北京科技大学 计算机与通信工程学院

实 验 报 告

实验名称：时序逻辑实验

一、实验目的与实验要求

1、实验目的

本实验的目的是学习时序逻辑模块在数字系统中的综合应用与程序编写并且掌握实验平台的外部功能模块在数字系统设计中的应用。学习 SRAM 的使用，为设计复杂的数字电路，尤其是 CPU 设计奠定基础。

2、实验要求

1） 在实验报告中提交系统级模块图、设计代码、仿真程序（部分模块）、仿真结果截图、实测验证结果照片。

其中，系统级模块设计图要求给出整个系统的数据输出信号，系统内各个子模块的输入输出信号和模块间的连接关系。

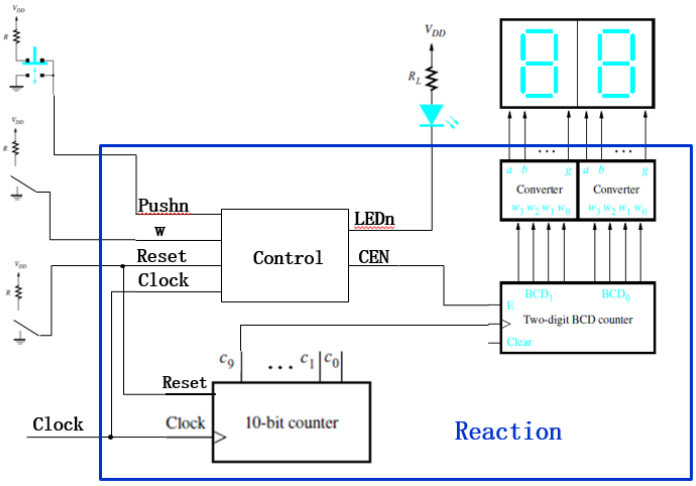


图 1：系统级模块设计图

2） 提交实验报告和每个实验的完整工程文件。

二、实验设备（环境）及要求

（1） Xilinx Ego1 实验平台。

（2） OS：Win7 64 位

（3） Software：Vivado15.4 开发工具

三、实验内容与步骤

1、实验 1

（1）实验内容

在七段数码管上滚动显示学号：按照“4 实验步骤——在七段数码管上滚动显示学号” 完成本实验。以学号 40123456 为例：

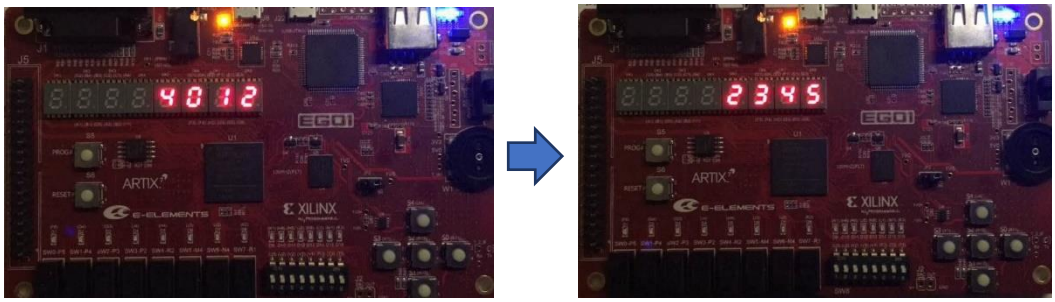
a. 首先将学号中的数字被存储在一个 32 位的寄存器 msgArray 中；

b. 4 个数码管始终显示寄存器的低 16 位数据；

c. 用频率为 3Hz 的时钟控制 7 段数码循环显示：在时钟的上升沿进行向左循环移动 4 位，并显示。注意：记得要把 msgArray 中的内容，即 msgArray [3:0] 的内容移到了 msgArray [31:28]中。

d. 复位时，寄存器恢复原始存储状态，7 段数码管显示第一组 4 位字符（寄存器中的低 16 位），即 4012；

下图显示了通过 4 个数码管滚动显示学号（40123456），初始时刻和经过 3 个周期以后的效果：



t(0)时刻　　t(3)时刻

（2）主要步骤

a) 新建工程，完成七段数码管的设计与板级验证

b) 滚动七段数码管模块代码如下：

```
module clkDiv(
    input clk100mhz,
    output clk190hz,
    output clk3hz
```

```
        );
          reg[25:0] count =0;
          assign clk190hz = count[18];
          assign clk3hz =count[25];
          always@(posedge clk100mhz)
                count<= count+1;
endmodule

module GPU(
      input clk3hz,
      input clr,
      output [15:0]dataBus
            );

            reg[31:0] msgArray;
            parameter NUMBER = 32'h41524119;
            assign dataBus = msgArray[31:16];
            always@(posedge clk3hz or posedge clr)
                  if(!clr)
                     msgArray<=NUMBER;
                  else begin
                       msgArray[3:0]<= msgArray[31:28];
                       msgArray[31:4]<= msgArray[27:0];
                  end
endmodule

module segMsg(
      input clk190hz,
      input [15:0] dataBus,
      output reg [3:0] pos,
```

```
output reg [7:0] seg
    );
      reg [1:0] posC;
      reg [3:0] dataP;
      always @(posedge clk190hz)begin
          case(posC)
            0:begin
                  pos    <= 4'b0001;
                  dataP <= dataBus[3:0];
               end

            1:begin
              pos <=4'b0010;
              dataP<= dataBus [7:4];
            end

             2:begin
              pos <=4'b0100;
              dataP<= dataBus [11:8];
             end
             3:begin
             pos    <=4'b1000;
             dataP<= dataBus[15:12];
             end
      endcase
      posC=posC + 1;
   end
   always @(dataP)
         case(dataP)
              0:seg =8'b0011_1111;
```

```
                1:seg =8'b0000_0110;
                2:seg =8'b0101_1011;
                3:seg =8'b0100_1111;
                4:seg =8'b0110_0110;
                5:seg =8'b0110_1101;
                6:seg =8'b0111_1101;
                7:seg =8'b0000_0111;
                8:seg =8'b0111_1111;
                9:seg =8'b0110_1111;
                10:seg =8'b0100_0000;
                11:seg =8'b0000_0000;
                default: seg =8'b0000_1000;
            endcase
endmodule


module top(
    input clk100mhz,
    input clr,
    output [3:0] pos,
    output [7:0] seg
        );
        wire clk190hz,clk3hz;
        wire [15:0]dataBus;

        clkDiv U1(clk100mhz,clk190hz,clk3hz);
        GPU    U2(clk3hz,clr,dataBus);
        segMsg U3(clk190hz,dataBus,pos,seg);

 endmodule
```

```
`timescale 1ns / 1ps
module GPU_tb();
    reg clk;
    reg clr;
    wire[15:0]dataBus;
        initial begin
            clk=0;
            clr=0;
            #100 clr=1;
        end
        always #5 clk=~clk;
        GPU tb1(clk,clr,dataBus);
endmodule
```

c) 管脚约束代码：

```
set_property IOSTANDARD LVCMOS33 [get_ports {pos[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property PACKAGE_PIN G1 [get_ports {pos[3]}]
```

```
set_property PACKAGE_PIN F1 [get_ports {pos[2]}]
set_property PACKAGE_PIN E1 [get_ports {pos[1]}]
set_property PACKAGE_PIN G6 [get_ports {pos[0]}]
set_property PACKAGE_PIN P17 [get_ports clk100mhz]
set_property PACKAGE_PIN P5 [get_ports clr]
set_property IOSTANDARD LVCMOS33 [get_ports clr]
set_property IOSTANDARD LVCMOS33 [get_ports clk100mhz]
set_property PACKAGE_PIN H2 [get_ports {seg[7]}]
set_property PACKAGE_PIN D2 [get_ports {seg[6]}]
set_property PACKAGE_PIN E2 [get_ports {seg[5]}]
set_property PACKAGE_PIN F3 [get_ports {seg[4]}]
set_property PACKAGE_PIN F4 [get_ports {seg[3]}]
set_property PACKAGE_PIN D3 [get_ports {seg[2]}]
set_property PACKAGE_PIN E3 [get_ports {seg[1]}]
set_property PACKAGE_PIN D4 [get_ports {seg[0]}]
```

d) RTL 门级结构示意图：

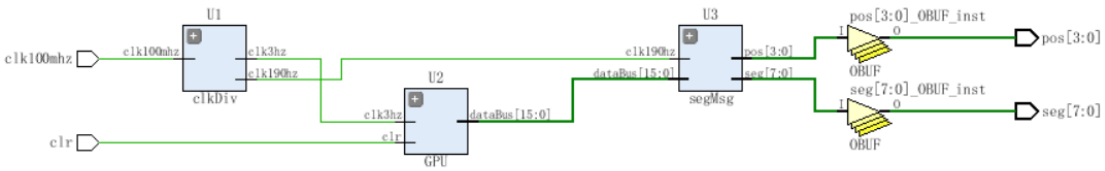


图 2：滚动显示 RTL 示意图

2、实验 2

（1）实验内容

通过按键输入学号，并循环显示：电路功能描述：通过 Ego1 上的按键输入自己

的学号（8 位 10 进制数），并存储在 32 位的寄存器中；8 位 10 进制数输入完成后，实现类似实验 3.1 中的滚动显示效果。

除了要求实现上述功能外，还包括如下具体实现上的要求：

a. 整个电路具有复位功能；

b. 8 个数码管中，4 个数码管用于输入数据后的循环显示，另外 4 个用于显示当前正在输入的数据。

c. 输入过程可控、实时可视：

- 用按键或者开关控制输入开始和结束，输入开始后用于循环显示的 4 个数码管停止循环显示，用于显示输入数据的 4 个数码管开始显示输入数据；输入结束后用于循环显示的 4 个数码管开始循环显示输入的学号，用于显示输入数据的 4 个数码管停止显示输入数据。
- 用按键切换当前要输入的数据位，切换结果通过用于显示输入数据的 4 个数码管展示出来，即用户能够通过显示输入数据的 4 个数码管看出来当前正在输入的是哪一位 10 进制数。

d. 输入数字的时候进行按键消抖（关于按键防抖的原理，请参考本实验手册的“5.1 实验 3.2 相关说明”部分）；

e. **设计具有开放性**，下图仅仅为参考的一种实现效果，不作强求实现一样的效果。

f. 数码管、按键、开关的管脚和控制方式等信息请参考本文件“5.1 实验 3.2 相关说明”以及”EGo1 使用手册.pdf 和 EGO1 电路原理图.pdf”

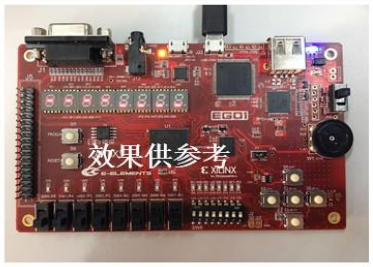
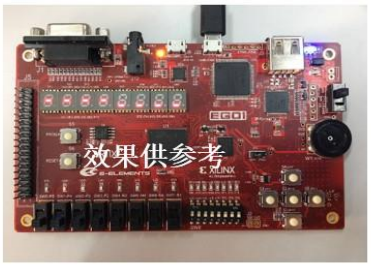
<table>
<tr><td> 
1.初始状态（未打开总开关）</td><td> 
2.打开开关准备输入，左侧四个数码管闪烁</td></tr>
<tr><td> 
3.在右侧数码管输入前四个数据</td><td> 
4.输入后四个数据</td></tr>
<tr><td> 
5.完成输入时，右侧四个数码管进入等待读取状态</td><td> 
6.打开读取开关，左侧四个数码管进行滚动输出</td></tr>
<tr><td> 
7.当前为滚动输出状态</td><td> 
8.关闭读取和总开关，恢复原状态，等待下一次开始。</td></tr>
</table>

图 3： 滚动显示照片

（2）主要步骤

a) 确定实验参考图相关知识及确定对应管脚：

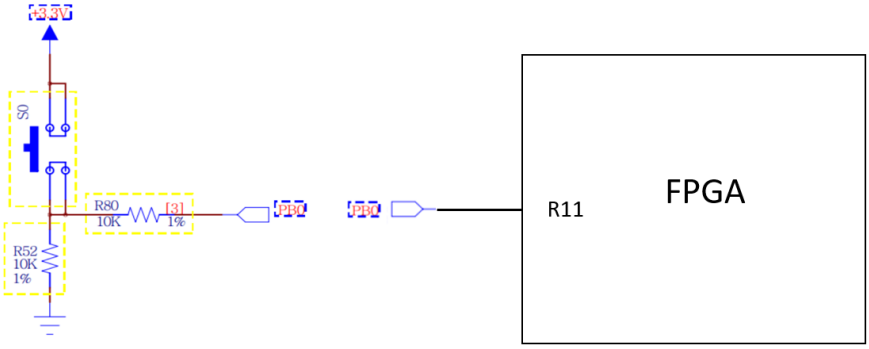


图 4： FPGA 连接示意图

按键名称	FPGA 管脚
S0	R11
S1	R17
S2	R15
S3	V1
S4	U4

图 5： 管脚对应表

图 6： 防抖说明

b) 新建工程，catchPress 即为输入滚动数码管显示器。

c) 模块代码

```
`timescale 1ns / 1ps

module catchPress(input clk190hz, input press, output signal);
    reg delay1, delay2, delay3, delay4, delay5;

    always @(posedge clk190hz)
    begin
        delay5 = delay4;
        delay4 = delay3;
        delay3 = delay2;
        delay2 = delay1;
```

```
            delay1 = press;
        end

        assign signal = delay1 & delay2 & delay3 & delay4 & delay5;
endmodule


module clkDiv(input clk100mhz, output clk190hz, output clk3hz);
        reg [25:0] cnt = 0;
        assign clk190hz = cnt[18];
        assign clk3hz = cnt[25];
        always @(posedge clk100mhz)
            cnt = cnt + 1;
endmodule
```

```
`timescale 1ns / 1ps

module dataRead(
        input clk190hz,
        input rst,
        input leftMove,
        input rightMove,
        input inc,
        output [3:0] pos,
        output [7:0] seg,
        output [31:0] myID,
        output [2:0] temp
        );

        reg [31:0] msgArray = 0;
        reg [2:0] cur = 7;
        reg [15:0] dataBus;
        reg lmv_delay = 0, rmv_delay = 0, inc_delay = 0;

        assign myID = msgArray;
       assign temp = cur;
        always @(posedge clk190hz)
        begin
                if (!rst) begin
                        msgArray = 0;
                        cur = 7;
                        lmv_delay = 0;
                        rmv_delay = 0;
                        inc_delay = 0;
                end
                else begin
```

```
            if (leftMove == 1 && lmv_delay == 0) cur = cur + 1;
              else if (rightMove == 1 && rmv_delay == 0) cur = cur - 1;
              else if (inc == 1 && inc_delay == 0) begin
                    msgArray = msgArray + (4'b0001 << (4*cur));
                    if (((msgArray >> (4*cur)) & 4'b1111) == 10)
                          msgArray = msgArray - (4'b1010 << (4*cur));
              end
              lmv_delay = leftMove;
              rmv_delay = rightMove;
              inc_delay = inc;
        end

        if (cur<4) dataBus = msgArray[15:0];
              else dataBus = msgArray[31:16];
    end

    segMsg2 u(clk190hz, rst, dataBus, cur[1:0], pos, seg);
endmodule

`timescale 1ns / 1ps

module dataShow(
    input clk190hz,
    input clk3hz,
    input rst,
    input [31:0] myID,
    output [3:0] pos,
    output [7:0] seg
    );

    wire [15:0] dataBus;
    GPU         u1(clk3hz, rst, myID, dataBus);
    segMsg1 u2(clk190hz, rst, dataBus, pos, seg);
endmodule
```

```
`timescale 1ns / 1ps

module GPU(input clk3hz, input rst, input [31:0] myID, output [15:0] dataBus);

	reg[31:0] msgArray;
	assign dataBus = msgArray[31:16];
	always @(posedge clk3hz)
	begin
		if (!rst) msgArray <= myID;
		else begin
			msgArray[3:0] <= msgArray[31:28];
			msgArray[31:4] <= msgArray[27:0];
		end
	end
endmodule

`timescale 1ns / 1ps

module segMsg1(
	input clk190hz,
	input rst,
	input [15:0] dataBus,
	output reg [3:0] pos,
	output reg [7:0] seg
	);

	reg [1:0] posC;
	reg [3:0] dataP;

	always @(posedge clk190hz)
	begin
		if (!rst)
		begin
			pos = 4'b0000;
			posC = 0;
		end
```

```
else begin
                case (posC)
                    0: begin
                        pos     <= 4'b0001;
                        dataP <= dataBus[3:0];
                    end
                    1: begin
                        pos     <= 4'b0010;
                        dataP <= dataBus[7:4];
                    end
                    2: begin
                        pos     <= 4'b0100;
                        dataP <= dataBus[11:8];
                    end
                    3: begin
                        pos     <= 4'b1000;
                        dataP <= dataBus[15:12];
                    end
                endcase
                posC = posC + 1;
            end
        end

        always @(dataP)
            case (dataP)
                0: seg = 8'b0011_1111;
                1: seg = 8'b0000_0110;
                2: seg = 8'b0101_1011;
                3: seg = 8'b0100_1111;
                4: seg = 8'b0110_0110;
                5: seg = 8'b0110_1101;
                6: seg = 8'b0111_1101;
                7: seg = 8'b0000_0111;
                8: seg = 8'b0111_1111;
                9: seg = 8'b0110_1111;
                10:seg = 8'b0100_0000;
                11:seg = 8'b0000_0000;
                default: seg = 8'b0000_1000;
            endcase
endmodule
```

```
`timescale 1ns / 1ps

module segMsg2(
	input clk190hz,
	input rst,
	input [15:0] dataBus,
	input [1:0] decimalPoint,
	output reg [3:0] pos,
	output reg [7:0] seg
	);

	reg [1:0] posC;
	reg [3:0] dataP;

	always @(posedge clk190hz)
	begin
		if (!rst)
		begin
			pos = 4'b0000;
			posC = 3;
		end
		else begin
			posC = posC + 1;
			case (posC)
				0: begin
					pos  <= 4'b0001;
					dataP <= dataBus[3:0];
				end
				1: begin
					pos  <= 4'b0010;
					dataP <= dataBus[7:4];
				end
				2: begin
					pos  <= 4'b0100;
					dataP <= dataBus[11:8];
				end
				3: begin
					pos  <= 4'b1000;
					dataP <= dataBus[15:12];
				end
			endcase
		end
	end
```

```
    always @(posC)
    begin
        case (dataP)
            0: seg = 8'b0011_1111;
            1: seg = 8'b0000_0110;
            2: seg = 8'b0101_1011;
            3: seg = 8'b0100_1111;
            4: seg = 8'b0110_0110;
            5: seg = 8'b0110_1101;
            6: seg = 8'b0111_1101;
            7: seg = 8'b0000_0111;
            8: seg = 8'b0111_1111;
            9: seg = 8'b0110_1111;
            10:seg = 8'b0100_0000;
            11:seg = 8'b0000_0000;
            default: seg = 8'b0000_1000;
        endcase

        if (posC == decimalPoint) seg = seg | 8'b1000_0000;
    end
endmodule
```

```
`timescale 1ns / 1ps

module top(
	input clk100mhz,
	input rst,
	input sel,
	input left_move,
	input right_move,
	input increase,
	output [3:0] pos1,
	output [7:0] seg1,
	output [3:0] pos2,
	output [7:0] seg2,
	output [2:0] temp
	);
	wire clk190hz, clk3hz;
	wire [31:0] myID;
	wire [15:0] dataBus;
	wire leftMove, rightMove, inc;

	clkDiv		u1(clk100mhz, clk190hz, clk3hz);
	dataShow	 u2(clk190hz, clk3hz, rst & sel, myID, pos1, seg1);
	dataRead	u3(clk190hz, rst & (~sel), leftMove, rightMove, inc, pos2, seg2,
myID, temp);

	catchPress c1(clk190hz, left_move, leftMove);
	catchPress c2(clk190hz, right_move, rightMove);
	catchPress c3(clk190hz, increase, inc);
endmodule
```

```
`timescale 1ns / 1ps


module dataRead_test();
    reg clk190hz, rst, leftMove, rightMove, inc;
    wire [31:0] myID;
    wire [15:0] dataBus;

    dataRead u1(clk190hz, rst, leftMove, rightMove, inc, myID, dataBus);

    initial begin
        clk190hz = 0;
        rst = 0;
        leftMove = 0;
        rightMove = 0;
        inc = 0;
        #10 rst = 1;
        #5 leftMove = 1;
        #3 leftMove = 0;
        #7 inc = 1;
    end

    always #5 clk190hz = ~clk190hz;

endmodule
```

d) FPGA 板级验证管脚约束文件：

```
set_property PACKAGE_PIN P17 [get_ports clk100mhz]
set_property PACKAGE_PIN P5 [get_ports rst]
set_property PACKAGE_PIN P4 [get_ports sel]
set_property PACKAGE_PIN R15 [get_ports increase]
set_property PACKAGE_PIN V1 [get_ports left_move]
set_property PACKAGE_PIN R11 [get_ports right_move]
set_property PACKAGE_PIN G1 [get_ports {pos1[3]}]
set_property PACKAGE_PIN F1 [get_ports {pos1[2]}]
set_property PACKAGE_PIN E1 [get_ports {pos1[1]}]
set_property PACKAGE_PIN G6 [get_ports {pos1[0]}]
set_property PACKAGE_PIN H2 [get_ports {seg1[7]}]
set_property PACKAGE_PIN D2 [get_ports {seg1[6]}]
set_property PACKAGE_PIN E2 [get_ports {seg1[5]}]
set_property PACKAGE_PIN F3 [get_ports {seg1[4]}]
set_property PACKAGE_PIN F4 [get_ports {seg1[3]}]
set_property PACKAGE_PIN D3 [get_ports {seg1[2]}]
set_property PACKAGE_PIN E3 [get_ports {seg1[1]}]
```

```
set_property PACKAGE_PIN D4 [get_ports {seg1[0]}]
set_property PACKAGE_PIN G2 [get_ports {pos2[3]}]
set_property PACKAGE_PIN C2 [get_ports {pos2[2]}]
set_property PACKAGE_PIN C1 [get_ports {pos2[1]}]
set_property PACKAGE_PIN H1 [get_ports {pos2[0]}]
set_property PACKAGE_PIN D5 [get_ports {seg2[7]}]
set_property PACKAGE_PIN B2 [get_ports {seg2[6]}]
set_property PACKAGE_PIN B3 [get_ports {seg2[5]}]
set_property PACKAGE_PIN A1 [get_ports {seg2[4]}]
set_property PACKAGE_PIN B1 [get_ports {seg2[3]}]
set_property PACKAGE_PIN A3 [get_ports {seg2[2]}]
set_property PACKAGE_PIN A4 [get_ports {seg2[1]}]
set_property PACKAGE_PIN B4 [get_ports {seg2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos2[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg2[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk100mhz]
set_property IOSTANDARD LVCMOS33 [get_ports increase]
set_property IOSTANDARD LVCMOS33 [get_ports left_move]
set_property IOSTANDARD LVCMOS33 [get_ports right_move]
set_property IOSTANDARD LVCMOS33 [get_ports rst]
set_property IOSTANDARD LVCMOS33 [get_ports sel]

set_property PACKAGE_PIN F6 [get_ports temp]
set_property IOSTANDARD LVCMOS33 [get_ports {temp[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {temp[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {temp[0]}]

set_property PACKAGE_PIN F6 [get_ports {temp[2]}]
set_property PACKAGE_PIN G4 [get_ports {temp[1]}]
```

set_property PACKAGE_PIN G3 [get_ports {temp[0]}]

e) 滚动数码管 RTL 门级结构图

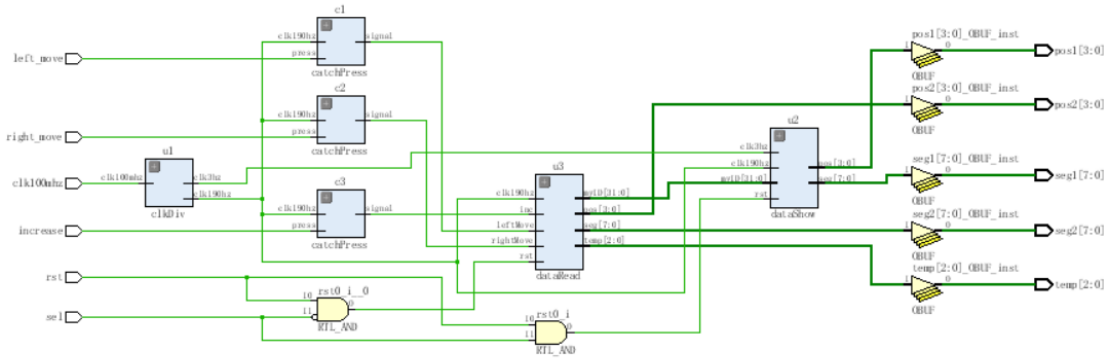


图 7： 数码管 RTL 门级结构图

f) 得到板级验证数码管显示情况，见实验结果与分析。

3、实验 3

（1）实验内容

SRAM 芯片数据存取：在实验 3.2 的基础上实现将输入的数据存入 FPGA 的 SRAM 芯片上，当进行读取时将数据从 SRAM 芯片中取出来。

（2）主要步骤

a) 查询 SRAM 说明书，理解 SRAM 相关必备知识。

SRAM 静态随机存储器，通常作为 CPU 的存储器使用。板卡搭载的 IS61WV12816DBLL SRAM 芯片，总容量 2Mbit。该 SRAM 为异步式 SRAM，最高存取时间可达 8ns。 操控简单，易于读写。

图 8： EGO 芯片示意图

I/O0-15	数据总线
A00-18	地址线
OE	输出使能
CE	芯片使能
WE	写使能
UB	高八位选通
LB	低八位选通

图 9： SRAM 管脚配置

TRUTH TABLE

Mode	$\overline{WE}$	$\overline{CE}$	$\overline{OE}$	$\overline{LB}$	$\overline{UB}$	I/O PIN I/O0-I/O7	I/O PIN I/O8-I/O15	V_{DD} Current
Not Selected	X	H	X	X	X	High-Z	High-Z	I_{SB1}, I_{SB2}
Output Disabled	H	L	H	X	X	High-Z	High-Z	I_{CC}
	X	L	X	H	H	High-Z	High-Z	
Read	H	L	L	L	H	D_{OUT}	High-Z	I_{CC}
	H	L	L	H	L	High-Z	D_{OUT}	
	H	L	L	L	L	D_{OUT}	D_{OUT}	
Write	L	L	X	L	H	D_{IN}	High-Z	I_{CC}
	L	L	X	H	L	High-Z	D_{IN}	
	L	L	X	L	L	D_{IN}	D_{IN}	

图 10：SRAM 控制信号真值表

一种 SRAM 写操作时序如下：

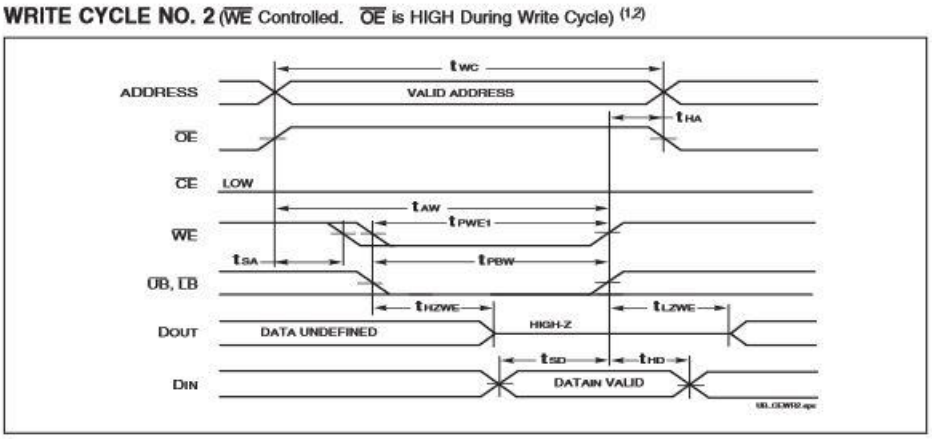


图 11：操作时序示意图

b) 新建工程，即 SRAM 写读文件。

c) SRAM 模块代码如下：

```
`timescale 1ns / 1ps
module changeAddr(
input clk100mhz,
input changeaddr,
output reg[18:0]addrBus1
      );
      always@(posedge clk100mhz)begin
      if(changeaddr==0) addrBus1 = 19'b100_0000_0000_0000_0000;
      else if(changeaddr)    addrBus1 = 19'b100_0000_0000_0000_0001;
      end
endmodule

`timescale 1ns / 1ps
module clkDiv(
input clk100mhz,
output clk190hz,
output clk6hz,
output clk3hz
);
      reg [25:0] count = 0;
      assign clk190hz =count[18];
      assign clk3hz =count[25];
      assign clk6hz =count[24];
      always@(posedge clk100mhz)
                  count <= count + 1;
endmodule
```

```
`timescale 1ns / 1ps
module GPU(
input [31:0]number,
input clk3hz,
input clr,
output [15:0]dataBus
      );
      reg[31:0] msgArray;
      assign dataBus = msgArray[31:16];
      always @ (posedge clk3hz or posedge clr)
                  if(!clr )
                  msgArray <= number;
                  else begin
                  msgArray[3:0] <= msgArray[31:28];
                  msgArray[31:4]<=msgArray[27:0];
                  end
endmodule

`timescale 1ns / 1ps
module segMsg(
input start,
input clk190hz,
input [15:0] dataBus,
input [15:0] dataBus1,
output reg [3:0] pos,
output reg [3:0] pos1,
output reg [7:0] seg,
output reg [7:0] seg1
);
reg  [1:0]     posC;
reg  [1:0]     posC1;
reg  [3:0]     dataP;
reg  [3:0]     dataP1;
always @(posedge clk190hz )begin
```

```
case(posC1)
        0:begin
        pos1    <=  4'b0001;
            if(start) dataP1 <=  dataBus1[3:0];
            else dataP1<= 10;
        end
        1:begin
        pos1    <=  4'b0010;
        if(start)   dataP1      <=      dataBus1[7:4];
        else dataP1<= 10;
        end
        2:begin
        pos1    <=4'b0100;
        if(start) dataP1        <=      dataBus1[11:8];
        else dataP1<= 10;
        end
        3:begin
        pos1    <=  4'b1000;
        if(start) dataP1        <=      dataBus1[15:12];
        else dataP1<= 10;
        end
    endcase
    posC1 = posC1   + 1;
    case(posC)
            0:begin
            pos     <=      4'b0001;
            if(start==0) dataP      <=      dataBus[3:0];
            else dataP<= 10;
            end
            1:begin
            pos     <=      4'b0010;
            if(start==0) dataP      <=      dataBus[7:4];
            else dataP<= 10;
            end
            2:begin
            pos     <=4'b0100;
            if(start==0) dataP      <=      dataBus[11:8];
            else dataP<= 10;
            end
            3:begin
            pos     <=      4'b1000;
            if(start==0) dataP      <=      dataBus[15:12];
            else dataP<= 10;
            end
        endcase
```

```
            posC = posC        + 1;
        end
        always@(dataP1 or dataP) begin
        case(dataP1)
        0: seg1 = 8'b0011_1111;
        1: seg1 = 8'b0000_0110;
        2: seg1 = 8'b0101_1011;
        3: seg1 = 8'b0100_1111;
        4: seg1 = 8'b0110_0110;
        5: seg1 = 8'b0110_1101;
        6: seg1 = 8'b0111_1101;
        7: seg1 = 8'b0000_0111;
        8: seg1 = 8'b0111_1111;
        9: seg1 = 8'b0110_1111;
        10:seg1 = 8'b0100_0000;
        11:seg1 = 8'b0000_0000;
        default: seg1 = 8'b0000_1000;
        endcase
            case(dataP)
            0: seg = 8'b0011_1111;
            1: seg = 8'b0000_0110;
            2: seg = 8'b0101_1011;
            3: seg = 8'b0100_1111;
            4: seg = 8'b0110_0110;
            5: seg = 8'b0110_1101;
            6: seg = 8'b0111_1101;
            7: seg = 8'b0000_0111;
            8: seg = 8'b0111_1111;
            9: seg = 8'b0110_1111;
            10:seg = 8'b0100_0000;
            11:seg = 8'b0000_0000;
            default: seg = 8'b0000_1000;
            endcase
            end
endmodule
```

```
`timescale 1ns / 1ps
module sram(
input clk,clr,rEn,
input [31:0] iDataBus,
inout [15:0] dataBus,
output reg[18:0] addrBus,
output ce,ub,lb,
output reg oe,w,
output reg [31:0]oDataBus
);

reg [15:0] tDataBus;
reg [31:0] rDataBus;
reg [3:0] countR ;
reg [3:0] countW ;

assign dataBus    = w ? 16'hzzzz : tDataBus;
assign ce = 0;
assign ub = 0;
assign lb = 0;

always @(posedge clk or posedge clr)
      if(!clr)begin
            countW <= 0;
            countR <= 0;
      end
      else begin
            case(countW)
                  2:begin
                        w <= 1'b0;
                        oe <= 1'b1;
                        tDataBus <= iDataBus[15:0];
                        addrBus <= 19'd100_000;
                     end
                  4:begin
                        w <= 1'b0;
                        oe <= 1'b1;
                        tDataBus <= iDataBus[31:16];
                        addrBus <= 19'd100_001;
                   end
                  default:begin
                        w <= 1'b1;
                        oe <= 1'b0;
                  end
```

```
            endcase
            case(countR)
                2:begin
                    w <= 1'b1;
                    addrBus<= 19'd100_000;
                    rDataBus[15:0] <= dataBus;
                  end
                4:begin
                     w <= 1'b1;
                     addrBus<= 19'd100_001;
                     rDataBus[31:16] <= dataBus;
                  end
                8:begin
                    oDataBus = rDataBus;
                end
             endcase
           if((countW<8)&&(rEn==0))
                countW <= countW + 1;
           if((countR<12)&&(rEn==1))
                countR <= countR + 1;
      end
endmodule

`timescale 1ns / 1ps
module sramR(
input clk100mhz,
input w,ce,
input [15:0] dataBus,
output [15:0]LED
      );
      reg [15:0]data;
      always@(posedge clk100mhz)
      begin
      if(w&&(ce==0))begin
      data <= dataBus;
      end
      end
      assign LED=data;
endmodule
```

```
`timescale 1ns / 1ps
module sramWR(
input clk100mhz,
input    w,ce,changeaddr,
input [15:0]number,
inout [15:0] dataBus,
output reg[15:0]LED,
output oe,lb,ub,
output reg[ 18:0]addrBus
      );
      reg [15:0] tDataBus;
      assign oe=0;assign lb =0;assign ub =0;
      assign dataBus    = w ? 16'hzzzz : tDataBus;
      always@(posedge clk100mhz)begin
            if(w==0&&ce==0)begin
               tDataBus <= number;
            end
            else if(w&&(ce==0))begin
            LED <= dataBus;
            end
            case(changeaddr)
            1: addrBus= 19'b100_0000_0000_0000_0000;
            0: addrBus= 19'b100_0000_0000_0000_0001;
            endcase
            end
endmodule

`timescale 1ns / 1ps
module xiaodou(
      input clk_190Hz,
      input btnIn,
      output btnOut
      );
        reg delay1;
        reg delay2;
        reg delay3;

            always @(posedge clk_190Hz)begin
                  delay3 = delay2;
                  delay2 = delay1;
                  delay1 = btnIn;
            end
            assign btnOut = delay1 & delay2 & delay3;
endmodule
```

d) RTL 门级结构图

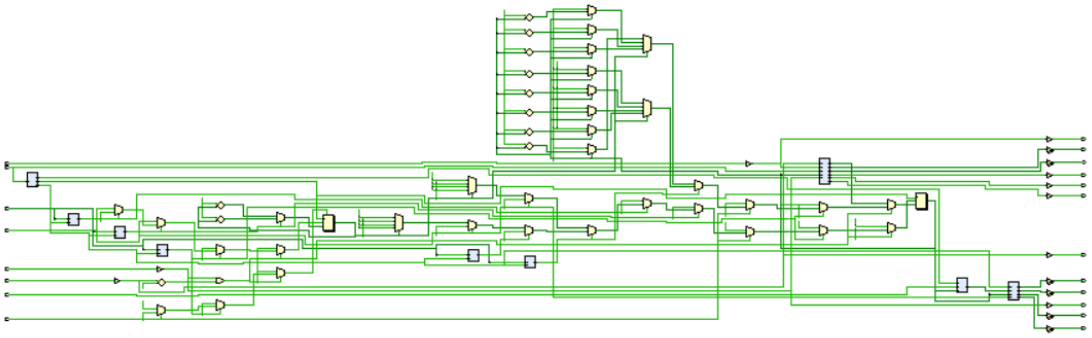


图 12： SRAM 的 RTL 门级结构图

e) SRAM 管脚约束代码

```
set_property PACKAGE_PIN R15 [get_ports {click[4]}]
set_property PACKAGE_PIN R11 [get_ports {click[3]}]
set_property PACKAGE_PIN V1 [get_ports {click[2]}]
set_property PACKAGE_PIN R17 [get_ports {click[1]}]
set_property PACKAGE_PIN U4 [get_ports {click[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {click[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {click[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {click[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {click[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {click[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {pos1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk100mhz]
set_property IOSTANDARD LVCMOS33 [get_ports clr]
set_property IOSTANDARD LVCMOS33 [get_ports res]
set_property IOSTANDARD LVCMOS33 [get_ports start]
set_property PACKAGE_PIN G2 [get_ports {pos[3]}]
set_property PACKAGE_PIN C2 [get_ports {pos[2]}]
set_property PACKAGE_PIN C1 [get_ports {pos[1]}]
set_property PACKAGE_PIN H1 [get_ports {pos[0]}]
set_property PACKAGE_PIN G1 [get_ports {pos1[3]}]
set_property PACKAGE_PIN F1 [get_ports {pos1[2]}]
set_property PACKAGE_PIN E1 [get_ports {pos1[1]}]
set_property PACKAGE_PIN G6 [get_ports {pos1[0]}]
set_property PACKAGE_PIN D5 [get_ports {seg[7]}]
set_property PACKAGE_PIN B3 [get_ports {seg[5]}]
set_property PACKAGE_PIN B1 [get_ports {seg[3]}]
set_property PACKAGE_PIN H2 [get_ports {seg1[7]}]
set_property PACKAGE_PIN B2 [get_ports {seg[6]}]
set_property PACKAGE_PIN A1 [get_ports {seg[4]}]
set_property PACKAGE_PIN A3 [get_ports {seg[2]}]
set_property PACKAGE_PIN A4 [get_ports {seg[1]}]
set_property PACKAGE_PIN B4 [get_ports {seg[0]}]
set_property PACKAGE_PIN D2 [get_ports {seg1[6]}]
set_property PACKAGE_PIN E2 [get_ports {seg1[5]}]
set_property PACKAGE_PIN F3 [get_ports {seg1[4]}]
set_property PACKAGE_PIN F4 [get_ports {seg1[3]}]
set_property PACKAGE_PIN D3 [get_ports {seg1[2]}]
set_property PACKAGE_PIN E3 [get_ports {seg1[1]}]
set_property PACKAGE_PIN D4 [get_ports {seg1[0]}]
set_property PACKAGE_PIN P17 [get_ports clk100mhz]
set_property PACKAGE_PIN P5 [get_ports clr]
set_property PACKAGE_PIN P3 [get_ports res]
set_property PACKAGE_PIN P4 [get_ports start]

set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[18]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[17]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[16]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[15]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[14]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[13]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[12]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {addrBus[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[15]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[14]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[13]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[12]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dataBus1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports ce]
set_property IOSTANDARD LVCMOS33 [get_ports control]
set_property IOSTANDARD LVCMOS33 [get_ports clt]
set_property IOSTANDARD LVCMOS33 [get_ports lb]
set_property IOSTANDARD LVCMOS33 [get_ports oe]
set_property PACKAGE_PIN U17 [get_ports {dataBus1[0]}]
set_property PACKAGE_PIN U18 [get_ports {dataBus1[1]}]
set_property PACKAGE_PIN U16 [get_ports {dataBus1[2]}]
set_property PACKAGE_PIN V17 [get_ports {dataBus1[3]}]
set_property PACKAGE_PIN T11 [get_ports {dataBus1[4]}]
set_property PACKAGE_PIN U11 [get_ports {dataBus1[5]}]
```

```
set_property PACKAGE_PIN U12 [get_ports {dataBus1[6]}]
set_property PACKAGE_PIN V12 [get_ports {dataBus1[7]}]
set_property PACKAGE_PIN V10 [get_ports {dataBus1[8]}]
set_property PACKAGE_PIN V11 [get_ports {dataBus1[9]}]
set_property PACKAGE_PIN U14 [get_ports {dataBus1[10]}]
set_property PACKAGE_PIN V14 [get_ports {dataBus1[11]}]
set_property PACKAGE_PIN T13 [get_ports {dataBus1[12]}]
set_property PACKAGE_PIN U13 [get_ports {dataBus1[13]}]
set_property PACKAGE_PIN T9 [get_ports {dataBus1[14]}]
set_property PACKAGE_PIN T10 [get_ports {dataBus1[15]}]
set_property PACKAGE_PIN T15 [get_ports {addrBus[0]}]
set_property PACKAGE_PIN T14 [get_ports {addrBus[1]}]
set_property PACKAGE_PIN N16 [get_ports {addrBus[2]}]
set_property PACKAGE_PIN N15 [get_ports {addrBus[3]}]
set_property PACKAGE_PIN M17 [get_ports {addrBus[4]}]
set_property PACKAGE_PIN M16 [get_ports {addrBus[5]}]
set_property PACKAGE_PIN P18 [get_ports {addrBus[6]}]
set_property PACKAGE_PIN N17 [get_ports {addrBus[7]}]
set_property PACKAGE_PIN P14 [get_ports {addrBus[8]}]
set_property PACKAGE_PIN N14 [get_ports {addrBus[9]}]
set_property PACKAGE_PIN T18 [get_ports {addrBus[10]}]
set_property PACKAGE_PIN R18 [get_ports {addrBus[11]}]
set_property PACKAGE_PIN M13 [get_ports {addrBus[12]}]
set_property PACKAGE_PIN R13 [get_ports {addrBus[13]}]
set_property PACKAGE_PIN R12 [get_ports {addrBus[14]}]
set_property PACKAGE_PIN M18 [get_ports {addrBus[15]}]
set_property PACKAGE_PIN L18 [get_ports {addrBus[16]}]
set_property PACKAGE_PIN L16 [get_ports {addrBus[17]}]
set_property PACKAGE_PIN L15 [get_ports {addrBus[18]}]
set_property PACKAGE_PIN V15 [get_ports ce]
set_property PACKAGE_PIN P2 [get_ports clt]
set_property PACKAGE_PIN R2 [get_ports control]
set_property PACKAGE_PIN R10 [get_ports lb]
set_property PACKAGE_PIN T16 [get_ports oe]
set_property PACKAGE_PIN V16 [get_ports w]
set_property PACKAGE_PIN R16 [get_ports ub]
set_property IOSTANDARD LVCMOS33 [get_ports ub]
set_property IOSTANDARD LVCMOS33 [get_ports w]

set_property IOSTANDARD LVCMOS33 [get_ports changeaddr]
set_property PACKAGE_PIN M4 [get_ports changeaddr]

set_property PACKAGE_PIN J3 [get_ports changeaddr1]
set_property IOSTANDARD LVCMOS33 [get_ports changeaddr1]
```

f)　FPGA 板级验证图像，见实验结果与分析所示

四：实验结果与分析

（1）　滚动显示数码管顶层模块示意图：

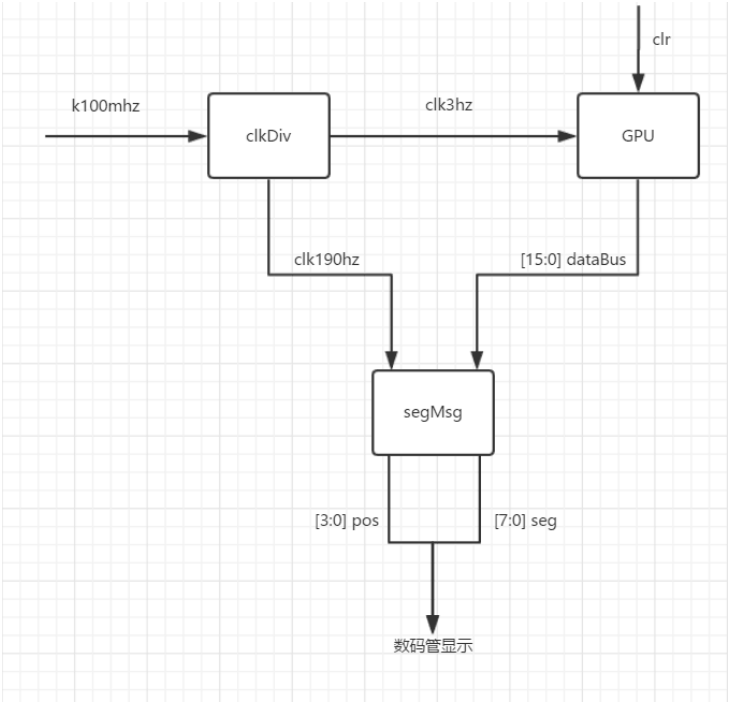


图 13　模块框图示意图

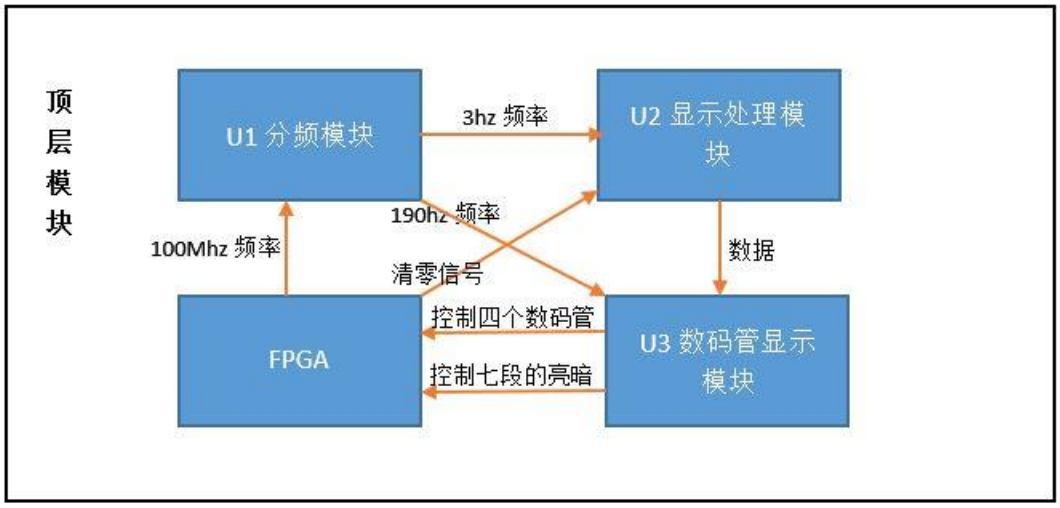


图 14：模块示意图

（2）　板级验证图像

i.　实验 3.1 FPGA 板级验证图像

图 15： 在七段数码管上滚动显示学号

ii. 实验 3.2FPGA 板级验证图像

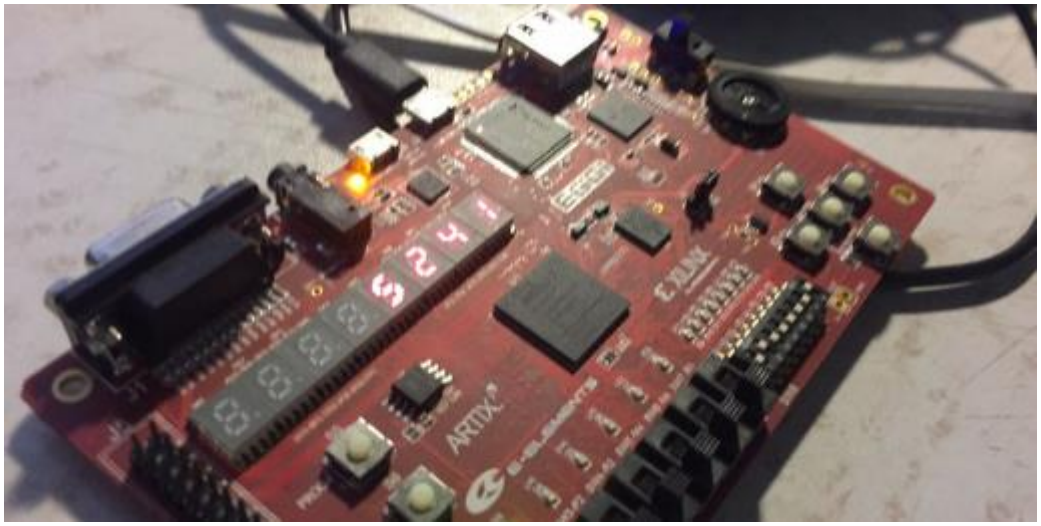


图 16： 在七段数码管上滚动显示学号

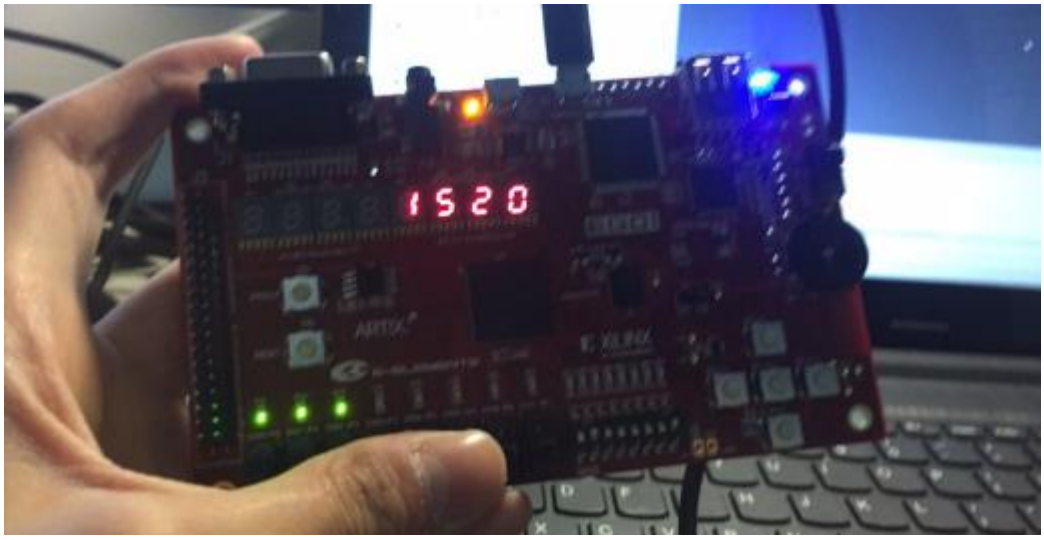


图 17： 通过按键输入学号，并循环显示

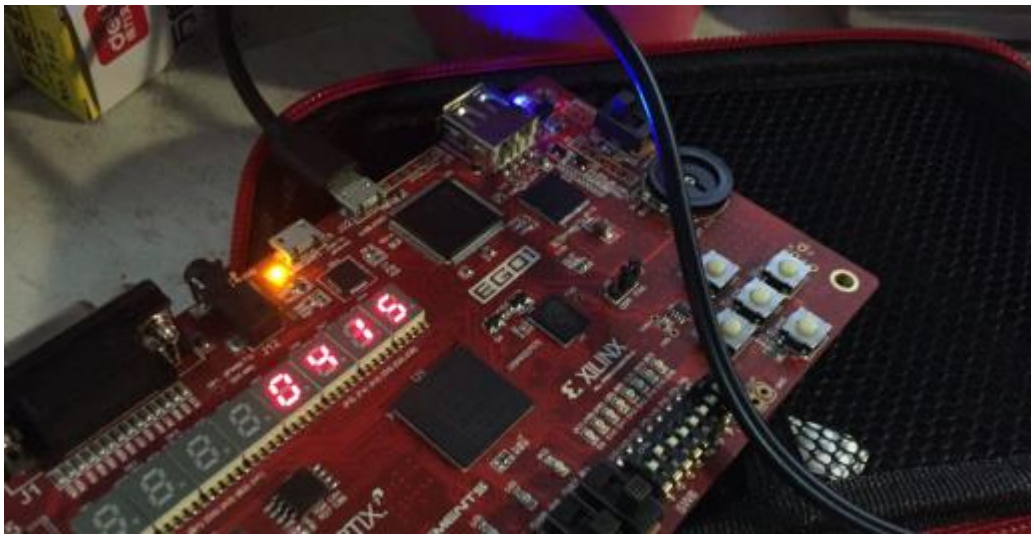


图 18： 通过按键输入学号，并循环显示

iii. 实验 3.3 SRAM 芯片数据存取示意图情况

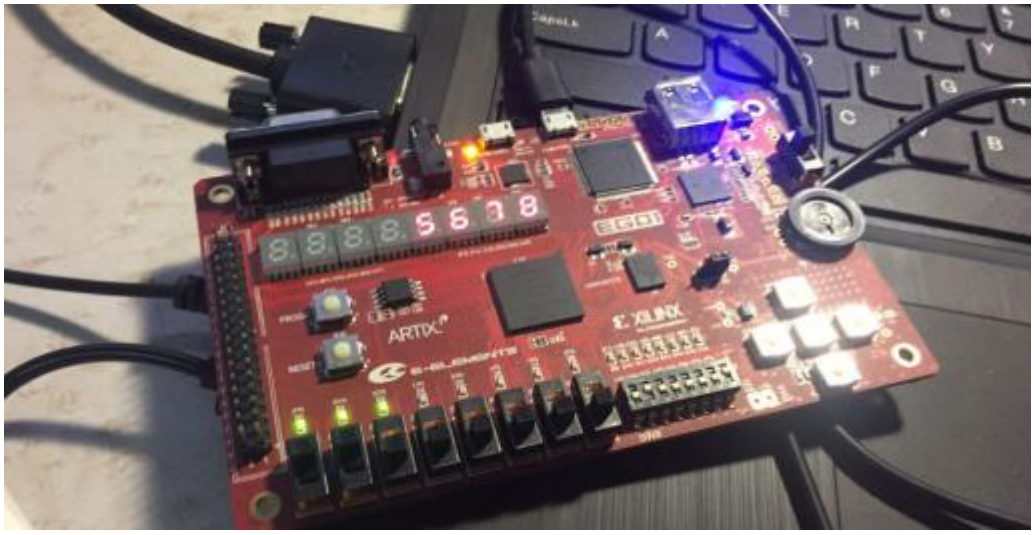


图 19： SRAM 芯片数据存取

五：结论（讨论）

1、实验结论

电路的输出同时依赖于电路之前的状态和当前输入值的一类常用电路，这种电路称为时序电路。在大部分的情况下，由一个时钟信号控制的时序电路成为同步时序电路，而没有时钟控制信号的时序的电路称为异步时序电路。同步电路易于设计，并且得到广泛的实际应用。但是缺点也是显而易见的，功耗大，时钟是高频率信号，而时钟必须分布到各个触发器，而不管触发器是否要工作，并且，由于最大的时钟频率是由电路中最慢的逻辑路径所决定的，限制了工作的最高频率。但是，相比之下，同步时序电路还是可以获得比异步时序电路更高的工作可靠性和工作速度。

SRAM 是静态随机存储器，通常作为 CPU 的存储器使用。板卡搭载的 IS61WV12816DBLL SRAM 芯片，总容量 2Mbit。该 SRAM 为异步式 SRAM，最高存取时间可达 8ns。操控简单，易于读写。RAM (Random Access Memory 随机存贮器）是指通过指令可以随机地、个别地对每个存储单元进行访问、访问所需时间基本固定、且与存储单元地址无关的可以读写的存储器。几乎所有的计算机系统和智能电子产品中，都是采用 RAM 作为主存。

在系统内部，RAM 是仅次于 CPU 的最重要的器件之一。它们之间的关系，

就如人的大脑中思维与记忆的关系一样，实际上是密不可分的。但在计算机内部，它们却是完全独立的器件，沿着各自的道路向前发展。在 CPU 和 RAM 之间有一条高速数据通道，CPU 所要处理的数据和指令必须先放到 RAM 中等待。而 CPU 也把大部分正在处理的中间数据暂时放置在 RAM 中，这就要求 RAM 和 CPU 之间的速度保持匹配。

2、讨论

本次试验比较成功，通过本次试验，我掌握了 SRAM 设计方法与测试方法和 FPGA 板上数码的设置方法，还掌握了常用中规模集成计数器的逻辑功能和使用方法，在整个大模块框图下，把每个功能划分成不同的小模块，其中，很大一部分都是组合逻辑电路的关键知识的结合实现不同的功能，通过 wire 进行连接，最终完成了整个系统的设计，使我对数字电路设计有了更加深刻的理解，加深了我对相关数字逻辑理论的认识。

本次实验中，关于分频电路，SRAM 设计上都存在另外的设计思路，同时，数字滚动的设计也体现了数字电路设计中的多样性，因此在实际设计中，我们应该努力拓展自己的思路，应用多种角度去思考，以便设计出更加合理高效的电路。

六、教师评审

教师评语	实验成绩
签名： 日期：	