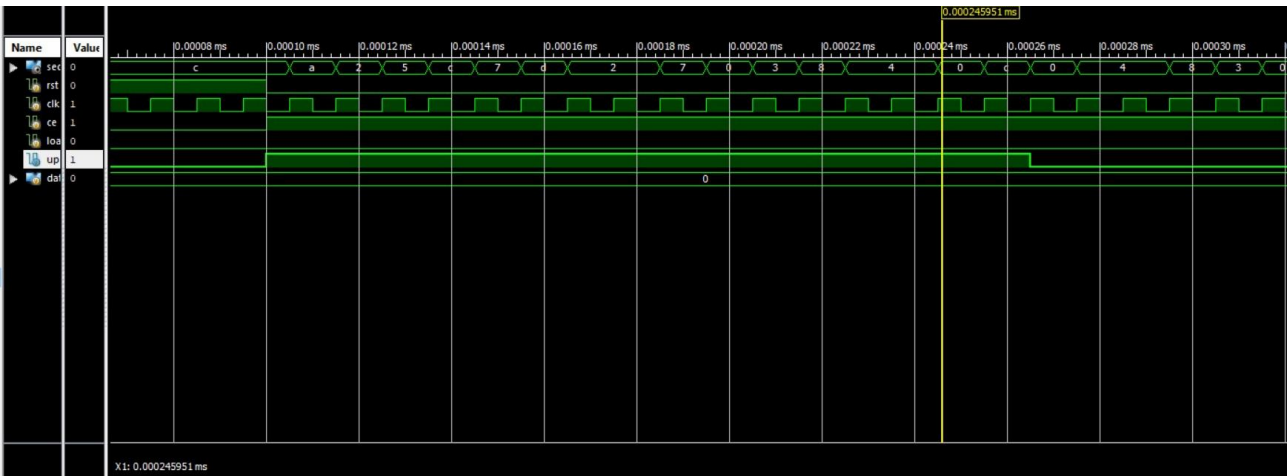


Рисунок 4 — Вывод iSim

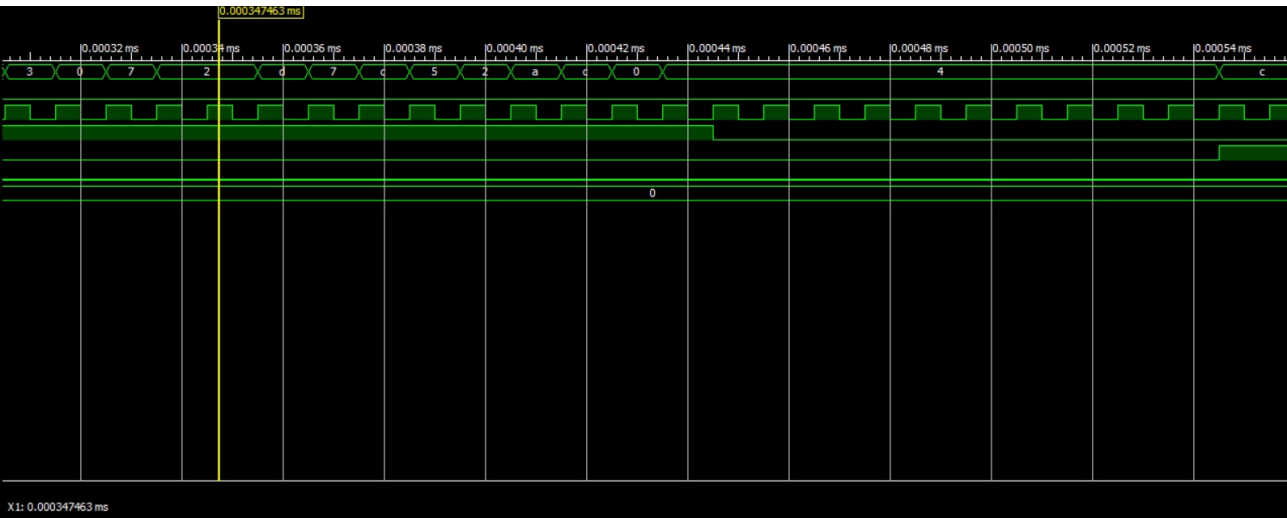


Рисунок 5 — Вывод iSim

Приведем структуру проекта. См. рис. 6.

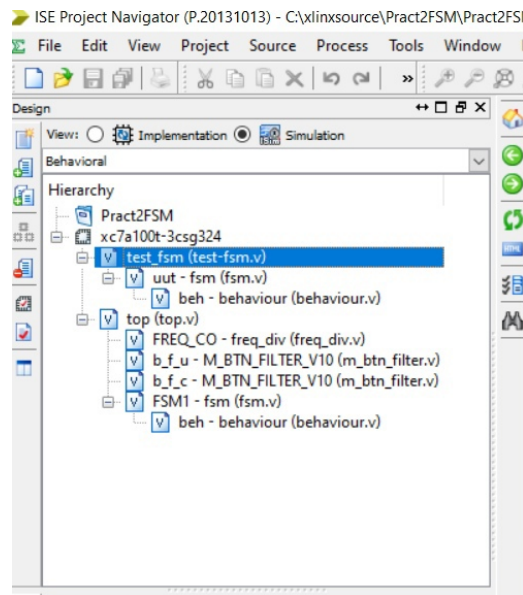


Рисунок 6 — Иерархия проекта

2 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 конечный автомат, представляющий собой генератор фиксированной последовательности логических сигналов, в виде синтезируемой модели.