实验 3：序列检测器设计

林莉淇 PB22051128

1 实验内容

首先编程设计两个序列产生器（即有限状态机），一个是包含“111010011”的序列产生器，另一个是不包含“111010011”的序列产生器，同时编程设计一个序列检测器（也可以看作是一个状态机）。通过选择器选一路序列送入序列检测器，如果输入的序列包含“111010011”，则序列检测器输出状态“1”，表示检测到被测序列，否则序列检测器输出状态“0”，表示没有检测到被测序列。

再编写 Test Bench 程序并完成功能仿真。接着查看 RTL 电路结构图、为设计工程分配管脚。最后进行硬件验证：观察 LED 灯，查看结果是否正确，并通过切换输入序列验证序列检测器是否正确。

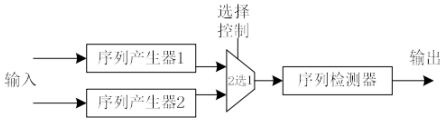

图 1: 实验结构图

2 设计分析

需要针对两个序列产生器、选择器、序列检测器编写不同模块。思路上，两个序列产生器序列检测器都可以用有限状态机的方法编写。先将两个序列产生器与选择器相连，再接到序列检测器上，得到输出。其中，CLK 信号由分频器分频。

具体模块如下：

- 序列产生器
  - 输入：使复位信号 reset、时钟信号 clk（实际输入是经过分频后的 clkD）；
  - 输出：数据序列；
- 序列检测器
  - 输入：使复位信号 reset、时钟信号 clk（实际输入是经过分频后的 clkD）、输入数据；
  - 输出：检测结果；
- 选择器
  - 输入：2 个数据 A,B、选择信号 select；
  - 输出：选择结果 Q；
- 分频器
  - 输入：原始时钟 clk、复位信号 reset;
  - 输出：分频后 clkD;

3 Verilog 源代码

```
module FSM1_llq(clk,reset,Moore);
	input clk,reset;
	output [1:0] Moore;
	reg [1:0] Moore;
	parameter S0=0,S1=1,S2=2,S3=3,S4=4,S5=5,S6=6,S7=7,S8=8,S9=9;
	reg [3:0]st;

	always @(posedge clk or negedge reset) begin
		if (!reset) begin
			st <= S0;
		end
		else begin
			case (st)
				S0:begin st <= S1;end
				S1:begin st <= S2;end
				S2:begin st <= S3;end
				S3:begin st <= S4;end
				S4:begin st <= S5;end
				S5:begin st <= S6;end
				S6:begin st <= S7;end
				S7:begin st <= S8;end
				S8:begin st <= S9;end
				S9:begin st <= S0;end
				default st <= S9;
			endcase
		end
	end

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

代码 1: 序列产生器 1（包含 111010011）

```
module FSM2_llq(clk,reset,Moore);
	input clk,reset;
	output [1:0] Moore;
	reg [1:0] Moore;
	parameter S0=0,S1=1,S2=2,S3=3,S4=4,S5=5,S6=6,S7=7,S8=8,S9=9;
	reg [3:0]st;

	always @(posedge clk or negedge reset) begin
		if (!reset) begin
			st <= S0;
		end
		else begin
			case (st)
				S0:begin st <= S1;end
				S1:begin st <= S2;end
				S2:begin st <= S3;end
				S3:begin st <= S4;end
				S4:begin st <= S5;end
				S5:begin st <= S6;end
				S6:begin st <= S7;end
				S7:begin st <= S8;end
				S8:begin st <= S9;end
				S9:begin st <= S0;end
				default st <= S9;
			endcase
		end
	end

	always @(st) begin
```

```
30              case (st)
31                      S0:Moore <= 1'b1;
32                      S1:Moore <= 1'b0;
33                      S2:Moore <= 1'b1;
34                      S3:Moore <= 1'b0;
35                      S4:Moore <= 1'b0;
36                      S5:Moore <= 1'b0;
37                      S6:Moore <= 1'b0;
38                      S7:Moore <= 1'b0;
39                      S8:Moore <= 1'b1;
40                      S9:Moore <= 1'b0;
41                      default Moore <= 1'b0;
42              endcase
43      end
44  endmodule
```

代码 2: 序列产生器 2（不包含 111010011）

```
1   module seq_detector_llq(clk,reset,Moore,strl);
2       input clk,reset,strl;
3       output [1:0] Moore;
4       reg [1:0] Moore;
5       parameter S0=0,S1=1,S2=2,S3=3,S4=4,S5=5,S6=6,S7=7,S8=8,S9=9;
6       reg [3:0]st;
7
8       always @(posedge clk or negedge reset) begin
9               if (!reset) begin
10                      st <= S0;
11              end
12              else begin
13                      case (st)
14                              S0:begin
15                                      if(strl == 1'b1) st <= S1;
16                                      else st <= S0; end
17                              S1:begin
18                                      if(strl == 1'b1) st <= S2;
19                                      else st <= S0; end
20                              S2:begin
21                                      if(strl == 1'b1) st <= S3;
22                                      else st <= S0; end
23                              S3:begin
24                                      if(strl == 1'b0) st <= S4;
25                                      else st <= S3; end
26                              S4:begin
27                                      if(strl == 1'b1) st <= S5;
28                                      else st <= S0; end
29                              S5:begin
30                                      if(strl == 1'b0) st <= S6;
31                                      else st <= S2; end
32                              S6:begin
33                                      if(strl == 1'b0) st <= S7;
34                                      else st <= S1; end
35                              S7:begin
36                                      if(strl == 1'b1) st <= S8;
37                                      else st <= S0; end
38                              S8:begin
39                                      if(strl == 1'b1) st <= S9;
40                                      else st <= S0; end
41                              S9:begin
42                                      if(strl == 1'b1) st <= S1;
43                                      else st <= S0; end
44                              default st <= S0;
45                      endcase
46              end
47      end
48
49      always @(st) begin
50              case (st)
51                      S0:Moore <= 1'b0;
52                      S1:Moore <= 1'b0;
53                      S2:Moore <= 1'b0;
54                      S3:Moore <= 1'b0;
55                      S4:Moore <= 1'b0;
56                      S5:Moore <= 1'b0;
57                      S6:Moore <= 1'b0;
58                      S7:Moore <= 1'b0;
59                      S8:Moore <= 1'b0;
60                      S9:Moore <= 1'b1;
61                      default Moore <= 1'b0;
```

```
62              endcase
63      end
64  endmodule
```

代码 3: 序列检测器

```
1   module MUX2TO1_llq(A,B,select,Q);
2       input A,B,select;
3       output Q;
4       reg Q;
5
6       always @(A or B or select) begin
7           begin
8               case (select)
9                   1'b0:Q <= A;
10                  1'b1:Q <= B;
11              endcase
12          end
13      end
14  endmodule
```

代码 4: 选择器

```
1   module FD_llq(clk,clkD,reset);
2       input clk;
3       input reset;
4       output clkD;
5       reg [20:0] cnt = 0;
6       reg clkD;
7       always @(posedge clk) begin
8           if(reset == 1'b0) begin
9               cnt <= 20'd0;
10              clkD <=0;
11          end
12          else begin
13              if(cnt==20'd1000000) begin
14                  cnt <=20'd0;
15                  clkD <=~clkD;
16              end
17              else begin
18                  cnt <=cnt+1;
19              end
20          end
21      end
22  endmodule
```

代码 5: 分频器

```
1   module FPGA_EXP3_LLQ(clk,reset,select,Q);
2       input clk,reset,select;
3       output Q;
4       wire x1,x2,x,clkD;
5
6       //FSM1_llq FSM1(clk,reset,x1);
7       //FSM2_llq FSM2(clk,reset,x2);
8       //MUX2TO1_llq MUX2TO1(x1,x2,select,x); //select=0,选FSM1
9       //seq_detector_llq seq_detector(clk,reset,Q,x);
10
11      FD_llq FD(clk,clkD,reset);
12      FSM1_llq FSM1(clkD,reset,x1);
13      FSM2_llq FSM2(clkD,reset,x2);
14      MUX2TO1_llq MUX2TO1(x1,x2,select,x); //select=0,选FSM1
15      seq_detector_llq seq_detector(clkD,reset,Q,x);
16
17  endmodule
```

代码 6: 顶层函数 (注释的是仿真所用顶层函数)

```
1   `timescale 10ns/1ns
2   module FPGA_EXP3_LLQ_tb();
3       reg clk,reset,select;
4       wire Q;
5
6       initial clk=0;
7       always #1 clk =~ clk;
8
```

```
9         initial begin
10                #0 reset = 1'b0;
11                #2 reset = 1'b1;
12                #2 select = 1'b0;
13                #70 select = 1'b1;
14        end
15
16        FPGA_EXP3_LLQ U1(clk,reset,select,Q);
17
18  endmodule
```

代码 7: 仿真代码

仿真了 FPGA_EXP3_LLQ 模块（去掉分频器）。0-2s，reset=0，统一复位；2s 之后，reset=1，使能两个序列产生器、序列检测器。片选信号在 0-70 时间单位内为 0，选择序列发生器 1；在 70 时间单位之后为 1，选择序列发生器 2。

4 仿真结果记录

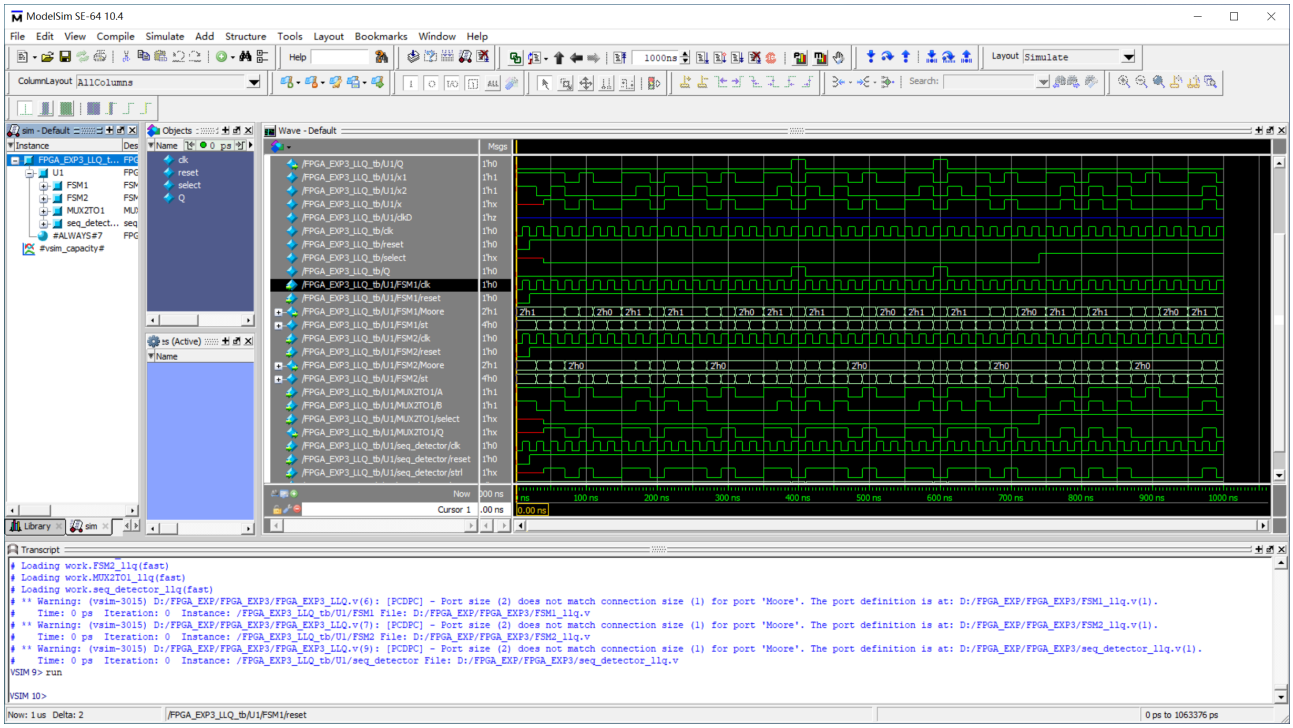

图 2: 仿真结果

由图可见：

- reset=0，使能序列发生器和检测器未使能
- reset=1，使能序列发生器和检测器
  - 2-70 单位时间内：select=0，选序列发生器 1，此时间段内序列检测器检测到 2 次 111010011，产生两次高电平;
  - 70 单位时间后：select=1，选序列发生器 2，此时间段内序列检测器未检测到 111010011，不会产生高电平;

5 RTL 结构图

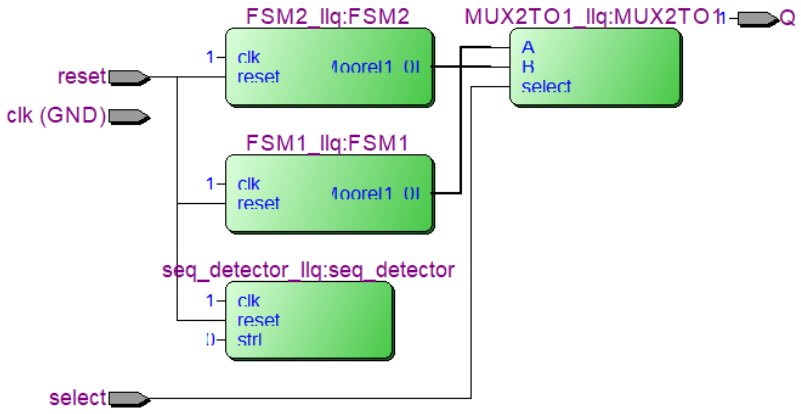

图 3: 总程序

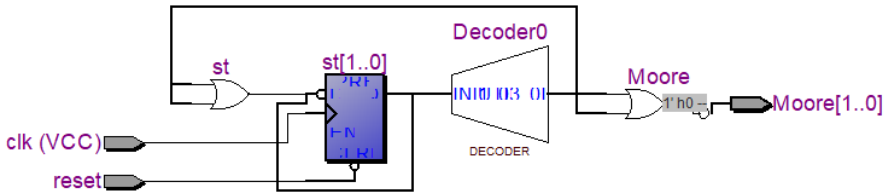

图 4: 序列产生器 1

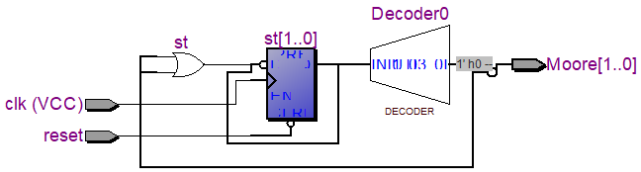

图 5: 序列产生器 2

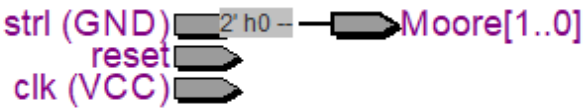

图 6: 序列检测器

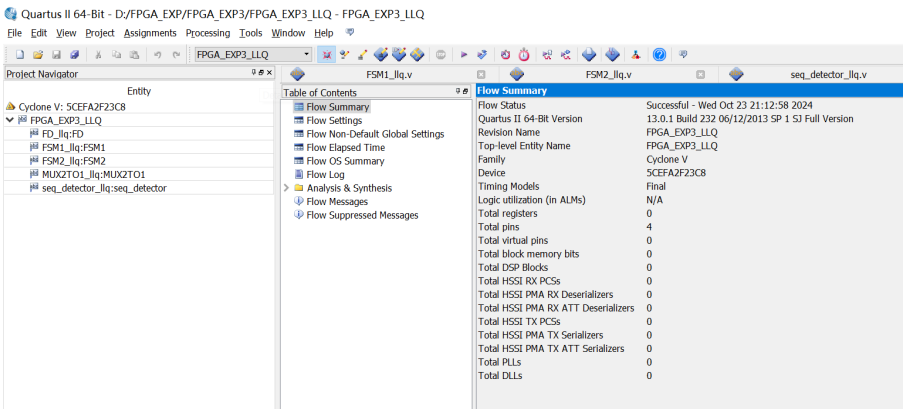

图 7: 全编译之后资源占用信息

6 FPGA 验证结果记录

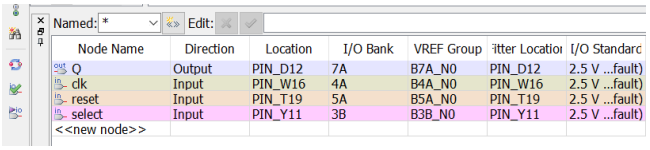

图 8: 管脚锁定情况

动态结果的文字描述：先拨动 DIP15=0，复位。再拨动 DIP15=1 并保持不变，使能序列发生器和检测器。

DIP0=0 时，选择含有 111010011 的序列，LED0 周期性闪烁；DIP0=1 时，选择含有 111010011 的序列，LED0 保持熄灭状态；

7 实验总结

通过这次实验，我明白了分模块调用的重要性。针对不同器件写不同模块再进行调用，可以使代码结构更加清晰。

其中，我碰到去掉分频器仿真时，Q 一直无输出波形的情况。首先是时钟信号传入错误，仍传入 clkD，应该是未分频的 clk。其次时 ctrl 只能从 0 变到 3，原因是“reg [1:0] crtl”, 这样只有 2 位，只能表示 0-3。要让 ctrl 到 9，需要至少 4 位。

我还碰到上板时 LED0 一直没有反应的情况。经检查，是顶层函数的 x 和 Q 传参时位置错误的原因。

在这次实验中，我收获良多。下次实验会更注意让代码结构更清晰，仿真更加全面，以便于最后的硬件验证。