# 实验4 – 集成替换CPU核实验报告

**——IP核设计CPU/IP2CPU**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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： | | 2018-3-26 | 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. 复习寄存器传输控制技术

2. 掌握CPU的核心组成：数据通路与控制器

3. 设计数据通路的功能部件

4. 进一步了解计算机系统的基本结构

5. 熟练掌握**IP**核的使用方法

# 二、实验内容和原理

## 2.1 实验任务

**1. 用IP核集成CPU并替换实验三的CPU核**

** 选用教材提供的IP核集成实现CPU**

** 此实验在Exp03的基础上完成**

**2. 设计数据通路子部件并作时序仿真：**

** ALU**

** Register Files**

**3. 熟练掌握IP核的使用方法**

## 2.2 Computer Organization

Computer System:

* Software
* Hardware
  + CPU
    - Control unit
    - Datapath
      * Path: multiplexors
      * ALU
      * Registers
  + Memory
  + I/O interface
    - Input: PS2
    - Bidirectional: Storage
    - Output: VGA

# 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

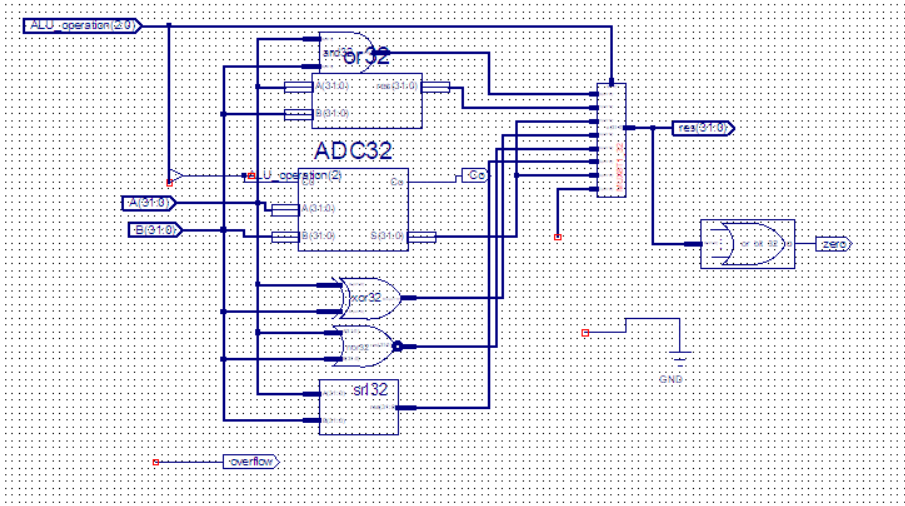
# 四、实验实现方法、步骤与调试

## 4.1 OExp04-IP2CPU

1. 清理Exp03工程
2. 新建SCPU模块，使用IP核按原理图绘制。

## 4.2 ALU

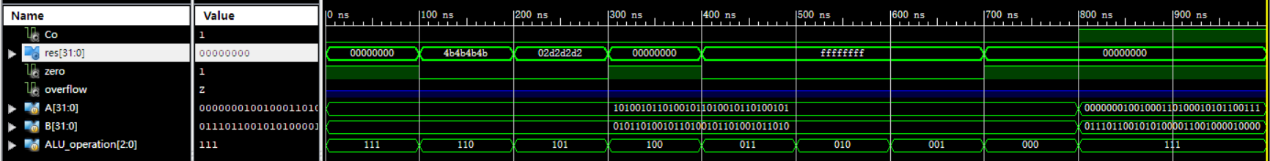
1. 按原理图描述ALU，仿真通过后封装逻辑符号。



仿真代码如下：

|  |
| --- |
| `timescale 1ns / 1ps  module ALU\_ALU\_sch\_tb();  // Inputs  reg [31:0] A;  reg [31:0] B;  reg [2:0] ALU\_operation;  // Output  wire Co;  wire [31:0] res;  wire zero;  wire overflow;  // Bidirs  // Instantiate the UUT  ALU UUT (  .A(A),  .B(B),  .Co(Co),  .ALU\_operation(ALU\_operation),  .res(res),  .zero(zero),  .overflow(overflow)  );  // Initialize Inputs  initial begin  A = 32'hA5A5A5A5;  B = 32'h5A5A5A5A;  ALU\_operation = 3'b111;  #100;  ALU\_operation = 3'b110;  #100;  ALU\_operation = 3'b101;  #100;  ALU\_operation = 3'b100;  #100;  ALU\_operation = 3'b011;  #100;  ALU\_operation = 3'b010;  #100;  ALU\_operation = 3'b001;  #100;  ALU\_operation = 3'b000;  #100;  A = 32'h01234567;  B = 32'h76543210;  ALU\_operation = 3'b111;  end  endmodule |

仿真结果如下：



2．用Verilog语言描述Regs，通过仿真。

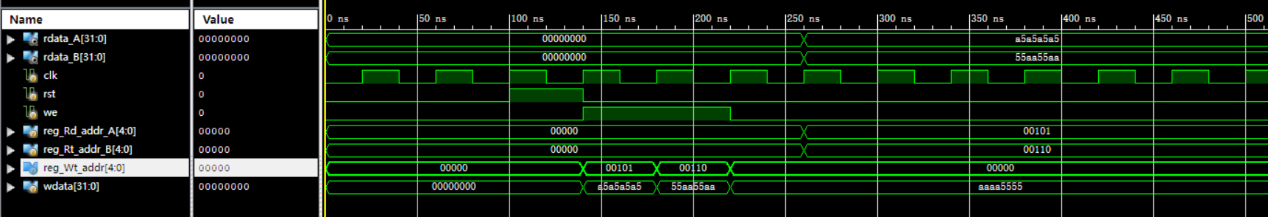
设计代码如下：

|  |
| --- |
| module Regs(  input clk, rst, we,  input [4:0] reg\_Rd\_addr\_A, reg\_Rt\_addr\_B, reg\_Wt\_addr,  input [31:0] wdata,  output [31:0] rdata\_A, rdata\_B  );  reg [31:0] register [1:31];  integer i;  assign rdata\_A = (reg\_Rd\_addr\_A==0) ? 0:register[reg\_Rd\_addr\_A];  assign rdata\_B = (reg\_Rt\_addr\_B==0) ? 0:register[reg\_Rt\_addr\_B];    always @(posedge clk or posedge rst)  begin if(rst==1)  for(i=1;i<32;i=i+1)  register[i]<=0;  else if((reg\_Wt\_addr!=0) && (we==1))  register[reg\_Wt\_addr] <= wdata;  end  endmodule |

仿真代码如下:

|  |
| --- |
| module Reg\_Test;  // Inputs  reg clk;  reg rst;  reg we;  reg [4:0] reg\_Rd\_addr\_A;  reg [4:0] reg\_Rt\_addr\_B;  reg [4:0] reg\_Wt\_addr;  reg [31:0] wdata;  // Outputs  wire [31:0] rdata\_A;  wire [31:0] rdata\_B;  // Instantiate the Unit Under Test (UUT)  Regs uut (  .clk(clk),  .rst(rst),  .we(we),  .reg\_Rd\_addr\_A(reg\_Rd\_addr\_A),  .reg\_Rt\_addr\_B(reg\_Rt\_addr\_B),  .reg\_Wt\_addr(reg\_Wt\_addr),  .wdata(wdata),  .rdata\_A(rdata\_A),  .rdata\_B(rdata\_B)  );  initial begin  // Initialize Inputs  clk = 0;  rst = 0;  we = 0;  reg\_Rd\_addr\_A = 0;  reg\_Rt\_addr\_B = 0;  reg\_Wt\_addr = 0;  wdata = 0;  fork  forever #20 clk<= ~clk;  begin  #100;  rst = 1;  #40;  rst = 0;  we = 1;  wdata = 32'ha5a5a5a5;  reg\_Wt\_addr = 32'h5;  #40;  wdata = 32'h55aa55aa;  reg\_Wt\_addr = 32'h6;  #40;  we = 0;  wdata = 32'haaaa5555;  reg\_Wt\_addr = 32'h0;  #40;  reg\_Rd\_addr\_A = 32'h5;  reg\_Rt\_addr\_B = 32'h6;  end  join    end    endmodule |

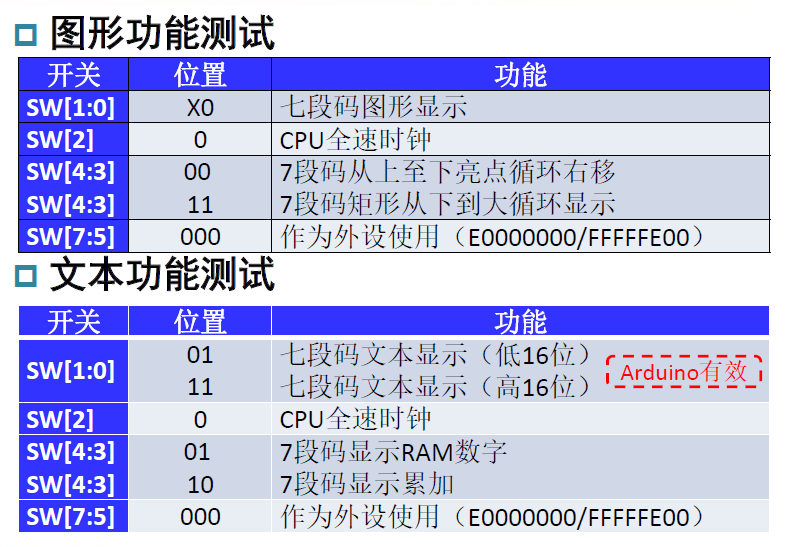
仿真结果如下：



# 五、实验结果与分析

5.1 OexpExp04\_IP2SCPU

能够实现下表功能。



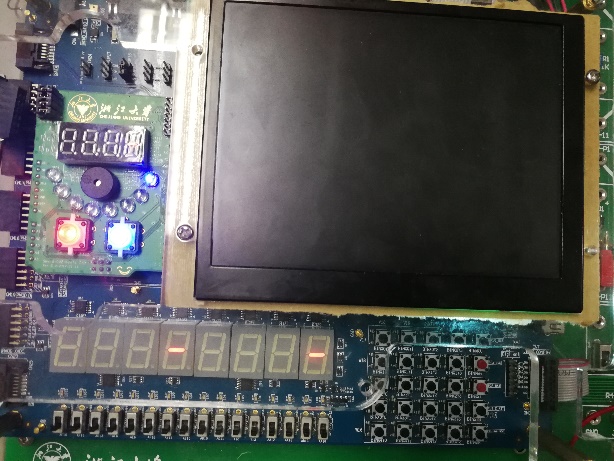


Figure 实验结果(1) 跑马灯

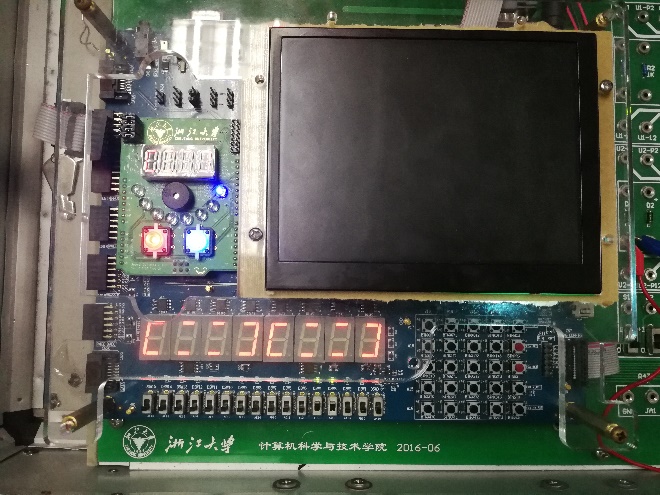


Figure 实验结果(2) 矩形

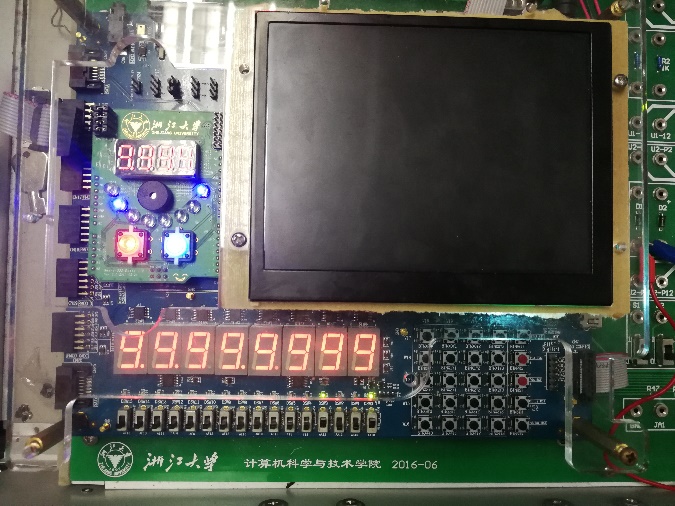


Figure 实验结果(3) RAM

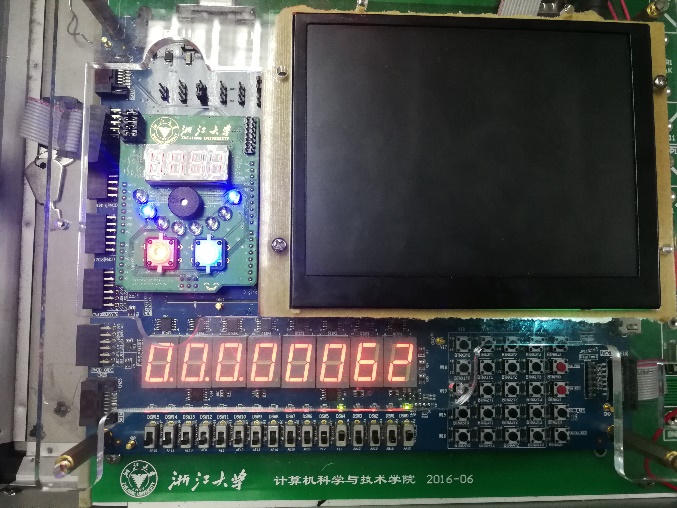


Figure 实验结果(4) 累加

5.2 ALU

通过仿真测试

# 六、讨论、心得

通过对SCPU的拆分和对ALU等的设计构建对结构有了更深的了解。