





module alu(

input [7:0] in\_a , //input a

input [7:0] in\_b , //input b

input [3:0] opcode , //opcode input

output reg [7:0] alu\_out , //alu output

output reg alu\_zero , //logic '1' when alu\_output [7:0] is all zeros

output reg alu\_carry //indicates a carry out from ALU

);

//Declaration of OP code macro

//c\_ to mark the identifier as a const

parameter c\_add = 4'h1;

parameter c\_sub = 4'h2;

parameter c\_inc = 4'h3;

parameter c\_dec = 4'h4;

parameter c\_or = 4'h5;

parameter c\_and = 4'h6;

parameter c\_xor = 4'h7;

parameter c\_shr = 4'h8;

parameter c\_shl = 4'h9;

parameter c\_onescomp = 4'hA;

parameter c\_twoscomp = 4'hB;

//Creating a temporary wire

reg [8:0] temp;

always\_comb

case (opcode)

c\_add : begin

temp = in\_a + in\_b;

alu\_out[7:0] = temp[7:0];

alu\_carry = temp[8];

end

c\_sub : begin

temp = in\_a - in\_b;

alu\_out[7:0] = temp[7:0];

alu\_carry = temp[8];

end

c\_inc : begin

temp = in\_a + 1;

alu\_out[7:0] = temp[7:0];

alu\_carry = temp[8];

end

c\_dec : begin

temp = in\_a - 1;

alu\_out[7:0] = temp[7:0];

alu\_carry = temp[8];

end

c\_or : begin

alu\_out[7:0] = in\_a | in\_b;

alu\_carry = 0;

end

c\_and : begin

alu\_out[7:0] = in\_a & in\_b;

alu\_carry = 0;

end

c\_xor : begin

alu\_out[7:0] = in\_a ^ in\_b;

alu\_carry = 0;

end

c\_shr : begin

alu\_out[7:0] = in\_a[7:0]>>1;

alu\_carry = 0;

end

c\_shl : begin

temp = in\_a[7:0]<<1;

alu\_out[7:0] = temp[7:0];

alu\_carry = temp[8];

end

c\_onescomp:begin

alu\_out = ~in\_a;

alu\_carry = 0;

end

c\_twoscomp:begin

temp = ~in\_a + 1'b1;

alu\_out = temp[7:0];

alu\_carry = temp[8];

end

default:begin

alu\_out = in\_a;

alu\_carry = 0;

end

endcase

assign alu\_zero = !alu\_out;

endmodule

