徐震 PB20051102 序列检测器设计

实验内容

利用有限状态机的原理设计序列检测器和两个序列产生器，中间通过二选一选择器相连，分别产生一个有效序列和一个无效序列，通过切换选择器的选择信号实现被检测序列的切换，检测到有效序列时输出高电平脉冲。

设计分析

用有限状态机的原理设计序列检测器和产生器，使用三 always 结构和 Moore 状态机模式，

源码

```
module seq(sel,RESET,CLK,detector_out);//顶层实体
   input RESET,CLK,sel;
   output detector_out;
   wire SEQ_EX,SEQ_IN,DATA_DET,CLK6D;

   SIXFD U0(CLK,CLK6D);
   SEQ_GENERATE_EXCLUDE U1(CLK6D,RESET,SEQ_EX);
   SEQ_GENERATE_INCLUDE U2(CLK6D,RESET,SEQ_IN);
   SELECT U3(sel,SEQ_EX,SEQ_IN,DATA_DET);
   SEQ_DETECT U4(CLK6D,RESET,DATA_DET,detector_out);

endmodule
```

```
module SIXFD(clk,clk6d);//分频器
  input clk;
  output clk6d;
  reg clk6d,clk6d1,clk6d2;
  reg [499:0] cnt1,cnt2,cnt;

  always@(posedge clk)begin
  if(cnt1==10'd499)begin cnt1<=10'd000;clk6d1=~clk6d1;end
  else cnt1<=cnt1+10'd001;
  end

  always@(posedge clk6d1)begin
  if(cnt2==10'd499)begin cnt2<=10'd000;clk6d2=~clk6d2;end
  else cnt2<=cnt2+10'd001;
  end

  always@(posedge clk6d2)begin
  if(cnt==10'd009)begin cnt<=10'd000;clk6d=~clk6d;end
  else cnt<=cnt+10'd001;
  end

endmodule
```

```
module SEQ_GENERATE_EXCLUDE(CLK,RESET,SEQ_EXCLUDE);//序列产生器，无效
   input CLK,RESET;
   output SEQ_EXCLUDE;
   reg SEQ_EXCLUDE;
   reg [2:0]current_st,next_st;
   parameter S0=3'b000,S1=3'b001,S2=3'b010,S3=3'b011,S4=3'b100,S5=3'b101,S6=3'b110,S7=3'b111;

   always@(posedge CLK or negedge RESET)begin
      if(!RESET)current_st<=S0;
      else current_st<=next_st;
   end

   always@(current_st)begin
      case(current_st)
         S0: next_st<=S1;
         S1: next_st<=S2;
         S2: next_st<=S3;
         S3: next_st<=S4;
         S4: next_st<=S5;
         S5: next_st<=S6;
         S6: next_st<=S7;
         S7: next_st<=S0;
         default: next_st<=S0;
      endcase
   end

   always@(current_st)begin
      case(current_st)
         S0: SEQ_EXCLUDE<=1'b1;
         S1: SEQ_EXCLUDE<=1'b1;
         S2: SEQ_EXCLUDE<=1'b0;
         S3: SEQ_EXCLUDE<=1'b0;
         S4: SEQ_EXCLUDE<=1'b0;
         S5: SEQ_EXCLUDE<=1'b1;
         S6: SEQ_EXCLUDE<=1'b1;
         S7: SEQ_EXCLUDE<=1'b0;
         default: SEQ_EXCLUDE<=1'b0;
      endcase
   end

endmodule
```

```
module SEQ_GENERATE_INCLUDE(CLK,RESET,SEQ_IN);//序列产生器，有效
   input CLK,RESET;
   output SEQ_IN;
   reg SEQ_IN;
   reg [3:0]current_st,next_st;
   parameter S0=4'b0000,S1=4'b0001,S2=4'b0010,S3=4'b0011,S4=4'b0100,S5=4'b0101,S6=4'b0110,S7=4'b0111,S8=4'b1000,S9=4'b1001;

   always@(posedge CLK or negedge RESET)begin
      if(!RESET)current_st<=S0;
      else current_st<=next_st;
   end

   always@(current_st)begin
      case(current_st)
         S0: next_st<=S1;
         S1: next_st<=S2;
         S2: next_st<=S3;
         S3: next_st<=S4;
         S4: next_st<=S5;
         S5: next_st<=S6;
         S6: next_st<=S7;
         S7: next_st<=S8;
         S8: next_st<=S9;
         S9: next_st<=S0;
         default: next_st<=S0;
      endcase
   end

   always@(current_st)begin
      case(current_st)
         S0: SEQ_IN<=1'b1;
         S1: SEQ_IN<=1'b1;
         S2: SEQ_IN<=1'b1;
         S3: SEQ_IN<=1'b0;
         S4: SEQ_IN<=1'b1;
         S5: SEQ_IN<=1'b0;
         S6: SEQ_IN<=1'b0;
         S7: SEQ_IN<=1'b1;
         S8: SEQ_IN<=1'b1;
         S9: SEQ_IN<=1'b0;
         default: SEQ_IN<=1'b0;
      endcase
   end

endmodule
```

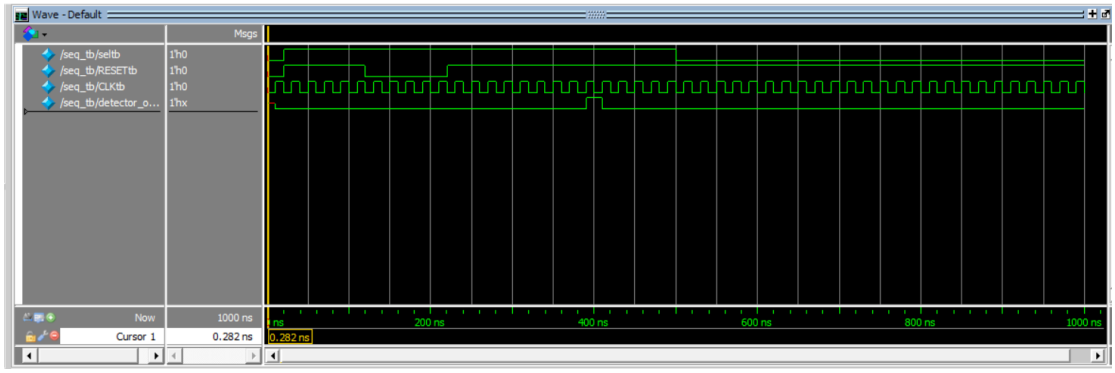
```
module SELECT(sel,SEQ_EX,SEQ_IN,DATA_DET);//选择器
   input sel,SEQ_EX,SEQ_IN;
   output DATA_DET;
   reg DATA_DET;

   always@(sel or SEQ_EX or SEQ_IN)begin
      case(sel)
         1'b1: DATA_DET<=SEQ_IN;
         1'b0: DATA_DET<=SEQ_EX;
      endcase
   end
endmodule
```

```
module SEQ_DETECT(CLK,RESET,datainput,detector_out);//检测器
   input CLK,datainput,RESET;
   output detector_out;
   reg detector_out;
   reg [3:0] current_st,next_st;
   parameter S0=4'b0000,S1=4'b0001,S2=4'b0010,S3=4'b0011,S4=4'b0100,S5=4'b0101,S6=4'b0110,S7=4'b0111,S8=4'b1000,S9=4'b1001;

   always@(posedge CLK)begin
      if(!RESET)current_st<=S0;
      else current_st<=next_st;
   end

   always@(current_st,datainput)begin
      case(current_st)
         S0: begin
             if(datainput==1'b1)next_st<=S1;
             else next_st<=S0;end
         S1: begin
             if(datainput==1'b1)next_st<=S2;
             else next_st<=S0;end
         S2: begin
             if(datainput==1'b1)next_st<=S3;
             else next_st<=S0;end
         S3: begin
             if(datainput==1'b0)next_st<=S4;
             else next_st<=S3;end
         S4: begin
             if(datainput==1'b1)next_st<=S5;
             else next_st<=S0;end
         S5: begin
             if(datainput==1'b0)next_st<=S6;
             else next_st<=S2;end
         S6: begin
             if(datainput==1'b0)next st<=S7;
             else next_st<=S1;end
         S7: begin
             if(datainput==1'b1)next_st<=S8;
             else next_st<=S0;end
         S8: begin
             if(datainput==1'b1)next_st<=S9;
             else next_st<=S0;end
         S9: begin
             if(datainput==1'b0)next_st<=S0;
             else next_st<=S1;end
        default: next_st<=S0;
      endcase
   end

   always@(current_st)begin
      if(current_st==S9)detector_out<=1'b1;
      else detector_out<=1'b0;
   end

endmodule
```

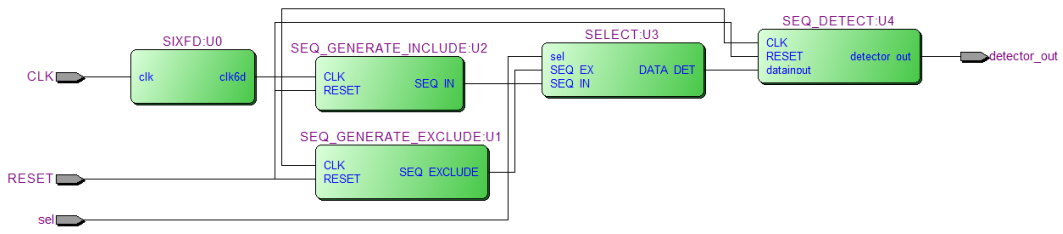
仿真

```
module seq(sel,RESET,CLK,detector_out);//顶层实体
   input RESET,CLK,sel;
   output detector_out;
   wire SEQ_EX,SEQ_IN,DATA_DET;//CLK6D;

   //SIXFD U0(CLK,CLK6D);
   SEQ_GENERATE_EXCLUDE U1(CLK,RESET,SEQ_EX);
   SEQ_GENERATE_INCLUDE U2(CLK,RESET,SEQ_IN);
   SELECT U3(sel,SEQ_EX,SEQ_IN,DATA_DET);
   SEQ_DETECT U4(CLK,RESET,DATA_DET,detector_out);

endmodule
```

```
1  `timescale 10ns/1ns
2  module seq_tb();
3     reg seltb,RESETtb,CLKtb;
4     wire detector_out_tb;
5
6     initial begin
7     CLKtb<=1'b0;
8     end
9     always #1 CLKtb = ~CLKtb;
10
11    initial begin
12    RESETtb<=1'b0;
13    #2 RESETtb<=1'b1;
14    #10 RESETtb<=1'b0;
15    #10 RESETtb<=1'b1;
16    end
17
18    initial begin
19    seltb<=1'b0;
20    #2 seltb<=1'b1;
21    #48 seltb<=1'b0;
22    end
23
24    seq U0(seltb,RESETtb,CLKtb,detector_out_tb);
25 endmodule
```

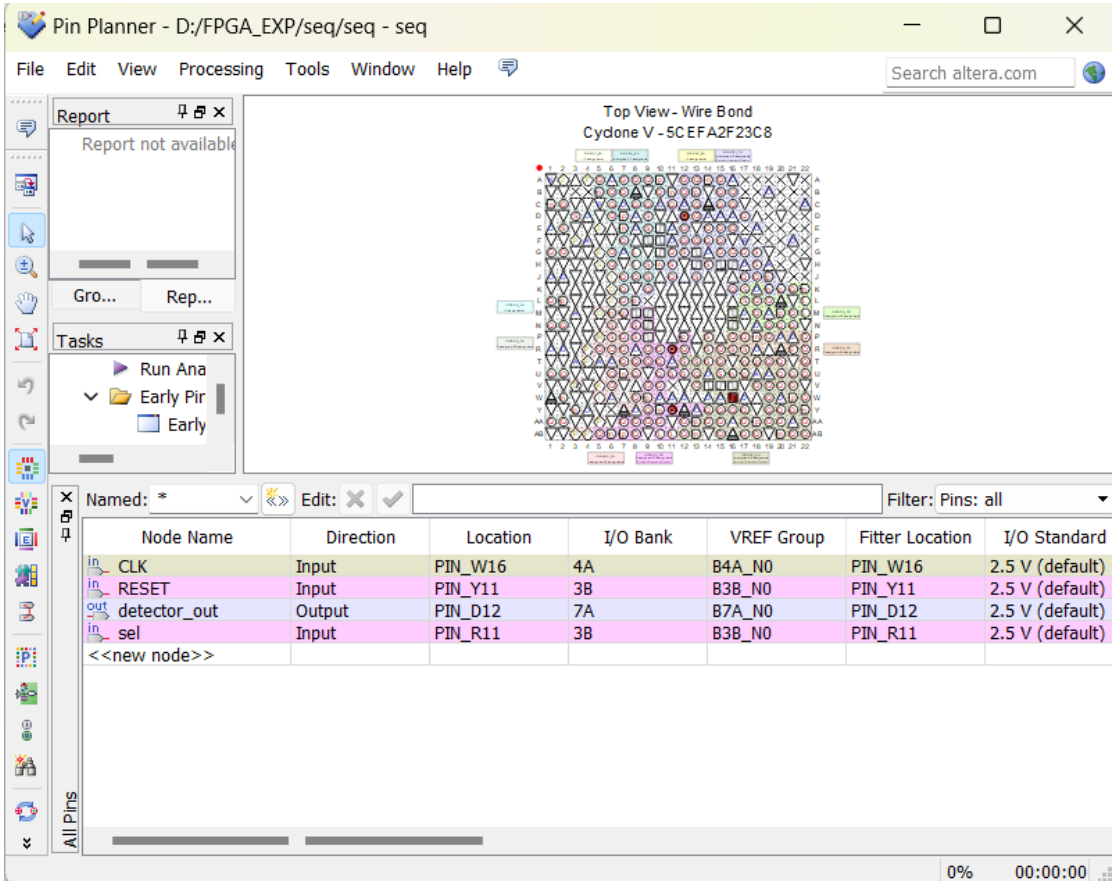


中间结果



Flow Summary	
Flow Status	Successful - Fri Oct 27 12:45:47 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	seq
Top-level Entity Name	seq
Family	Cyclone V
Device	5CEFA2F23C8
Timing Models	Final
Logic utilization (in ALMs)	1,127 / 9,430 (12 %)
Total registers	1533
Total pins	4 / 224 (2 %)
Total virtual pins	0
Total block memory bits	0 / 1,802,240 (0 %)
Total DSP Blocks	0 / 25 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI PMA RX ATT Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total HSSI PMA TX ATT Serializers	0
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 4 (0 %)

FPGA 验证



只有当 sel 信号和 reset 信号都是高电平时才有 led 灯闪烁。

总结

Always 函数的敏感信号要包括检测的输入信号，不是只有时钟与 reset！