实验 2：七段数码管动态显示电路设计

林莉淇 PB22051128

1 实验内容

首先编写 Verilog 程序，实现七段数码管动态显示控制电路（原理如下图），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，最后完成硬件验证。

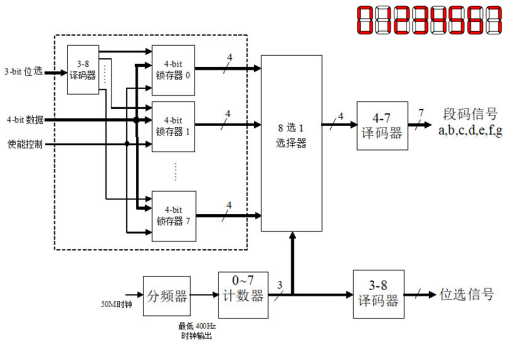

图 1: 总框图

2 设计分析

需要针对锁存器、3-8 译码器、4-7 译码器、8 选 1 选择器、分频器编写不同模块。思路上，先将 3-8 译码器与 8 个 4 位锁存器相连，再用八选一选择器对锁存器做出选择。其中，选择器的选择信号由 clk 经过分频器和计数器给出。之后，把选择得到的输出接到 4-7 译码器上，得到段码信号。此外，计数器的输出经过 3-8 译码器，得到位选信号。

具体模块如下：

- 锁存器
  - 输入：使能信号 en、片选信号 cs、数据 D；
  - 输出：数据 Q；
- 3-8 译码器
  - 输入：二进制数 A；
  - 输出：译码结果 Y；
- 8 选 1 数据选择器
  - 输入：8 个数据 A,B,C,D,E,F,G,H、选择信号 select；
  - 输出：选择结果 Q；
- 7 段数码管
  - 输入：4 位二进制信号 q;
  - 输出：段码信号 A,B,C,D,E,F,G;

- 分频器
  - 输入：原始时钟 clk、复位信号 RST;
  - 输出：分频后 clkD;

3 Verilog 源代码

```
module latch4_llq(D,en,cs,Q);
	input en,cs;
	input [3:0] D;
	output [3:0] Q;

	reg [3:0] Q;

	always @ (*)
	begin
		if(cs == 1)
			if(en == 1)
				Q <= D;
			else
				Q <= Q;
		else
			Q <= Q;
	end
endmodule
```

代码 1: 4 位锁存器

```
module decoder3to8_llq(A,Y);
	input [2:0] A;
	output [7:0] Y;
	reg [7:0] Y;
	always @(A)
	begin
		case (A)
			3'b000: Y = 8'b00000001;
			3'b001: Y = 8'b00000010;
			3'b010: Y = 8'b00000100;
			3'b011: Y = 8'b00001000;
			3'b100: Y = 8'b00010000;
			3'b101: Y = 8'b00100000;
			3'b110: Y = 8'b01000000;
			3'b111: Y = 8'b10000000;
		endcase
	end
endmodule
```

代码 2: 3-8 译码器

```
module select8to1_llq(A,B,C,D,E,F,G,H,select,Q);
	input [3:0] A,B,C,D,E,F,G,H;
	input [2:0] select;
	output [3:0] Q;
	reg [3:0] Q;
	always @(A or B or C or D or E or F or G or H or select)
		begin
			case (select)
				3'b000: Q <= A;
				3'b001: Q <= B;
				3'b010: Q <= C;
				3'b011: Q <= D;
				3'b100: Q <= E;
				3'b101: Q <= F;
				3'b110: Q <= G;
				3'b111: Q <= H;
				default: Q <= A;
			endcase
		end
endmodule
```

代码 3: 8 选 1 数据选择器

```
module func_llq(select,count,data,en,Q);
       input [2:0] select;
       input [2:0] count;
       input [3:0] data;
       input en;
       wire [3:0] Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7;
       wire [7:0] CS;
       output [3:0] Q;

       decoder3to8_llq Decoder(.A(select),.Y(CS));
       latch4_llq LATCH0(.en(en),.D(data),.Q(Q0),.cs(CS[0]));
       latch4_llq LATCH1(.en(en),.D(data),.Q(Q1),.cs(CS[1]));
       latch4_llq LATCH2(.en(en),.D(data),.Q(Q2),.cs(CS[2]));
       latch4_llq LATCH3(.en(en),.D(data),.Q(Q3),.cs(CS[3]));
       latch4_llq LATCH4(.en(en),.D(data),.Q(Q4),.cs(CS[4]));
       latch4_llq LATCH5(.en(en),.D(data),.Q(Q5),.cs(CS[5]));
       latch4_llq LATCH6(.en(en),.D(data),.Q(Q6),.cs(CS[6]));
       latch4_llq LATCH7(.en(en),.D(data),.Q(Q7),.cs(CS[7]));

       select8to1_llq MUX(.A(Q0),.B(Q1),.C(Q2),.D(Q3),.E(Q4),.F(Q5),.G(Q6),.H(Q7),.select(count),.Q(Q));

endmodule
```

代码 4: 锁存器、3-8 译码器、8 选一数据选择器综合模块（待仿真模块）

```
module Seg7Led(q,A,B,C,D,E,F,G);
       input [3:0] q;
       output A,B,C,D,E,F,G;
       reg [6:0] ATOG;
       always@(q)
              begin
                     ATOG = 7'b0000000;
                     case(q)
                            4'b0000:ATOG = 7'b0000001;
                            4'b0001:ATOG = 7'b1001111;
                            4'b0010:ATOG = 7'b0010010;
                            4'b0011:ATOG = 7'b0000110;
                            4'b0100:ATOG = 7'b1001100;
                            4'b0101:ATOG = 7'b0100100;
                            4'b0110:ATOG = 7'b0100000;
                            4'b0111:ATOG = 7'b0001111;
                            4'b1000:ATOG = 7'b0000000;
                            4'b1001:ATOG = 7'b0001000;
                            4'b1010:ATOG = 7'b0001000;
                            4'b1011:ATOG = 7'b1100000;
                            4'b1100:ATOG = 7'b0110001;
                            4'b1101:ATOG = 7'b1000010;
                            4'b1110:ATOG = 7'b0110000;
                            4'b1111:ATOG = 7'b0111000;
                            default:ATOG = 7'b1111111;
                     endcase
              end

              assign A = ATOG[6];
              assign B = ATOG[5];
              assign C = ATOG[4];
              assign D = ATOG[3];
              assign E = ATOG[2];
              assign F = ATOG[1];
              assign G = ATOG[0];

endmodule
```

代码 5: 7 段数码管

```
module FD(clk,clkD,RST);
       input clk;
       input RST;
       output clkD;
       reg [17:0] cnt = 0;
       reg clkD;
       always @(posedge clk) begin
              if(RST == 1'b0) begin
                     cnt <= 18'd0;
                     clkD <=0;
              end
              if(cnt==18'd2) begin
```

```
                    cnt <=18'b000000000000000000;
                    clkD <=~clkD;
             end
             else begin
                    cnt <=cnt+1;
             end
      end
endmodule
```

代码 6: 分频器

```
module FPGA_EXP2_llq(en,data,select,clk,LED_S,A,B,C,D,E,F,G,RST);
      input clk,en,RST;
      input [2:0] select;
      input [3:0] data;
      wire clkD;
      reg [2:0] count;
      output [7:0] LED_S;
      output A,B,C,D,E,F,G;

      wire [3:0] Q;

      FD FD(clk,clkD,RST);

      always@(posedge clkD) begin
             count = count+1;
      end

      decoder3to8_llq Decoder(.A(count),.Y(LED_S));
      func_llq func(select,count,data,en,Q);
      Seg7Led Seg7Led(Q,A,B,C,D,E,F,G);

endmodule
```

代码 7: 顶层函数

```
`timescale 10ns/1ns
module func_llq_tb();
      reg [2:0] select,count;
      reg [3:0] data;
      reg en;
      wire [3:0] Q;

      initial begin
             #0 en=1'b1;
      end

      always begin
             #0 select=3'b000;
             #2 select=3'b001;
             #2 select=3'b010;
             #2 select=3'b011;
             #2 select=3'b100;
             #2 select=3'b101;
             #2 select=3'b110;
             #2 select=3'b111;
      end

      always begin
             #0 data=4'b0000;
             #2 data=4'b0001;
             #2 data=4'b0010;
             #2 data=4'b0011;
      end

      func_llq U1(select,select, data, en, Q);

endmodule
```

代码 8: 仿真代码

仿真了 func 模块，即包含 8 个 4 位锁存器、3-8 译码器、8 选一数据选择器的综合模块（实验内容框图中虚线框内部分）。

- 使能信号 en 一直为 1；

- 为了便于仿真，令 8 选一数据选择器的 count 和 3-8 译码器的 select 一同变化（在顶层函数调用该模块时，会将数据选择器的 count 接到分频器上而不是 select 上，此处仅为了便于仿真）。

4 仿真结果记录

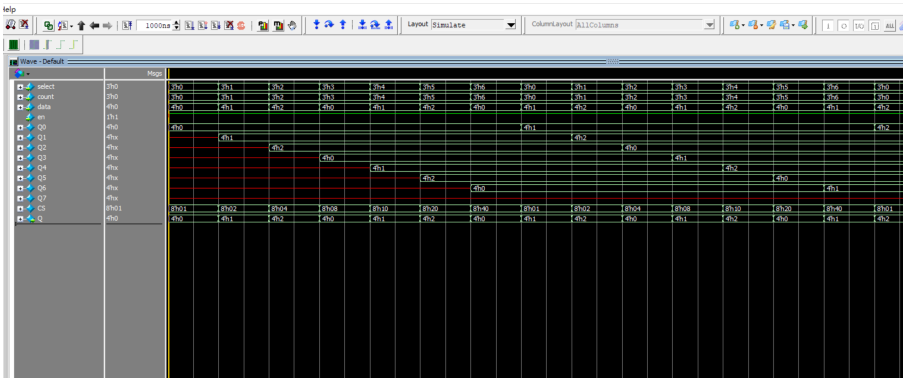

图 2: 仿真结果

由图可见：

- select=0，此时 data=0，存入寄存器 0
- select=1，此时 data=1，存入寄存器 1
- select=2，此时 data=2，存入寄存器 2
- elect=3，此时 data=0，存入寄存器 3
- ……

可以看出，在使能信号 en 一直为 1 后，select 信号使不同的 data 数据传入到不同的锁存器中。同时，count 接到 select 上，使得 8 选一选择器选择的锁存器与 select 选择的相同，令 data 的改变能反馈到 Q 上。

5 RTL 结构图

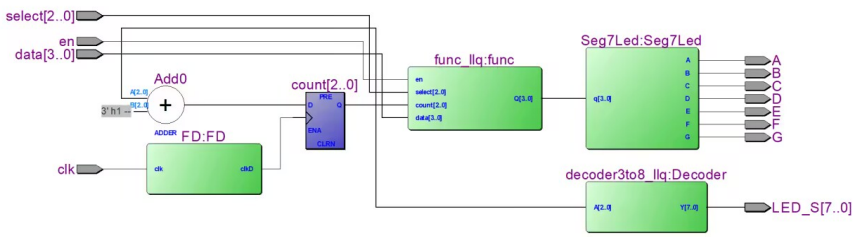

图 3: 总程序

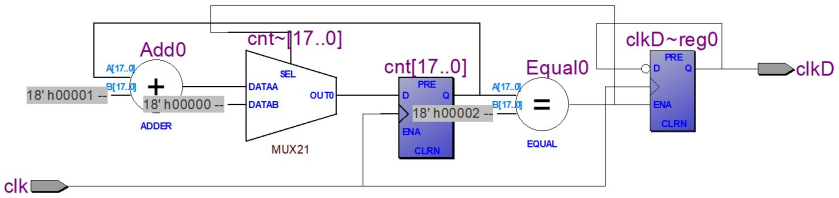

图 4: 分频器

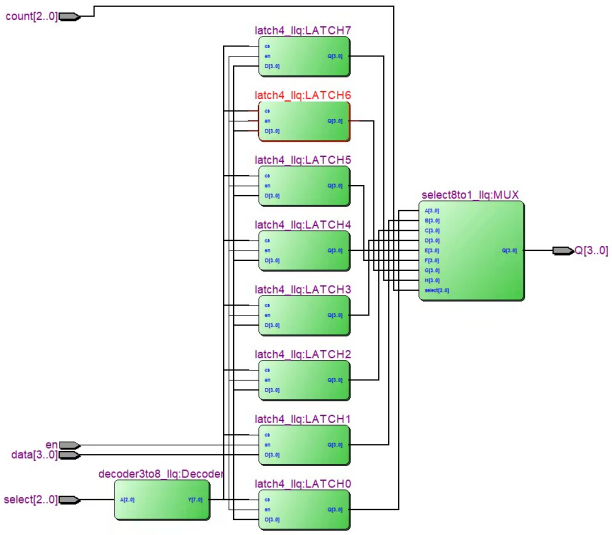

图 5: 仿真模块

select 接到 decoder3to8_llq 上，count 接到 select8to1_llq 上，en 接到每一个锁存器上。

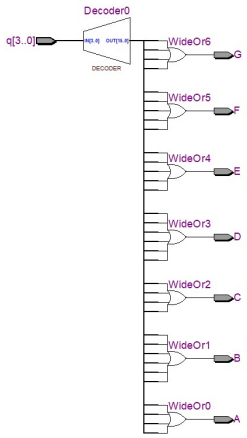

图 6: 7 段数码管

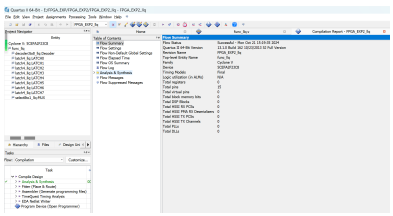

图 7: 全编译之后资源占用信息

6 FPGA 验证结果记录

Node Name	Direction	Location	I/O Bank	REF Grou	ter Locati	O Standa	Reserved	ent Strer	Slew Rate
A	Output	PIN_K19	7A	B7A_N0	PIN_K19	2.5 ...ult)		12m...lt)	1 (d...ult)
B	Output	PIN_H18	7A	B7A_N0	PIN_H18	2.5 ...ult)		12m...lt)	1 (d...ult)
C	Output	PIN_K20	7A	B7A_N0	PIN_K20	2.5 ...ult)		12m...lt)	1 (d...ult)
D	Output	PIN_M18	5B	B5B_N0	PIN_M18	2.5 ...ult)		12m...lt)	1 (d...ult)
E	Output	PIN_E16	7A	B7A_N0	PIN_E16	2.5 ...ult)		12m...lt)	1 (d...ult)
F	Output	PIN_G13	7A	B7A_N0	PIN_G13	2.5 ...ult)		12m...lt)	1 (d...ult)
G	Output	PIN_G17	7A	B7A_N0	PIN_G17	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[7]	Output	PIN_E12	7A	B7A_N0	PIN_E12	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[6]	Output	PIN_D13	7A	B7A_N0	PIN_D13	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[5]	Output	PIN_A13	7A	B7A_N0	PIN_A13	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[4]	Output	PIN_C11	7A	B7A_N0	PIN_C11	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[3]	Output	PIN_F13	7A	B7A_N0	PIN_F13	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[2]	Output	PIN_E14	7A	B7A_N0	PIN_E14	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[1]	Output	PIN_B15	7A	B7A_N0	PIN_B15	2.5 ...ult)		12m...lt)	1 (d...ult)
LED_S[0]	Output	PIN_B16	7A	B7A_N0	PIN_B16	2.5 ...ult)		12m...lt)	1 (d...ult)
RST	Input	PIN_T19	5A	B5A_N0	PIN_T19	2.5 ...ult)		12m...lt)	
clk	Input	PIN_W16	4A	B4A_N0	PIN_W16	2.5 ...ult)		12m...lt)	
data[3]	Input	PIN_Y11	3B	B3B_N0	PIN_Y11	2.5 ...ult)		12m...lt)	
data[2]	Input	PIN_R11	3B	B3B_N0	PIN_R11	2.5 ...ult)		12m...lt)	
data[1]	Input	PIN_U12	3B	B3B_N0	PIN_U12	2.5 ...ult)		12m...lt)	
data[0]	Input	PIN_V13	4A	B4A_N0	PIN_V13	2.5 ...ult)		12m...lt)	
en	Input	PIN_AB13	4A	B4A_N0	PIN_AB13	2.5 ...ult)		12m...lt)	
select[2]	Input	PIN_Y14	4A	B4A_N0	PIN_Y14	2.5 ...ult)		12m...lt)	
select[1]	Input	PIN_Y15	4A	B4A_N0	PIN_Y15	2.5 ...ult)		12m...lt)	
select[0]	Input	PIN_AA15	4A	B4A_N0	PIN_AA15	2.5 ...ult)		12m...lt)	

图 8: 管脚锁定情况

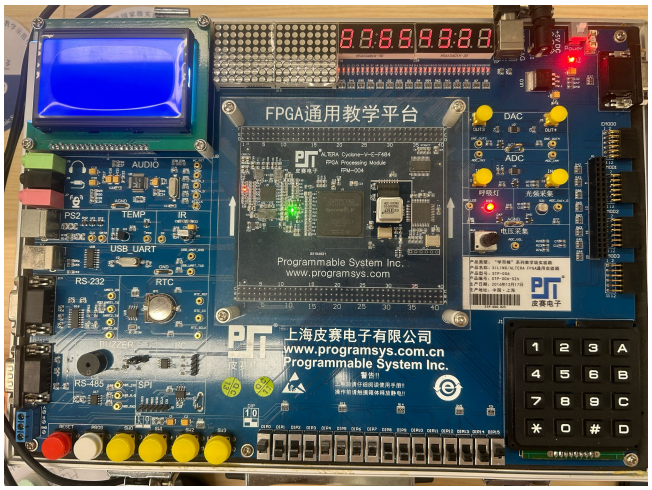

图 9: 硬件验证图

动态结果的文字描述：先拨动 DIP15 复位。拨动 DIP4 使能后，拨动 DIP5、DIP6、DIP7 选定想要改变的锁存器，再拨动 DIP0、DIP1、DIP2、DIP3，改变数字。

例如，当 DIP0、1、2、3、4、5、6、7 分别为 0001 1 010 时，从右往左第三个数码管显示的数字为 1。此时再把 DIP5、6、7 调整为 001，把 DIP0、1、2、3 调整为 0、1、0、0，则第二个数码管显示 4，同时第三个数码管依然显示 1。

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，我碰到编译时报错” 端口未定义” 的情况，module 内定义的所有输入输出端口都要写入 module 后面的括号声明中。

我还碰到分频器没有加复位信号，模块中各个变量没有赋初值的情况。

在这次实验中，我收获良多。下次实验会提前写好程序，成功编译、仿真，在实验室再进行最后的接线验证。