计算机组成原理 模块一

1120141831 朴泉宇

1. 实验目的

了解现代计算机硬件设计的基本流程和方法。

了解典型的RISC 处理器MIPS 的体系结构。

了解汇编语言到机器语言到计算机执行软硬的逻辑关系。

掌握处理器的设计原理和方法。

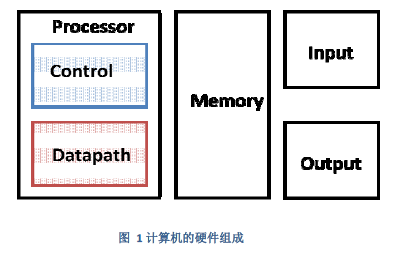
培养寄存器级硬件故障的检错和排错能力。

1. 实验内容

单周期处理器设计与实现

设计单周期控制的处理器，支持MIPS 指令子集：Lui，Addiu，Add，Lw，Sw，Beq，j，以及一条随机抽取的指令。为了简化模型，指令可以不支持溢出。

1. 设计要求
2. 整个project必须采用模块化和层次化设计。层次结构如图1所示：



详细的工程组织结构参见素材“结构示例”。其中处理器由datapath（数据通路）和controller（控制器）组成。数据通路包含：pc（程序计数器）、nPC（下一个PC 计算单元）、gpr（通用寄存器组，也称为寄存器文件、寄存器堆）、ALU（算术逻辑单元）、ext（扩展单元）、IM（指令存储器，容量32bit×1024字）、DM（数据存储器，容量32bit×1024字）、mux多路选择器等，每一个模块独立一个文件。

1. 顶层设计文件命名为mips.v
   1. PC必须被实例化命名：U\_PC。
   2. 指令存储器必须被实例化命名：U\_IM。
   3. 数据存储器必须被实例化命名：U\_DM。
   4. 寄存器文件必须被实例化命名：U\_RF。
2. code.txt中存储的是指令码，仿真时以$readmemh命令读入机器码
3. PC复位后初值为0x00003000，对应MARS的Memory Configuration设置。
4. 代码分析
5. code.txt
6. 24080000
7. 24090000
8. 21280080

其含义是：

1. addiu $t0, $zero, 0
2. addiu $t1, $zero, 0
3. addi $t0, $t1, 128
4. ctrl.v
5. module ctrl (clk,rst,op,funct,beqout,ALUctr,DMWrite,npc\_sel,RegWrt,ExtOp,mux4\_5sel,mux4\_32sel,mux2sel);//ALUctr[1:0]
6. input clk,rst;
7. input beqout;//made by alu
8. input [5:0] op,funct;
10. //control signal out
11. output [2:0] ALUctr;
12. output DMWrite;
13. output [1:0] npc\_sel;
14. output RegWrt;
15. output [1:0] ExtOp;
17. output [1:0] mux4\_5sel;
18. output [1:0] mux4\_32sel;
19. output mux2sel;
21. wire Rtype,add,addiu,lw,sw,j,beq,addi;//Rtype,addu,subu,ori,lw,sw,beq,lui,addi,addiu,slt,j,jal,jr,lb,lbu,lh,lhu,sb,sh,slti; //former
22. //wire add,sub,sll,srl,sra,sllv,srlv,srav,AND,OR,XOR,NOR,andi,xori,sltiu,bne,blez,bgtz,bltz,bgez,jalr; //ADDED
23. //???
24. assign Rtype = (op==6'b000000)?1:0;
25. assign add = (Rtype&&funct==6'b100000)?1:0; //&& ???it's better to be included in "head.v"
26. assign sub = (Rtype&&funct==6'b100010)?1:0;
27. assign lw = (op==6'b100011)?1:0;
28. assign sw = (op==6'b101011)?1:0;
29. assign beq = (op==6'b000100)?1:0;
30. assign lui = (op==6'b001111)?1:0;
31. assign addiu = (op==6'b001001)?1:0;
32. assign j = (op==6'b000010)?1:0;
33. assign addi = (op==6'b001000)?1:0; //ADDED

36. //(add||addiu||lw||sw)?3'b001:(beq)?3'b010:3'b000;
37. assign DMWrite = (sw)?'b1:'b0;
38. assign npc\_sel= (j)?2'b01:(beq&&beqout)?2'b11:2'b00; //singular valuebeq?11
39. assign RegWrt =(add||addiu||lw||lui||addi)?'b1:'b0;//JAL??????????????? //ADDED
40. assign ExtOp =(lui)?2'b00:(lw||sw||addi)?2'b10:2'b10; //ADDED EXTEND
41. assign mux4\_5sel = (addiu||lui||lw||addi)?2'b00:2'b01;//add //ADDED RD RT
42. assign mux2sel = (lw||sw||addiu||addi)?'b1:'b0;//beq //ADDED ALUsrc
43. assign mux4\_32sel = (lui)?2'b11:(add||addiu||addi)?2'b00:(lw)?2'b01:2'b10;//??????pc+4????JAL //ADDED MemtoReg
45. assign ALUctr = (add||addiu||lw||sw||addi)?3'b001:(beq)?3'b010:3'b000; //ADDED
47. endmodule

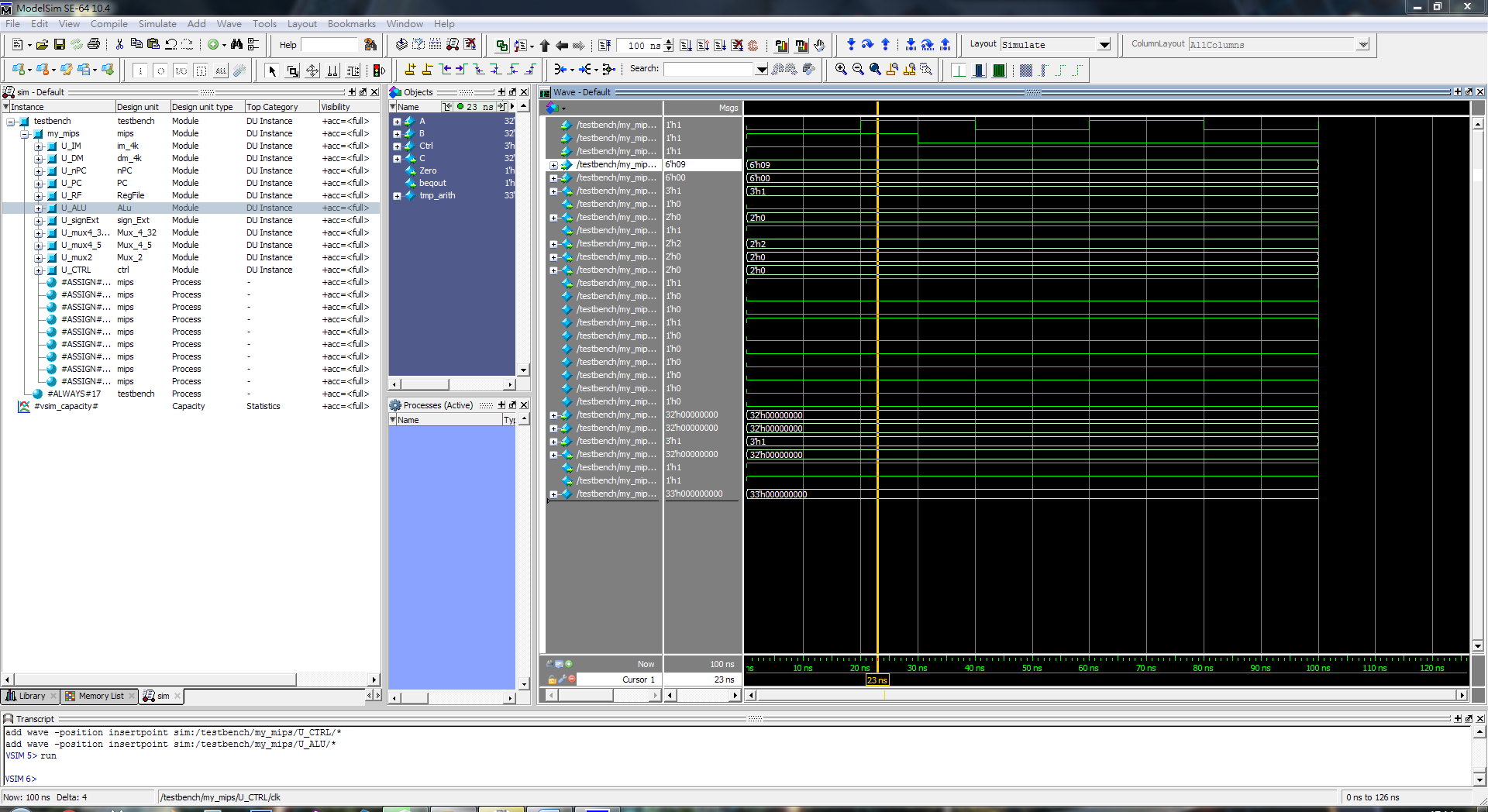
所有有关addi的相关的信号的添加，已用注释标明。

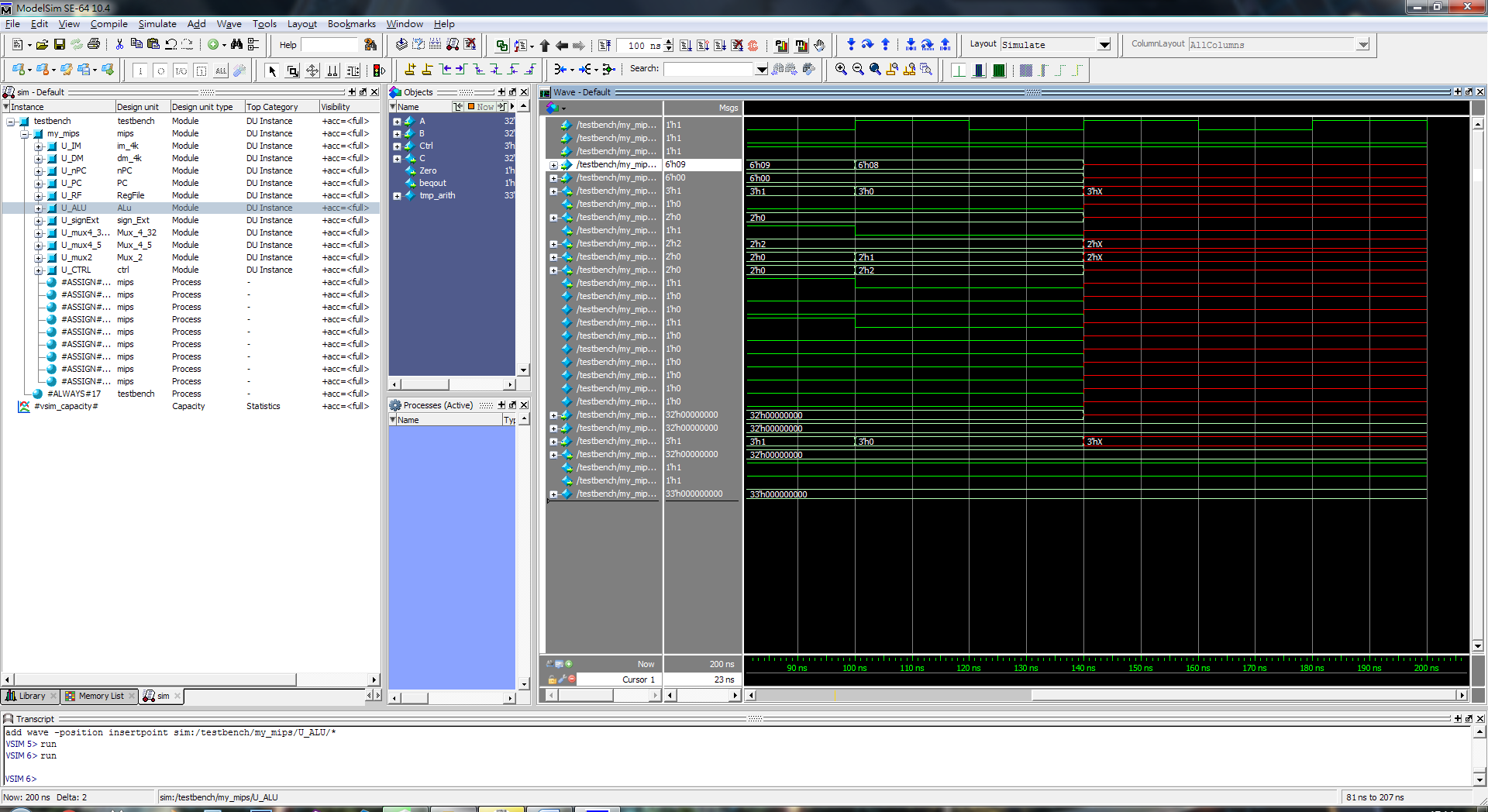
在PDF文件资料中，我查到，addi需要符号拓展Extend，需要RD、RT的寄存器操作，需要ALUsrc的立即数与寄存器的操作，需要最后再存入寄存器的MemtoReg。

1. mips.v与ctrl.v中注释的对应

上边文字所说的Extend，ALUsrc等，在mips.v文件中已作定义，但由于和PDF资料文件中给出的名字不同，我们需要根据电路逻辑来标明。标明对应的结果如下：

1. //module Extender ( EXTop,imm16,extout);
2. sign\_Ext U\_signExt(ExtOp,imm\_16,extout); // ADDED ExtendOp
3. //module Mux\_4\_32 (muxSel,in1,in2,in3,in4,Mout);
4. Mux\_4\_32 U\_mux4\_32(mux4\_32sel,ALUout,DMout,PC\_add\_4,extout,mux4\_32out); // ADDED MemtoReg
6. Mux\_4\_5 U\_mux4\_5(mux4\_5sel,RT,RD,mux4\_5out); // ADDED RD RT
8. Mux\_2 U\_mux2(mux2sel,RS2out,extout,mux2out); // ADDED ALUsrc
9. 效果展示





完成了上述的操作：

1. addiu $t0, $zero, 0
2. addiu $t1, $zero, 0
3. addi $t0, $t1, 128
4. 心得体会

本次实验，在已有基础上添加一条指令，体会了底层的CPU指令的实现。

计算机的本质，就是在复杂电路中，基于电路逻辑实现一条一条简单的指令，并巧妙地运用这些指令进行更复杂的操作。一切的基础在CPU，以及与CPU有关的底层运算中。将这些操作逐个实现的过程，复杂且精细。

通过这次实验，我进一步了解了计算机硬件与底层，也锻炼了我精细思考及编码的能力，获益匪浅。