COD_LAB2 数据通路与状态机

黄业琦 PB17000144

March 25, 2019

Contents

1 实验目的 **2**
1.1 排序 . . . 2
1.2 除法运算 . . . 2

2 实验环境 **3**

3 逻辑设计 **3**
3.1 Sort 设计 . . . 3
3.2 Fibonacci 设计和累计求和 . . . 3

4 仿真截图 **4**

5 性能评测截图 **5**

6 实验代码 **6**
6.1 Sort 实现代码 . . . 6
6.2 DIV 实现代码 . . . 9

7 FSM 实现 **12**
7.1 排序 FSM 设计 . . . 12
7.2 除法 FSM 设计 . . . 12

8 三段式状态机代码 **13**
8.1 SortFSM 实现代码 . . . 13
8.2 DivFSM 实现代码 . . . 17

1 实验目的

1.1 排序

s0 - s3 是 x0 - x3 的排序结果。

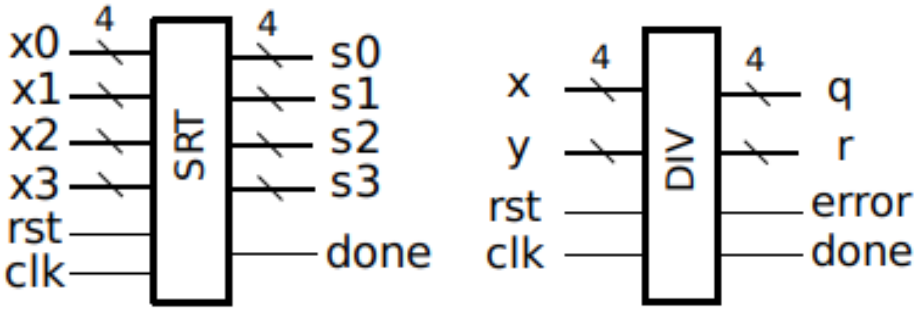

Figure 1: Sort & Div

1.2 除法运算

x / y = q ···r

2 实验环境

Linux 下编程调试和仿真，使用 IVerilog，GtkWave 系列工具。
Windows 下用于生成比特流文件，使用 Vivado 2018.2，Verilog HDL
所有下载均在 Nexsy4-DDR 实验板完成

3 逻辑设计

3.1 Sort 设计

采用六次比较以实现我们的排序工作。
详细见附录代码。

3.2 Fibonacci 设计和累计求和

采用移位算法实现我们的除法操作：
1、被除数为 x, 除数为 y, 商为 qout, 余数为 rout；
2、将被除数赋给寄存器变量 q，变量 r 初始值为 0，t 为 q,r;
3、先将 t 左移一位；
4、比较 r 与除数的大小，如果 r> 除数则 r=r-除数并且 t 再左移一位并且 q[0]=1，反之 r 不变 t 左移一位 q[0]=0。
5、继续上述过程，循环至无法移位了；
6、此时商 qout=q，而余数 rout=r»1(即余数等于 r 向右移一位)。

4 仿真截图

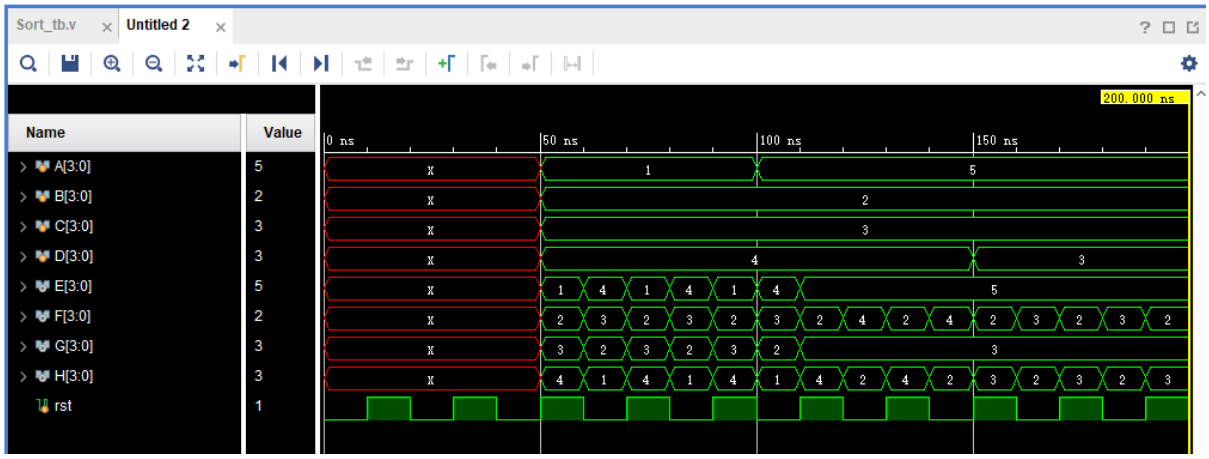

Figure 2: SORT_sim

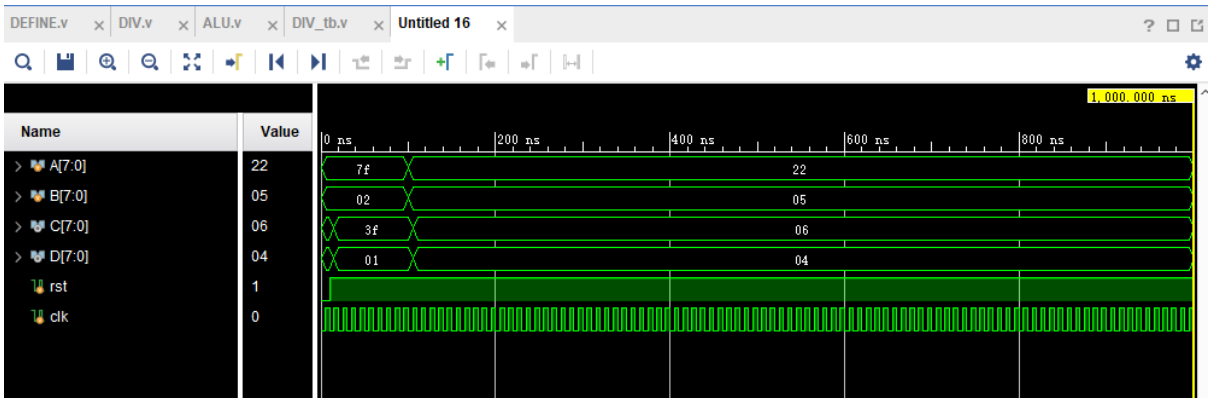

Figure 3: DIV_sim

5 性能评测截图

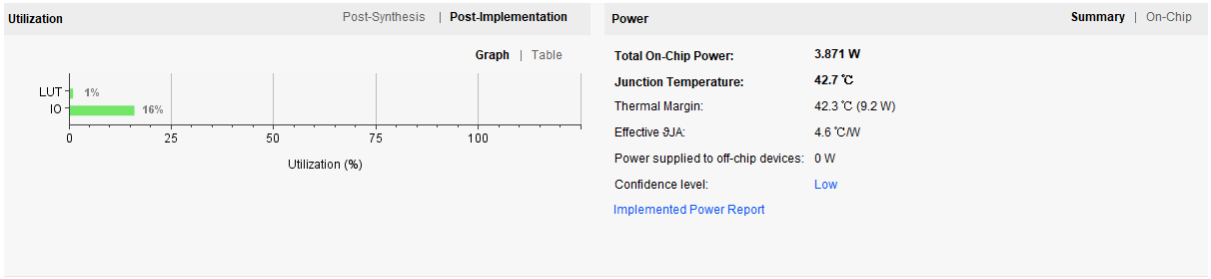

Figure 4: Sort_Performance1

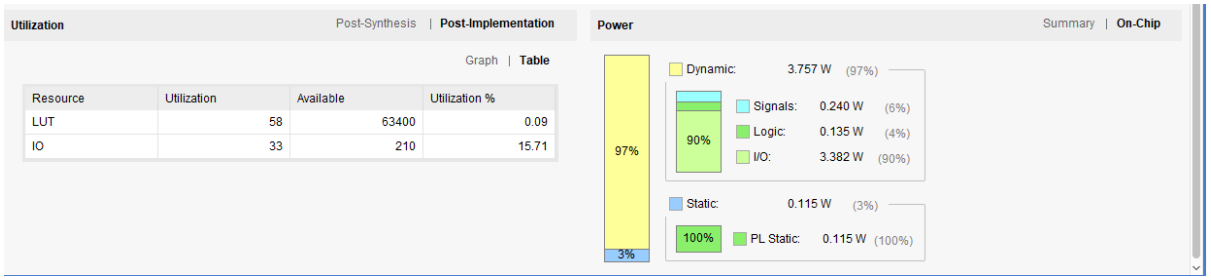

Figure 5: Sort_Performance2

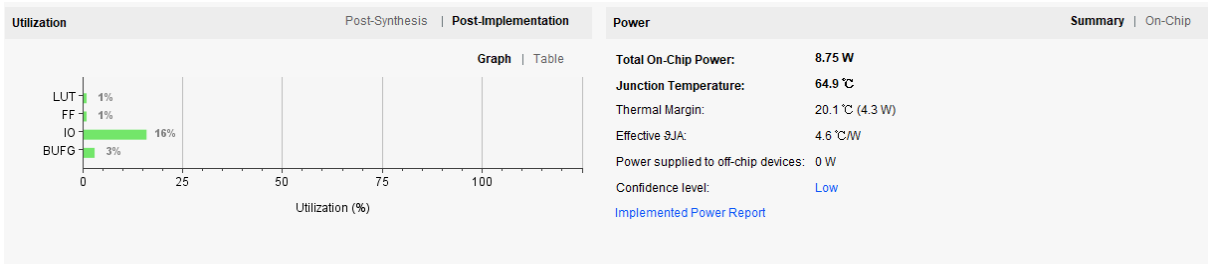

Figure 6: DIV_Performance1

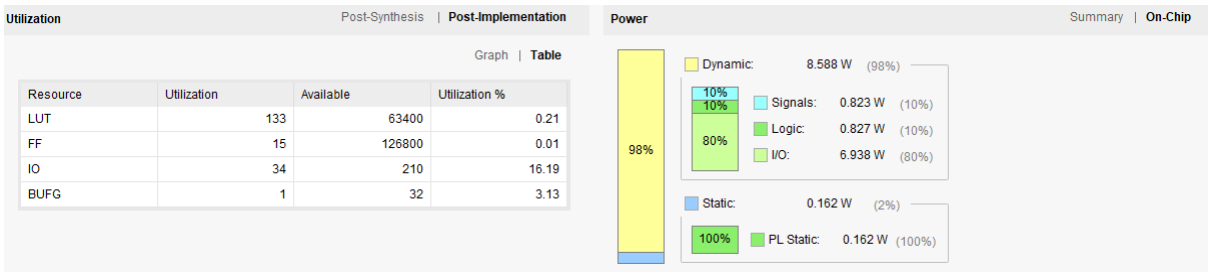

Figure 7: DIV_Performance2

6 实验代码

6.1 Sort 实现代码

DEFINE 和 ALU 同上一次的实验，不再重复给出。

Listing 1: CMP.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/03/23 22:14:52
// Design Name:
// Module Name: CMP
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module CMP(
    input [3:0] A,
    input [3:0] B,
    output reg C
    );
    reg [`OPECODE_BUS] SUB;
    wire [`OPERAND_BUS] CT;
    wire [`OPECODE_BUS] F;

    initial SUB = `EXE_SUB;

    ALU ALUER(SUB,A,B,CT,F);

    always @* C = F[2];
endmodule
```

Listing 2: Sort.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
```

```
6  // Create Date: 2019/03/23 22:14:02
7  // Design Name:
8  // Module Name: Sort
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22
23 module Sort(
24     input rst,
25     input [3:0] sIN1,
26     input [3:0] sIN2,
27     input [3:0] sIN3,
28     input [3:0] sIN4,
29     output reg [3:0] sOUT1,
30     output reg [3:0] sOUT2,
31     output reg [3:0] sOUT3,
32     output reg [3:0] sOUT4
33     );
34
35     wire T12,T13,T14,T23,T24,T34;
36
37     reg [3:0] TMP;
38     reg [3:0] tOUT1,tOUT2,tOUT3,tOUT4;
39
40     always@*
41     begin
42             sOUT1=tOUT1;
43             sOUT2=tOUT2;
44             sOUT3=tOUT3;
45             sOUT4=tOUT4;
46     end
47
48     always @*
49     begin
50         if (rst)
51         begin
52                     tOUT1 = sIN1;
53                     tOUT2 = sIN2;
54                     tOUT3 = sIN3;
55                     tOUT4 = sIN4;
56         end
```

```
57          else
58          begin
59          if (T12) begin TMP = tOUT1; tOUT1 = tOUT2; tOUT2 = TMP; end
60          if (T13) begin TMP = tOUT1; tOUT1 = tOUT3; tOUT3 = TMP; end
61          if (T14) begin TMP = tOUT1; tOUT1 = tOUT4; tOUT4 = TMP; end
62          if (T23) begin TMP = tOUT2; tOUT2 = tOUT3; tOUT3 = TMP; end
63          if (T24) begin TMP = tOUT2; tOUT2 = tOUT4; tOUT4 = TMP; end
64          if (T34) begin TMP = tOUT3; tOUT3 = tOUT4; tOUT4 = TMP; end
65          end
66      end
67
68          CMP c12(tOUT1,tOUT2,T12);
69          CMP c13(tOUT1,tOUT3,T13);
70          CMP c14(tOUT1,tOUT4,T14);
71          CMP c23(tOUT2,tOUT3,T23);
72          CMP c24(tOUT2,tOUT4,T24);
73          CMP c34(tOUT3,tOUT4,T34);
74  endmodule
```

Listing 3: Sort_tb.v

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date: 2019/03/23 23:14:36
7   // Design Name:
8   // Module Name: Sort_tb
9   // Project Name:
10  // Target Devices:
11  // Tool Versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22  `include "DEFINE.v"
23
24  module Sort_tb (
25      );
26
27
28      reg  [`OPERAND_BUS] A;
29      reg  [`OPERAND_BUS] B;
30      reg  [`OPERAND_BUS] C;
31      reg  [`OPERAND_BUS] D;
```

```
32      wire [`OPERAND_BUS] E;
33      wire [`OPERAND_BUS] F;
34      wire [`OPERAND_BUS] G;
35      wire [`OPERAND_BUS] H;
36      reg rst;
37
38      initial begin
39      rst = 0;
40      forever #10 rst = ~rst;
41      end
42
43     initial begin
44      #50 A = 1; B = 2; C = 3; D = 4;
45      #50 A = 5; B = 2; C = 3; D = 4;
46      #50 A = 5; B = 2; C = 3; D = 3;
47      #50 $stop;
48     end
49
50     Sort S(rst,A,B,C,D,E,F,G,H);
51 endmodule
```

6.2 DIV 实现代码

Listing 4: DIV.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2019/03/23 23:56:50
7  // Design Name:
8  // Module Name: DIV
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21 `include "DEFINE.v"
22
23 module DIV(
24     input clk,
25     input rst,
```

```
26      input [`OPERAND_BUS] A,
27      input [`OPERAND_BUS] B,
28      output [`OPERAND_BUS] C,
29      output [`OPERAND_BUS] D
30  );
31
32      reg [`OPERAND_BUS]q;
33      reg [`OPERAND_BUS]r;
34      reg [`OPERAND_BUS]t;
35
36      integer i;
37
38      always@(posedge clk or negedge rst)
39      begin
40      if(!rst)
41      begin
42          q=0;
43          r=0;
44      end
45      else
46      begin
47          q=A;
48          t={4'b0000,B};
49          r=8'b00000000;
50          {r,q}={r,q}<<1;
51
52          for(i=0;i<8;i=i+1)
53          if(r>=t)
54          begin
55              r=r-t;
56              {r,q}={r,q}<<1;
57              q[0]=1;
58          end
59          else
60          begin
61              r=r;
62              {r,q}={r,q}<<1;
63              q[0]=0;
64          end
65      end
66      end
67      assign C=q;
68      assign D=r>>1;
69  endmodule
```

Listing 5: DIV_tb.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
```

```
// Create Date: 2019/03/24 13:01:00
// Design Name:
// Module Name: DIV_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//

 `timescale 1ns/1ns
module divider_module_tb  ;

 reg  [7:0]  A   ;
 reg  [7:0]  B   ;
 wire  [7:0]  C   ;
 wire  [7:0]  D   ;
 reg    rst   ;
 reg    clk   ;

 initial
 begin
   rst<=0;
   #10 rst<=1;
 end

 initial clk<=0;
 always #5 clk<=~clk;
 initial
 begin
   A<=8'd127;
   B<=4'd2;
   #100
   A=8'd34;
   B=5;
   #100;
 end

 DIV DUT ( clk,rst,A,B,C,D);

endmodule
```

7 FSM 实现

7.1 排序 FSM 设计

我们设计存在三个状态:

1. 初始状态，加载数据
2. 一趟排序流程（冒泡一趟）
3. 终

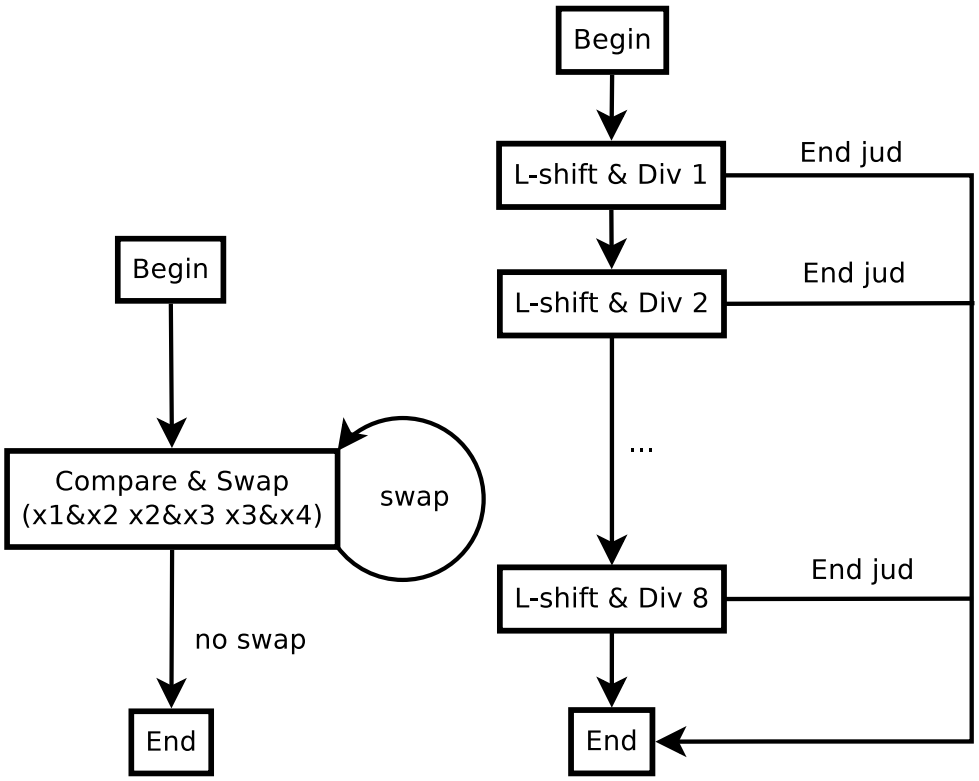

Figure 8: Sort & Div

7.2 除法 FSM 设计

我们设计存在三个状态:

1. 初始状态，加载数据
2. 一趟移位除法操作，同之前算法描述处介绍
3. 终

之前的 i 循环变量视为状态编号就可以将之前的代码视作两段式状态机的程序。

8 三段式状态机代码

仿真图像和之前一致，不再重复给出。这里在 DivFSM 中加入了 err 判定以及我们的 Done 判定。(之前写漏了)

8.1 SortFSM 实现代码

Listing 6: Sort.v

```
`timescale 1ns / 1ps

module Sort(
    input rst,
    input clk,
    input [3:0] sIN1,
    input [3:0] sIN2,
    input [3:0] sIN3,
    input [3:0] sIN4,
    output [7:0] an,
    output [7:0] seg
    );

    wire T12,T13,T14,T23,T24,T34;
    reg [3:0] TMP;
    reg [3:0] tOUT1,tOUT2,tOUT3,tOUT4;
    reg [3:0] nOUT1,nOUT2,nOUT3,nOUT4;
    reg [3:0] sOUT1,sOUT2,sOUT3,sOUT4;
    reg jud;

    always @(posedge clk or negedge rst)
    begin
        if (rst)
        begin
            nOUT1 = sIN1;
            nOUT2 = sIN2;
            nOUT3 = sIN3;
            nOUT4 = sIN4;
        end
        else
        begin
            jud = 0;
            nOUT1 = tOUT1;
            nOUT2 = tOUT2;
            nOUT3 = tOUT3;
            nOUT4 = tOUT4;
        end
    end

    CMP c12(nOUT1,nOUT2,T12);
    CMP c23(nOUT2,nOUT3,T23);
```

```
42      CMP c34(nOUT3,nOUT4,T34);
43
44      always @*
45      begin
46          if (jud)
47          begin
48          if (T12) begin TMP = nOUT1; nOUT1 = nOUT2; nOUT2 = TMP; jud=1; end
49          if (T23) begin TMP = nOUT2; nOUT2 = nOUT3; nOUT3 = TMP; jud=1; end
50          if (T34) begin TMP = nOUT3; nOUT3 = nOUT4; nOUT4 = TMP; jud=1; end
51          end
52      end
53
54      always @(posedge clk or negedge rst)
55      begin
56          tOUT1 = nOUT1;
57          tOUT2 = nOUT2;
58          tOUT3 = nOUT3;
59          tOUT4 = nOUT4;
60          if (!jud)
61          begin
62              sOUT1 = tOUT1;
63              sOUT2 = tOUT2;
64              sOUT3 = tOUT3;
65              sOUT4 = tOUT4;
66              SegOUT (clk,sOUT1,sOUT2,sOUT3,sOUT4,an,seg);
67          end
68      end
69  endodule
```

Listing 7: Seg_out.v

```
1   `timescale 1ns / 1ps
2
3   module Seg(
4       input [3:0] a,
5       output reg [0:6] seg);
6       always@*
7       begin
8       case (a)
9           4'd0:begin  seg[0]=0;seg[1]=0;seg[2]=0;seg[3]=0;
10                      seg[4]=0;seg[5]=0;seg[6]=1; end
11          4'd1:begin  seg[0]=1;seg[1]=0;seg[2]=0;seg[3]=1;
12                      seg[4]=1;seg[5]=1;seg[6]=1; end
13          4'd2:begin  seg[0]=0;seg[1]=0;seg[2]=1;seg[3]=0;
14                      seg[4]=0;seg[5]=1;seg[6]=0; end
15          4'd3:begin  seg[0]=0;seg[1]=0;seg[2]=0;seg[3]=0;
16                      seg[4]=1;seg[5]=1;seg[6]=0; end
17          4'd4:begin  seg[0]=1;seg[1]=0;seg[2]=0;seg[3]=1;
18                      seg[4]=1;seg[5]=0;seg[6]=0; end
19          4'd5:begin  seg[0]=0;seg[1]=1;seg[2]=0;seg[3]=0;
20                      seg[4]=1;seg[5]=0;seg[6]=0; end
21          4'd6:begin  seg[0]=0;seg[1]=1;seg[2]=0;seg[3]=0;
```

```
                    seg[4]=0;seg[5]=0;seg[6]=0; end
        4'd7:begin  seg[0]=0;seg[1]=0;seg[2]=0;seg[3]=1;
                    seg[4]=1;seg[5]=1;seg[6]=1; end
        4'd8:begin  seg[0]=0;seg[1]=0;seg[2]=0;seg[3]=0;
                    seg[4]=0;seg[5]=0;seg[6]=0; end
        4'd9:begin  seg[0]=0;seg[1]=0;seg[2]=0;seg[3]=0;
                    seg[4]=1;seg[5]=0;seg[6]=0; end
        4'd10:begin seg[0]=0;seg[1]=0;seg[2]=0;seg[3]=1;
                    seg[4]=0;seg[5]=0;seg[6]=0; end//A
        4'd11:begin seg[0]=1;seg[1]=1;seg[2]=0;seg[3]=0;
                    seg[4]=0;seg[5]=0;seg[6]=0; end//b
        4'd12:begin seg[0]=0;seg[1]=1;seg[2]=1;seg[3]=0;
                    seg[4]=0;seg[5]=0;seg[6]=1; end//C
        4'd13:begin seg[0]=1;seg[1]=0;seg[2]=0;seg[3]=0;
                    seg[4]=0;seg[5]=1;seg[6]=0; end//d
        4'd14:begin seg[0]=0;seg[1]=1;seg[2]=1;seg[3]=0;
                    seg[4]=0;seg[5]=0;seg[6]=0; end//E
        4'd15:begin seg[0]=0;seg[1]=1;seg[2]=1;seg[3]=1;
                    seg[4]=0;seg[5]=0;seg[6]=0; end//F
        default:seg=7'b0000000;
    endcase
    end
endmodule

module counter #(parameter cnt=9)
    (
    input clk,
    input enable,
    input reset,
    output carry,
    output reg [3:0] Q
    );
    assign carry=(Q==cnt)?1:0;
    initial
    begin
    Q<=0;
    end
    always@(posedge clk,posedge reset)
    begin
        if(reset)
        Q<=0;
        else if(enable)
        if(Q>=cnt)
            Q<=0;
        else
            Q=Q+1;
    end

endmodule

module segOutput(
```

```
73      input clk5MHZ,
74      input [3:0] Q0,
75      input [3:0] Q1,
76      input [3:0] Q2,
77      input [3:0] Q3,
78      output reg [7:0] an,
79      output reg [7:0] seg,
80      output dp
81      );
82      wire [3:0] num;
83      wire [7:0] seg0,seg1,seg2,seg3;
84      reg clk;
85      reg [12:0] count ;
86
87      Seg b0(Q0,seg0);
88      Seg b1(Q1,seg1);
89      Seg b2(Q2,seg2);
90      Seg b3(Q3,seg3);
91      initial
92      begin
93      an<=8'b11111111;
94      clk<=0;
95      count<=0;
96      end
97      always@(posedge clk5MHZ)
98          if(count>=13'd4999)
99          begin
100             clk<=~clk;
101             count=13'b0;
102         end
103         else
104         count=count+1;
105     counter#(3) c(clk,1,,,num);
106     always@(num)
107     case(num)
108         0:begin
109         an<=8'b00000001;
110         seg<=seg0;
111         end
112         1:begin
113         an<=8'b00000010;
114         seg<=seg1;
115         end
116         2:begin
117         an<=8'b00000100;
118         seg<=seg1;
119         end
120         3:begin
121         an<=8'b00001000;
122         seg<=seg1;
123         end
```

```
124     endcase
125 endmodule
126 
127 module SegOUT (
128     input CLK100MHZ,
129     input [3:0] A,
130     input [3:0] B,
131     input [3:0] C,
132     input [3:0] D,
133     output [7:0] an,
134     output [7:0] seg)
135 begin
136     wire clk0,locked,reset,clk_out1,JW1,JW2;
137     integer clk1;
138     wire [0:6] seg1,seg2,seg3,seg4;
139     clk_wiz0 C1(.clk_in1(CLK100MHZ),.reset(reset),
140                 .clk_out1(clk_out1),.locked(locked));
141     clock C2(clk_out1,locked,clk0);
142     numclock C3(clk_out1,locked,clk1);
143     Seg s1(A[3:0],seg1[0:6]);
144     Seg s2(B[3:0],seg2[0:6]);
145     Seg s2(C[3:0],seg3[0:6]);
146     Seg s2(D[3:0],seg4[0:6]);
147     segOutput so(clk5MHZ,seg1,seg2,seg3,seg4,an,seg);
148 end
```