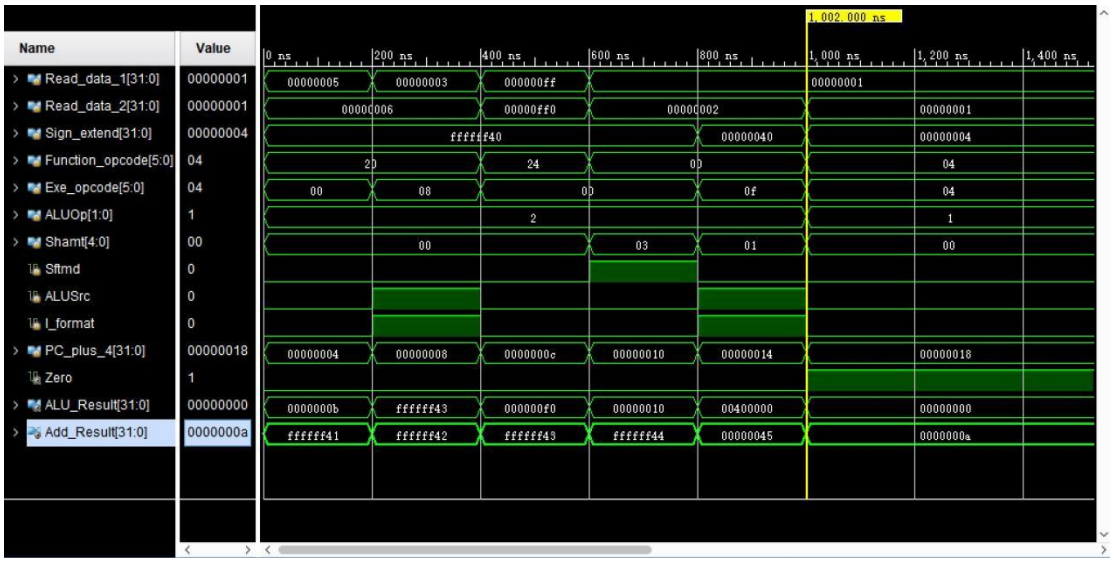
执行单元仿真时序

SEU-09019204-曹邹颖

1. 设计的执行单元的仿真波形图



2. executs32.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
module Executs32(Read_data_1,Read_data_2,Sign_extend,Function_opcode,
Exe_opcode,ALUOp,Shamt,ALUSrc,I_format,Zero,Sftmd,ALU_Result,
Add_Result, PC_plus_4);
    input[31:0]   Read_data_1;        // r-form rs
    input[31:0]   Read_data_2;        // r-form rt
    input[31:0]   Sign_extend;        // i-form
    input[5:0]    Function_opcode;    // r-form instructions[5..0]
    input[5:0]    Exe_opcode;         // op code
    input[1:0]    ALUOp;              // lw,sw:00;beq,bne:01;R 型,I-format:10
    input[4:0]    Shamt;
    input         Sftmd;
    input          ALUSrc;
    input          I_format;
    output         Zero;              //  为 1 表示计算值为 0
    output[31:0] ALU_Result;
    output[31:0] Add_Result;          // pc op   计算的地址结果
    input[31:0]   PC_plus_4;          //  来自取值单元的 PC+4

    reg[31:0] ALU_Result;
    wire[31:0] Ainput,Binput;
    reg[31:0] Sinput;
    reg[31:0] ALU_output_mux;
```

```
    wire[32:0] Branch_Add;
    wire[2:0] ALU_ctl;
    wire[5:0] Exe_code;
    wire[2:0] Sftm;
    wire Sftmd;

    assign Sftm = Function_opcode[2:0];  // 实际有用的只有低三位(移位指令)
    assign Exe_code = (I_format==0) ? Function_opcode:
{3'b000,Exe_opcode[2:0]};
    assign Ainput = Read_data_1;
    assign Binput = (ALUSrc == 0) ? Read_data_2 : Sign_extend[31:0];
    assign ALU_ctl[0] = (Exe_code[0] | Exe_code[3]) & ALUOp[1];
    assign ALU_ctl[1] = ((!Exe_code[2]) | (!ALUOp[1]));
    assign ALU_ctl[2] = (Exe_code[1] & ALUOp[1]) | ALUOp[0];

   always @* begin    // 6 种移位指令
       if(Sftmd)
        case(Sftm[2:0])
            3'b000:Sinput = Binput<<Shamt;          // Sll rd,rt,shamt    00000
            3'b010:Sinput = Binput>>Shamt;          // Srl rd,rt,shamt    00010
            3'b100:Sinput = Binput<<Ainput;         // Sllv rd,rt,rs 000100
            3'b110:Sinput = Binput>>Ainput;         // Srlv rd,rt,rs 000110
            3'b011:Sinput = $signed(Binput)>>>Shamt; // Sra rd,rt,shamt
            3'b111:Sinput = $signed(Binput)>>>Ainput; // Srav rd,rt,rs 00111
            default:Sinput = Binput;
        endcase
       else Sinput = Binput;
    end

    always @* begin
        if(((ALU_ctl[2:1]==2'b11) && (I_format==1))||((ALU_ctl==3'b111)
&& (Exe_code[3]==1))) //  所有 SLT 类
            ALU_Result = {31'd0,ALU_output_mux[31]};    //  符号位为 1 说
明(rs)<(rt)
        else if((ALU_ctl==3'b101) && (I_format==1))          // lui
            ALU_Result[31:0] = {Binput,16'd0};
        else if(Sftmd==1) ALU_Result = Sinput;                  //  移位
        else   ALU_Result = ALU_output_mux[31:0];           // otherwise
    end

    assign Branch_Add = PC_plus_4[31:2] + Sign_extend[31:0];
    assign Add_Result = Branch_Add[31:0];                       //  算出的下一个
PC 值已经做了除 4 处理，所以不需左移 16 位
    assign Zero = (ALU_output_mux[31:0]== 32'h00000000) ? 1'b1 : 1'b0;
```

```
    always @(ALU_ctl or Ainput or Binput) begin
        case(ALU_ctl)
            3'b000:ALU_output_mux = Ainput & Binput;              // and,andi
            3'b001:ALU_output_mux = Ainput | Binput;              // or,ori
            // add,addi,lw,sw
            3'b010:ALU_output_mux = $signed(Ainput) + $signed(Binput);
            3'b011:ALU_output_mux = Ainput + Binput;              // addu,addiu
            3'b100:ALU_output_mux = Ainput ^ Binput;              // xor,xori
            3'b101:ALU_output_mux = ~(Ainput | Binput);           // nor,lui
            // sub,slti,beq,bne
            3'b110:ALU_output_mux = $signed(Ainput) - $signed(Binput);
            3'b111:ALU_output_mux = Ainput - Binput;    // subu,sltiu,slt,sltu
            default:ALU_output_mux = 32'h00000000;
        endcase
    end
endmodule
```