姓名 ________________ 箱子号 ___80___ 专业 ________________

实验项目编号 __3__ 实验名称 ________________________________

CPU

- slti sltui andi ori xori sll srl sra pcaddu12i
- mul.w mulh.w mulh.wu div.w mod.w div.wu mod.w
- blt bge bltu bgeu
- ld.b ld.h ld.bu ld.hu st.b st.h

1:

	PC
	32 ,
	,
	,

1

ALU mul.v div.v

CPU

2

3

3.1

- pcaddu12i PC
- beq, bne
- load 2 ld.bu, ld.hu ld.b, ld.h store 2

3.2

mul.w/mulh.w/mulh.wu div.w/mod.w/div.wu/mod.wu EXE 32 XLEN=32 -

-

```
// mul.v 32x32 → 64      /
module mul #(
   parameter XLEN = 32
)(
   input wire                clk,
   input wire                resetn,
   input wire                start,      //
   input wire                signed_mode, // 1        0
   input wire [XLEN-1:0]    op_a,
   input wire [XLEN-1:0]    op_b,
   output wire               busy,       //
   output wire               done,       //
   output wire [2*XLEN-1:0] product     // 64
);
```

```
// div.v 32/32 → 32   + 32          /
module div #(
   parameter XLEN = 32
)(
   input wire          clk,
   input wire          resetn,
   input wire          start,        //
   input wire          signed_mode,  // 1         0
   input wire [XLEN-1:0] dividend,
   input wire [XLEN-1:0] divisor,
   output wire         busy,         //
   output wire         done,         //
   output wire         divide_by_zero, //     0
   output wire [XLEN-1:0] quotient,  //
   output wire [XLEN-1:0] remainder  //
);
```

start EXE busy busy done exe_ready_go/allowin busy/done 3.3 FSM

-

/

1. signed_mode=1 /
2. 64
3. | / divisor=0 divide_by_zero /

- **- Shift–Add**

  / -

  - acc_r multiplicand_r multiplier_r count_r COUNT_WIDTH=$clog2(XLEN)+1
  - multiplier_r[0] 1 acc_r += multiplicand_r multiplicand_r multiplier_r count_r--
  - count_r==1 / done=1 busy=0 product

    32/64 32

- **Restoring Division**

  - remainder_mag_r quotient_mag_r dividend_shift_r divisor_mag_r count_r
  -
    1. dividend_shift_r
    2. divisor_mag_r remainder_sub
    3. ge_candidate 1 1 remainder_sub 0
    4. count_r-- last_cycle

3.3

ALU 12 19 IDU alu_op alu_op[12] alu_op[18] mul.w mulh.w mulh.wu div.w mod.w div.wu mod.wu

```verilog
assign alu_op[12] = inst_mul_w; // MUL.W: 32x32->32 (low part)
assign alu_op[13] = inst_mulh_w; // MULH.W: 32x32->32 (high part, signed)
assign alu_op[14] = inst_mulh_wu; // MULH.WU: 32x32->32 (high part, unsigned)
assign alu_op[15] = inst_div_w; // DIV.W: signed division
assign alu_op[16] = inst_mod_w; // MOD.W: signed modulo
assign alu_op[17] = inst_div_wu; // DIV.WU: unsigned division
assign alu_op[18] = inst_mod_wu; // MOD.WU: unsigned modulo
```

```verilog
wire ty_MD     = op_31_26_d[0] & op_25_22_d[0] & ((inst[21:18] == 4'b0111) | (inst[21:18] ==
    4'b1000));
wire inst_mul_w = ty_MD & op_21_20_d[1] & (inst[17:15] == 3'b000);
wire inst_mulh_w = ty_MD & op_21_20_d[1] & (inst[17:15] == 3'b001);
wire inst_mulh_wu = ty_MD & op_21_20_d[1] & (inst[17:15] == 3'b010);
wire inst_div_w = ty_MD & op_21_20_d[2] & (inst[17:15] == 3'b000);
wire inst_mod_w = ty_MD & op_21_20_d[2] & (inst[17:15] == 3'b001);
wire inst_div_wu = ty_MD & op_21_20_d[2] & (inst[17:15] == 3'b010);
wire inst_mod_wu = ty_MD & op_21_20_d[2] & (inst[17:15] == 3'b011);
```

```verilog
assign is_mul_op = exe_alu_op[12] | exe_alu_op[13] | exe_alu_op[14];
assign is_div_op = exe_alu_op[15] | exe_alu_op[16] | exe_alu_op[17] | exe_alu_op[18];
assign is_multicycle_op = is_mul_op | is_div_op;
```

multicycle_executing

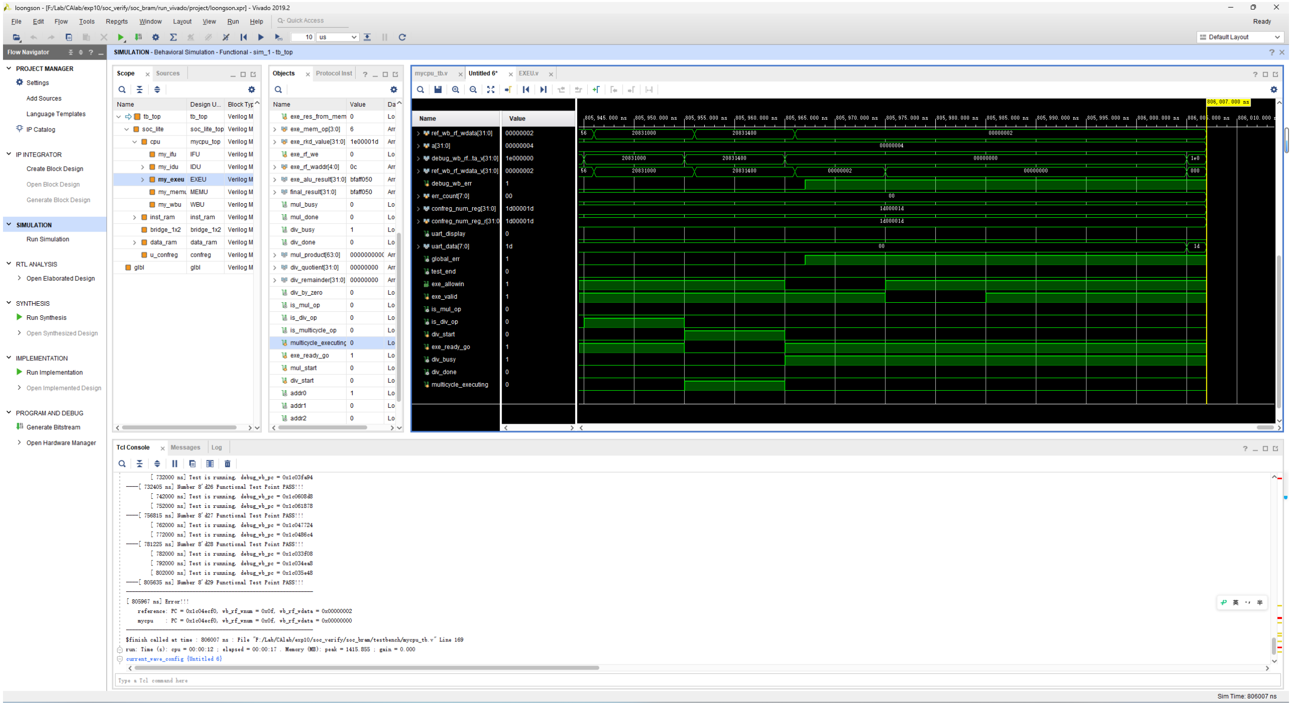
```verilog
always @(posedge clk) begin
   if (~resetn) begin
      multicycle_executing <= 1'b0;
   end else begin
      if (start_multicycle) begin
         // Start multi-cycle execution
         multicycle_executing <= 1'b1;
      end else if (multicycle_executing) begin
         // Wait for completion of multi-cycle operation
         if ((is_mul_op & mul_done) | (is_div_op & div_done))
            multicycle_executing <= 1'b0;
      end
   end
end
```

```verilog
assign exe_ready_go = ~start_multicycle & ~multicycle_executing;
assign exe_allowin = ~exe_valid | (exe_ready_go & mem_allowin);
assign exe_to_mem_valid = exe_valid & exe_ready_go;
```

```verilog
always @(posedge clk) begin
   if (~resetn) begin
      mul_start <= 1'b0;
      div_start <= 1'b0;
   end else begin
      // Start signal active for one cycle when new multicycle instruction arrives
      mul_start <= is_mul_op & start_multicycle;
      div_start <= is_div_op & start_multicycle;
   end
end
```

```verilog
assign final_result = exe_alu_op[12] ? mul_product[31:0] : // MUL.W: low 32 bits
                   exe_alu_op[13] ? mul_product[63:32] : // MULH.W: high 32 bits (signed)
                   exe_alu_op[14] ? mul_product[63:32] : // MULH.WU: high 32 bits (unsigned)
                   exe_alu_op[15] ? div_quotient :    // DIV.W: quotient (signed)
                   exe_alu_op[16] ? div_remainder :   // MOD.W: remainder (signed)
                   exe_alu_op[17] ? div_quotient :    // DIV.WU: quotient (unsigned)
                   exe_alu_op[18] ? div_remainder :   // MOD.WU: remainder (unsigned)
                   exe_alu_result;                    // Regular ALU result
```

1 EXEU



```
    [ 732000 ns] Test is running, debug_wb_pc = 0x1c03fa94
——[ 732405 ns] Number 8'd26 Functional Test Point PASS!!!
    [ 742000 ns] Test is running, debug_wb_pc = 0x1c0608d8
    [ 752000 ns] Test is running, debug_wb_pc = 0x1c061878
——[ 756815 ns] Number 8'd27 Functional Test Point PASS!!!
    [ 762000 ns] Test is running, debug_wb_pc = 0x1c047724
    [ 772000 ns] Test is running, debug_wb_pc = 0x1c0486c4
——[ 781225 ns] Number 8'd28 Functional Test Point PASS!!!
    [ 782000 ns] Test is running, debug_wb_pc = 0x1c033f08
    [ 792000 ns] Test is running, debug_wb_pc = 0x1c034ea8
    [ 802000 ns] Test is running, debug_wb_pc = 0x1c035e48
——[ 805635 ns] Number 8'd29 Functional Test Point PASS!!!
--------------------------------------------------------------
[ 805967 ns] Error!!!
    reference: PC = 0x1c04ecf0, wb_rf_wnum = 0x0f, wb_rf_wdata = 0x00000002
    mycpu    : PC = 0x1c04ecf0, wb_rf_wnum = 0x0f, wb_rf_wdata = 0x00000000
--------------------------------------------------------------
$finish called at time : 806007 ns : File "F:/Lab/CAlab/exp10/soc_verify/soc_bram/testbench/mycpu_tb.v" Line 169
run: Time (s): cpu = 00:00:12 ; elapsed = 00:00:17 . Memory (MB): peak = 1415.855 ; gain = 0.000
current_wave_config {Untitled 6}
```

loongson - [F:/Lab/CAlab/exp10/soc_verify/soc_bram/run_vivado/project/loongson.xpr] - Vivado 2019.2
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_top
Flow Navigator
Tcl Console
Messages
Log
Scope
Objects
mycpu_tb.v
Untitled 6*
EXEU.v
Sim Time: 806007 ns

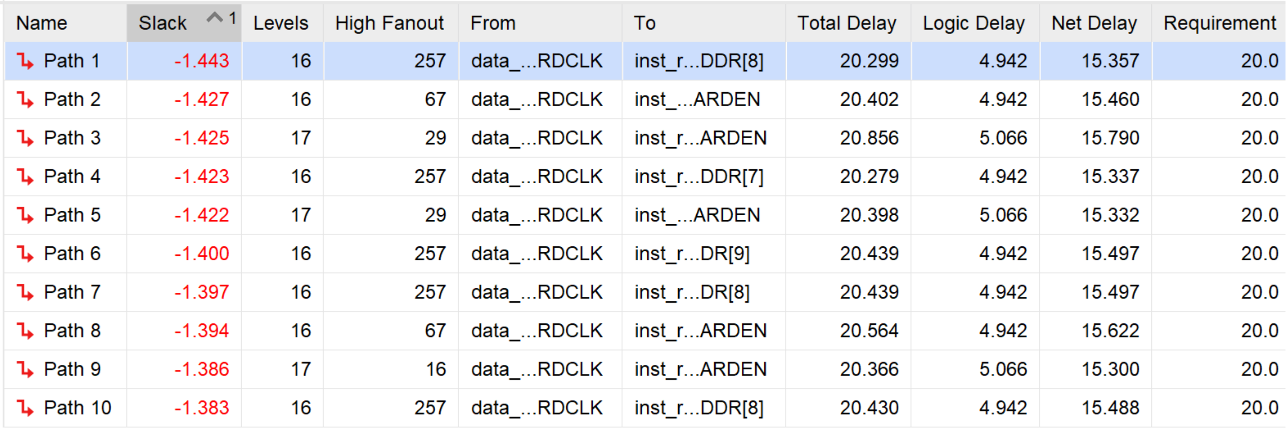
1: EXEU

EXEU allowin ready_go exe valid allowin EXEU ready_go ready_go 1 valid EXEU “ ” start_exe start_multicycle

```verilog
reg start_exe;
always @(posedge clk) begin
    if (id_to_exe_valid & exe_allowin) begin
        {exe_alu_op, exe_res_from_mem, exe_alu_src1, exe_alu_src2, exe_mem_op, exe_rf_we,
            exe_rf_waddr, exe_rkd_value, exe_pc} <= id_to_exe_zip;
        start_exe <= 1'b1;
    end else
        start_exe <= 1'b0;
end
wire  start_multicycle;
assign start_multicycle = is_multicycle_op & start_exe; // Special time stamp
```

start_multicycle

2



Name	Slack ⌃1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
↳ Path 1	-1.443	16	257	data_...RDCLK	inst_r...DDR[8]	20.299	4.942	15.357	20.0
↳ Path 2	-1.427	16	67	data_...RDCLK	inst_...ARDEN	20.402	4.942	15.460	20.0
↳ Path 3	-1.425	17	29	data_...RDCLK	inst_r...ARDEN	20.856	5.066	15.790	20.0
↳ Path 4	-1.423	16	257	data_...RDCLK	inst_r...DDR[7]	20.279	4.942	15.337	20.0
↳ Path 5	-1.422	17	29	data_...RDCLK	inst_...ARDEN	20.398	5.066	15.332	20.0
↳ Path 6	-1.400	16	257	data_...RDCLK	inst_r...DR[9]	20.439	4.942	15.497	20.0
↳ Path 7	-1.397	16	257	data_...RDCLK	inst_r...DR[8]	20.439	4.942	15.497	20.0
↳ Path 8	-1.394	16	67	data_...RDCLK	inst_r...ARDEN	20.564	4.942	15.622	20.0
↳ Path 9	-1.386	17	16	data_...RDCLK	inst_r...ARDEN	20.366	5.066	15.300	20.0
↳ Path 10	-1.383	16	257	data_...RDCLK	inst_r...DDR[8]	20.430	4.942	15.488	20.0

2:

load-to-use
Data BRAM ⟶ ⟶ IDU ⟶ ⟶ ⟶ Inst BRAM
load-to-use ID MEM MEM EXE WB

CPU

CPU