实验 2 – IO 部件(设备)扩展实验报告

——数字逻辑实验输出模块扩展二

姓名：林逸竹 学号：3160104229 专业：计算机科学与技术

课程名称：计算机组成与设计实验 同组学生姓名：无

实验时间：2018-3-12 实验地点：紫金港东 4-509 指导老师： 施青松，黎金洪

一、实验目的和要求

1.了解设备与接口
2.了解人机交互
3.了解计算机通讯
4.了解最简单的接口 **GPIO**
5.了解用 **GPIO** 实现简单的人机交互

二、实验内容和原理

2.1 实验任务

1. 优化逻辑实验输出的显示模块
- 将原理图转化为 HDL 结构化行为描述
- 增加七段码文本图形显示

2. 设计 **GPIO** 接口
GPIO 的输出 D17~D2 用驱动 LED

3. 在 **Exp01** 上修改验证
- 备份实验一工程
- 重建 Exp01 工程为 Exp02-IO

2.2 地址空间

- 存储器空间：0000000-……
- GPIO(LED)输出设备(写)：F0000000/FFFFFF00
- Switch、Button 输入设备(读)：F0000000/FFFFFF00

7-Segment 显示设备(写)：E0000000/FFFFFE00

2.3 七段显示模块优化

优化目标：

1. 用 HDL 描述重新实现并扩展相应的数据宽度
2. 建立大小合适的符号图封装供后继实验调用

用途：

1. 在 CPU 应用中可以作为一个外设
2. CPU 设计中用于测试调试显示
3. 数字系统输出显示

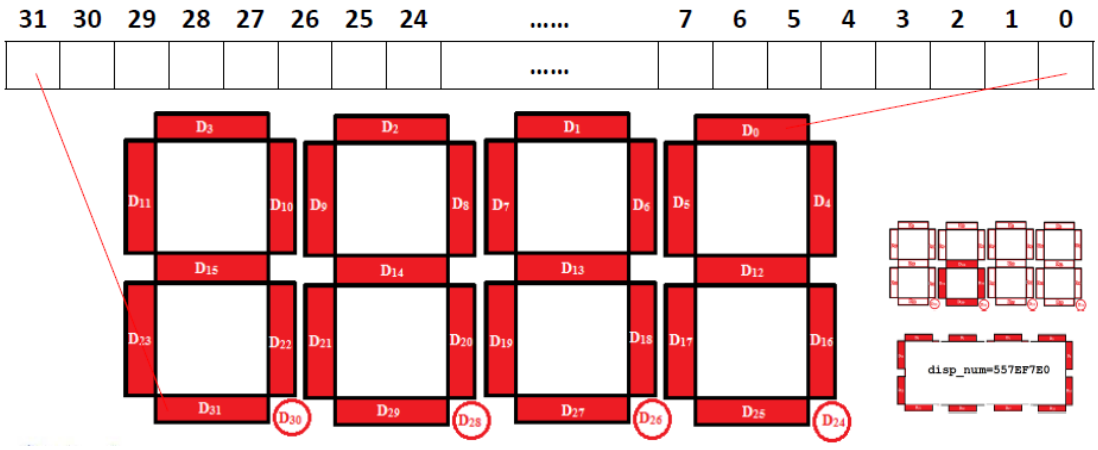


Figure 1 映射点阵显示

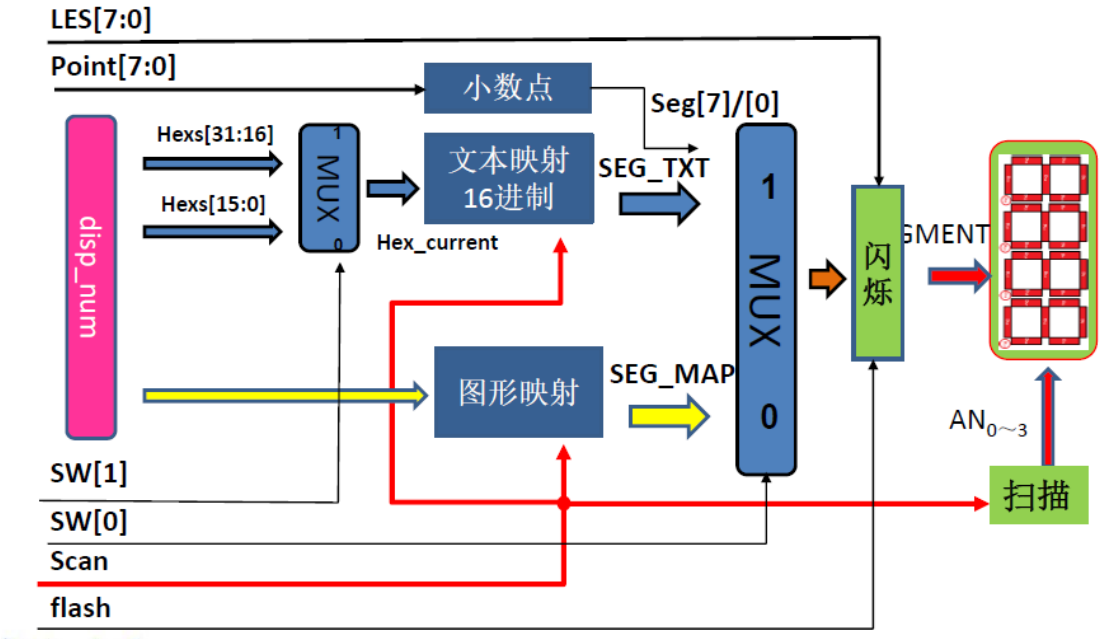


Figure 2 Display 逻辑结构

三、主要仪器设备

3.1 实验设备

1. 计算机（Intel Core i5 以上，4GB 内存以上）系统
2. 计算机软硬件课程贯通教学实验系统
3. Xilinx ISE14.4 及以上开发工具

3.2 材料

无

四、实验实现方法、步骤与调试

OExp02-IO

1. 设计实现八位七段显示器模块：Display.v

设计代码如下：

```
module Display(
    input clk,
    input rst,
    input Start,
    input Text,
    input flash,
    input [31:0] Hexs,
    input [7:0] point,
    input [7:0] LES,
    output seg_clk,
    output seg_sout,
    output SEG_PEN,
    output seg_clrn
    );
    wire [63:0] text;
    wire [63:0] map;
    wire [63:0] seg;

    HexTo8SEG SM1(.flash(flash), .Hexs(Hexs), .points(point),
        .LES(LES), .SEG_TXT(text));

    SSeg_map SM3(.Disp_num({Hexs,Hexs}), .Seg_map(map));

    MUX2T1_64 MUXSH2M(.a(text),.b(map), .o(seg), .s(Text));

    P2S M2(
        .clk(clk),
        .rst(rst),
        .Start(Start),
        .PData(seg),
        .sclk(seg_clk),
        .sout(seg_sout),
        .EN(SEG_PEN),
        .sclrn(seg_clrn)
        );
```

```
endmodule
```

其中 HexTo8SEG 代码如下：

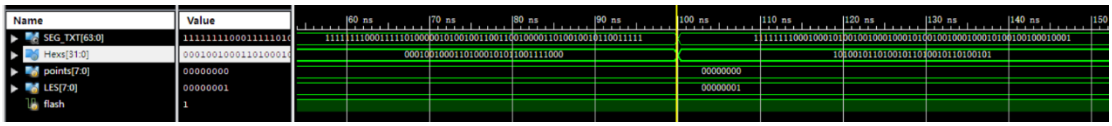
```
module HexTo8SEG(
    input [31:0] Hexs,
    input [7:0] points,
    input [7:0] LES,
    input flash,
    output [63:0] SEG_TXT
    );
Hex2Seg M0(Hexs[31:28],LES[7],points[7],flash,SEG_TXT[7:0]);
Hex2Seg M1(Hexs[27:24],LES[6],points[6],flash,SEG_TXT[15:8]);
Hex2Seg M2(Hexs[23:20],LES[5],points[5],flash,SEG_TXT[23:16]);
Hex2Seg M3(Hexs[19:16],LES[4],points[4],flash,SEG_TXT[31:24]);
Hex2Seg M4(Hexs[15:12],LES[3],points[3],flash,SEG_TXT[39:32]);
Hex2Seg M5(Hexs[11:8],LES[2],points[2],flash,SEG_TXT[47:40]);
Hex2Seg M6(Hexs[7:4],LES[1],points[1],flash,SEG_TXT[55:48]);
Hex2Seg M7(Hexs[3:0],LES[0],points[0],flash,SEG_TXT[63:56]);

endmodule

module Hex2Seg(input [3:0] Hex,
                  input LE,
                  input point,
                  input flash,
                  output [7:0] Segment
                  );
       wire EN=LE & flash;
       MC14495
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
```

仿真测试代码如下：

```
module HexTo8SEG_Test;

    // Inputs
    reg [31:0] Hexs;
    reg [7:0] points;
    reg [7:0] LES;
    reg flash;

    // Outputs
    wire [63:0] SEG_TXT;

    // Instantiate the Unit Under Test (UUT)
    HexTo8SEG uut (
        .Hexs(Hexs),
        .points(points),
        .LES(LES),
        .flash(flash),
        .SEG_TXT(SEG_TXT)
    );

    initial begin
        // Initialize Inputs
        Hexs = 0;
        points = 0;
        LES = 1;
        flash = 1;
```

```
        Hexs = 32'h12345678;
        #100;
        Hexs = 32'hA5A5A5A5;

    end

endmodule
```

测试结果：



SSeg_map 代码如下：

```
module SSeg_map(
    input [63:0]Disp_num,
   output [63:0]Seg_map
    );

    assign Seg_map =
        {Disp_num[0],   Disp_num[4],   Disp_num[16],   Disp_num[25],
Disp_num[17], Disp_num[5], Disp_num[12], Disp_num[24],
         Disp num[1],   Disp num[6],   Disp num[18],   Disp num[27],
Disp_num[19], Disp_num[7], Disp_num[13], Disp_num[26],
         Disp_num[2],   Disp_num[8],   Disp_num[20],   Disp_num[29],
Disp_num[21], Disp_num[9], Disp_num[14], Disp_num[28],
         Disp_num[3],    Disp_num[10],Disp_num[22],    Disp_num[31],
Disp_num[23], Disp_num[11],Disp_num[15], Disp_num[30],

         Disp_num[0],   Disp_num[4],   Disp_num[16],   Disp_num[25],
Disp_num[17], Disp_num[5], Disp_num[12], Disp_num[24],
         Disp_num[1],   Disp_num[6],   Disp_num[18],   Disp_num[27],
Disp_num[19], Disp_num[7], Disp_num[13], Disp_num[26],
         Disp_num[2],   Disp_num[8],   Disp_num[20],   Disp_num[29],
Disp_num[21], Disp_num[9], Disp_num[14], Disp_num[28],
         Disp_num[3],    Disp_num[10],Disp_num[22],    Disp_num[31],
Disp_num[23], Disp_num[11],Disp_num[15], Disp_num[30]};

endmoduleendmodule
```

MUX2T1 代码省略，P2S 调用 IP 核。

2. 设计实现并行输出兼 LED 显示模块：GPIO.v

GPIO 代码如下：

```
module GPIO(
    input clk,
    input rst,
    input Start,
    input EN,
    input [31:0] P_Data,
    output reg[1:0] counter_set,
    output [15:0] LED_out,
    output wire ledclk,
    output wire ledsout,
    output wire ledclrn,
    output wire LEDEN,
    output reg[13:0] GPIOf0
    );

    reg [15:0] LED;
    assign LED_out = LED;
```

```
    always @ (negedge clk or posedge rst)
    begin
        if (rst) begin LED <= 8'h2A; counter_set <= 2'b00; end
        else if (EN) {GPIOf0[13:0],LED,counter_set} <= P_Data;
        else   begin LED <= LED; counter_set <= counter_set; end
    end

    LEDP2S #(.DATA_BITS(16),.DATA_COUNT_BITS(4),.DIR(0))
        P2LED(.clk(clk),
              .rst(rst),
               .Start(Start),
               .PData(~LED),
               .sclk(ledclk),
               .sclrn(ledclrn),
               .sout(ledsout),
               .EN(LEDEN)
               );
Endmodule
```

LEDP2S 调用 IP 核。

3. 集成替换实验一的 U6, U7 核
4. 设计 U61 模块 Seg7_Dev.v

代码如下：

```
module Seg7_Dev(
    input [2:0] Scan,
    input SW0,
    input flash,
    input [31:0] Hexs,
    input [7:0] point,
    input [7:0] LES,
    output [7:0] SEGMENT,
    output [3:0] AN
    );

    wire [3:0] Hex;
    wire le;
    wire p, LE;
    wire [7:0] map;
    wire [7:0] SEG_TXT;
    assign LE = le & flash;

    Scansync M2(.Scan(Scan), .Hexs(Hexs), .point(point), .LES(LES),
        .Hexo(Hex), .LE(le), .p(p), .AN(AN));

    Seg_map M3(.Hexs(Hexs), .Scan(Scan), .Seg_map(map));

    MUX2T1_8 MUXHM(.I0(map), .I1(SEG_TXT), .o(SEGMENT), .s(SW0));

    MC14495 M1(.D3(Hex[3]), .D2(Hex[2]), .D1(Hex[1]), .D0(Hex[0]),

    .LE(LE), .point(p), .a(SEG_TXT[0]), .b(SEG_TXT[1]), .c(SEG_TXT[
2]),

    .d(SEG_TXT[3]), .e(SEG_TXT[4]), .f(SEG_TXT[5]), .g(SEG_TXT[6]),
.p(SEG_TXT[7]));

endmodule
```

其中 Scansync 代码如下：

```
module Scansync(
    input [31:0] Hexs,
```

```
    input [2:0] Scan,
    input [7:0] point,
    input [7:0] LES,
    output reg[3:0] Hexo,
    output reg p, LE,
    output reg [3:0] AN
    );
    always @ *
    begin
        case (Scan)
            3'b000: begin Hexo = Hexs[3:0]; AN = 4'b1110; p = point[0];
LE = LES[0]; end
            3'b001: begin Hexo = Hexs[7:4]; AN = 4'b1101; p = point[1];
LE = LES[1]; end
            3'b010: begin Hexo = Hexs[11:8]; AN = 4'b1011; p =
point[2]; LE = LES[2]; end
            3'b011: begin Hexo = Hexs[15:12]; AN = 4'b0111; p =
point[3]; LE = LES[3]; end
            3'b100: begin Hexo = Hexs[19:16]; AN = 4'b1110; p =
point[4]; LE = LES[4]; end
            3'b101: begin Hexo = Hexs[23:20]; AN = 4'b1101; p =
point[5]; LE = LES[5]; end
            3'b110: begin Hexo = Hexs[27:24]; AN = 4'b1011; p =
point[6]; LE = LES[6]; end
            3'b111: begin Hexo = Hexs[31:28]; AN = 4'b0111; p =
point[7]; LE = LES[7]; end
        endcase
    end
endmodule
```

Seg_map 代码如下：

```
module Seg_map(input [31:0] Hexs,
                   input [2:0] Scan,
                   output reg [7:0] Seg_map
    );
    always @ *
    begin
        case (Scan[1:0])
            2'b00:
                begin
                    Seg_map[0] = Hexs[0];
                    Seg_map[1] = Hexs[4];
                    Seg_map[2] = Hexs[16];
                    Seg_map[3] = Hexs[25];
                    Seg_map[4] = Hexs[17];
                    Seg_map[5] = Hexs[5];
                    Seg_map[6] = Hexs[12];
                    Seg_map[7] = Hexs[24];
                end
            2'b01:
                begin
                    Seg_map[0] = Hexs[1];
                    Seg_map[1] = Hexs[6];
                    Seg_map[2] = Hexs[18];
                    Seg_map[3] = Hexs[27];
                    Seg_map[4] = Hexs[19];
                    Seg_map[5] = Hexs[7];
                    Seg_map[6] = Hexs[13];
                    Seg_map[7] = Hexs[26];
                end
            2'b10:
                begin
```

```
                    Seg_map[0] = Hexs[2];
                    Seg_map[1] = Hexs[8];
                    Seg_map[2] = Hexs[20];
                    Seg_map[3] = Hexs[29];
                    Seg_map[4] = Hexs[21];
                    Seg_map[5] = Hexs[9];
                    Seg_map[6] = Hexs[14];
                    Seg_map[7] = Hexs[28];
                end
            2'b11:
                begin
                    Seg_map[0] = Hexs[3];
                    Seg_map[1] = Hexs[10];
                    Seg_map[2] = Hexs[22];
                    Seg_map[3] = Hexs[31];
                    Seg_map[4] = Hexs[23];
                    Seg_map[5] = Hexs[11];
                    Seg_map[6] = Hexs[15];
                    Seg_map[7] = Hexs[30];
                end
        endcase
    end

endmodule
```

Notes: MUX2T1 同上省略，MC14495 模块调用逻辑实验绘制的原理图。

五、实验结果与分析

完成 Seg7_Dev，GPIO 和 Display 模块，替换进实验一的工程文件中，实现基本显示和输入输出功能。

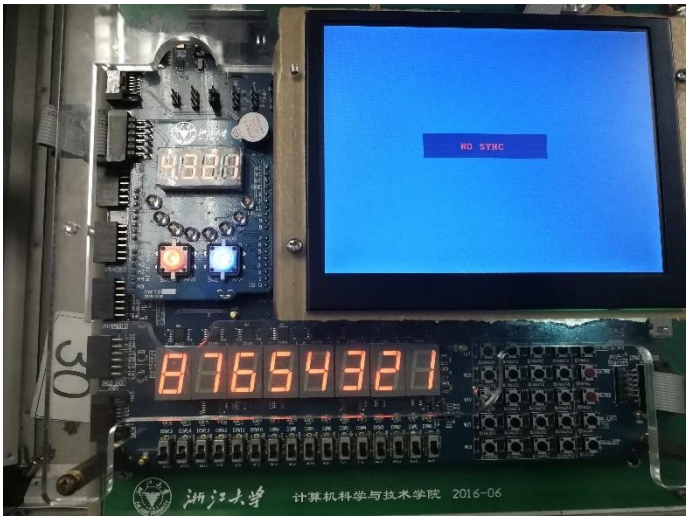


Figure 3 结果图 1(同实验一)

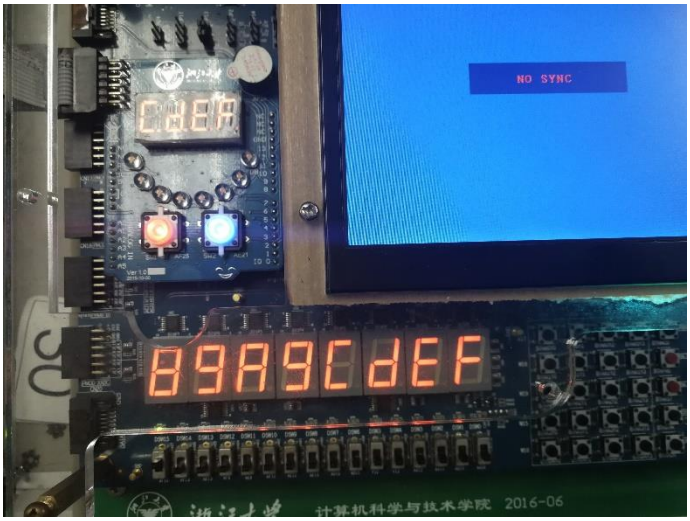


Figure 4 结果图 2(同实验一)

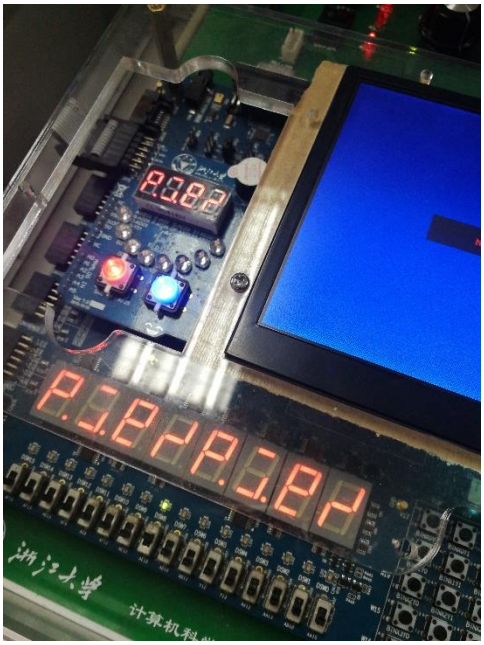


Figure 5 结果图 3(同实验一)

六、讨论、心得

刚开始觉得要将原理图转换成陌生的 Verilog 语言一定很难，但自己仔细一看发现并不难，相反只要细心看懂出错的地方就很容易纠正并完成。因此，最重要的还是要搞懂实验的原理所在。