实验报告 2 输入输出部件的使用编程

工程哈希：d3d4501c8fab7a6ff62be2aed573a8f6

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

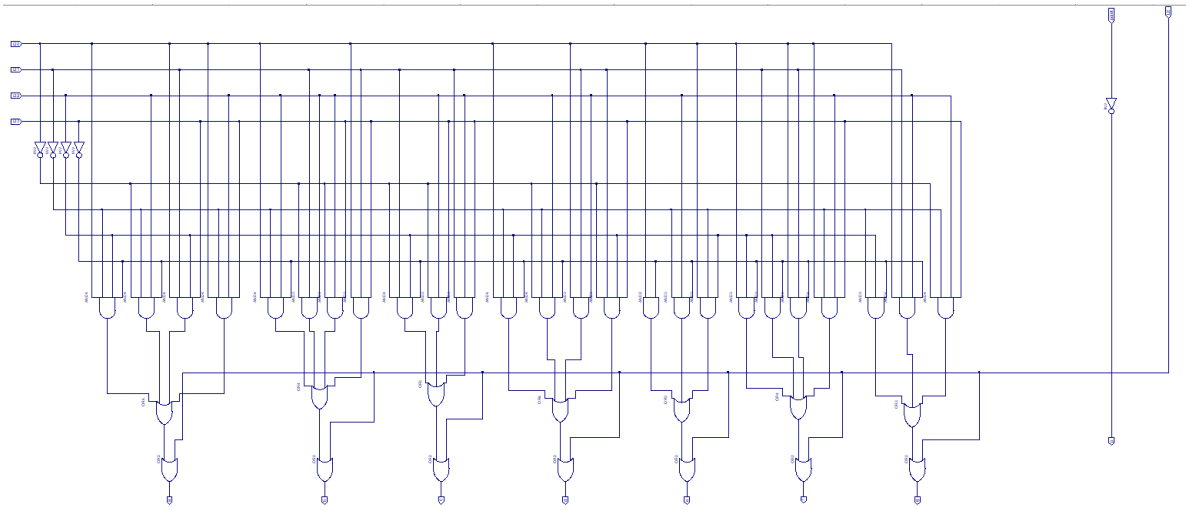
注：本节工程代码大部分来自小组成员于 2019 年秋冬学期数字逻辑设计课程中撰写的实验代码

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;
    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(clk), .SR(SR), .SL(SL),
                .S1(S1), .S0(S0), .D(D), .Q(Q));

    always @(posedge clk) // 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 clk 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 | clk; // 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);


    wire [15:0]P_Data_Input = {P_Data[0], P_Data[1],
P_Data[2], P_Data[3], P_Data[4], P_Data[5], P_Data[6],
P_Data[7],
                                    P_Data[8], P_Data[9],
P_Data[10], P_Data[11], P_Data[12], P_Data[13], P_Data[14],
P_Data[15]};
    assign led_sout = ~led_sout_temp;
    P2S #(.DATA_BITS(16), .DATA_COUNT_BITS(4)) // use P2S to
output
       LED_P2S (.clk(clk), .rst(rst), .Start(Start),

     .PData(P_Data_Input), .s_clk(led_clk), .s_clrn(led_clrn)
,
                .sout(led_sout_temp), .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：

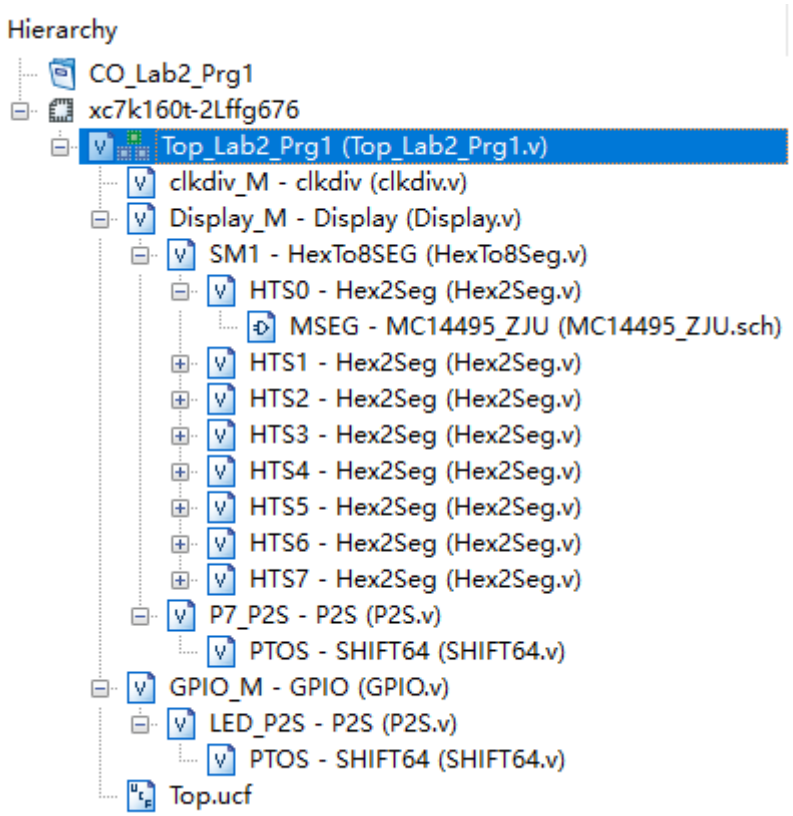


图 1 工程 1 层次结构图

- 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：

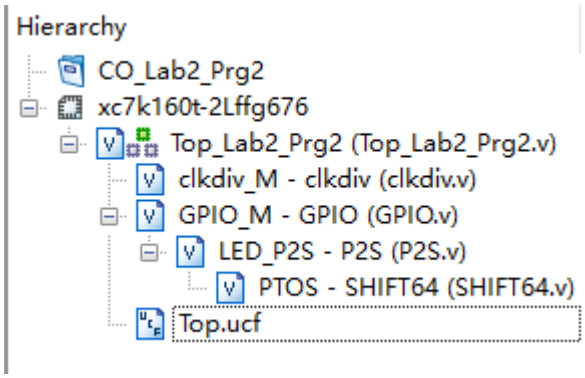


图 2 工程 2 层次结构图

- 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：

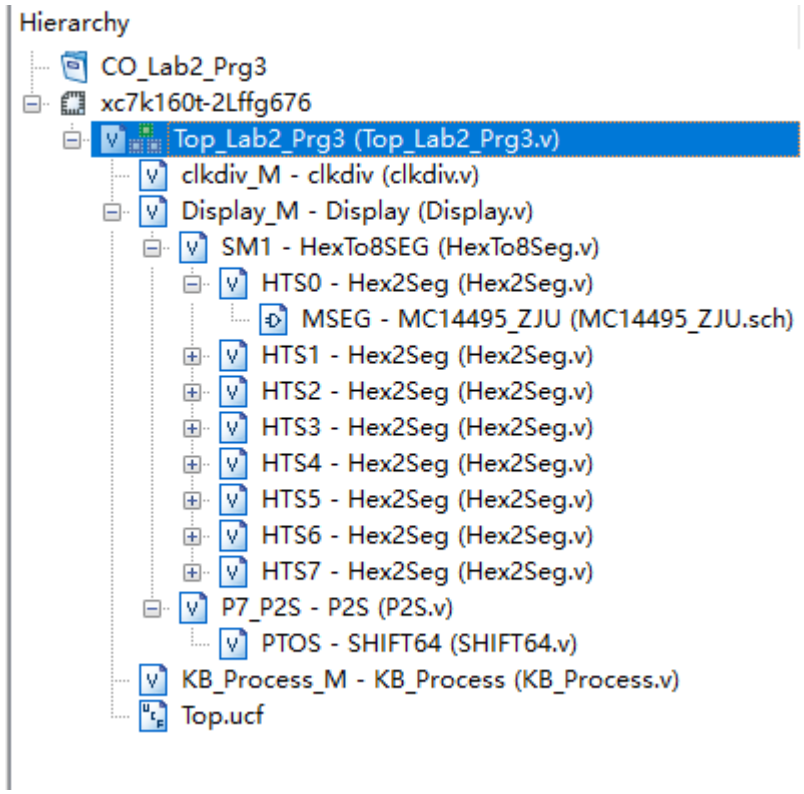


图 3 工程 3 层次结构图

- 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

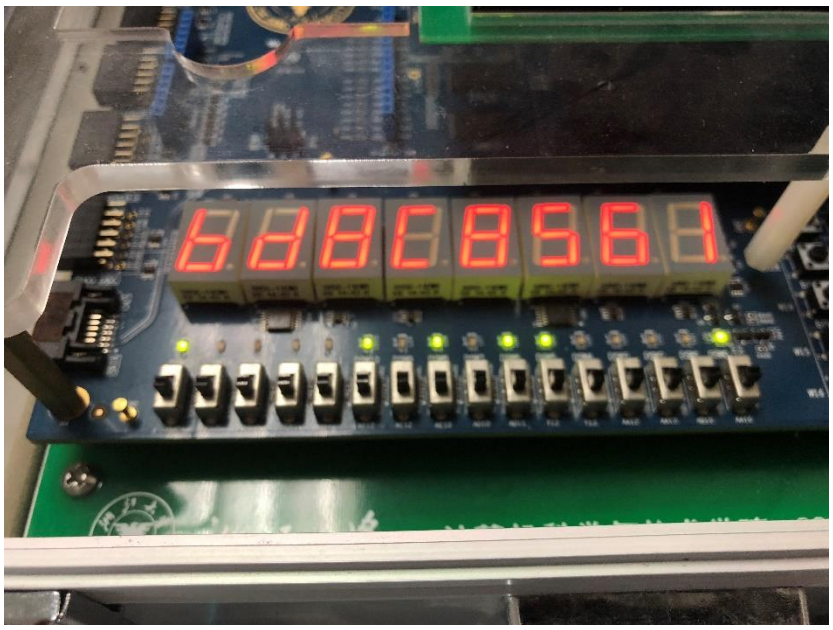


图 4 工程 1 物理测试结果图

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

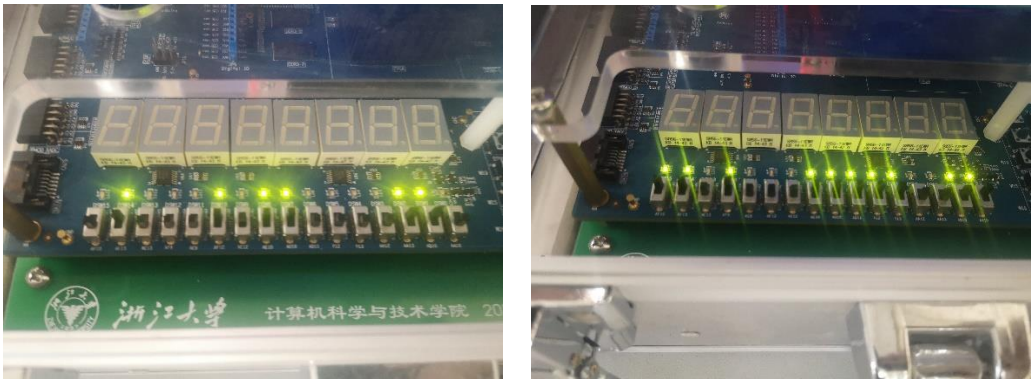


图 5 工程 2 物理测试结果图

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）。

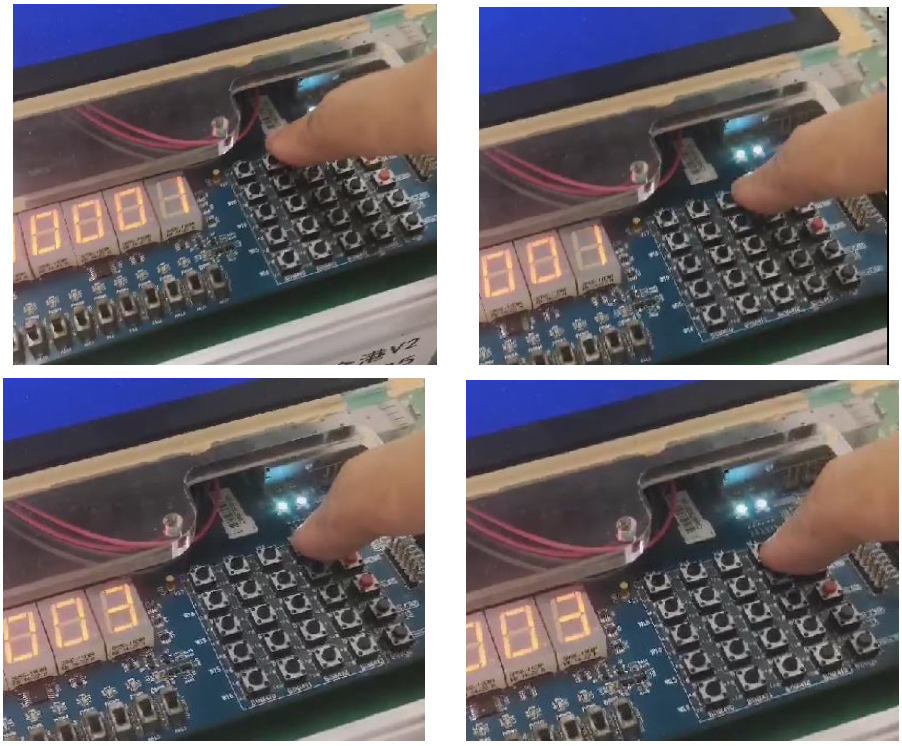


图 6 工程 3 物理测试结果图

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。