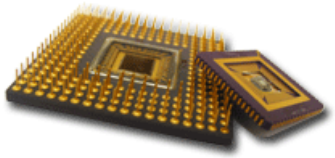
Lab 3: Simple ALU

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, 2024

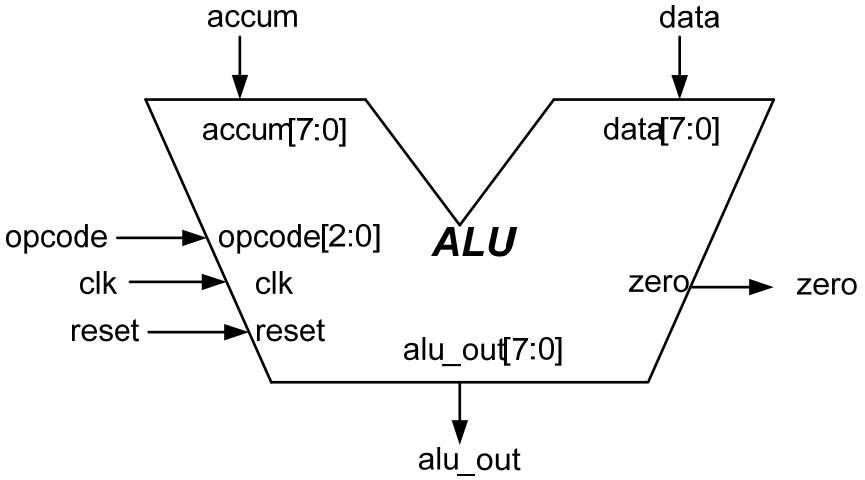
Lab 3 Goal: Simple ALU

- In this lab, you will practice Verilog to design one simple ALU.
- The lab file submission deadline is on 09/30 by 6:00pm.

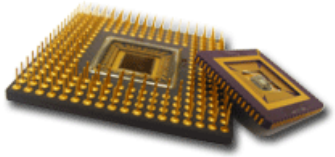


Simple ALU

- When reset=1, no matter what opcode is, reset ALU and alu_out will be 0.
- accum, data, and alu_out are represented by 2's complement.
- If overflow occurs when operating addition or subtraction, you should implement **saturation arithmetic algorithm**. You can assume that multiplication would not cause overflow during this lab.
- **When opcode input X(unknow), alu_out is 0.**



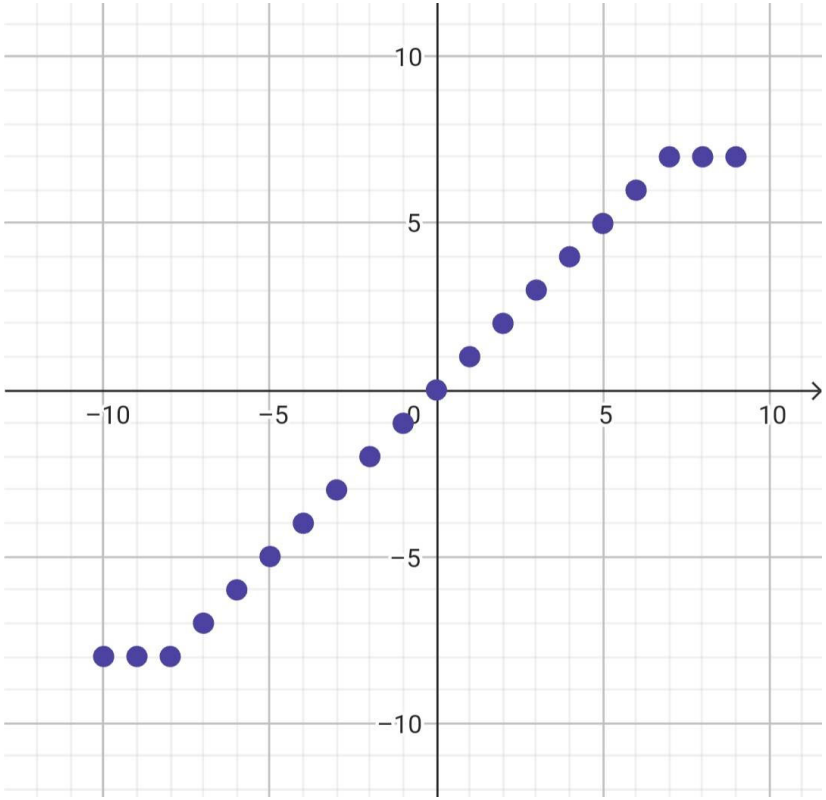
opcode	ALU operation
000	Pass accum
001	accum + data (add)
010	accum – data (subtraction)
011	accum arithmetic right shift data bit
100	accum XOR data (bit-wise XOR)
101	ABS(accum) (absolute value)
110	accum * data (MUL)
111	-(accum) (flip the sign)

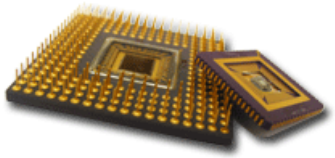


1. When the absolute operation is activated, accum[7]is the signed bit.
2. MUL is only for sign multiplication.

Saturation Mode

- You should design saturation mode to your ALU when doing addition or subtraction.
- Convert the value to MAX when overflow or MIN when underflow.

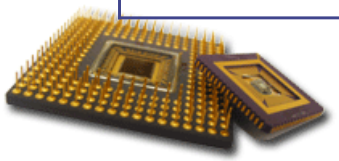




PSW

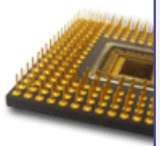
- In this lab, you will also need to design four PSW(Program Status Word) signal.

PSW signal	description
zero	When alu_out equals zero, pull up this signal. Otherwise set this signal to 0.
overflow	During addition or subtraction, if overflow occurs, pull up this signal. Otherwise set this signal to 0.
parity	Implement even parity on alu_out.
sign	If the sign bit of alu_out is negative, then pull up this signal.



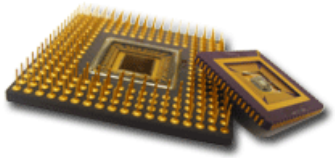
Testbench of the ALU Module


```
wire [7:0] alu_out;
  wire zero;
  wire overflow;
  wire parity;
  wire sign;

  reg  [7:0] data, accum;
  reg  [2:0] opcode;

  wire [7:0] mask;

  reg reset;
// 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), .overflow(overflow),
.parity(parity), .sign(sign),  .opcode(opcode), .data(data & mask),
.accum(accum & mask), .reset(reset));
// Define mnemonics to represent opcodes
  `define PASSA 3'b000
  `define ADD   3'b001
  ...
  `define FLIPSIGN 3'b111
```



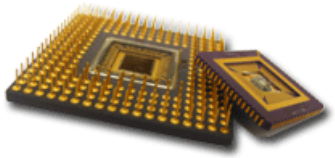
Testbench of the ALU Module


```
// Define a safe delay between each strobing of the ALU
inputs/outputs
`define strobe      10

// 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;
```



Testbench of the ALU Module


```
// pattern generate
initial begin
  reset = 0;
  # `strobe;
  reset = 1;
  # `strobe;
  reset = 0;

  // testcase 1
  opcode = 3'b000;
  accum = 8'h47;
  data = 8'h03;
  # 1
  check_outputs; // call this task to check output signal
  ……
end
```



Testbench of the ALU Module


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

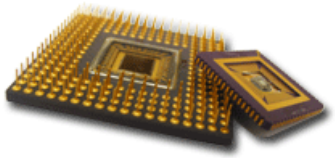
Testbench of the ALU Module

- You should design other testcases to check your circuit functional correctness.

	OPCODE	DATA	ACCUM	ALU_OUT	ZERO	OVERFLOW	PARITY	SIGN
ADD OPERATION :	001	00000001	01111111	01111111	0	1	1	0

	OPCODE	DATA	ACCUM	ALU_OUT	ZERO	OVERFLOW	PARITY	SIGN
SUB OPERATION :	010	11111111	10000000	10000001	0	0	0	1

	OPCODE	DATA	ACCUM	ALU_OUT	ZERO	OVERFLOW	PARITY	SIGN
MUL OPERATION :	110	11111100	00000111	11100100	0	0	0	1



Lab 3 Demo Guide

- A sample project, lab3.zip, is available on E3. Please do not modify the IO port in alu.v .
- You should upload your lab3 solution to E3 before the deadline.
- You should design your ALU only based on combinational circuit.
- During the demo time, TA will ask you to execute some hidden testcases to check your circuit correctness.
  - You can download your code from E3 during demo.