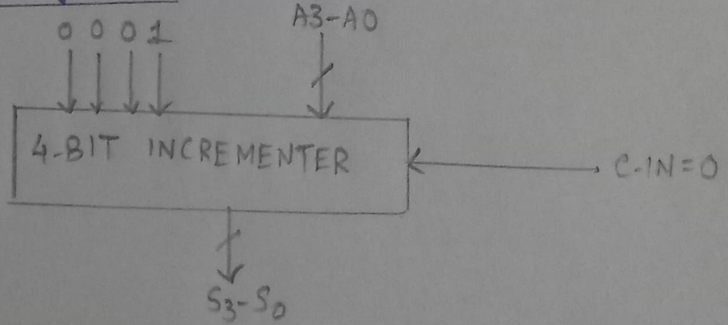
Name- Moitrush Maity Roll-48 CSE 4th Sem

EXPERIMENT-8

- **Title:** Arithmetic Operation
- **Objective:** Implementations of arithmetic operations.
- **Course Outcome:** CO2
- **Bloom's Level:** Evaluation

8.1 Incrementer Circuit

- **Problem Statement:** Write a verilog program to design, simulate and test the functionality of 4-bit incrementer circuit.
- **Theory:** Binary incrementer increases value of a 4-bit number by 001 (1). It simply add one to the existing value of a 4 bit number. If a given 4 bit binary no. is 0011(3) then incrementer circuit changes its value to 0100(4).
- **Logic Diagram:**



- **Function Table:**

INPUTS				OUTPUTS				
A3	A2	A1	A0	C-OUT	S3	S2	S1	S0
L	H	L	L	L	L	H	L	H
L	L	H	H	L	L	H	L	L
L	H	L	H	L	L	H	H	L
L	H	H	L	L	L	H	H	H
L	H	H	H	L	H	L	L	L
H	L	L	L	L	H	L	L	H
H	L	L	H	L	H	L	H	L
H	H	L	L	L	H	H	L	H

Design Code:

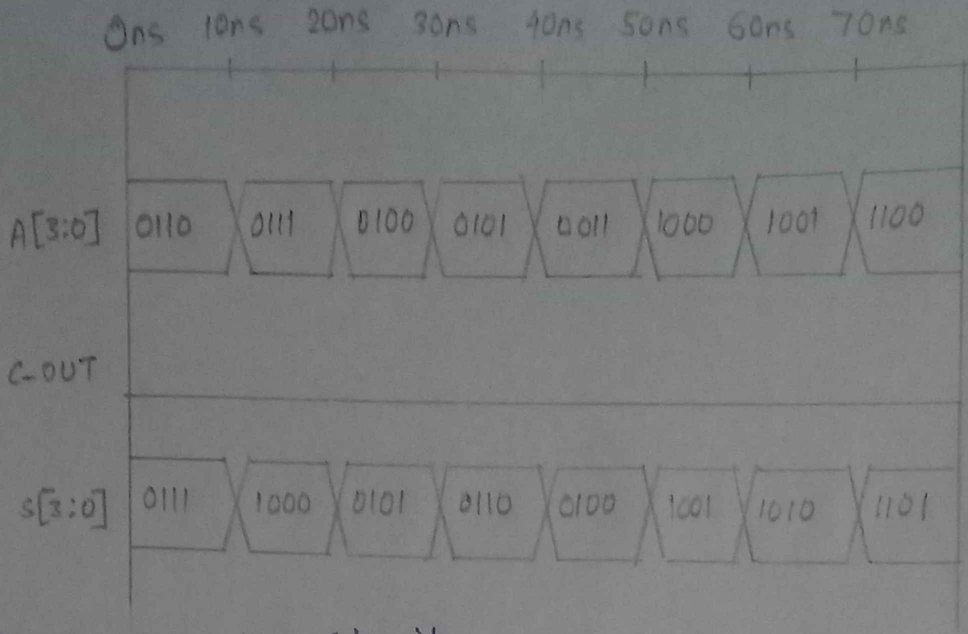
```
module full-adder(input a,b,c, output s,cout);
    assign s = a ^ b ^ c;
    assign cout = (a&b) | (c&(a^b));
endmodule
module incrementer(input [3:0]a, input [3:0]b, input c0,
                        output [3:0]s, output cout);

    wire c1,c2,c3;
    wire [3:0]bnew;
    assign bnew[0] = c0 ^ b[0];
    assign bnew[1] = c0 ^ b[1];
    assign bnew[2] = c0 ^ b[2];
    assign bnew[3] = c0 ^ b[3];
    full-adder f1 (a[0], bnew[0], c0, s[0], c1);
    full-adder f2 (a[1], bnew[1], c1, s[1], c2);
    full_adder f3 (a[2], bnew[2], c2, s[2], c3);
    full-adder f4 (a[3], bnew[3], c3, s[3], cout);
endmodule
```

Testbench Code:

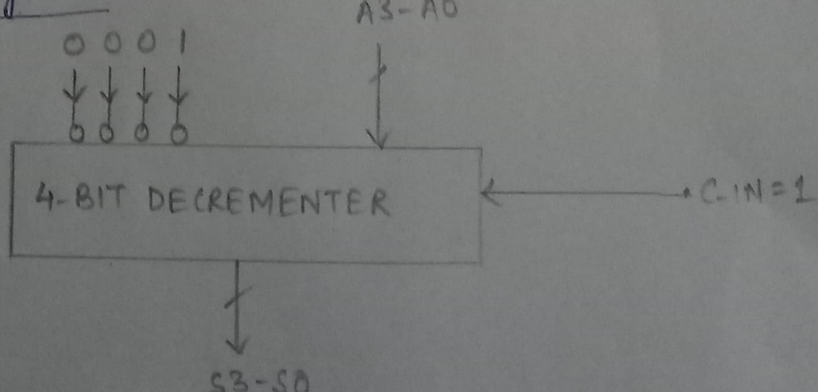
```
module incrementer_tb();
    reg [3:0]A;
    wire [3:0]S;  wire C_OUT;
    incrementer uut(.a(A), .b(4'b0001), .c0(1'b0),
                        .s(S), .cout(C_OUT));
    initial
      begin
        $dumpfile("dump.vcd"); $dumpvars;
        A = 4'b0110; #10;
        A = 4'b0111; #10;
        A = 4'b0100; #10;
        A = 4'b0101; #10;
        A = 4'b0011; #10;
        A = 4'b1000; #10;
        A = 4'b1001; #10;
        A = 4'b1100; #10;
      end
endmodule
```

Timing Diagram:



8.2 Decrementer Circuit

- **Problem statement:** Write a verilog program to design, simulate and test the functionality of a 4-bit decrementer circuit.
- **Theory:** Binary decrementer decreases value of a 4-bit binary number by 1 (001). It simply subtract 1 (001) from existing value of a 4-bit number using 2's compliment method. If a given 4-bit binary number is 0011(3), then decrementer circuit changes its value to 0010(2).
- **Logic Diagram:**



Function Table:

INPUTS				OUTPUTS				
A3	A2	A1	A0	C-OUT	S3	S2	S1	S0
L	H	L	L	H	L	L	H	H
L	L	H	H	H	L	L	H	L
L	H	L	H	H	L	H	L	L
L	H	H	L	H	L	H	L	H
L	H	H	H	H	L	H	H	L
H	L	L	L	H	L	H	H	H
H	L	L	H	H	H	L	L	L
H	H	L	L	H	H	L	H	H

Design Code:

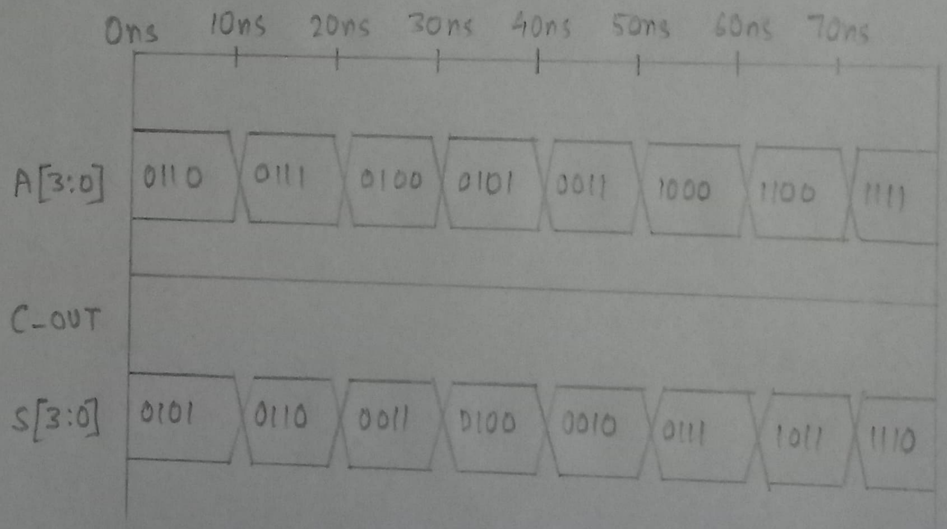
```
module full_adder (input a,b,c, output s,cout);
        assign s = a ^ b ^ c;
        assign cout = (a & b) | (c & (a ^ b));
endmodule
module decrementer (input [3:0]a, input [3:0]b, input c0,
                    output [3:0]s , output cout);

   wire c1, c2, c3;
   wire [3:0]bnew;
   assign bnew[0] = c0 ^ b[0];
   assign bnew[1] = c0 ^ b[1];
   assign bnew[2] = c0 ^ b[2];
   assign bnew[3] = c0 ^ b[3];
   full_adder f1(a[0], bnew[0], c0, s[0], c1);
   full_adder f2(a[1], bnew[1], c1, s[1], c2);
   full_adder f3(a[2], bnew[2], c2, s[2], c3);
   full_adder f4(a[3], bnew[3], c3, s[3], cout);
endmodule
```

Testbench code:

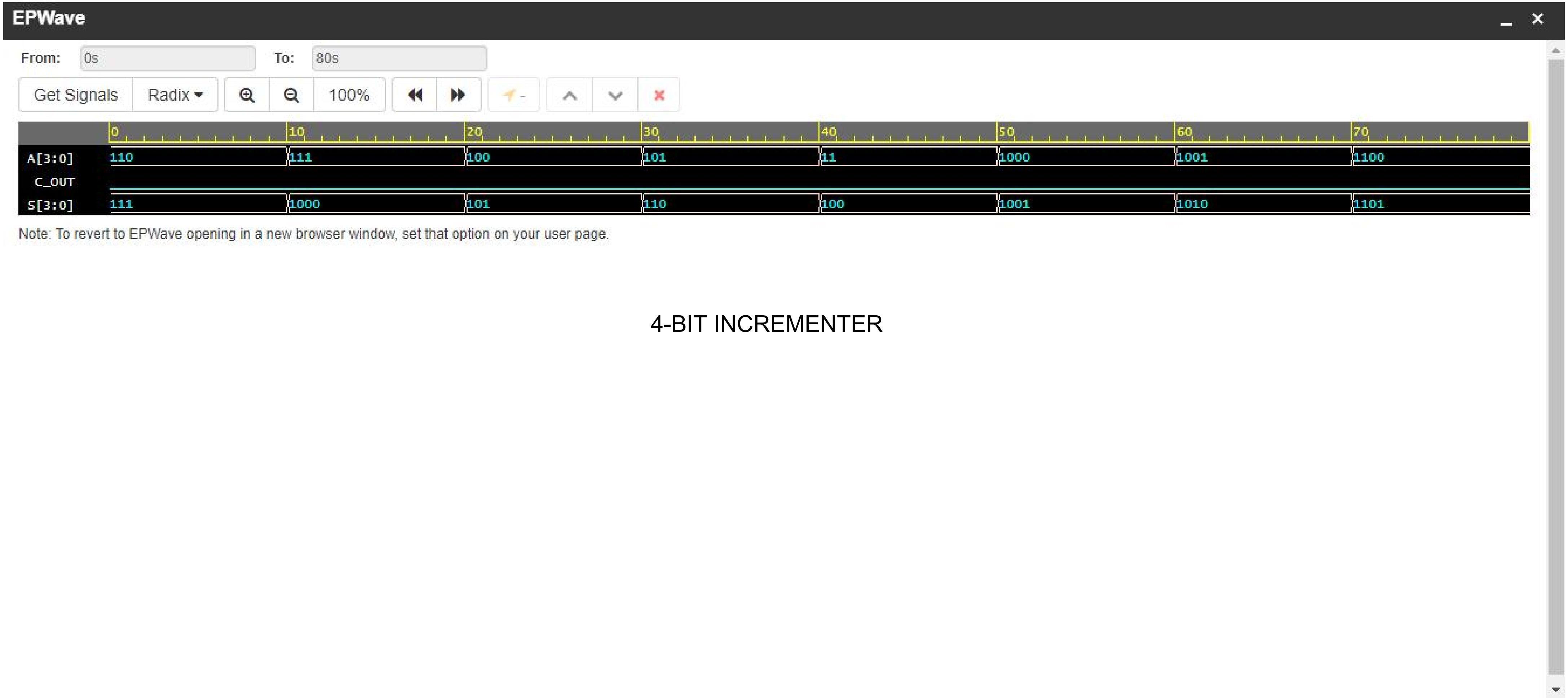
```
module decrementer_tb();
    reg [3:0]A;
    wire [3:0]S;
    wire C_OUT;
    decrementer out (.a(A),.b(4'b0001),.c0(1'b1),.s(S),.cout(cout));
    Initial begin
        $dumpfile ("dump.vcd");
        $dumpvars;
        A=4'b0110; #10;
        A=4'b0111; #10;
        A=4'b0100; #10;
        A=4'b0101; #10;
        A=4'b0011; #10;
        A=4'b1000; #10;
        A=4'b1100; #10;
        A=4'b1111; #10;
    end
endmodule
```

Timing Diagram:

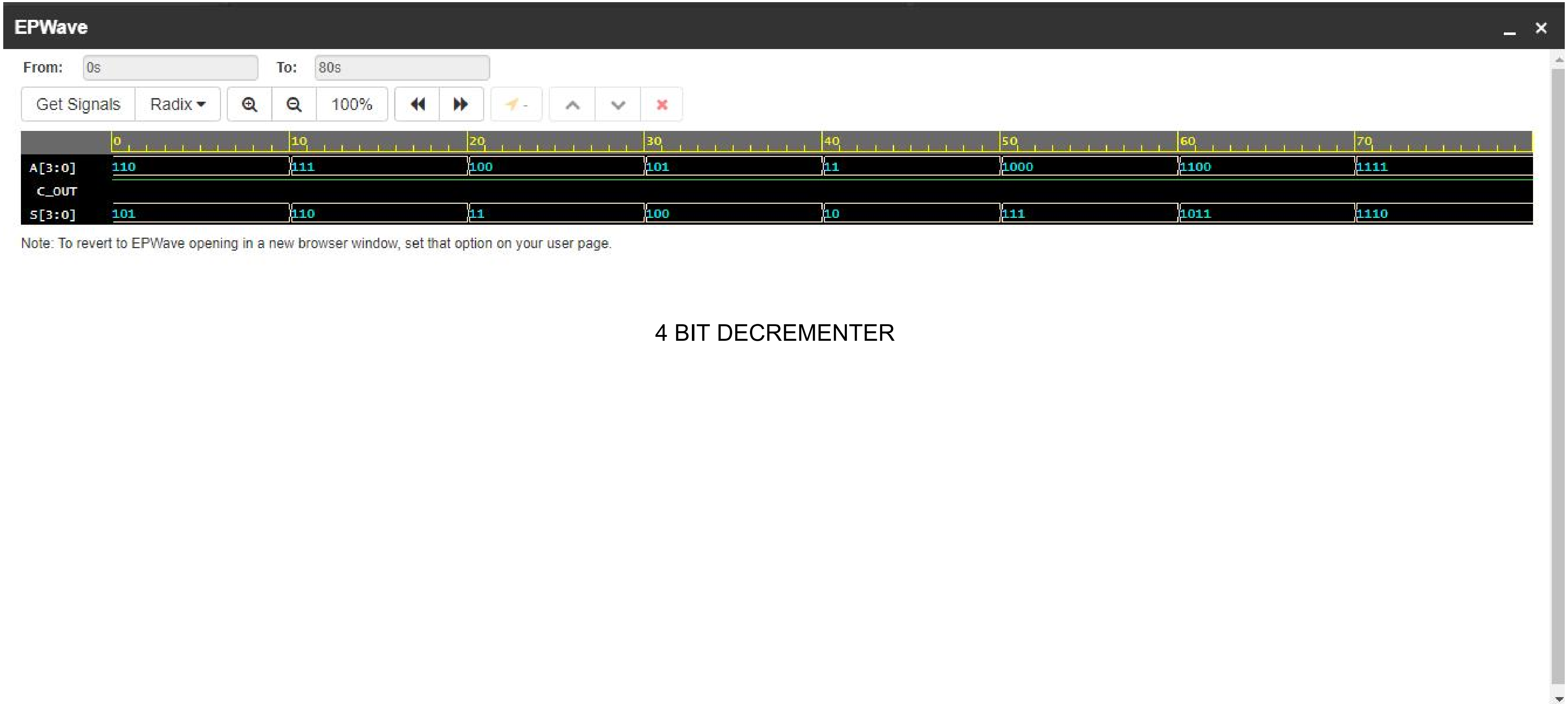


Discussion: In this experiment, we have implemented various arithmetic operations like incrementation, decrementation etc. We have also learned various HDL keywords & logic also.

□ Justification of CO: In this experiment, we have implemented 4 bit incrementer and decrementer circuit which are actually combinational circuits. Thus, CO2 is justified.



4-BIT INCREMENTER



4 BIT DECREMENTER