# Computer Architecture

Lab 5: Building Blocks of Processor

## Getting Started

- Open a web browser and go to https://www.edaplayground.com/

# Lab Assignment 1: ALU

- Write a SystemVerilog code to implement the ALU
  - Its functionality is given in the table on the right.
  - Additional requirement: the ALU should generate a signal that indicates whether the result is zero or not
- The ALU has the following inputs and outputs
  - iA (32 bits): source operand 1
  - iB (32 bits): source operand 2
  - iF (3 bits): control signal
  - oY (32 bits): result
  - oZero (1 bit): 1 if oY==0, 0 otherwise
- Use the testbench in the following slide
- Submit your code as plain text to Blackboard

| $iF_{2:0}$ | Function |
|---|---|
| 000 | iA & iB |
| 001 | iA \| iB |
| 010 | iA + iB |
| 011 | not used |
| 100 | iA & ~iB |
| 101 | iA \| ~iB |
| 110 | iA – iB |
| 111 | SLT |

# Testbench for ALU

```
module testbench_alu();
  logic [31:0] a, b, y;
  logic [2:0] f;
  logic zero;
  logic result;

  alu dut(
    .iA (a),
    .iB (b),
    .iF (f),
    .oY (y),
    .oZero (zero)
  );

  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    result = 1'b1;

    a = 32'h1234_5678; b = 32'h0000_ffff; f=3'b000; #10;
    if(y!=32'h0000_5678) begin $display("000 failed."); result=1'b0; end

    a = 32'h1234_5678; b = 32'h0000_ffff; f=3'b001; #10;
    if(y!=32'h1234_ffff) begin $display("001 failed."); result=1'b0; end

    a = 32'h1234_5678; b = 32'h1111_2222; f=3'b010; #10;
    if(y!=32'h2345_789a) begin $display("010 failed."); result=1'b0; end

    a = 32'h1234_5678; b = 32'h0000_ffff; f=3'b100; #10;
    if(y!=32'h1234_0000) begin $display("100 failed."); result=1'b0; end

    a = 32'h1234_5678; b = 32'h0000_ffff; f=3'b101; #10;
    if(y!=32'hffff_5678) begin $display("101 failed."); result=1'b0; end

    a = 32'h1234_5678; b = 32'h1111_2222; f=3'b110; #10;
    if(y!=32'h0123_3456) begin $display("110 failed."); result=1'b0; end

    a = 32'h0000_5678; b = 32'h0000_ffff; f=3'b111; #10;
    if(y!=32'h0000_0001) begin $display("111 failed."); result=1'b0; end

    a = 32'h0000_0000; b = 32'h0000_0000; f=3'b000; #10;
    if(zero!=1) begin $display("zero failed."); result=1'b0; end

    if(result)    $display("SUCCESS!");
    else                             $display("FAILURE!");
  end
endmodule
```

# Lab Assignment 2: Register File

- Write a SystemVerilog code to implement a register file
- It has 32 32-bit registers and the register 0 should be always 0
- It has two read ports and one write port
- It has the following inputs and outputs
  - iClk (1 bit): clock (rising edge)
  - iReset (1 bit): reset signal
  - iRaddr1 (5 bits): the register number of read port 1
  - oRdata1 (32 bits): the output of read port 1
  - iRaddr2 (5 bits): the register number of read port 2
  - oRdata2 (32 bits): the output of read port 2
  - iWaddr (5 bits): the register number of the write port
  - iWdata (32 bits): the input data for the write port
  - iWe (1 bit): the write enable signal
- Use the testbench in the following slide
- Submit your code as plain text to Blackboard

# Testbench for Register File

```
module testbench_regfile();
  logic clk;
  logic reset;
  logic [4:0] raddr1, raddr2, waddr;
  logic we;
  logic [31:0] wdata;
  logic [31:0] rdata1, rdata2;
  logic result;
  logic [31:0] regs[31:0];
  integer i;

  regfile dut(
    .iClk              (clk),
    .iReset            (reset),
    .iRaddr1(raddr1),
    .iRaddr2(raddr2),
    .iWaddr            (waddr),
    .iWe               (we),
    .iWdata            (wdata),
    .oRdata1(rdata1),
    .oRdata2(rdata2)
  );

  always     // no sensitivity list, so it always executes
    begin
      clk = 1; #5; clk = 0; #5;
    end

  initial begin
    result = 1;
    $dumpfile("dump.vcd"); $dumpvars;
    reset = 0; #21;
    raddr1 = 0; raddr2 = 0; waddr = 0; we = 0; wdata = 0;
    for(i=0; i<32; i=i+1) regs[i]=0;
    reset = 1; #10;
    reset = 0; #10;
    for(i=0; i<32; i=i+1) begin
      waddr = i; we = 1; wdata = $random; regs[i] = wdata; #10;
    end
    waddr = 0; we = 0; wdata = 0;
    for(i=0; i<32; i=i+1) begin
      raddr1 = i; raddr2 = i; #1
      if(i==0) begin
        if(rdata1 != 0 | rdata2 != 0) begin
          $display("Read data from r0 failed");
          result = 0;
        end
      end else begin
        if(rdata1 != regs[i]) begin
          $display("Read data from port 1 at address %d failed %x %x", i, rdata1, regs[i]);
          result = 0;
        end
        if(rdata2 != regs[i]) begin
          $display("Read data from port 2 at address %d failed", i);
          result = 0;
        end
      end
      #9;
    end
    if(result)         $display("SUCCESS!");
    else                                      $display("FAILURE!");
    #10; $stop;
  end
endmodule
```