**实验报告2 输入输出部件的使用编程**

# 1. 基本信息

## 1.1 小组成员信息

|  |  |  |
| --- | --- | --- |
| 序号 | 姓名 | 学号 |
| ① | 钟添芸 | 3180103009 |
| ② | 江昊瀚 | 3180101995 |
| ③ | 徐震 | 3180105504 |
| ④ | 麦昌楷 | 3180101982 |

## 1.2 声明

* 本工程目标平台与物理测试平台为Sword Kintex7实验平台
* 本工程主要代码采用Verilog编写，同时辅助有Schematic图形
* 本工程综合编译环境为ISE-14.7 (nt64)
* 本工程Simulation环境为Windows10 (64bit)

# 2. 实验要求

注：该节以下内容引用自实验要求Word文档

## 2.1 数据准备-要求

你的学号通过手工计算得到对应的16进制数A，A的长度要求为8个16进制数位（例如学号3160105212作为10进制数，用windows系统自带的计算器转换成16进制是 0xBC5B60FC，于是A=0xBC5B60FC），多于8个则截掉高位，少于8个则高位补0。

## 2.2 工程1-要求

将2.1中得到的16进制数A显示在8个7段数码管上（7段数码管右边是低位），A的4个低位数位（例如上面的0x60FC）转换成2进制数B（例如0110 0000 1111 1100），把B的16个2进制数位显示在16个LED上，LED位于拨动开关上部，右边是低位。

## 2.3 工程2-要求

16个拨动开关的值显示在16个对应的LED上，上拨为1（LED亮），下拨为0（LED不亮）。

## 2.4 工程3-要求

5\*5的按键小键盘只用左边4列，每个按键下面都有位置标识，例如左下角的按键是BTNX4Y0，其位置坐标就是（4,0）; 右上角的按键是BTNX0Y3，其位置坐标就是（0,3）其余按键的坐标（M,N）按此规则获取，每按1个按键，把相应的坐标中的N显示在8个7段数码管的最右面1个数码管上，并一直显示，直到按另一键为止。

# 3. 实验步骤

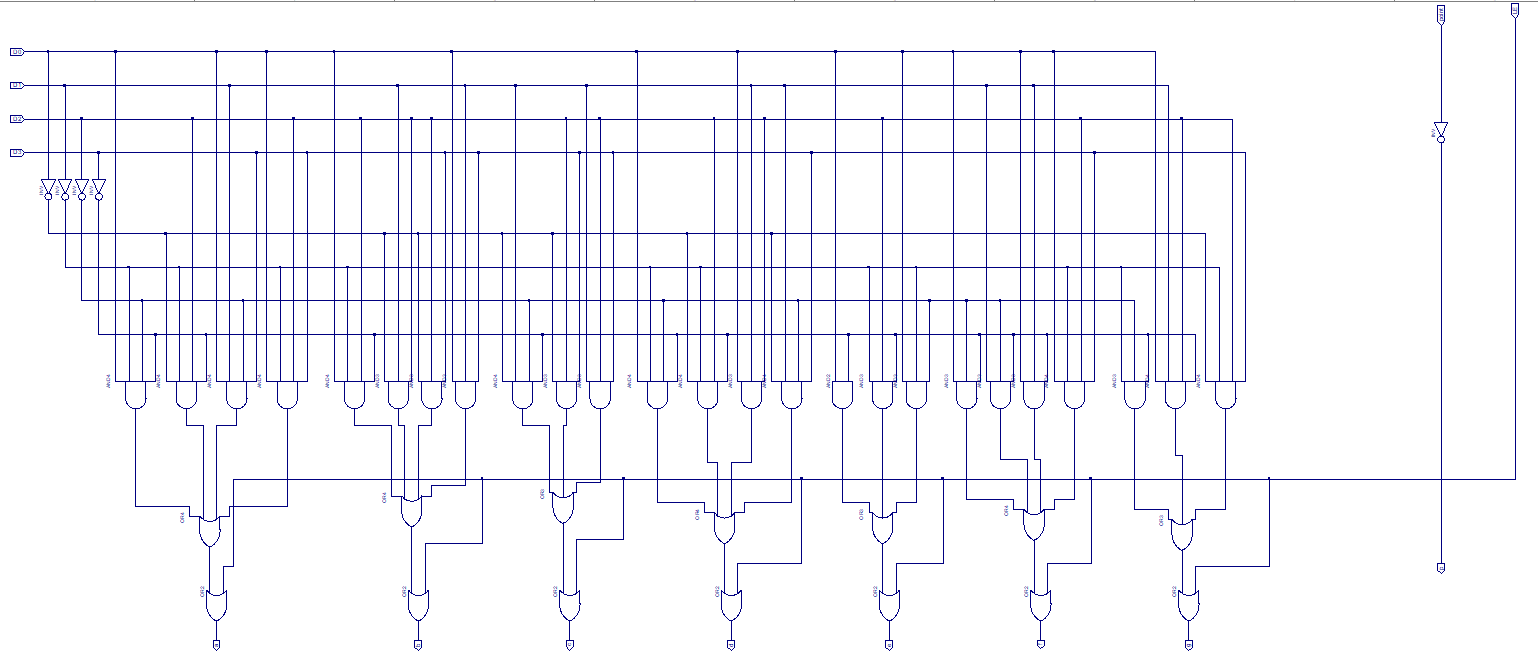
注：本节工程代码大部分来自小组成员于2019年秋冬学期数字逻辑设计课程中撰写的实验代码

## 3.1 数据准备-步骤

|  |  |
| --- | --- |
| 学号（十进制） | 3180103009 |
| 学号（十六进制） | BD8C 8561 |
| 学号（二进制） | 1011 1101 1000 1100 1000 0101 0110 0001 |

## 3.2 工程1-步骤

* 创建ISE工程，工程名为CO\_Lab2\_Prg1
* 添加并编辑Schematic文件MC14495\_ZJU.sch（详见工程目录）：



* 添加并编辑Verilog文件Hex2Seg.v：

|  |
| --- |
| `timescale 1ns / 1ps  // transform single Hex number to 7-segment  module Hex2Seg(  input[3:0] Hex,  input LE,  input point,  input flash,  output[7:0] Segment  );  wire en = LE & flash; // flash logic    // use MC14495 module to display a Hex number in 7-segment  MC14495\_ZJU MSEG(.D3(Hex[3]), .D2(Hex[2]), .D1(Hex[1]), .D0(Hex[0]), .LE(en), .point(point),  .a(a), .b(b), .c(c), .d(d), .e(e), .f(f), .g(g), .p(p));    assign Segment = {a, b, c, d, e, f, g, p};  endmodule |

* 添加并编辑Verilog文件HexTo8Seg.v：

|  |
| --- |
| `timescale 1ns / 1ps  // transform 8 Hex numbers to 7-segment  module HexTo8SEG (  input [31:0] Hexs,  input [7:0] points,  input [7:0] LES,  input flash,  output[63:0] SEG\_TXT  );  // output every number  Hex2Seg HTS0 (Hexs[31:28], LES[7], points[7], flash, SEG\_TXT[7:0]);  Hex2Seg HTS1 (Hexs[27:24], LES[6], points[6], flash, SEG\_TXT[15:8]);  Hex2Seg HTS2 (Hexs[23:20], LES[5], points[5], flash, SEG\_TXT[23:16]);  Hex2Seg HTS3 (Hexs[19:16], LES[4], points[4], flash, SEG\_TXT[31:24]);  Hex2Seg HTS4 (Hexs[15:12], LES[3], points[3], flash, SEG\_TXT[39:32]);  Hex2Seg HTS5 (Hexs[11:8], LES[2], points[2], flash, SEG\_TXT[47:40]);  Hex2Seg HTS6 (Hexs[7:4], LES[1], points[1], flash, SEG\_TXT[55:48]);  Hex2Seg HTS7 (Hexs[3:0], LES[0], points[0], flash, SEG\_TXT[63:56]);    endmodule |

* 添加并编辑Verilog文件Display.v：

|  |
| --- |
| `timescale 1ns / 1ps  // display the Hex number using the eight 7-segments  module Display(input wire clk,  input wire rst,  input wire Start,  input wire flash,  input wire [31:0]Hexs,  input wire [7:0]point,  input wire [7:0]LES,  output wire seg\_clk,  output wire seg\_sout,  output wire seg\_pen,  output wire seg\_clrn);    wire [63:0] PData;  wire [63:0] SM3\_Din;    HexTo8SEG SM1(.flash(flash), .Hexs(Hexs), .points(point),  .LES(LES), .SEG\_TXT(PData));    P2S #(.DATA\_BITS(64), .DATA\_COUNT\_BITS(4))  P7\_P2S (.clk(clk), .rst(rst), .Start(Start),  .PData(PData), .s\_clk(seg\_clk), .s\_clrn(seg\_clrn),  .sout(seg\_sout), .EN(seg\_pen));  endmodule |

* 添加并编辑Verilog文件SHIFT64.v：

|  |
| --- |
| `timescale 1ns / 1ps  // shifter with a changable length  module SHIFT64(clk, SR, SL, S1, S0, D, Q);  input wire clk;  input wire SR;  input wire SL;  input wire S1;  input wire S0;  input wire [DATA\_BITS:0]D;  output reg [DATA\_BITS:0]Q;    parameter DATA\_BITS = 16;  parameter DATA\_COUNT\_BITS = 4;    always @(posedge clk)  begin  case ({S1,S0})  2'b00: Q <= Q; //S1S0 = 00, hold  2'b01: Q <= {SR, Q[DATA\_BITS: 1]}; //S1S0 = 01, right shift  2'b10: Q <= {Q[DATA\_BITS - 1: 0], SL}; //S1S0 = 10, left shift  2'b11: Q <= D; //parallel to input  endcase  end  endmodule |

* 添加并编辑Verilog文件P2S.v：

|  |
| --- |
| `timescale 1ns / 1ps  module P2S(clk, rst, Start, PData, s\_clk, s\_clrn, sout, EN);  //parallel to serial  input wire clk;  input wire rst;  input wire Start;  input wire [DATA\_BITS-1: 0] PData;  output wire s\_clk;  output wire s\_clrn;  output wire sout;  output reg EN;    parameter  DATA\_BITS = 64, // data length  DATA\_COUNT\_BITS = 4, // data shift bits  DIR = 1; // Shift direction = 0 左移    wire S1, S0, SL, SR, clkn;  wire [DATA\_BITS:0] D, Q;  reg [1:0] Go = 00, S = 00;  assign clkn = ~clk;  assign {SR, SL} = 2'b11; // 左移右移初值为1，用于传输标志  assign {S1, S0} = DIR ? {S[0], S[1]} : S; // 调整移位方向  assign D = DIR ? {1'b0, PData} : {PData, 1'b0}; // 置移位末端标志“0”  wire finish = DIR ? &Q[DATA\_BITS:1] : &Q[DATA\_BITS - 1:0]; // 移位传输结束：全“1”  assign sout = DIR ? Q[0] : Q[DATA\_BITS]; // 串行输出，右移输出Q[0]    SHIFT64 #(.DATA\_BITS(DATA\_BITS)) // 调用移位寄存器，宽度=DATA\_BITS  PTOS (.clk(clkn), .SR(SR), .SL(SL),  .S1(S1), .S0(S0), .D(D), .Q(Q));    always @(posedge clkn) // 采样Start上升沿  Go <= {Go[0], Start};  assign shift = (Go == 2'b01) ? 1'b1 : 1'b0; //移位启动采样    always @(posedge clkn or posedge rst)  begin  if (rst)  begin  EN <= 1;  S <= 2'b11;  end else  begin  if (shift) // 并行置入  begin  EN <= 0;  S <= 2'b11;  end else  begin  if (!finish) // 启动移位输入  begin  EN <= 0;  S <= 2'b10;  end else  begin // 结束移位  EN <= 1;  S <= 2'b00;  end  end  end  end  assign s\_clk = finish | clkn; // 禁止74LS164时钟  assign s\_clrn = 1;  endmodule |

* 添加并编辑Verilog文件GPIO.v：

|  |
| --- |
| `timescale 1ns / 1ps  module GPIO(input wire clk,  input wire rst,  input wire Start,  input wire [15:0] P\_Data, // input data  output wire led\_clk, // output LED signal  output wire led\_sout,  output wire led\_clrn,  output wire led\_pen);    P2S #(.DATA\_BITS(16), .DATA\_COUNT\_BITS(4)) // use P2S to output  LED\_P2S (.clk(clk), .rst(rst), .Start(Start),  .PData(P\_Data), .s\_clk(led\_clk), .s\_clrn(led\_clrn),  .sout(led\_sout), .EN(led\_pen));  endmodule |

* 添加并编辑Verilog文件clkdiv.v：

|  |
| --- |
| `timescale 1ns / 1ps  module clkdiv(input clk,  output reg[31:0]clkdiv);  always @ (posedge clk) // increase 1 every tick  clkdiv <= clkdiv + 1'b1;  endmodule |

* 添加并编辑Verilog文件Top\_Lab2\_Prg1.v：

|  |
| --- |
| `timescale 1ns / 1ps  module Top\_Lab2\_Prg1(input wire clk\_100mhz, // clock signal  output wire SEGCLK, // 7-segment output signal  output wire SEGDT,  output wire SEGEN,  output wire SEGCLR, // LED input signal  output wire LEDCLK,  output wire LEDDT,  output wire LEDEN,  output wire LEDCLR);      wire [31:0]SID = 32'hBD8C\_8561; // SID  wire [31:0]Div; // clock div signal    // clock div module  clkdiv clkdiv\_M(.clk(clk\_100mhz),  .clkdiv(Div[31:0]));    // 7-segment output module  Display Display\_M(.LES(8'h00),  .point(8'h00),  .Hexs(SID), // connect SID to 7-segment  .flash(Div[25]),  .rst(1'b0),  .Start(Div[10]),  .clk(clk\_100mhz),  .seg\_clrn(SEGCLR),  .seg\_sout(SEGDT),  .seg\_pen(SEGEN),  .seg\_clk(SEGCLK));    // LED output module  GPIO GPIO\_M (.clk(clk\_100mhz),  .rst(1'b0),  .Start(Div[20]),  .P\_Data(SID[15:0]), // connect SID to LED  .led\_clk(LEDCLK),  .led\_sout(LEDDT),  .led\_clrn(LEDCLR),  .led\_pen(LEDEN));    endmodule |

* 添加并编辑UCF文件Top.ucf：

|  |
| --- |
| NET "clk\_100mhz" LOC = AC18 | IOSTANDARD = LVCMOS18;  NET "clk\_100mhz" TNM\_NET = TM\_CLK;  TIMESPEC TS\_CLK\_100M = PERIOD "TM\_CLK" 10 ns HIGH 50%;  #七段码串行接口  NET "SEGCLK" LOC = M24 | IOSTANDARD = LVCMOS33 ;  NET "SEGCLR" LOC = M20 | IOSTANDARD = LVCMOS33 ;  NET "SEGDT" LOC = L24 | IOSTANDARD = LVCMOS33 ;  NET "SEGEN" LOC = R18 | IOSTANDARD = LVCMOS33 ;  #SWord LED移位接口 SPLIO输出，低16位对应LED  NET "LEDCLK" LOC = N26 | IOSTANDARD = LVCMOS33 ;  NET "LEDCLR" LOC = N24 | IOSTANDARD = LVCMOS33 ;  NET "LEDDT" LOC = M26 | IOSTANDARD = LVCMOS33 ;  NET "LEDEN" LOC = P18 | IOSTANDARD = LVCMOS33 ; |

* 检查并调整层次结构：

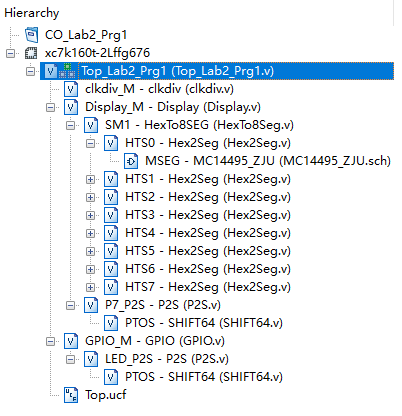


图 1 工程1层次结构图

* 以Top\_Lab2\_Prg1.v为顶层进行Generate Programming File
* 将top\_lab2\_prg1.bit文件下载至Sword Kintex7平台进行物理测试

## 3.3 工程2-步骤

* 创建ISE工程，工程名为CO\_Lab2\_Prg2
* 将工程1中的SHIFT64.v, P2S.v, GPIO.v, clkdiv.v复制并添加进本工程（因代码一致不再在本报告中赘述，可详见工程目录）
* 添加并编辑Top\_Lab2\_Prg2.v

|  |
| --- |
| `timescale 1ns / 1ps  module Top\_Lab2\_Prg2(input wire clk\_100mhz, // clock signal  input wire [15:0]SW, // switch input  output wire LEDCLK, // LED output  output wire LEDDT,  output wire LEDEN,  output wire LEDCLR);    wire [31:0]Div; // clock div    // clock div module  clkdiv clkdiv\_M(.clk(clk\_100mhz),  .clkdiv(Div[31:0]));    // LED output module  GPIO GPIO\_M (.clk(clk\_100mhz),  .rst(1'b0),  .Start(Div[20]),  .P\_Data(SW), // connect switch signal to LED  .led\_clk(LEDCLK),  .led\_sout(LEDDT),  .led\_clrn(LEDCLR),  .led\_pen(LEDEN));  endmodule |

* 添加并编辑UCF文件Top.ucf：

|  |
| --- |
| NET "clk\_100mhz" LOC = AC18 | IOSTANDARD = LVCMOS18;  NET "clk\_100mhz" TNM\_NET = TM\_CLK;  TIMESPEC TS\_CLK\_100M = PERIOD "TM\_CLK" 10 ns HIGH 50%;  #SWord LED移位接口 SPLIO输出，低16位对应LED  NET "LEDCLK" LOC = N26 | IOSTANDARD = LVCMOS33 ;  NET "LEDCLR" LOC = N24 | IOSTANDARD = LVCMOS33 ;  NET "LEDDT" LOC = M26 | IOSTANDARD = LVCMOS33 ;  NET "LEDEN" LOC = P18 | IOSTANDARD = LVCMOS33 ;  #switch  NET "SW[0]" LOC = AA10 | IOSTANDARD = LVCMOS15; #to AN[3]  NET "SW[1]" LOC = AB10 | IOSTANDARD = LVCMOS15; #to AN[2]  NET "SW[2]" LOC = AA13 | IOSTANDARD = LVCMOS15; #to AN[1]  NET "SW[3]" LOC = AA12 | IOSTANDARD = LVCMOS15; #to AN[0]  NET "SW[4]" LOC = Y13 | IOSTANDARD = LVCMOS15; #D3  NET "SW[5]" LOC = Y12 | IOSTANDARD = LVCMOS15; #D2  NET "SW[6]" LOC = AD11 | IOSTANDARD = LVCMOS15; #D1  NET "SW[7]" LOC = AD10 | IOSTANDARD = LVCMOS15; #D0  NET "SW[8]" LOC = AE10 | IOSTANDARD = LVCMOS15 ;  NET "SW[9]" LOC = AE12 | IOSTANDARD = LVCMOS15 ;  NET "SW[10]" LOC = AF12 | IOSTANDARD = LVCMOS15 ;  NET "SW[11]" LOC = AE8 | IOSTANDARD = LVCMOS15 ;  NET "SW[12]" LOC = AF8 | IOSTANDARD = LVCMOS15 ;  NET "SW[13]" LOC = AE13 | IOSTANDARD = LVCMOS15 ;  NET "SW[14]" LOC = AF13 | IOSTANDARD = LVCMOS15 ;  NET "SW[15]" LOC = AF10 | IOSTANDARD = LVCMOS15 ; |

* 检查并调整层次结构：

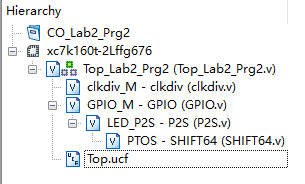


图 1 工程2层次结构图

* 以Top\_Lab2\_Prg2.v为顶层进行Generate Programming File
* 将top\_lab2\_prg2.bit文件下载至Sword Kintex7平台进行物理测试

## 3.4 工程3-步骤

* 创建ISE工程，工程名为CO\_Lab2\_Prg3
* 将工程1中的SHIFT64.v, P2S.v, Hex2Seg.v, HexTo8SEG.v, Display.v, clkdiv.v复制并添加进本工程（因代码一致不再在本报告中赘述，可详见工程目录）
* 添加并编辑KB\_Process.v

|  |
| --- |
| `timescale 1ns / 1ps  // transform K\_ROL signal to its Hex form  module KB\_Process(input wire clk,  input wire [3:0]K\_COL,  output reg [2:0]KB\_Hex\_COL);    initial KB\_Hex\_COL = 3'd0; // initial value    always @(posedge clk)  begin  if (K\_COL[0] == 0) // when press key and row = 0  KB\_Hex\_COL <= 3'd0;  else if (K\_COL[1] == 0) // when press key and row = 1  KB\_Hex\_COL <= 3'd1;  else if (K\_COL[2] == 0) // when press key and row = 2  KB\_Hex\_COL <= 3'd2;  else if (K\_COL[3] == 0) // when press key and row = 3  KB\_Hex\_COL <= 3'd3;  else  KB\_Hex\_COL <= KB\_Hex\_COL; // when no press  end  endmodule |

* 添加并编辑Top\_Lab2\_Prg3.v

|  |
| --- |
| `timescale 1ns / 1ps  module Top\_Lab2\_Prg3(input wire clk\_100mhz, // clock signal  input wire [3:0]K\_COL, // keyboard row input  output wire SEGCLK, // LED output  output wire SEGDT,  output wire SEGEN,  output wire SEGCLR,  output wire [4:0]K\_ROW);    wire [31:0]Div; // clock div  wire [2:0]KB\_Hex\_COL; // Hex form of K\_ROL signal  wire [15:0]Hexs = {13'd0, KB\_Hex\_COL}; // real output value of 7-segment    assign K\_ROW = 5'b00000;  // clock div module  clkdiv clkdiv\_M(.clk(clk\_100mhz),  .clkdiv(Div[31:0]));    // 7-segment output module  Display Display\_M(.LES(8'h00),  .point(8'h00),  .Hexs(Hexs), // connect Hexs to 7-segment  .flash(Div[25]),  .rst(1'b0),  .Start(Div[10]),  .clk(clk\_100mhz),  .seg\_clrn(SEGCLR),  .seg\_sout(SEGDT),  .seg\_pen(SEGEN),  .seg\_clk(SEGCLK));    // transform K\_ROL signal to its Hex form  KB\_Process KB\_Process\_M(.clk(clk\_100mhz),  .K\_COL(K\_COL),  .KB\_Hex\_COL(KB\_Hex\_COL));  endmodule |

* 添加并编辑UCF文件Top.ucf：

|  |
| --- |
| NET "clk\_100mhz" LOC = AC18 | IOSTANDARD = LVCMOS18;  NET "clk\_100mhz" TNM\_NET = TM\_CLK;  TIMESPEC TS\_CLK\_100M = PERIOD "TM\_CLK" 10 ns HIGH 50%;  #七段码串行接口  NET "SEGCLK" LOC = M24 | IOSTANDARD = LVCMOS33 ;  NET "SEGCLR" LOC = M20 | IOSTANDARD = LVCMOS33 ;  NET "SEGDT" LOC = L24 | IOSTANDARD = LVCMOS33 ;  NET "SEGEN" LOC = R18 | IOSTANDARD = LVCMOS33 ;  #阵列键盘列  NET "K\_COL[0]" LOC = V18 | IOSTANDARD = LVCMOS18;  NET "K\_COL[1]" LOC = V19 | IOSTANDARD = LVCMOS18;  NET "K\_COL[2]" LOC = V14 | IOSTANDARD = LVCMOS18;  NET "K\_COL[3]" LOC = W14 | IOSTANDARD = LVCMOS18;  #阵列键盘行  NET "K\_ROW[0]" LOC = V17 | IOSTANDARD = LVCMOS18;  NET "K\_ROW[1]" LOC = W18 | IOSTANDARD = LVCMOS18;  NET "K\_ROW[2]" LOC = W19 | IOSTANDARD = LVCMOS18;  NET "K\_ROW[3]" LOC = W15 | IOSTANDARD = LVCMOS18;  NET "K\_ROW[4]" LOC = W16 | IOSTANDARD = LVCMOS18; |

* 检查并调整层次结构：

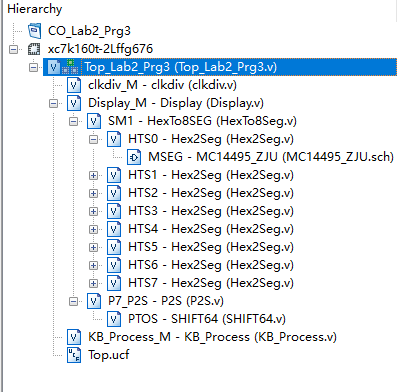


图 1 工程3层次结构图

* 以Top\_Lab2\_Prg3.v为顶层进行Generate Programming File
* 将top\_lab2\_prg3.bit文件下载至Sword Kintex7平台进行物理测试

# 4. 实验结果

注：本次实验为输入输出测试，Simulation意义较小，此处仅给出物理测试结果

## 4.1 工程1-结果

图 1 工程1物理测试结果图

如上图所示，该工程在物理条件下正确地显示了8位7段数码管数字（BD8C 8561）以及二进制表示的LED灯结果（1000 0101 0110 0001）。

因此，我们可以认为：该模块在一定程度上可以完成8位7段数码管以及LED灯的正确显示。

## 4.2 工程2-结果

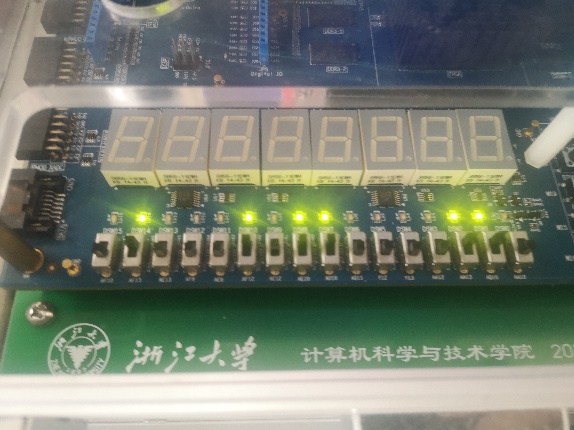
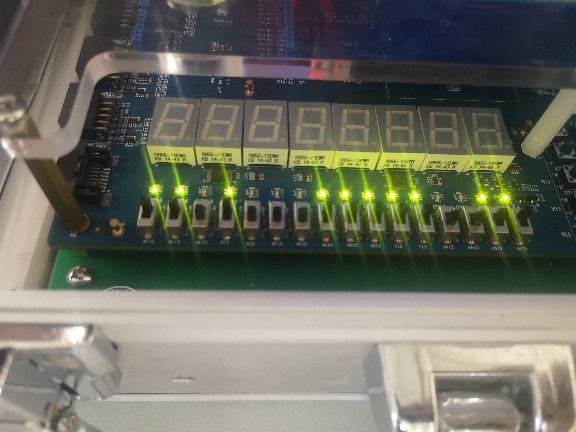
 

图 2 工程2物理测试结果图

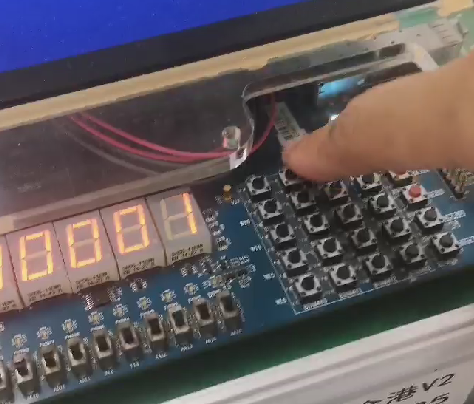
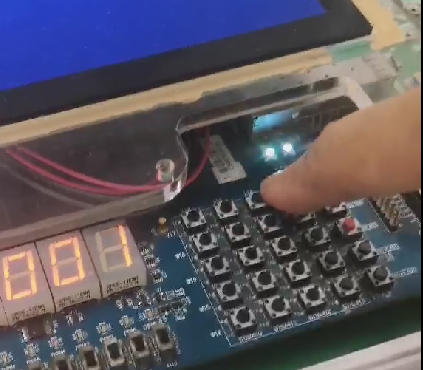
如上图所示，该工程在物理条件下用LED灯正确地显示开关的实际情况（0-开-亮，1-闭-灭）：

* 当开关情况为0100 0101 1000 0110时（左图），LED显示为0100 0101 1000 0110
* 当开关情况为1101 0001 1111 0011时（右图），LED显示为1101 0001 1111 0011

上述测试均正确通过。因此，我们可以认为：该模块在一定程度上可以完成对于开关的输入检测以及LED灯的精确反馈。

## 4.3 工程3-结果

16个拨动开关的值显示在16个对应的LED上，上拨为1（LED亮），下拨为0（LED不亮）。

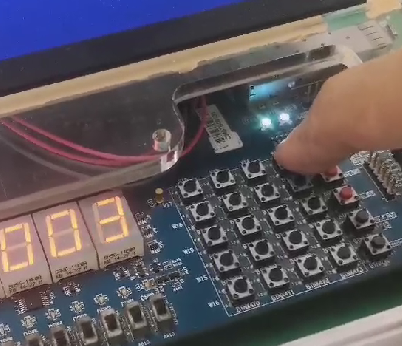
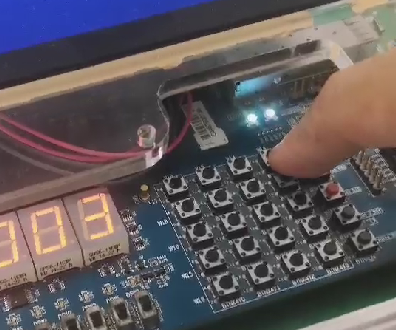
 

图 3 工程3物理测试结果图

如上图所示，该工程在物理条件下用数码管正确地显示阵列键盘的按下情况：

* 当按下第二行开关（ROW = 1）时，数码管显示为数字1
* 弹起第二行开关后，数码管保持显示数字1
* 当按下第四行开关（ROW = 3）时，数码管显示为数字3
* 弹起第四行开关后，数码管保持显示数字3

因此，我们可以认为：该模块可以完成对于阵列键盘的输入检测以及7段数码管的精确反馈。

# 5. 实验心得

由于本次实验内容较为简单并且小组成员均在上个学期参与了数字逻辑设计课的学习，因此我们比较轻松快速地完成了实验任务。当然，我们也有一定收获——通过编写Verilog代码，我们良好地回忆了数字逻辑课的知识，并进一步与计算机组成的知识相结合，深化了对于计算机硬件设计的理解。