中国科学技术大学计算机学院

《数字电路实验》报告

实验题目：寄存器堆

学生姓名：柯志伟

学生学号：PB20061338

完成日期：2021 年 11 月 27 日

计算机实验教学中心制

2020 年 09 月

【实验题目】

寄存器堆

【实验目的】

- 寄存器堆
- 数据输入/输出
- 数据排序

【实验环境】

- PC 机
- Vivado 平台
- Nexys4 开发板

【实验过程】

- 设计模块实现的状态图

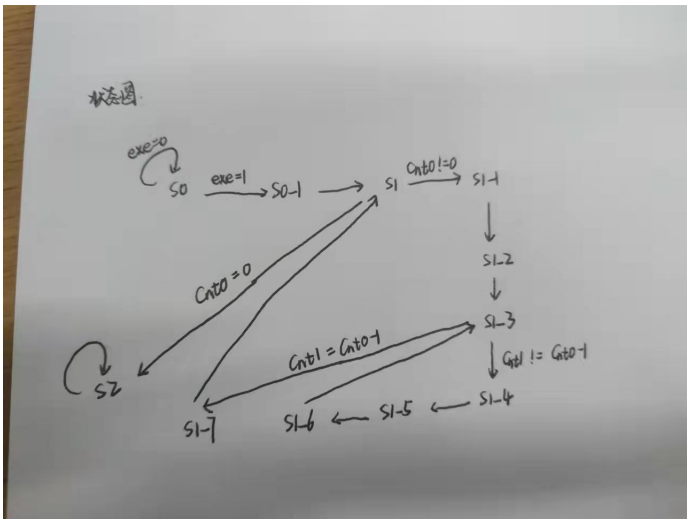

图 1: 输入数据并实现排序的状态图

- 用 Verilog 分别实现寄存器堆、数据输入/输出、数据排序的模块
  代码如下:
  - 寄存器堆

```
module  register_file # (
    parameter AW = 5,           //地址宽度
    parameter DW = 16           //数据宽度
```

```
    )(
        input clk,                          //时钟
        input [AW-1:0] ra0, ra1,                    //读地址
        output [DW-1:0] rd0, rd1,           //读数据
        input [AW-1:0] wa,                  //写地址
        input [DW-1:0] wd,                  //写数据
        input we                            //写使能
    );
    reg [DW-1:0] rf [0: (1<<AW)-1];                 //寄存器堆
    assign rd0 = rf[ra0], rd1 = rf[ra1];        //读操作

    always @(posedge clk)
        if (we) rf[wa] <= wd;                   //写操作
    endmodule
```

– 数据输入/输出

```
module DIO(input clk,rstn,
        input [15:0] d,
        input pre,
        input bs,
        input nxt,
        output [7:0] an,
        output [6:0] seg
);

    //对输入去抖动并同步化
    wire DS_bs,DS_pre,DS_nxt;
    wire [15:0] DS_d;
    wire [3:0] En_d;

    reg [15:0] DR;
    reg [4:0] AR;
    reg [15:0] DReg;
    reg [15:0] DNxt;

    reg [7:0] Count;

    reg [4:0] wa;

    reg [15:0] wd;

    reg we;
    wire [15:0] RD;
```

```
28
29      reg [2:0] CS,NS;
30
31      reg [1:0] T;    // {nxt,pre}
32
33      parameter s0 = 3'b000,s1 = 3'b001,s2 = 3'b010,
34      s3 = 3'b011,s4 = 3'b100,s5 = 3'b101;
35
36
37      deSyn desyn_bs(clk,rstn,bs,DS_bs);
38      deSyn desyn_pre(clk,rstn,pre,DS_pre);
39      deSyn desyn_nxt(clk,rstn,nxt,DS_nxt);
40
41      deSyn desyn_d0(clk,rstn,d[0],DS_d[0]);
42      deSyn desyn_d1(clk,rstn,d[1],DS_d[1]);
43      deSyn desyn_d2(clk,rstn,d[2],DS_d[2]);
44      deSyn desyn_d3(clk,rstn,d[3],DS_d[3]);
45      deSyn desyn_d4(clk,rstn,d[4],DS_d[4]);
46      deSyn desyn_d5(clk,rstn,d[5],DS_d[5]);
47      deSyn desyn_d6(clk,rstn,d[6],DS_d[6]);
48      deSyn desyn_d7(clk,rstn,d[7],DS_d[7]);
49      deSyn desyn_d8(clk,rstn,d[8],DS_d[8]);
50      deSyn desyn_d9(clk,rstn,d[9],DS_d[9]);
51      deSyn desyn_d10(clk,rstn,d[10],DS_d[10]);
52      deSyn desyn_d11(clk,rstn,d[11],DS_d[11]);
53      deSyn desyn_d12(clk,rstn,d[12],DS_d[12]);
54      deSyn desyn_d13(clk,rstn,d[13],DS_d[13]);
55      deSyn desyn_d14(clk,rstn,d[14],DS_d[14]);
56      deSyn desyn_d15(clk,rstn,d[15],DS_d[15]);
57
58
59
60
61      encode ECD(DNxt,En_d);
62
63      register_file RF(.clk(clk),.ra0(AR),.rd0(RD),
64      .wa(AR),.wd(DR),.we(we));
65
66      dynimage DIS(clk,rstn,Count,DR,an,seg);
67
68      always @(posedge clk,negedge rstn) begin
69      if(!rstn)
```

```
70          CS <= s0;
71      else CS <= NS;
72
73      end
74
75      /*
76      s0:默认状态
77      s1:读出数字并显示
78      s2:输入数字
79      s3:保存数字
80      s4:退格
81      s5:显示上一个数字或下一个数字
82      */
83
84      always @(*) begin
85          case(CS)
86              s0: NS = s1;
87              s1: NS = s2;
88              s2: begin
89              if(T[1]||T[0])   NS = s5;
90              else if(DS_bs) NS = s4;
91              else if(DNxt) NS = s3;
92              else NS = s2;
93              end
94              s3: NS = s2;
95              s4: NS = s2;
96              s5: NS = s1;
97          endcase
98      end
99
100
101 always @(posedge clk,negedge rstn) begin
102 if(!rstn) begin
103   DNxt <= 0;
104   DReg <= 0;
105   AR <= 0;
106   DR <= 0;
107   Count <= 0;
108   we <= 0;
109 end
110 else begin
111   DNxt <= DS_d - DReg;
```

```
112     DReg <= DS_d;
113     case(NS)
114     s1:begin
115          DR <= RD;
116          we <= 0;
117     end
118     s2:begin
119         T <= {DS_nxt,DS_pre};
120         we <= 0;
121     end
122     s3:begin
123          DR <= (DR<<4)+En_d;
124          we <= 1;
125     end
126     s4:begin
127         DR <=  DR >> 4;
128         we <= 1;
129     end
130     s5:begin
131         we = 0;
132        if(T[1]) begin
133          AR <= AR +1;
134          Count <= Count +1;
135        end
136        if(T[0]) begin
137          AR <= AR -1;
138          Count <= Count -1;
139        end
140     end
141     endcase
142 end
143 end
144 
145 endmodule
146 
147 //其他模块
148 
149 module deSyn(input clk,rstn,input x,output y);
150      wire de_x;
151      debounce DB(clk,rstn,x,de_x);
152      Synchron SYN(de_x,clk,rstn,y);
153 endmodule
```

```
154
155 module debounce(input clk,rstn,
156                  input x,
157                  output reg y);
158     parameter  Cnt0 = 3'b000, Cnt1 = 3'b001,
159     Assign0 = 3'b010,Assign1=3'b011,Default = 3'b111;
160       //刚开始进入默认状态，两个计数状态，两个赋值状态
161     reg [2:0] CS,NS;
162     reg [19:0] Count;
163     reg y_reg;
164 always @(posedge clk,negedge rstn) begin //异步复位
165     if(!rstn)begin
166       CS <= Default;
167       y_reg <= 0;
168     end
169     else begin
170       CS <= NS;
171     y_reg <= y;
172     end
173 end
174
175 always @(posedge clk,negedge rstn) begin
176   if(!rstn) Count <=0;
177   else begin
178     case(CS)
179         Default: Count <= 0;
180     Cnt0 :begin
181       Count <= Count+1;
182     end
183     Cnt1: Count <= Count+1;
184     Assign0: Count <= 0;
185     Assign1: Count <= 0;
186
187   endcase
188   end
189
190
191 end
192
193 always @(*) begin
194     y = y_reg;
195     NS = Default;
```

```
196     case (CS)
197     Default: begin
198       y = 0;
199       if(x == 1) NS = Cnt1;
200       else NS = Cnt0;
201     end
202     Cnt0:begin
203         if(Count == 100000) begin
204           NS = Assign0;
205         end
206         else if(x == 0)begin
207           NS = Cnt0;
208         end
209         else begin
210           NS = Default;
211         end
212     end
213     Cnt1:begin
214         if(Count == 100000) begin
215           NS = Assign1;
216         end
217         else if(x == 1)begin
218           NS = Cnt1;
219         end
220         else begin
221           NS = Default;
222         end
223     end
224     Assign0: begin
225             y = 0;
226       if(x == 0)
227         NS = Assign0;
228       else
229           NS = Default;   //只有极个别才在此刻为1
230     end
231     Assign1:begin
232       y = 1;
233       if(x == 1) NS = Assign1;
234       else NS = Default;   //只有极个别才在此刻为1
235     end
236     endcase
237 end
```

```
238 endmodule
239
240 module Synchron(input x,input clk,rstn,
241                 output y);
242 reg s1,s2;
243 reg s;
244
245 always @(posedge clk) begin
246     if(~rstn ) begin
247       s1 <= 0;
248       s2 <= 0;
249       s <= 0;
250     end
251     else begin
252       s1 <= x;
253       s2 <= s1;
254       s <= s2;
255     end
256 end
257 assign y = (!s)&s2;
258 endmodule
259
260 module  register_file # (
261     parameter AW = 5,               //地址宽度
262     parameter DW = 16               //数据宽度
263 )(
264     input clk,                      //时钟
265     input [AW-1:0] ra0, ra1,                //读地址
266     output [DW-1:0] rd0, rd1,       //读数据
267     input [AW-1:0] wa,              //写地址
268     input [DW-1:0] wd,              //写数据
269     input we                        //写使能
270 );
271     reg [DW-1:0] rf [0: (1<<AW)-1];             //寄存器堆
272     assign rd0 = rf[ra0], rd1 = rf[ra1];        //读操作
273
274     always @(posedge clk)
275         if (we) rf[wa] <= wd;                   //写操作
276 endmodule
277
278 module encode(input [15:0] In,output reg [3:0] Out );
279     always @(*) begin
```

```
280             case(In)
281             16'b0000_0000_0000_0001: Out = 4'b0000;
282             16'b0000_0000_0000_0010: Out = 4'b0001;
283             16'b0000_0000_0000_0100: Out = 4'b0010;
284             16'b0000_0000_0000_1000: Out = 4'b0011;
285             16'b0000_0000_0001_0000: Out = 4'b0100;
286             16'b0000_0000_0010_0000: Out = 4'b0101;
287             16'b0000_0000_0100_0000: Out = 4'b0110;
288             16'b0000_0000_1000_0000: Out = 4'b0111;
289             16'b0000_0001_0000_0000: Out = 4'b1000;
290             16'b0000_0010_0000_0000: Out = 4'b1001;
291             16'b0000_0100_0000_0000: Out = 4'b1010;
292             16'b0000_1000_0000_0000: Out = 4'b1011;
293             16'b0001_0000_0000_0000: Out = 4'b1100;
294             16'b0010_0000_0000_0000: Out = 4'b1101;
295             16'b0100_0000_0000_0000: Out = 4'b1110;
296             16'b1000_0000_0000_0000: Out = 4'b1111;
297             endcase
298         end
299
300 endmodule
301
302 module dynimage(
303                     input clk,rstn,
304                     input [7:0] Cnt,
305                     input [15:0] d,
306                     output reg [7:0] an,
307                     output reg [6:0] seg);
308
309                     wire clkd;          //分频时钟
310                     reg [3:0] DIn;
311                     wire [6:0] DOut;
312             frequdivision  FreDivClk(clk,~rstn,clkd);
313             Decoder7Seg decoder(DIn,DOut);
314
315 parameter s0 = 3'b000, s1 = 3'b001,s2 = 3'b010,
316 s3 = 3'b011,s4 = 3'b100,s5 = 3'b101;
317 reg [2:0] CS,NS;
318
319
320 always @(posedge clkd,negedge rstn) begin  //异步初始化
321     if(~rstn) CS <= s0;
```

```
322       else CS <= NS;
323 end
324
325 always @(*) begin
326         case(CS)
327         s0 : begin
328               DIn = d[3:0];
329               an = 8'b1111_1110;
330               seg = DOut;
331               NS = s1;
332         end
333         s1 : begin
334           DIn = d[7:4];
335           seg = DOut;
336           NS = s2;
337           an = 8'b1111_1101;
338         end
339         s2 : begin
340           DIn = d[11:8];
341           seg = DOut;
342           NS = s3;
343           an = 8'b1111_1011;
344
345         end
346         s3 : begin
347           DIn = d[15:12];
348           seg = DOut;
349           NS = s4;
350           an = 8'b1111_0111;
351         end
352         s4 :begin
353           DIn = Cnt[3:0];
354           seg = DOut;
355           NS = s5;
356           an = 8'b1011_1111;
357         end
358         s5 :begin
359           DIn = Cnt[7:4];
360           seg = DOut;
361           NS = s0;
362           an = 8'b0111_1111;
363         end
```

```
364         endcase
365 end
366
367
368 endmodule
369
370
371 module frequdivision #(parameter N = 200000,
372 RST_VLU = 0)(input clk,rst, output reg out);
373   //分频器 N = 100000 ~ 2000000
374 reg [19:0] cnt ;
375 always @(posedge clk) begin
376     if(rst)  cnt <= RST_VLU;
377     else if(cnt == (N-1)) cnt <= 0;
378     else cnt <= cnt + 1;
379 end
380 always @(posedge clk) begin
381      if(rst) out <= 0;
382      else if(cnt == (N-2)) out <= 1;
383      else out <= 0;
384 end
385 endmodule
386
387 module Decoder7Seg(                 //7段译码管
388 input wire [3:0] In,
389 output reg [6:0] Out
390      );
391 always @ (*)
392      begin
393      case(In)
394           4'b0000: Out = 7'b000_0001;
395           4'b0001: Out = 7'b100_1111;
396           4'b0010: Out = 7'b001_0010;
397           4'b0011: Out = 7'b000_0110;
398           4'b0100: Out = 7'b100_1100;
399           4'b0101: Out = 7'b010_0100;
400           4'b0110: Out = 7'b010_0000;
401           4'b0111: Out = 7'b000_1111;
402           4'b1000: Out = 7'b000_0000;
403           4'b1001: Out = 7'b000_0100;
404           4'b1010: Out = 7'b000_1000;   //A
405           4'b1011: Out = 7'b110_0000;   //b
```

```
406             4'b1100: Out = 7'b011_0001;   //C
407             4'b1101: Out = 7'b100_0010;   //d
408             4'b1110: Out = 7'b011_0000;   //E
409             4'b1111: Out = 7'b011_1000;   //F
410         endcase
411         end
412 endmodule
```

– 数据排序

```
1             module DataSort(input clk,rstn, //冒泡法排序
2             input [15:0] d,
3             input bs,
4             input pre,
5             input nxt,
6             input exe,
7             output [6:0] seg,
8             output  [7:0] an,
9             output reg busy,
10            output reg [15:0] delay
11 );
12        parameter s0 = 4'b0000,s0_1=4'b0001,s1= 4'b0010,
13        s1_1=4'b0011,s1_2 = 4'b0100,s1_3 =4'b0101,
14        s1_4 = 4'b0110,s1_5=4'b0111,s1_6=4'b1000,
15        s1_7 = 4'b1001,s2 = 4'b1010;
16
17
18        reg [4:0] RA0,RA1;
19        wire [15:0] RD0,RD1;
20
21        reg [4:0] WA;
22        reg [15:0] WD;
23
24        wire DIo_we;
25
26        wire [4:0] DIo_RA,DIo_WA;
27        wire [15:0] DIo_WD;
28        wire [4:0] Count;
29
30        reg [3:0] CS,NS;
31        reg we;
32        reg [15:0] rd0,rd1;
33
34 //  reg [15:0] ComIn0,ComIn1;
```

```
35  // wire [15:0] ComOut0,ComOut1;
36
37      register_file RF(.clk(clk),.ra0(RA0),.ra1(RA1),
38      .rd0(RD0),.rd1(RD1),.wa(WA),.wd(WD),.we(we));
39      DIOx dio(clk,rstn,d,pre,bs,nxt,RD0,DIo_RA,
40      DIo_WA,DIo_WD,DIo_we,an,seg,Count);
41      // CompareTwo comparetwo(ComIn0,ComIn1,
42      // ComOut0,ComOut1);
43
44      always @(posedge clk,negedge rstn) begin
45          if(!rstn) CS <= s0;
46          else CS <= NS;
47      end
48
49      reg [4:0] Cnt1,Cnt0;
50
51  always @(*) begin
52      case(CS)
53          s0:begin
54              if(exe) NS = s0_1;
55              else NS = s0;
56          end
57          s0_1:begin
58              NS = s1;
59          end
60          s1:begin
61              if(Cnt0 == 0) NS = s2;
62              else  NS = s1_1;
63          end
64          s1_1:begin
65              NS = s1_2;
66          end
67          s1_2:begin
68              NS = s1_3;
69          end
70          s1_3:begin
71              if(Cnt1 == Cnt0-1) begin
72              NS = s1_7;
73              end
74              else   NS = s1_4;
75          end
76          s1_4:begin
```

```
77                  NS = s1_5;
78              end
79              s1_5:begin
80                  NS = s1_6;
81              end
82              s1_6:begin
83                  NS = s1_3;
84              end
85              s1_7:begin
86                  NS = s1;
87              end
88              s2:begin
89                  NS = s2;
90              end
91          endcase
92
93  end
94
95          always @(posedge clk) begin
96          case(NS)
97          s0:begin                    //输入数据阶段
98              RA0 <= DIo_RA;
99              WA <= DIo_WA;
100             WD <= DIo_WD;
101             we <= DIo_we;
102             delay <= 0;
103             busy <= 0;
104         end
105         s0_1:begin
106         Cnt0 <= Count;
107         delay <= delay + 1;
108         end
109         s1:begin                //外层循环
110         Cnt1 <= 0;
111         busy <= 1;
112         we <=0;
113         delay <= delay+1;
114         end
115         s1_1:begin
116         RA0 <= Cnt1;
117         RA1 <= Cnt1+1;
118         we <=0;
```

```
119         delay <= delay+1;
120         end
121         s1_2:begin
122         rd1 <= (RD1 > RD0)?RD0:RD1;          //rd1为较小数
123         rd0 <= (RD0 > RD1)?RD0:RD1;
124         we <=0;
125         delay <= delay+1;
126         end
127         s1_3:begin
128             WA <= Cnt1;
129             WD <= rd1;
130             we <= 1;
131         delay <= delay+1;
132         end
133         s1_4:begin
134         Cnt1 <= Cnt1+1;
135             we <=0;
136         delay <= delay+1;
137         end
138         s1_5:begin
139         RA1 <= Cnt1+1;
140             we <=0;
141         delay <= delay+1;
142         end
143         s1_6:begin
144             rd1 <= (RD1 > rd0)?rd0:RD1;
145             rd0 <= (rd0 > RD1)?rd0:RD1;
146             we <=0;
147             delay <= delay+1;
148         end
149         s1_7:begin
150         Cnt0 <= Cnt0-1;
151             we <=1;
152         delay <= delay+1;
153        WA <= Cnt0 ;
154        WD <= rd0;
155         end
156         s2:begin
157         busy <= 0;
158        RA0 <= DIo_RA;
159        WA <= DIo_WA;
160        WD <= DIo_WD;
```

```
161         we <= 0;
162         end
163
164         endcase
165
166         end
167
168 endmodule
169
170
171
172
173
174
175
176 module DIO(input clk,rstn,
177         input [15:0] d,
178         input pre,
179         input bs,
180         input nxt,
181         output [7:0] an,
182         output [6:0] seg
183 );
184
185         //对输入去抖动并同步化
186         wire DS_bs,DS_pre,DS_nxt;
187         wire [15:0] DS_d;
188         wire [3:0] En_d;
189
190         reg [15:0] DR;
191         reg [4:0] AR;
192         reg [15:0] DReg;
193         reg [15:0] DNxt;
194
195         reg [7:0] Count;
196
197         reg [4:0] wa;
198
199         reg [15:0] wd;
200
201         reg we;
202         wire [15:0] RD;
```

```
203
204     reg [2:0] CS,NS;
205
206     reg [1:0] T;    // {nxt,pre}
207
208     parameter s0 = 3'b000,s1 = 3'b001,s2 = 3'b010,s3 = 3'b011,s4 = 3'b100
209
210     deSyn desyn_bs(clk,rstn,bs,DS_bs);
211     deSyn desyn_pre(clk,rstn,pre,DS_pre);
212     deSyn desyn_nxt(clk,rstn,nxt,DS_nxt);
213
214     deSyn desyn_d0(clk,rstn,d[0],DS_d[0]);
215     deSyn desyn_d1(clk,rstn,d[1],DS_d[1]);
216     deSyn desyn_d2(clk,rstn,d[2],DS_d[2]);
217     deSyn desyn_d3(clk,rstn,d[3],DS_d[3]);
218     deSyn desyn_d4(clk,rstn,d[4],DS_d[4]);
219     deSyn desyn_d5(clk,rstn,d[5],DS_d[5]);
220     deSyn desyn_d6(clk,rstn,d[6],DS_d[6]);
221     deSyn desyn_d7(clk,rstn,d[7],DS_d[7]);
222     deSyn desyn_d8(clk,rstn,d[8],DS_d[8]);
223     deSyn desyn_d9(clk,rstn,d[9],DS_d[9]);
224     deSyn desyn_d10(clk,rstn,d[10],DS_d[10]);
225     deSyn desyn_d11(clk,rstn,d[11],DS_d[11]);
226     deSyn desyn_d12(clk,rstn,d[12],DS_d[12]);
227     deSyn desyn_d13(clk,rstn,d[13],DS_d[13]);
228     deSyn desyn_d14(clk,rstn,d[14],DS_d[14]);
229     deSyn desyn_d15(clk,rstn,d[15],DS_d[15]);
230
231
232
233
234 encode ECD(DNxt,En_d);
235
236 register_file RF(.clk(clk),.ra0(AR),.rd0(RD),
237 .wa(AR),.wd(DR),.we(we));
238
239 dynimage DIS(clk,rstn,Count,DR,an,seg);
240
241 always @(posedge clk,negedge rstn) begin
242     if(!rstn)
243     CS <= s0;
244     else CS <= NS;
```

```
245
246     end
247
248     /*
249     s0:默认状态
250     s1:读出数字并显示
251     s2:输入数字
252     s3:保存数字
253     s4:退格
254     s5:显示上一个数字或下一个数字
255     */
256
257     always @(*) begin
258     case(CS)
259         s0: NS = s1;
260         s1: NS = s2;
261         s2: begin
262         if(DS_nxt || DS_pre)  NS = s5;
263         else if(bs) NS = s4;
264         else if(DNxt) NS = s3;
265         else NS = s2;
266         end
267         s3: NS = s2;
268         s4: NS = s2;
269         s5: NS = s1;
270     endcase
271     end
272
273
274     always @(posedge clk,negedge rstn) begin
275         if(!rstn) begin
276             DNxt <= 0;
277             DReg <= 0;
278             AR <= 0;
279             DR <= 0;
280             Count <= 0;
281             we <= 0;
282         end
283         else begin
284             DNxt <= DS_d - DReg;
285             DReg <= DS_d;
286             case(CS)
```

```
287                     s0:begin
288                         AR <= 0;
289                         DR <= 0;
290                         we <= 0;
291                     end
292                     s1:begin
293                         DR <= RD;
294                         we <= 0;
295                     end
296                     s2:begin
297                         T <= {DS_nxt,DS_pre};
298                         we <= 0;
299                     end
300                     s3:begin
301                         DR <= (DR<<4)+En_d;
302                         we <= 1;
303                     end
304                     s4:begin
305                         DR <=  DR >> 4;
306                         we <= 1;
307                     end
308                     s5:begin
309                         we = 0;
310                         if(T[1]) begin
311                             AR <= AR +1;
312                             Count <= Count +1;
313                         end
314                         if(T[0]) begin
315                             AR <= AR -1;
316                             Count <= Count -1;
317                         end
318                     end
319                 endcase
320             end
321         end
322
323 endmodule
324
325 module DIOx(input clk,rstn,
326             input [15:0] d,
327             input pre,
328             input bs,
```

```
329                 input  nxt,
330                 input   [15:0] RD,
331                 output reg [4:0] RA,WA,
332                 output reg [15:0] WD,
333                 output reg we,
334                 output  [7:0] an,
335                 output  [6:0] seg,
336                 output  [4:0] Countx
337  );
338 //对输入去抖动并同步化
339 wire DS_bs,DS_pre,DS_nxt;
340 wire [15:0] DS_d;
341 wire [3:0] En_d;
342 
343 reg [7:0] Count;
344 
345 reg [15:0] DReg,DNxt;
346 
347 reg [2:0] CS,NS;
348 
349 reg [1:0] T;
350 
351 parameter s0 = 3'b000,s1 = 3'b001,s2 = 3'b010,
352 s3 = 3'b011,s4 = 3'b100,s5 = 3'b101;
353 
354 deSyn desyn_bs(clk,rstn,bs,DS_bs);
355 deSyn desyn_pre(clk,rstn,pre,DS_pre);
356 deSyn desyn_nxt(clk,rstn,nxt,DS_nxt);
357 
358 deSyn desyn_d0(clk,rstn,d[0],DS_d[0]);
359 deSyn desyn_d1(clk,rstn,d[1],DS_d[1]);
360 deSyn desyn_d2(clk,rstn,d[2],DS_d[2]);
361 deSyn desyn_d3(clk,rstn,d[3],DS_d[3]);
362 deSyn desyn_d4(clk,rstn,d[4],DS_d[4]);
363 deSyn desyn_d5(clk,rstn,d[5],DS_d[5]);
364 deSyn desyn_d6(clk,rstn,d[6],DS_d[6]);
365 deSyn desyn_d7(clk,rstn,d[7],DS_d[7]);
366 deSyn desyn_d8(clk,rstn,d[8],DS_d[8]);
367 deSyn desyn_d9(clk,rstn,d[9],DS_d[9]);
368 deSyn desyn_d10(clk,rstn,d[10],DS_d[10]);
369 deSyn desyn_d11(clk,rstn,d[11],DS_d[11]);
370 deSyn desyn_d12(clk,rstn,d[12],DS_d[12]);
```

```
371 deSyn desyn_d13(clk,rstn,d[13],DS_d[13]);
372 deSyn desyn_d14(clk,rstn,d[14],DS_d[14]);
373 deSyn desyn_d15(clk,rstn,d[15],DS_d[15]);
374
375 assign Countx = Count[4:0];
376
377 encode ECD(DNxt,En_d);
378
379 dynimage DIS(clk,rstn,Count,WD,an,seg);
380
381 always @(posedge clk,negedge rstn) begin
382     if(!rstn)
383         CS <= s0;
384     else CS <= NS;
385
386 end
387
388 always @(*) begin
389     case(CS)
390     s0: NS = s1;
391     s1: NS = s2;
392     s2: begin
393       if(T[1]||T[0])  NS = s5;
394       else if(DS_bs) NS = s4;
395       else if(DNxt) NS = s3;
396       else NS = s2;
397     end
398     s3: NS = s2;
399     s4: NS = s2;
400     s5: NS = s1;
401     endcase
402 end
403
404 always @(posedge clk,negedge rstn) begin
405     if(!rstn) begin
406       DNxt <= 0;
407       DReg <= 0;
408       RA <= 0;
409       Count <= 0;
410       we <= 0;
411     end
412     else begin
```

```
413         DNxt <= DS_d - DReg;
414         DReg <= DS_d;
415         WA <= RA;
416         case(NS)
417         s1:begin
418               we <= 0;
419         end
420         s2:begin
421             WD <= RD;
422             T <= {DS_nxt,DS_pre};
423             we <= 0;
424         end
425         s3:begin
426               WD <= (WD<<4)+En_d;
427               we <= 1;
428         end
429         s4:begin
430             WD <= WD >> 4;
431             we <= 1;
432         end
433         s5:begin
434             we = 0;
435            if(T[1]) begin
436              RA <= RA +1;
437               Count <= Count +1;
438            end
439            if(T[0]) begin
440              RA <= RA -1;
441               Count <= Count -1;
442            end
443         end
444         endcase
445     end
446 end
447
448
449 endmodule
450
451 module deSyn(input clk,rstn,input x,output y);
452 wire de_x;
453 debounce DB(clk,rstn,x,de_x);
454 Synchron SYN(de_x,clk,rstn,y);
```

```
455 endmodule
456
457 module debounce(input clk,rstn,
458                 input x,
459                 output reg y);
460 parameter  Cnt0 = 3'b000, Cnt1 = 3'b001,
461 Assign0 = 3'b010,Assign1=3'b011,Default = 3'b111;
462 //刚开始进入默认状态，两个计数状态，两个赋值状态
463 reg [2:0] CS,NS;
464 reg [19:0] Count;
465 reg y_reg;
466 always @(posedge clk,negedge rstn) begin //异步复位
467     if(!rstn)begin
468       CS <= Default;
469       y_reg <= 0;
470     end
471     else begin
472       CS <= NS;
473     y_reg <= y;
474     end
475 end
476
477 always @(posedge clk,negedge rstn) begin
478   if(!rstn) Count <=0;
479   else begin
480     case(CS)
481         Default: Count <= 0;
482     Cnt0 :begin
483       Count <= Count+1;
484     end
485     Cnt1: Count <= Count+1;
486     Assign0: Count <= 0;
487     Assign1: Count <= 0;
488
489   endcase
490   end
491
492
493 end
494
495 always @(*) begin
496     y = y_reg;
```

```
497     NS = Default;
498     case (CS)
499     Default: begin
500       y = 0;
501       if(x == 1) NS = Cnt1;
502       else NS = Cnt0;
503     end
504     Cnt0:begin
505         if(Count == 100000) begin
506           NS = Assign0;
507         end
508         else if(x == 0)begin
509           NS = Cnt0;
510         end
511         else begin
512           NS = Default;
513         end
514     end
515     Cnt1:begin
516         if(Count == 100000) begin
517           NS = Assign1;
518         end
519         else if(x == 1)begin
520           NS = Cnt1;
521         end
522         else begin
523           NS = Default;
524         end
525     end
526     Assign0: begin
527             y = 0;
528       if(x == 0)
529         NS = Assign0;
530       else
531           NS = Default;   //只有极个别才在此刻为1
532     end
533     Assign1:begin
534       y = 1;
535       if(x == 1) NS = Assign1;
536       else NS = Default; //只有极个别才在此刻为1
537     end
538     endcase
```

```
end
endmodule

module Synchron(input x,input clk,rstn,
                output y);
reg s1,s2;
reg s;

always @(posedge clk) begin
    if(~rstn ) begin
      s1 <= 0;
      s2 <= 0;
      s <= 0;
    end
    else begin
      s1 <= x;
      s2 <= s1;
      s <= s2;
    end
end
assign y = (!s)&s2;
endmodule




module  register_file # (
    parameter AW = 5,           //地址宽度
    parameter DW = 16           //数据宽度
)(
    input clk,                  //时钟
    input [AW-1:0] ra0, ra1,            //读地址
    output [DW-1:0] rd0, rd1,   //读数据
    input [AW-1:0] wa,          //写地址
    input [DW-1:0] wd,          //写数据
    input we                    //写使能
);
    reg [DW-1:0] rf [0: (1<<AW)-1];             //寄存器堆
    assign rd0 = rf[ra0], rd1 = rf[ra1];        //读操作

    always @(posedge clk)
```

```
581         if (we)   rf[wa]  <=  wd;                    //写操作
582 endmodule
583
584 module encode(input [15:0] In,output reg [3:0] Out );
585     always @(*) begin
586         case(In)
587         16'b0000_0000_0000_0001: Out = 4'b0000;
588         16'b0000_0000_0000_0010: Out = 4'b0001;
589         16'b0000_0000_0000_0100: Out = 4'b0010;
590         16'b0000_0000_0000_1000: Out = 4'b0011;
591         16'b0000_0000_0001_0000: Out = 4'b0100;
592         16'b0000_0000_0010_0000: Out = 4'b0101;
593         16'b0000_0000_0100_0000: Out = 4'b0110;
594         16'b0000_0000_1000_0000: Out = 4'b0111;
595         16'b0000_0001_0000_0000: Out = 4'b1000;
596         16'b0000_0010_0000_0000: Out = 4'b1001;
597         16'b0000_0100_0000_0000: Out = 4'b1010;
598         16'b0000_1000_0000_0000: Out = 4'b1011;
599         16'b0001_0000_0000_0000: Out = 4'b1100;
600         16'b0010_0000_0000_0000: Out = 4'b1101;
601         16'b0100_0000_0000_0000: Out = 4'b1110;
602         16'b1000_0000_0000_0000: Out = 4'b1111;
603         endcase
604     end
605
606 endmodule
607
608 module dynimage(
609                 input clk,rstn,
610                 input [7:0] Cnt,
611                 input [15:0] d,
612                 output reg [7:0] an,
613                 output reg [6:0] seg);
614
615                 wire clkd;          //分频时钟
616                 reg [3:0] DIn;
617                 wire [6:0] DOut;
618         frequdivision  FreDivClk(clk,~rstn,clkd);
619         Decoder7Seg decoder(DIn,DOut);
620
621 parameter s0 = 3'b000, s1 = 3'b001,s2 = 3'b010,
622 s3 = 3'b011,s4 = 3'b100,s5 = 3'b101;
```

```
623 reg [2:0] CS,NS;
624
625
626 always @(posedge clkd,negedge rstn) begin  //异步初始化
627     if(~rstn) CS <= s0;
628     else CS <= NS;
629 end
630
631 always @(*) begin
632       case(CS)
633       s0 : begin
634            DIn = d[3:0];
635            an = 8'b1111_1110;
636            seg = DOut;
637            NS = s1;
638       end
639       s1 : begin
640         DIn = d[7:4];
641         seg = DOut;
642         NS = s2;
643         an = 8'b1111_1101;
644       end
645       s2 : begin
646         DIn = d[11:8];
647         seg = DOut;
648         NS = s3;
649         an = 8'b1111_1011;
650
651       end
652       s3 : begin
653         DIn = d[15:12];
654         seg = DOut;
655         NS = s4;
656         an = 8'b1111_0111;
657       end
658       s4 :begin
659         DIn = Cnt[3:0];
660         seg = DOut;
661         NS = s5;
662         an = 8'b1011_1111;
663       end
664       s5 :begin
```

```
665             DIn = Cnt[7:4];
666             seg = DOut;
667             NS = s0;
668             an = 8'b0111_1111;
669         end
670         endcase
671 end
672 
673 
674 endmodule
675 
676 module frequdivision #(parameter N = 200000,RST_VLU = 0)
677 (input clk,rst, output reg out);
678  //分频器 N = 100000 ~ 2000000
679 reg [19:0] cnt ;
680 always @(posedge clk) begin
681    if(rst)  cnt <= RST_VLU;
682    else if(cnt == (N-1)) cnt <= 0;
683    else cnt <= cnt + 1;
684 end
685 always @(posedge clk) begin
686     if(rst) out <= 0;
687     else if(cnt == (N-2)) out <= 1;
688     else out <= 0;
689 end
690 endmodule
691 
692 module Decoder7Seg(                 //7段译码管
693 input wire [3:0] In,
694 output reg [6:0] Out
695     );
696 always @ (*)
697     begin
698     case(In)
699         4'b0000: Out = 7'b000_0001;
700         4'b0001: Out = 7'b100_1111;
701         4'b0010: Out = 7'b001_0010;
702         4'b0011: Out = 7'b000_0110;
703         4'b0100: Out = 7'b100_1100;
704         4'b0101: Out = 7'b010_0100;
705         4'b0110: Out = 7'b010_0000;
706         4'b0111: Out = 7'b000_1111;
```

```
        4'b1000: Out = 7'b000_0000;
        4'b1001: Out = 7'b000_0100;
        4'b1010: Out = 7'b000_1000;   //A
        4'b1011: Out = 7'b110_0000;   //b
        4'b1100: Out = 7'b011_0001;   //C
        4'b1101: Out = 7'b100_0010;   //d
        4'b1110: Out = 7'b011_0000;   //E
        4'b1111: Out = 7'b011_1000;   //F
    endcase
    end
endmodule


// module CompareTwo(input [15:0] In0,In1,
//                   output reg [15:0] Out0,Out1
//                     );
// always @(*) begin
//           if(In0 > In1) begin
//             Out0 = In0;
//             Out1 = Out1;
//           end
//           else begin
//             Out0 = In1;
//             Out1 = In0;
//           end
//         end

// endmodule
```

- 完成 8 个数码管显示空白分隔的 2 个 16 进制数的下载、测试及电路图、使用资源查看
  - 代码如下:

```
    module display2num(input clk,input rstn,
    input [7:0] a,d,
    output reg [7:0] an,
    output reg [6:0] seg);
    wire clkd;
    reg [3:0] DIn;
    wire [6:0] DOut;

    frequdivision  FreDivClk(clk,~rstn,clkd);
    Decoder7Seg decoder(DIn,DOut);
parameter s0 = 2'b00,s1 = 2'b01,s2=2'b10,s3=2'b11;
```

```
12 reg [1:0] CS,NS;
13 always @(posedge clkd,negedge rstn) begin
14     if(!rstn) CS <= s0;
15     else    CS <= NS;
16 end
17 always @(*) begin
18     case (CS)
19         s0: NS = s1;
20         s1: NS = s2;
21         s2: NS = s3;
22         s3: NS = s0;
23     endcase
24 end
25 always @(*) begin
26       case(CS)
27       s0:begin
28         DIn = d[3:0];
29         seg = DOut;
30         an = 8'b11111110;
31       end
32       s1:begin
33         DIn = d[7:4];
34         seg = DOut;
35         an = 8'b11111101;
36       end
37       s2:begin
38         DIn = a[3:0];
39         seg = DOut;
40         an = 8'b10111111;
41       end
42       s3:begin
43         DIn = a[7:4];
44         seg = DOut;
45         an = 8'b01111111;
46       end
47       endcase
48 end
49
50 endmodule
51
52 module frequdivision #(parameter N = 200000,RST_VLU = 0)
53 (input clk,rst,output reg out);
```

```
 //分频器 N = 100000 ~ 2000000
reg [19:0] cnt ;
always @(posedge clk) begin
    if(rst)  cnt <= RST_VLU;
    else if(cnt == (N-1)) cnt <= 0;
    else cnt <= cnt + 1;
end
always @(posedge clk) begin
     if(rst) out <= 0;
     else if(cnt == (N-2)) out <= 1;
     else out <= 0;
end
endmodule

module Decoder7Seg(                 //7段译码管
input wire [3:0] In,
output reg [6:0] Out
     );
always @ (*)
     begin
     case(In)
          4'b0000: Out = 7'b000_0001;
          4'b0001: Out = 7'b100_1111;
          4'b0010: Out = 7'b001_0010;
          4'b0011: Out = 7'b000_0110;
          4'b0100: Out = 7'b100_1100;
          4'b0101: Out = 7'b010_0100;
          4'b0110: Out = 7'b010_0000;
          4'b0111: Out = 7'b000_1111;
          4'b1000: Out = 7'b000_0000;
          4'b1001: Out = 7'b000_0100;
          4'b1010: Out = 7'b000_1000;   //A
          4'b1011: Out = 7'b110_0000;   //b
          4'b1100: Out = 7'b011_0001;   //C
          4'b1101: Out = 7'b100_0010;   //d
          4'b1110: Out = 7'b011_0000;   //E
          4'b1111: Out = 7'b011_1000;   //F
     endcase
     end
endmodule
```

– 下载测试

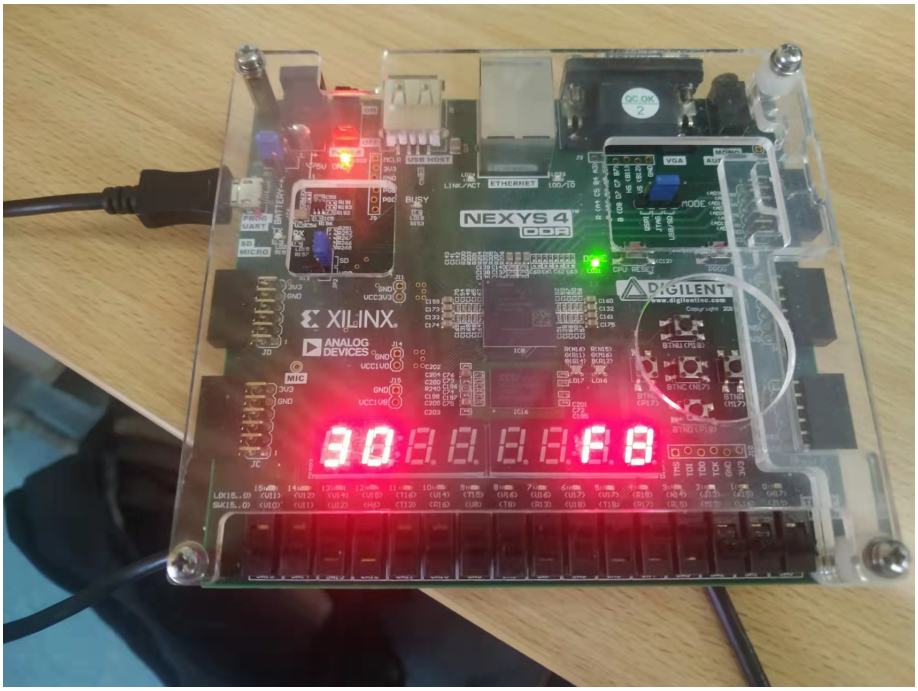

图 2: 8 个数码管显示空白分隔的 2 个 16 进制数下载测试

– 查看 RTL 分析和综合后电路图

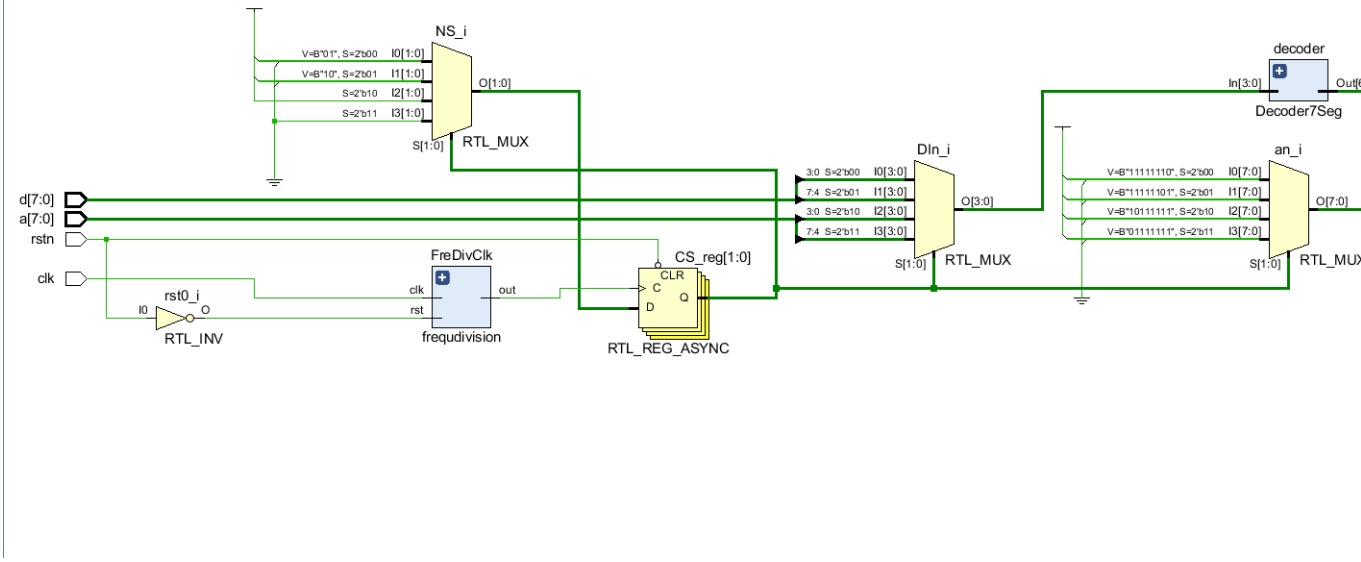

图 3: 8 个数码管显示空白分隔的 2 个 16 进制数 RTL 分析后电路图

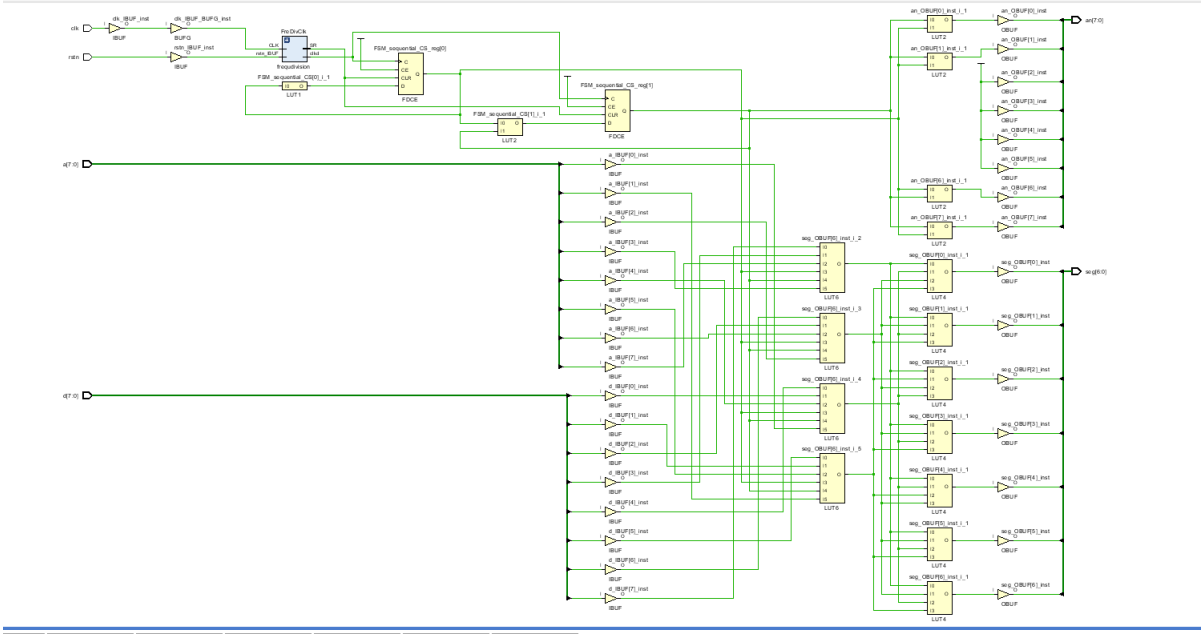

图 4: 8 个数码管显示空白分隔的 2 个 16 进制数综合后电路图

– 查看综合后电路资源使用情况

Hierarchy

Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
display2num	37	23	33	1
FreDivClk (frequdivision)	26	21	0	0

图 5: 8 个数码管显示空白分隔的 2 个 16 进制数综合后电路资源使用情况

- 完成一组 16 进制数的输入输出的下载、测试及电路、使用资源的查看

  – 下载测试

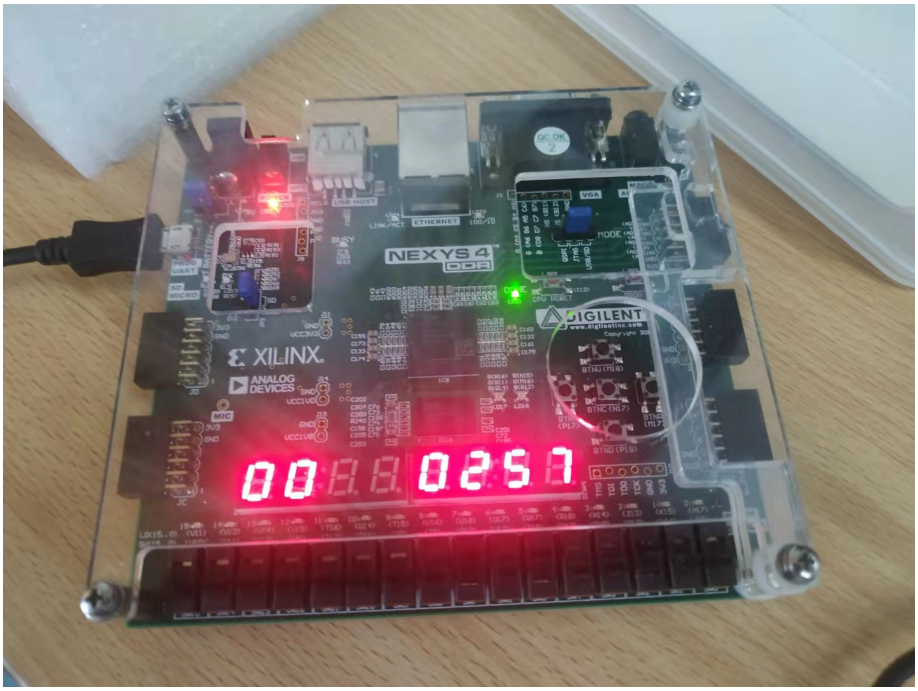

图 6: 开关输入抖动验证下载测试

– 查看 RTL 分析和综合后电路图

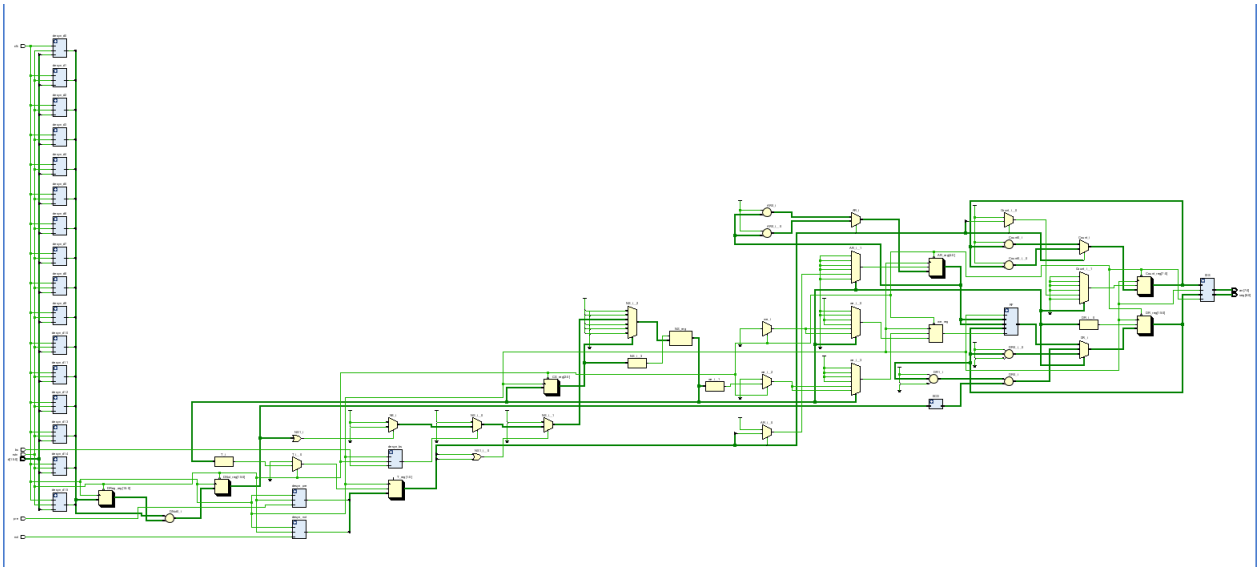

图 7: 开关输入抖动验证 RTL 分析后电路图

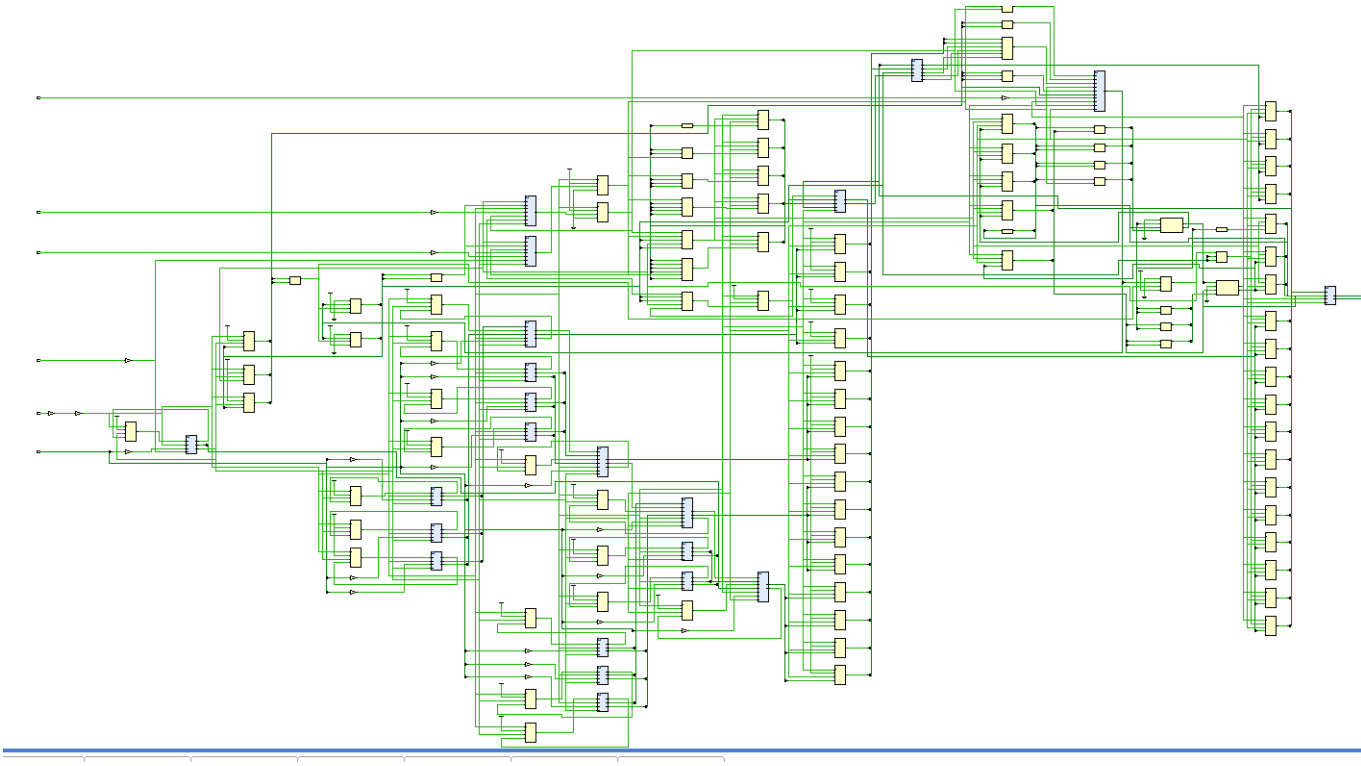

图 8: 开关输入抖动验证综合后电路图

– 查看综合后电路资源使用情况

Hierarchy

Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
DIO	525	631	36	1
desyn_bs (deSyn)	24	27	0	0
desyn_d0 (deSyn_0)	22	27	0	0
desyn_d1 (deSyn_1)	22	27	0	0
desyn_d2 (deSyn_8)	22	27	0	0
desyn_d3 (deSyn_9)	22	27	0	0
desyn_d4 (deSyn_10)	22	27	0	0
desyn_d5 (deSyn_11)	22	27	0	0
desyn_d6 (deSyn_12)	22	27	0	0
desyn_d7 (deSyn_13)	22	27	0	0
desyn_d8 (deSyn_14)	22	27	0	0
desyn_d9 (deSyn_15)	22	27	0	0
desyn_d10 (deSyn_2)	22	27	0	0
desyn_d11 (deSyn_3)	22	27	0	0
desyn_d12 (deSyn_4)	22	27	0	0
desyn_d13 (deSyn_5)	22	27	0	0
desyn_d14 (deSyn_6)	22	27	0	0
desyn_d15 (deSyn_7)	23	27	0	0
desyn_nxt (deSyn_16)	21	27	0	0
desyn_pre (deSyn_17)	21	27	0	0
DIS (dynimage)	42	44	0	0
ECD (encode)	18	4	0	0
RF (register_file)	28	0	0	0

图 9: 开关输入抖动验证综合后电路资源使用情况

- 完成一组数据的输入、排序的下载、测试及电路图、使用资源查看
  - 仿真与下载测试

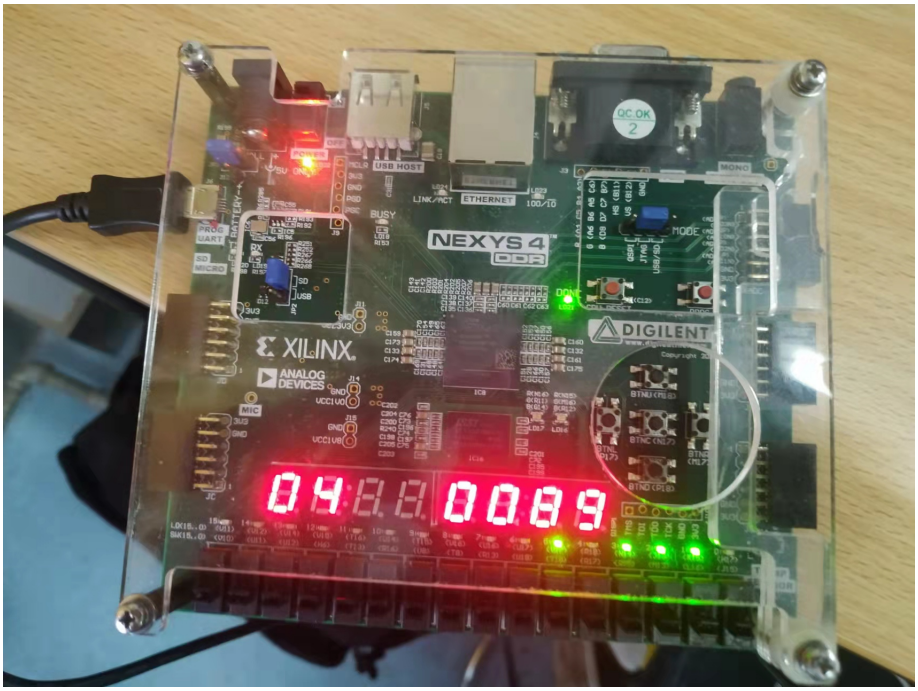

图 10: 开关输入去抖动后动态显示下载测试

– 查看 RTL 分析和综合后电路图

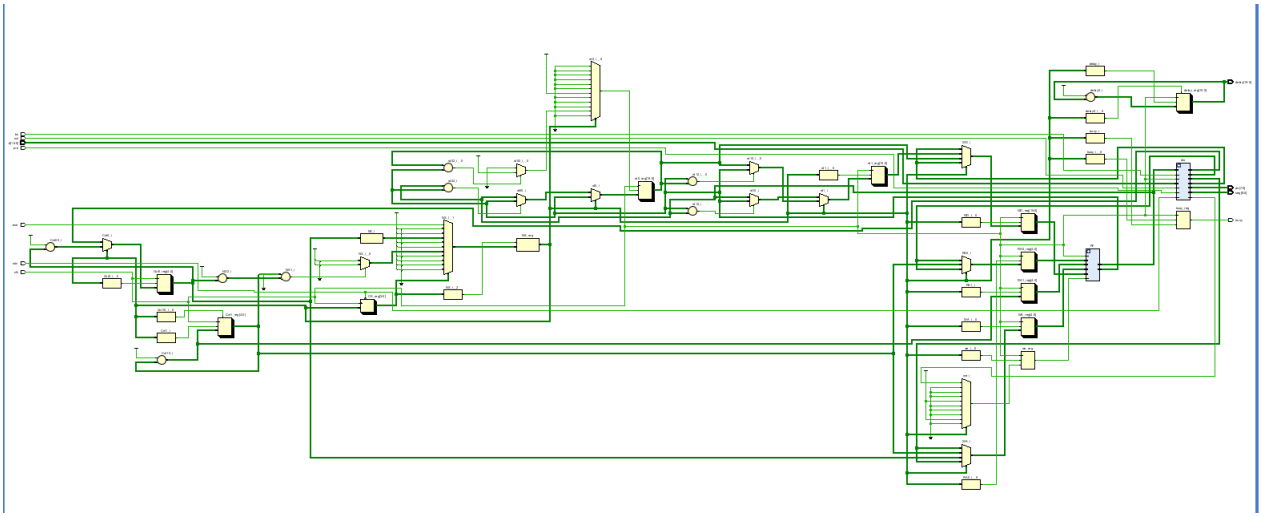

图 11: 开关输入去抖动后动态显示 RTL 分析后电路图

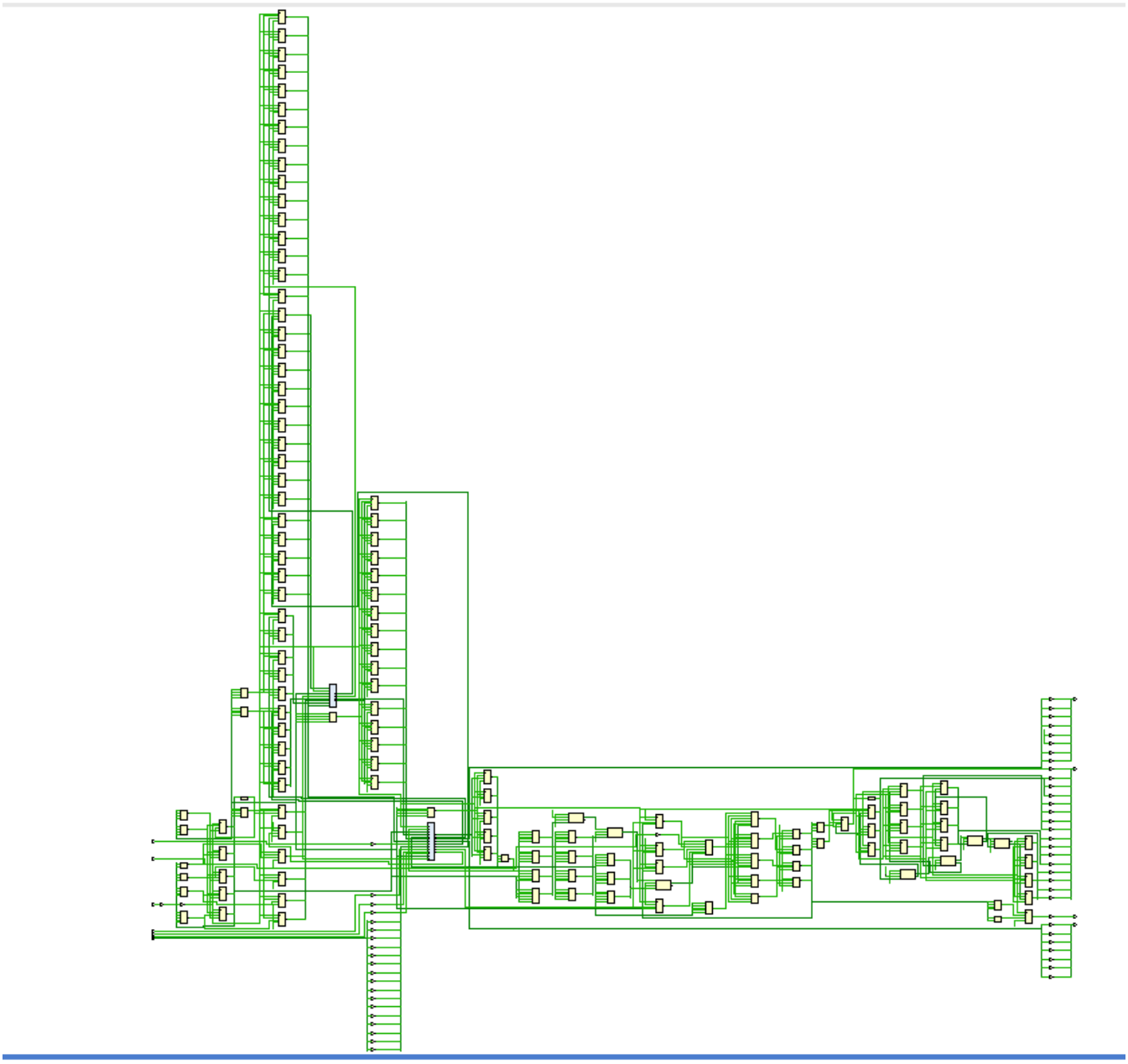

图 12: 开关输入去抖动后动态显示综合后电路图

– 查看综合后电路资源使用情况

Name 1	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
DataSort	655	735	54	1
dio (DIOx)	532	636	0	0
desyn_bs (deSy	24	27	0	0
desyn_d0 (deSy	23	27	0	0
desyn_d1 (deSy	22	27	0	0
desyn_d2 (deSy	22	27	0	0
desyn_d3 (deSy	22	27	0	0
desyn_d4 (deSy	22	27	0	0
desyn_d5 (deSy	22	27	0	0
desyn_d6 (deSy	22	27	0	0
desyn_d7 (deSy	22	27	0	0
desyn_d8 (deSy	22	27	0	0
desyn_d9 (deSy	22	27	0	0
desyn_d10 (deS	22	27	0	0
desyn_d11 (deS	22	27	0	0
desyn_d12 (deS	22	27	0	0
desyn_d13 (deS	22	27	0	0
desyn_d14 (deS	22	27	0	0
desyn_d15 (deS	22	27	0	0
desyn_nxt (deS	21	27	0	0
desyn_pre (deS	21	27	0	0
DIS (dynimage)	38	44	0	0
ECD (encode)	16	4	0	0
RF (register_file)	91	0	0	0

图 13: 开关输入去抖动后动态显示综合后电路资源使用情况

【总结与思考】

- 总结:
  - 通过此次实验，更加熟悉使用状态机实现所需的电路，熟悉时序部分与逻辑部分的区别
- 思考:
  - 在此次排序的实现过程中，未考虑实现排序的最优算法，以及如何更加提升效率
  - 在本次实验中，曾发生对状态的转化条件及要做的事错位（后移一个状态），导致出错
  - 在使用 Vivado 过程中，对其中的一些报错信息，警告信息不太理解，需要加强这方面的经验