



Lab 3: Simple ALU

Chien-Ming Wu

TSRI, NARL, Taiwan, R.O.C.

Lan-Da Van

Department of Computer Science

National Yang Ming Chiao Tung University

Taiwan, R.O.C.

Fall, 2022



Lab 3 Goal: Simple ALU

Lab 3

- ◆ In this lab, you will practice Verilog to design one simple ALU.
- ◆ The lab file submission deadline is on 10/17 by 6:00pm.

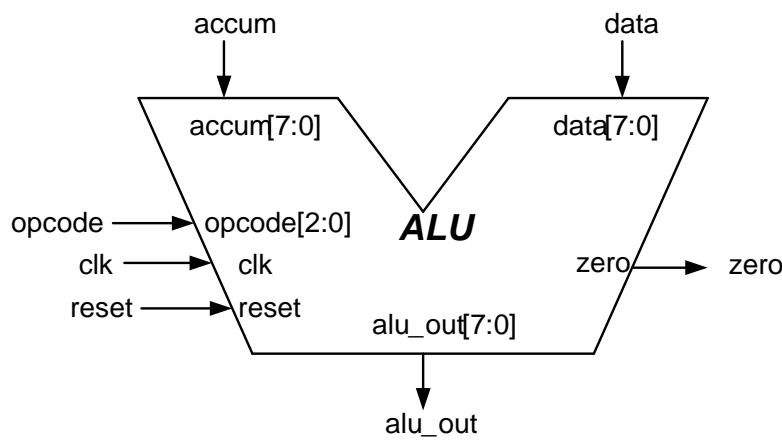




Simple ALU

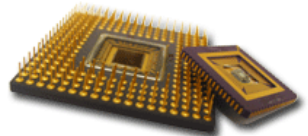
Lab 3

- ◆ All inputs and outputs except zero are synchronized with positive clock edge (rising edge).
- ◆ reset is a synchronized reset. When reset=1, reset ALU and alu_out will be 0.
- ◆ accum, data, and alu_out are represented by 2's complement.
- ◆ When accum=0, the zero output is 1. On the contrary, when accum≠0, the zero output is 0. zero and reset are independent.
- ◆ **When opcode input X(unknown), alu_out is 0.**



opcode	ALU operation	
000	Pass accum	
001	accum + data	(add)
010	accum – data	(subtraction)
011	accum AND data	(bit-wise AND)
100	accum XOR data	(bit-wise XOR)
101	ABS(accum)	(absolute value)
110	MUL	(multiplication)
111	Pass data	

1. When the absolute operation is activated, accum[7] is the signed bit.
2. MUL is only for sign multiplication.





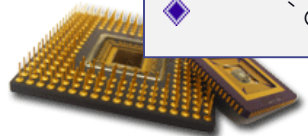
Testbench of the ALU Module

Lab 3

```
◆ wire [7:0] alu_out;
◆ reg [7:0] data, accum;
◆ reg [2:0] opcode;
◆ wire [7:0] mask;
◆ reg clk, reset;

◆
◆ parameter ranseed = 8; // Seed for the random function
◆                          // Modify the seed for different inputs
◆
◆ // Instantiate the ALU. Named mapping allows the designer to have
◆ freedom with the order of port declarations
◆ alu    alu1 (.alu_out(alu_out), .zero(zero), //outputs from ALU
◆             .opcode(opcode), .data(data & mask), //inputs to ALU
◆             .accum(accum & mask), .clk(clk), .reset(reset));

◆ // Define mnemonics to represent opcodes
◆ `define PASSA 3'b000
◆ `define ADD   3'b001
◆ ...
◆ `define PASSD 3'b111
```





Testbench of the ALU Module

Lab 3

```
◆ // Define a safe delay between each strobing of the ALU
  inputs/outputs

◆ `define strobe          20

◆ // To perform a 4-bit multiplication, set the first 4 bits of the
  input to 4'b0000 when opcode is 3'b110 (Multiplication)

◆ assign mask = (opcode == 3'b110)? 8'h0f: 8'hff;

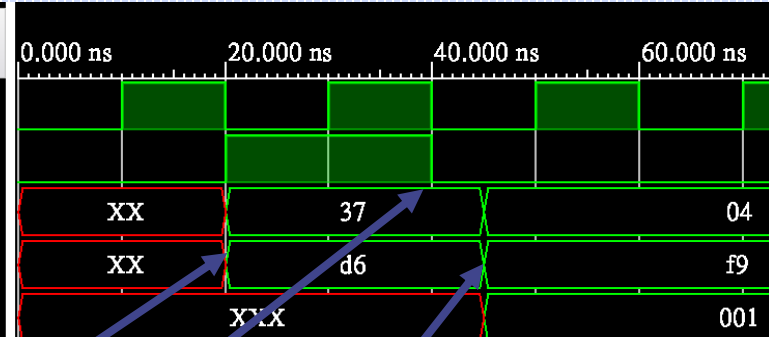
◆ // Clock generate
◆ initial    clk = 0;
◆ always #(`strobe/2) clk = ~clk;
```





Test

Name	Value
clk	1
reset	0
accum[7:0]	04
data[7:0]	f9
opcode[2:0]	001



```
// pattern generate
initial begin
    // SET UP THE OUTPUT FORMAT FOR THE TEXT DISPLAY
    $display("\t\t\t INPUTS\t\t\t REAL\t\t\t OUTPUT\t\t\t \n");
    $display("\t\t\t\t\t OPCODE\t\t\t DATA IN\t\t\t ACCUM IN\t\t\t ALU OUT\t\t\t ZERO BIT");
    $display("\t\t\t\t\t ----- \t\t\t ----- \t\t\t ----- \t\t\t -----");
    ...

    reset = 0;
    # `strobe;
    accum = 8'h37;
    data = 8'hD6;
    reset = 1; //reset the ALU
    # `strobe;
    reset = 0;
    #(`strobe/4) opcode = 3'b001; // Set operation code

    // APPLY STIMULUS TO THE INPUT PINS
    accum = $random % ranseed; //Set inputs to the ALU
    data = $random % ranseed;
    //Wait for ALU to process inputs
    #(`strobe/2) check_outputs; //call a task to verify outputs
end
```

\$random : Generate a 32-bit signed-integer random number

\$random % 8 : Generate a random number whose range is between -7 and 7



Testbench of the ALU Module

Lab 3

```

// SUBROUTINES TO DISPLAY THE ALU OUTPUTS
task check_outputs;
  casez (opcode)
    `PASSA : begin
      $display("PASS ACCUM OPERATION:",
               "      %b      %b      %b |      %b      %b",
               opcode, data, accum, alu_out, zero);
    end
    `ADD : begin
      $display("ADD OPERATION      :",
               "      %b      %b      %b |      %b      %b",
               opcode, data, accum, alu_out, zero);
    end
    ...
  end
end

```

	INPUTS			REAL	OUTPUT
	OPCODE	DATA IN	ACCUM IN	ALU OUT	ZERO BIT
ADD OPERATION	:	001	11111001	00000100 1	11111101 0



Lab 3 Demo Guide

Lab 3

- ◆ You can download the sample testbench file `alu_test.v` from E3, and create a Vivado project for it.
- ◆ You should upload your lab3 solution to E3 before the deadline.
- ◆ During the demo time, TA will ask you to modify the testbench to show different results.
 - You can download your code from E3 during demo.

