# 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

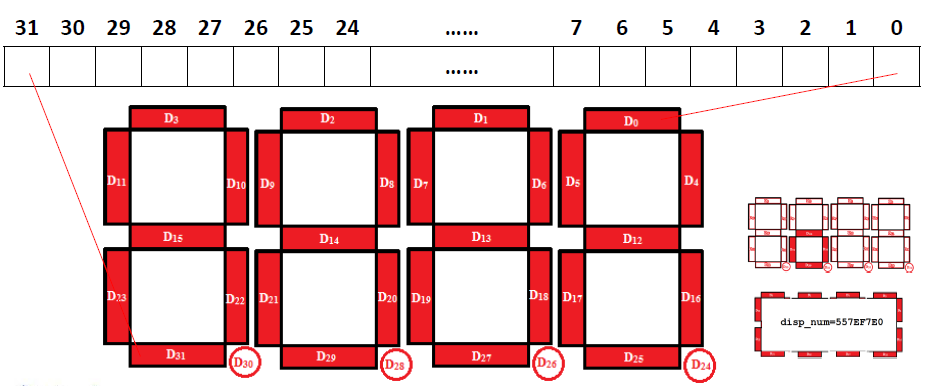
****

Figure 映射点阵显示

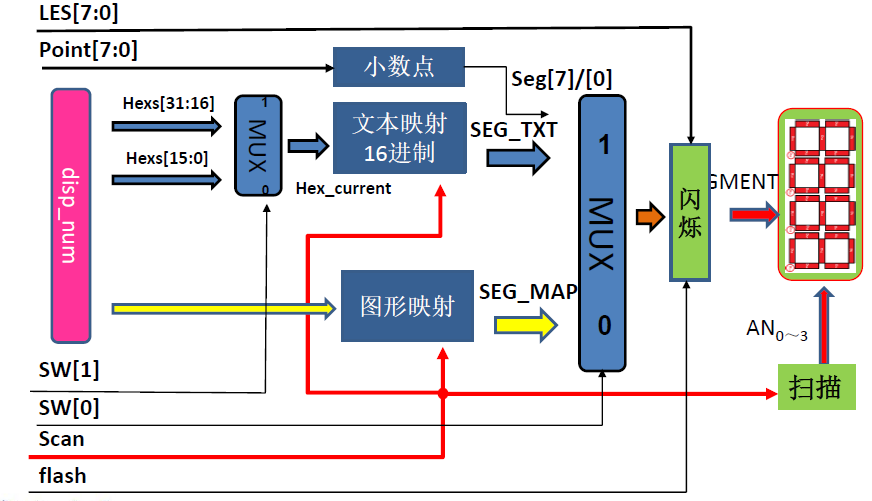


Figure Display逻辑结构

# 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 |

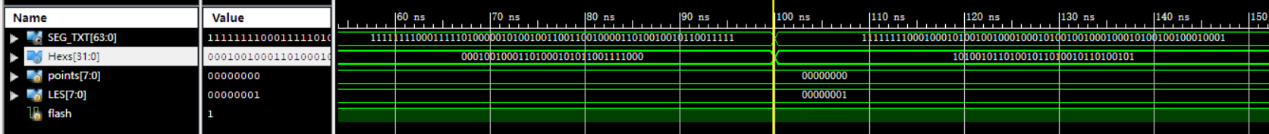
其中HexTo8SEG代码如下：

|  |
| --- |
| 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。

1. 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。

1. 集成替换实验一的U6, U7核
2. 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。

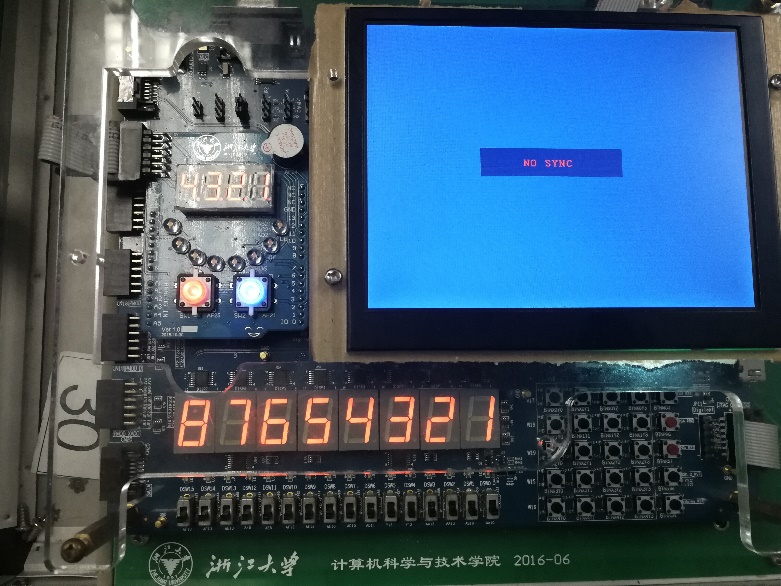


Figure 结果图1(同实验一)

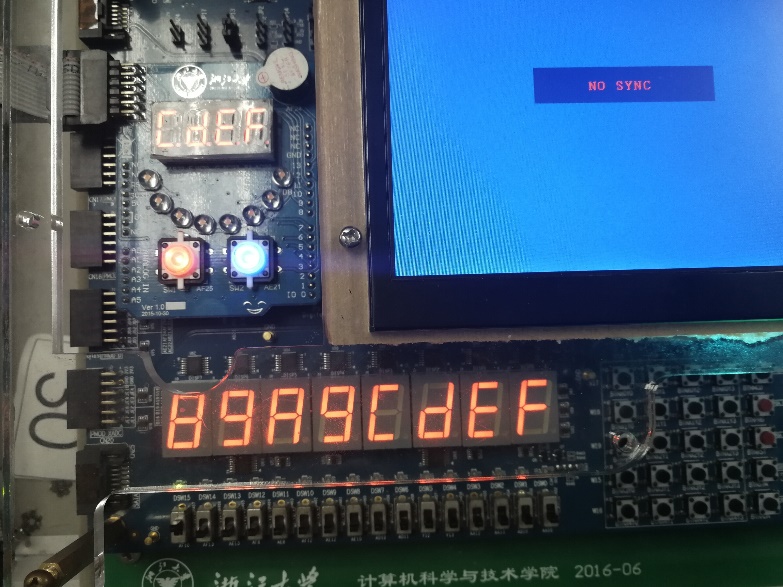


Figure 结果图2(同实验一)

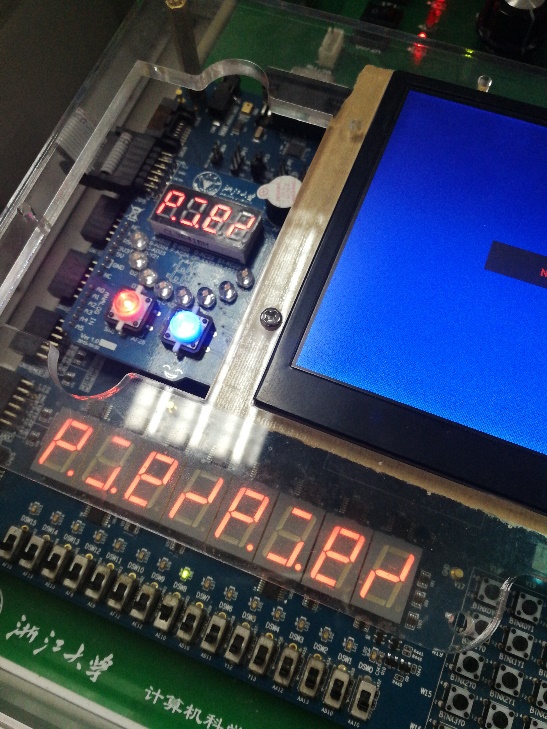


Figure 结果图3(同实验一)

# 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。