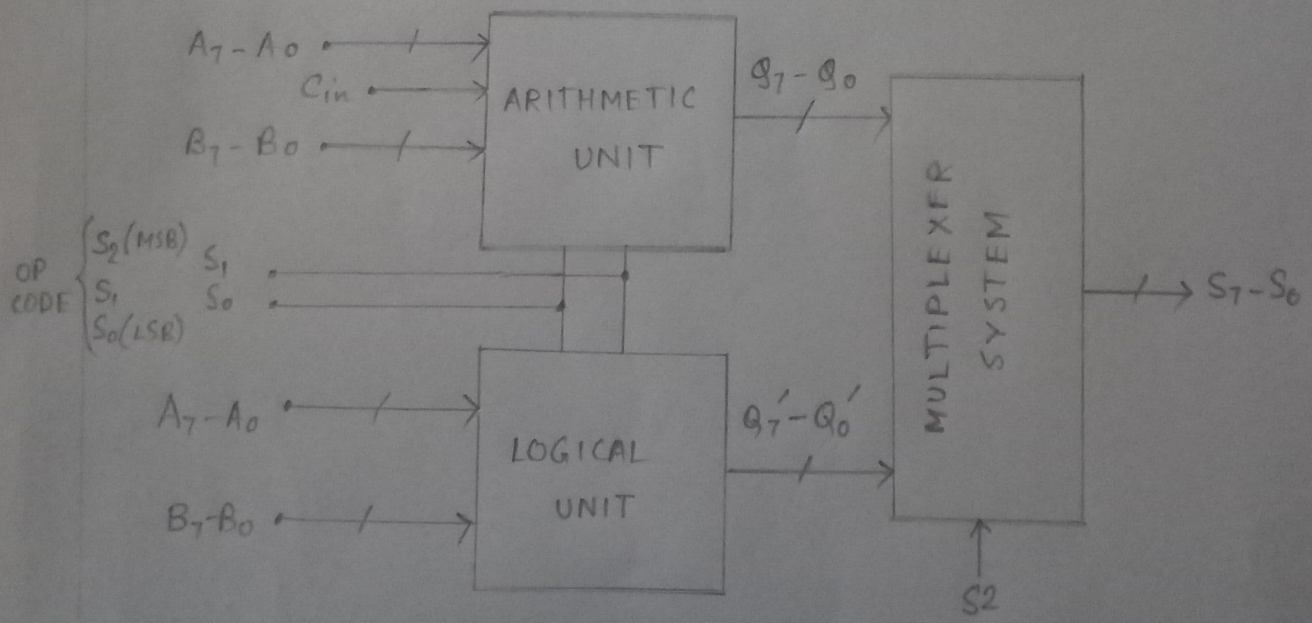
Name - Moitrish Maity Dept - CSE Roll-48 4th Sem

EXPERIMENT-13

- **Title:** Arithmetic and Logical Unit (ALU)
- **Objective:** Implementation of ALU
- **Course Outcome:** CO6
- **Bloom's Level:** Comprehension
- **Problem Statement:** Write a verilog program to implement an 8 bit ALU which will perform following function table.
- **Theory:** Arithmetic logic unit (ALU) is an important part of microprocessor. In digital processors logical and arithmetic operation executes using ALU. In this experiment we have implemented a 8 bit ALU circuit using MUX's and parallel adders. Based on selection lines of every MUX, a specific arithmatical or logical operation is selected and executed.
- **Logic Diagram:**



Function Table:

OPCODE	OPERATION	REMARKS
LLL	A-B	without carry
LLH	A+B	w/o carry
LHL	A+B+Cin	with carry
LHH	A-B-Cin	with carry
HLL	A ^ B	XOR operation
HLH	~B	NOT operation
HHL	A | B	OR operation
HHH	A & B	AND operation

(A & B are 2 8 bit no.)

OPCODE

S_2 S_1 S_0

↙ (MSB) ↓ (LSB)

Design Code:

```
module mux_4to1(input a0, input a1, input a2, input a3,
                input [1:0] s, output y);

  assign y = s[1] ? ( s[0] ? a[3] : a[2]) : (s[0] ? a[1] : a[0]) ;

endmodule
module mux_2to1 (input a, input b, input s, output y);
  assign y = s ? b : a ;
endmodule
module full_adder (input a, b, c, output s, cout);
  assign s = a ^ b ^ c;
  assign c_out = (a & b) | (c & (a ^ b));
endmodule
module adder_subtractor (input [7:0]a, input [7:0]b, input
                         c0, output [7:0]s, output cout);

  wire c1, c2, c3, c4, c5, c6, c7;
  full_adder f1(a[0], b[0], c[0], s[0], c1);
  full_adder f2(a[1], b[1], c[1], s[1], c2);
  full_adder f3(a[2], b[2], c[2], s[2], c3);
  full_adder f4(a[3], b[3], c[3], s[3], c4);
  full_adder f5(a[4], b[4], c[4], s[4], c5);
```

```
    full_adder f6 (a[5], b[5], c5, s[5], c6);
    full_adder f7 (a[6], b[6], c6, s[6], c7);
    full_adder f8 (a[7], b[7], c7, s[7], cout);
endmodule
module arithmetic (input [7:0] a, input [7:0] b, input cin,
                   input [1:0] sel, output [7:0] out0,
                   output cout);
    wire [7:0] y0;
    wire carry;
    mux_4to1 uut1 (~b[0], b[0], b[0], ~b[0], sel[1:0], y0[0]);
    mux_4to1 uut2 (~b[1], b[1], b[1], ~b[1], sel[1:0], y0[1]);
    mux_4to1 uut3 (~b[2], b[2], b[2], ~b[2], sel[1:0], y0[2]);
    mux_4to1 uut4 (~b[3], b[3], b[3], ~b[3], sel[1:0], y0[3]);
    mux_4to1 uut5 (~b[4], b[4], b[4], ~b[4], sel[1:0], y0[4]);
    mux_4to1 uut6 (~b[5], b[5], b[5], ~b[5], sel[1:0], y0[5]);
    mux_4to1 uut7 (~b[6], b[6], b[6], ~b[6], sel[1:0], y0[6]);
    mux_4to1 uut8 (~b[7], b[7], b[7], ~b[7], sel[1:0], y0[7]);
    assign carry = (sel[1] ^ sel[0]) ? cin : (~cin);
    adder_subtractor xx1 (a[7:0], y0[7:0], carry, out0[7:0], cout);
endmodule
module logical (input [7:0] a, input [7:0] b,
                output [7:0] out1);
    mux_4to1 ut1 ((a[0] ^ b[0]), (~b[0]), (a[0] | b[0]), (a[0] & b[0]),
                  sel[1:0], out1[0]);
    mux_4to1 ut2 ((a[1] ^ b[1]), (~b[1]), (a[1] | b[1]), (a[1] & b[1]),
                  sel[1:0], out1[1]);
    mux_4to1 ut3 ((a[2] ^ b[2]), (~b[2]), (a[2] | b[2]), (a[2] & b[2]),
                  sel[1:0], out1[2]);
    mux_4to1 ut4 ((a[3] ^ b[3]), (~b[3]), (a[3] | b[3]), (a[3] & b[3]),
                  sel[1:0], out1[3]);
    mux_4to1 ut5 ((a[4] ^ b[4]), (~b[4]), (a[4] | b[4]), (a[4] & b[4]),
                  sel[1:0], out1[4]);
    mux_4to1 ut6 ((a[5] ^ b[5]), (~b[5]), (a[5] | b[5]), (a[5] & b[5]),
                  sel[1:0], out1[5]);
```

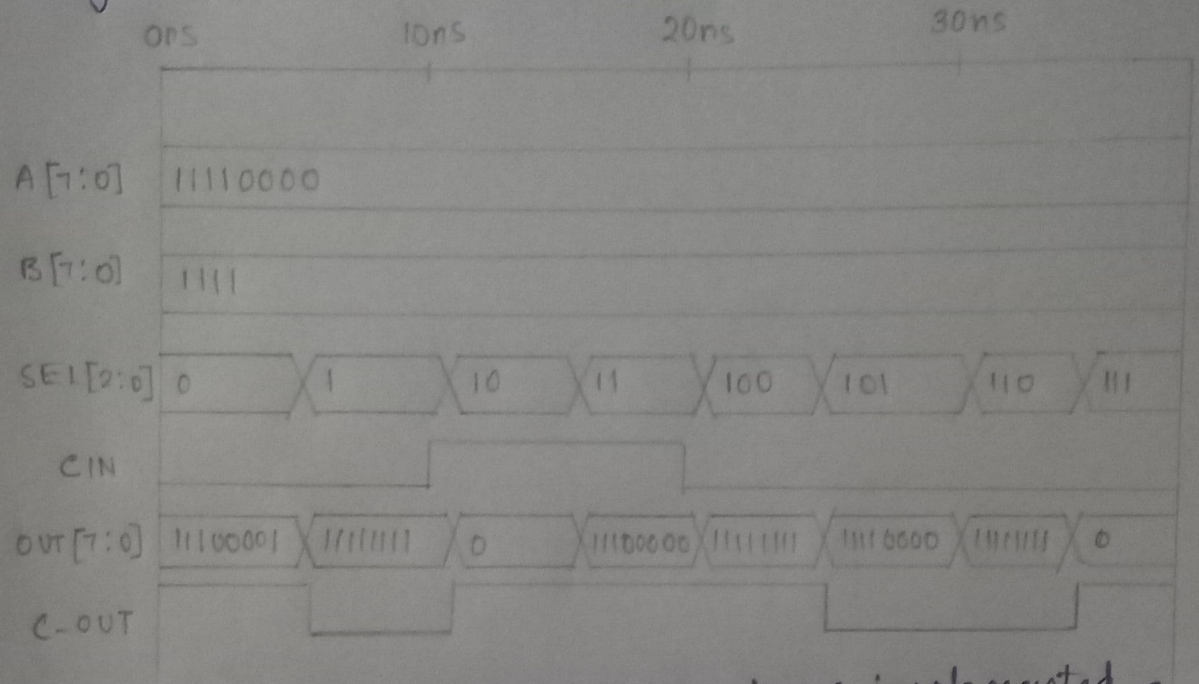
```
    mux_4to1 ut7((a[6]^b[6]), (~b[6]), (a[6]|b[6]), (a[6]&b[6]),
                                    sel[1:0], out1[6]);
    mux_4to1 ut8((a[7]^b[7]), (~b[7]), (a[7]|b[7]), (a[7]&b[7]),
                                    sel[1:0], out1[7]);
endmodule
module alu(input [7:0]a, input [7:0]b, input cin, input[2:0]sel,
            output [7:0]out, output [7:0]out1, output
                                    [7:0]out0, output cout);
    arithmetic f1(a[7:0], b[7:0], cin, sel[1:0], out0[7:0], cout);
    logical f2(a[7:0], b[7:0], sel[1:0], out1[7:0]);
    mux_2to1 uut1(out0[0], out1[0], sel[2], out[0]);
    mux_2to1 uut2(out0[1], out1[1], sel[2], out[1]);
    mux_2to1 uut3(out0[2], out1[2], sel[2], out[2]);
    mux_2to1 uut4(out0[3], out1[3], sel[2], out[3]);
    mux_2to1 uut5(out0[4], out1[4], sel[2], out[4]);
    mux_2to1 uut6(out0[5], out1[5], sel[2], out[5]);
    mux_2to1 uut7(out0[6], out1[6], sel[2], out[6]);
    mux_2to1 uut8(out0[7], out1[7], sel[2], out[7]);
endmodule
```

□ Testbench Code:

```
module alu_tb();
    reg [7:0]A;
    reg [7:0]B;
    reg CIN;
    reg [2:0]SEL;
    wire [7:0]OUT;
    wire C_OUT;
    alu uut(.a(A), .b(B), .cin(CIN), .sel(SEL), .out(OUT), .cout(C_OUT));
    initial begin
        $dumpfile("dump.vcd"); $dumpvars(1);
        A = 8'b11110000;
        A = 8'b00001111;
        CIN = 0
        SEL = 3'b000; #5;
```

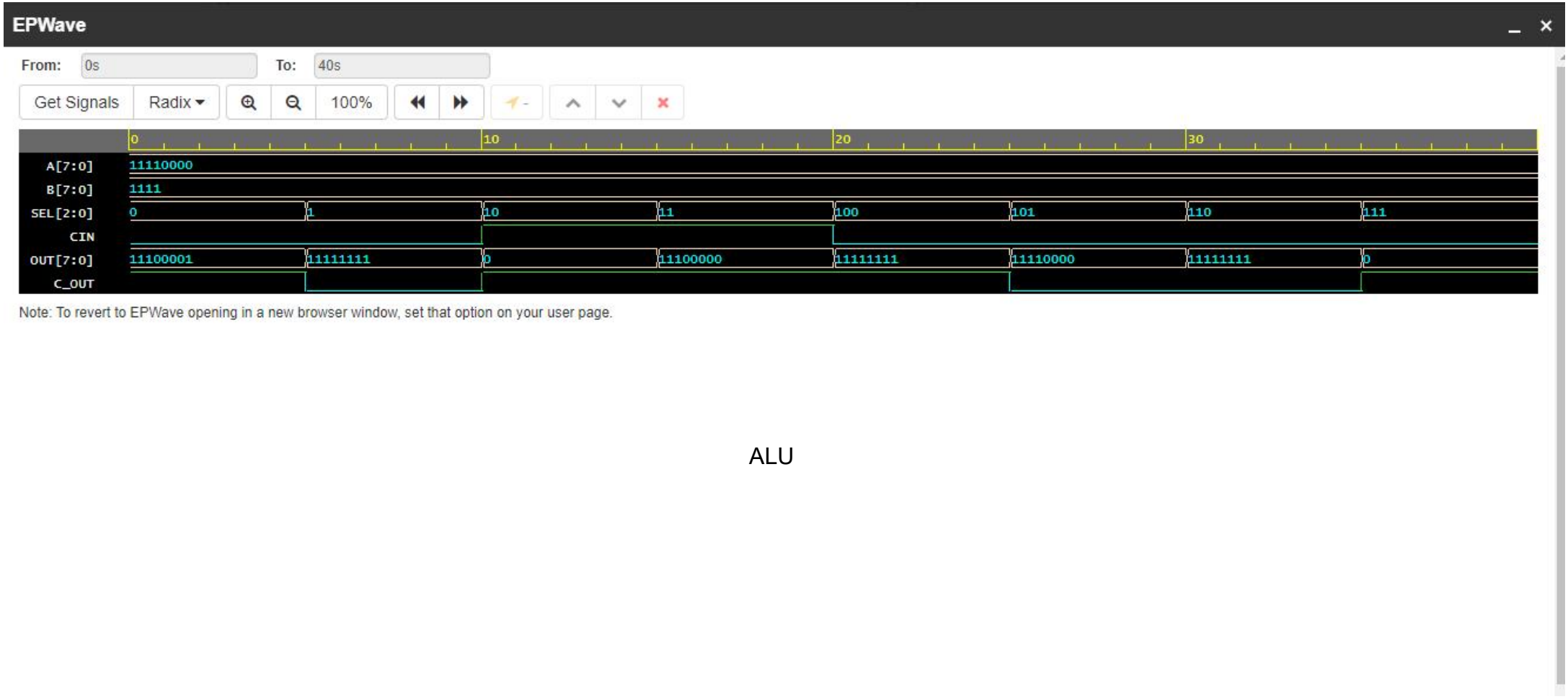
```
        SEL=3'b001 ; #5 ;
        CIN=1 ;
        SEL=3'b010 ; #5 ;
        SEL =3'b011 ; #5 ;
        CIN=0 ;
        SEL =3'b100 ; #5 ;
        SEL = 3'b101 ; #5 ;
        SEL = 3'b110 ; #5 ;
        SEL = 3'b111 ; #5 ;
    end
endmodule
```

□ Timing Diagram :



□ Discussion: In this experiment, we have implemented a 8bit ALU circuit using 4×1 and 2×1 MUX and a parallel adder-subtractor module. We came to know that this ALU works as a processing unit in computer system. We have learned various HDL terms also.

□ Justification of CO: In this experiment, we have designed an ALU circuit which is one of the most important processing element in a computer system. We made this circuit using verilog environment, which fulfils the conditions. Hence, CO6 is justified.


EPWave
From:
0s
To:
40s
Get Signals
Radix
100%
0
10
20
30
A[7:0]
11110000
B[7:0]
1111
SEL[2:0]
0
1
10
11
100
101
110
111
CIN
OUT[7:0]
11100001
11111111
0
11100000
11111111
11110000
11111111
0
C_OUT
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

ALU