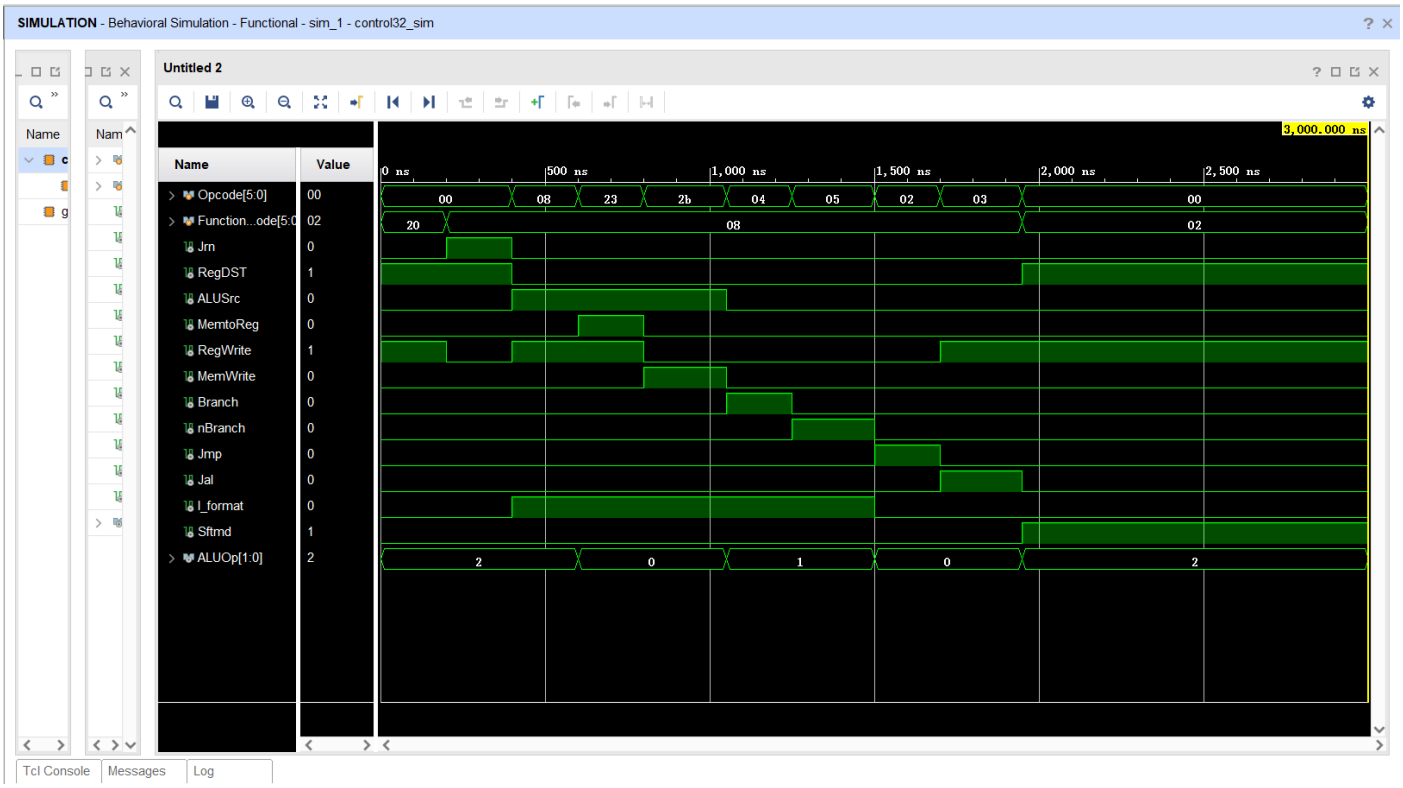
计算机系统综合设计

控制器的设计仿真时序

09017227 卓旭（SEU-本 18-09017227-卓旭）

仿真时序图：

control32_sim.v



代码：

control32.v

```
`include "public.v"

module control32(
  input wire[5:0] Opcode,            // 来自取指单元[31..26]
  input wire[5:0] Function_opcode,    // R 型指令的[5..0]
  output reg Jrn,                    // 为 1 表示下一 PC 来源于寄存器，否则来源于 PC 相关运算
  output reg RegDST,                 // 为 1 说明目标寄存器是 rd，否则是 rt
  output reg ALUSrc,                 // 为 1 表明第二个操作数是立即数，否则是寄存器（beq、bne 除
外）
  output reg MemtoReg,               // 寄存器组写入数据来源，1 为 Mem，0 为 ALU
  output reg RegWrite,               // 寄存器组写使能
  output reg MemWrite,               // DRAM 写使能
  output reg Branch,                 // 为 1 表明是 beq
  output reg nBranch,                // 为 1 表明是 bne
  output reg Jmp,                    // 为 1 表明是 j
  output reg Jal,                    // 为 1 表明是 jal
  output reg I_format,               // 是否为 I 型指令
  output reg Sftmd,                  // 为 1 表明是移位指令
  output reg[1:0] ALUOp              // LW/SW-00, BEQ/BNE-01, R-TYPE-10, I-TYPE=10
);
```

```
  reg R_format;

  always @(*) begin
    // 处理 Jrn
    Jrn <= Opcode == `OP_RTYPE && Function_opcode == `FUNC_JR;

    // 处理 RegDST
    R_format <= (Opcode == `OP_RTYPE) ? `Enable : `Disable; // 判断是否是 R 型指令
    RegDST <= R_format; // 为 1 说明目标寄存器是 rd，否则是 rt

    // 处理 I_format
    I_format <= Opcode != `OP_RTYPE && Opcode != `OP_J && Opcode != `OP_JAL;

    // 处理 ALUSrc
    ALUSrc <= I_format && Opcode != `OP_BEQ && Opcode != `OP_BNE;

    // 处理 MemtoReg
    MemtoReg <= Opcode == `OP_LW;

    // 处理 RegWrite
    RegWrite <= Opcode[5:3] == 3'b001 || (Opcode == `OP_RTYPE && Function_opcode != `FUNC_JR) || Opcode == `OP_LW || Opcode == `OP_JAL;

    // 处理 MemWrite
    MemWrite <= Opcode == `OP_SW;

    // 处理 Branch
    Branch <= Opcode == `OP_BEQ;

    // 处理 nBranch
    nBranch <= Opcode == `OP_BNE;

    // 处理 Jmp
    Jmp <= Opcode == `OP_J;

    // 处理 Jal
    Jal <= Opcode == `OP_JAL;

    // 处理 Sftmd
    Sftmd <= Opcode == `OP_RTYPE &&
      (Function_opcode == `FUNC_SLL || Function_opcode == `FUNC_SRL || Function_opcode == `FUNC_SRA
      || Function_opcode == `FUNC_SLLV || Function_opcode == `FUNC_SRLV || Function_opcode == `FUNC_SRAV);

    // 处理 ALUOp
    if (Opcode == `OP_LW || Opcode == `OP_SW) begin
      ALUOp <= 2'b00;
    end else if (Branch == `Enable || nBranch == `Enable) begin
```

```
      ALUOp <= 2'b01;
    end else if (I_format == `Enable || R_format == `Enable) begin
      ALUOp <= 2'b10;
    end else begin
      ALUOp <= 2'b00;
    end

  end

endmodule
```