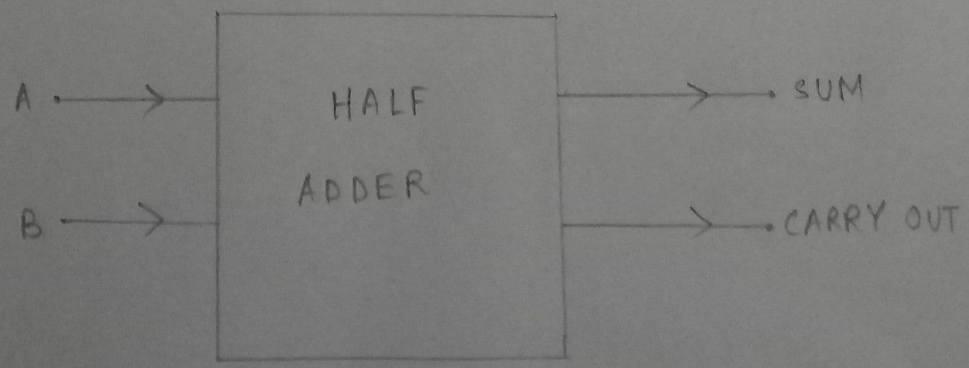
Name- Moitrish Maity CSE 4th Sem Roll-48

EXPERIMENT-7

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

7.1 Half-Adder :

- Problem Statement: Write a verilog program to design, simulate and test the functionality of a half adder circuit.
- Theory: The addition of 2 bits is done using a combinational circuit called half adder. The input variables are augend and addend bits and the output bits are sum and carry bits. If A and B are 2 input bits, then sum = $A \oplus B$ and carry out = $A \cdot B$.
- Logic Diagram:



Truth Table:

INPUTS		OUTPUTS	
A	B	SUM	CARRY OUT
L	L	L	L
L	H	H	L
H	L	H	L
H	H	L	H

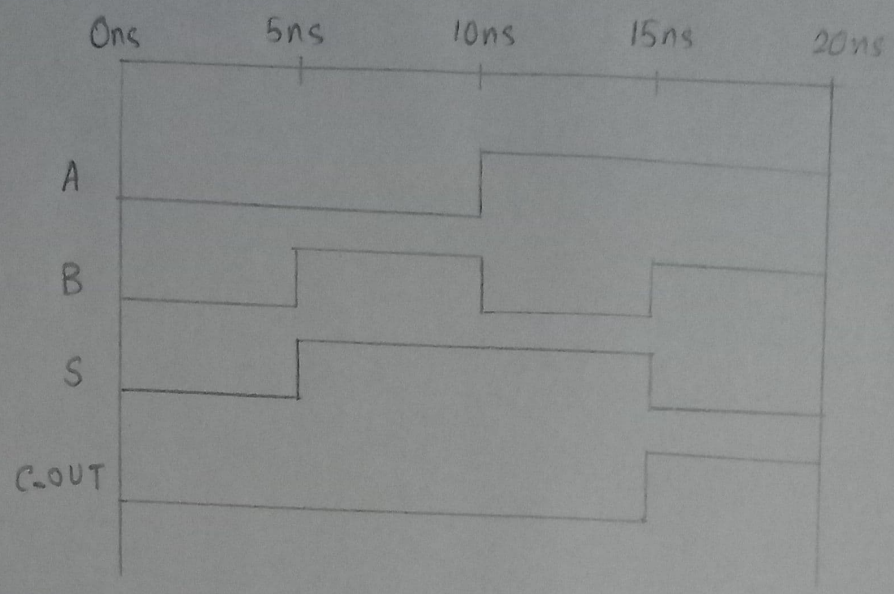
Design Code:

```
module half_adder(input a,b, output s,c);
    assign s = a ^ b;
    assign c = a & b;
endmodule
```

Testbench Code:

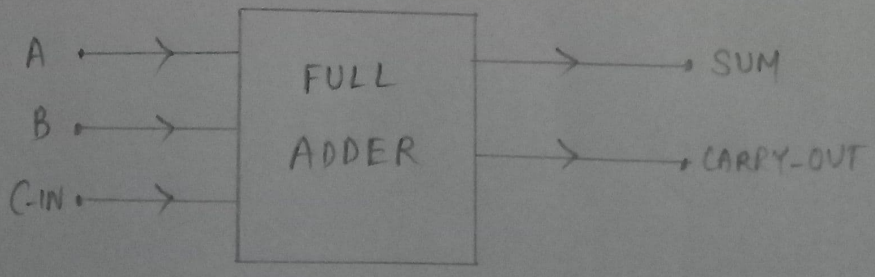
```
module half_adder_tb();
    reg A,B;
    wire S, C_OUT;
    half_adder uut (.a(A), .b(B), .s(S), .c(C_OUT));
    initial
        begin
        $dumpfile("dump.vcd"); $dumpvars;
        A=0 ; B=0 ; #5;
        A=0 ; B=1 ; #5;
        A=1 ; B=0 ; #5;
        A=1 ; B=1 ; #5;
        end
endmodule.
```

- Timing Diagram:



7.2 Full Adder

- Problem Statement: Write a verilog program to design, simulate and test the functionality of a full adder circuit.
- Theory: Full adder is an adder circuit which adds 3 inputs and produces 2 outputs. First two inputs are A and B and third one is an input carry as C_IN. Output carry is designated by C_OUT and normal output as Sum output. Sum = $A \oplus B \oplus C\_IN$
  Carry-out = $AB + C\_IN(A \oplus B)$.
- Logic Diagram:



Truth Table

INPUTS			OUTPUTS	
A	B	C_{IN}	SUM	C_{OUT}
L	L	L	L	L
L	L	H	H	L
L	H	L	H	L
L	H	H	L	H
H	L	L	H	L
H	L	H	L	H
H	H	L	L	H
H	H	H	H	H

Design Code :

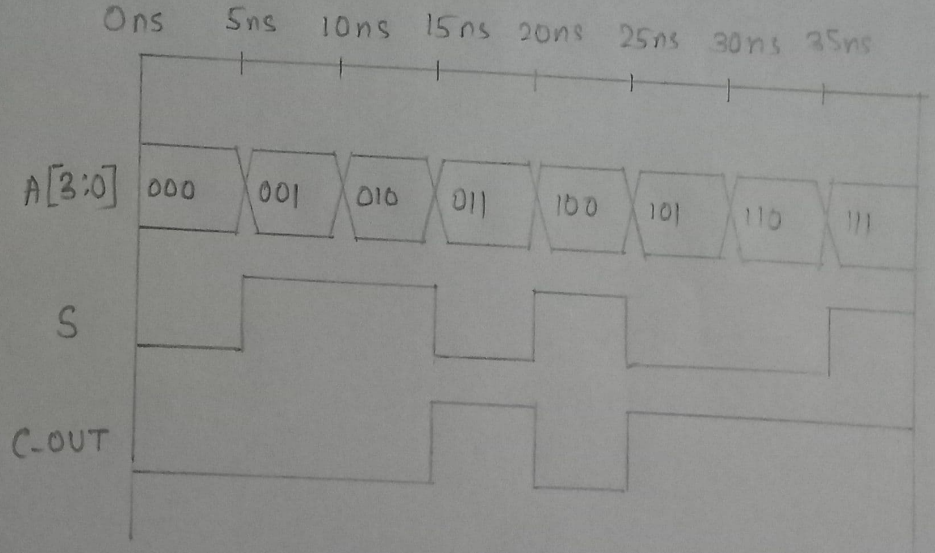
```
module half_adder (input a,b, output s,c);
    assign s = a ^ b;
    assign c = a & b;
endmodule
module full_adder (input [2:0]a, output s, output cout);
    wire c1, c2, x;
    half_adder f1 ( a[2], a[1], x, c1);
    half_adder f2 ( x, a[0], s, c2);
    assign cout = c1 | c2 ;
endmodule
```

Testbench Code :

```
module full_adder_tb();
    reg [2:0] A;
    wire S, C_OUT;
    full_adder uut (.a(A), .s(S), .cout(C_OUT));
```

```
initial
    begin
    $dumpfile("dump.vcd"); $dumpvars;
    A = 3'b000 ; #5 ;
    A = 3'b001 ; #5 ;
    A = 3'b010 ; #5 ;
    A = 3'b011 ; #5 ;
    A = 3'b100 ; #5 ;
    A = 3'b101 ; #5 ;
    A = 3'b110 ; #5 ;
    A = 3'b111 ; #5 ;
    end
endmodule
```

□ Timing Diagram:

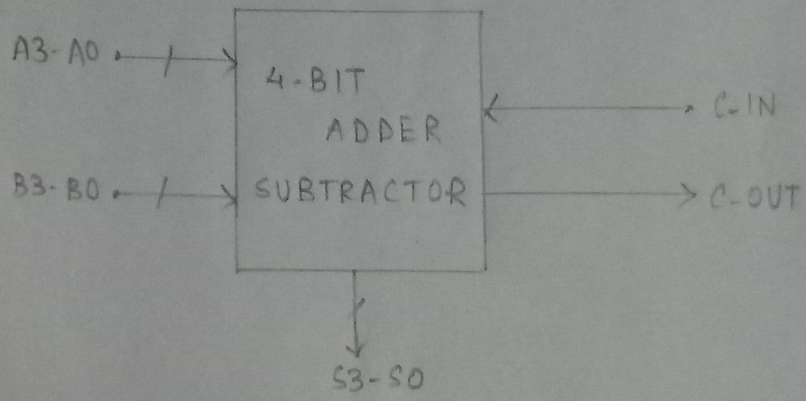


7.3 4-bit Adder Subtractor

□ Problem Statement: Write a hardware description to design, simulate and test the functionality of a 4-bit adder / subtractor circuit.

□ Theory: Binary adder/subtractor, is one which is capable of both addition and subtraction of binary numbers in one circuit itself. The operation being performed depends upon the value of control signal. If the control signal holds value '1' it performs subtraction operation by using 2's compliment method. And when control signal becomes '0', addition operation is performed.

□ Logic Diagram:-



□ Function Table:

INPUTS									OUTPUTS				
Cin	A3	A2	A1	A0	B3	B2	B1	B0	C_{out}	S3	S2	S1	S0
L	L	L	L	L	L	L	L	L	L	L	L	L	L
H	L	H	L	H	L	L	L	H	H	L	H	L	L
L	L	H	H	L	L	L	H	L	L	H	L	L	L
H	L	L	L	L	L	L	L	L	H	L	L	L	L
L	H	L	L	L	L	L	L	H	L	H	L	L	H
H	L	H	H	H	L	L	L	H	H	L	H	H	L
L	L	H	H	L	L	L	H	L	L	H	L	L	L
H	H	L	L	L	L	H	L	H	H	L	L	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
H	H	H	H	H	H	H	H	H	H	L	L	L	L

Cin: H → Subtraction, L → Addition.

Design Code:

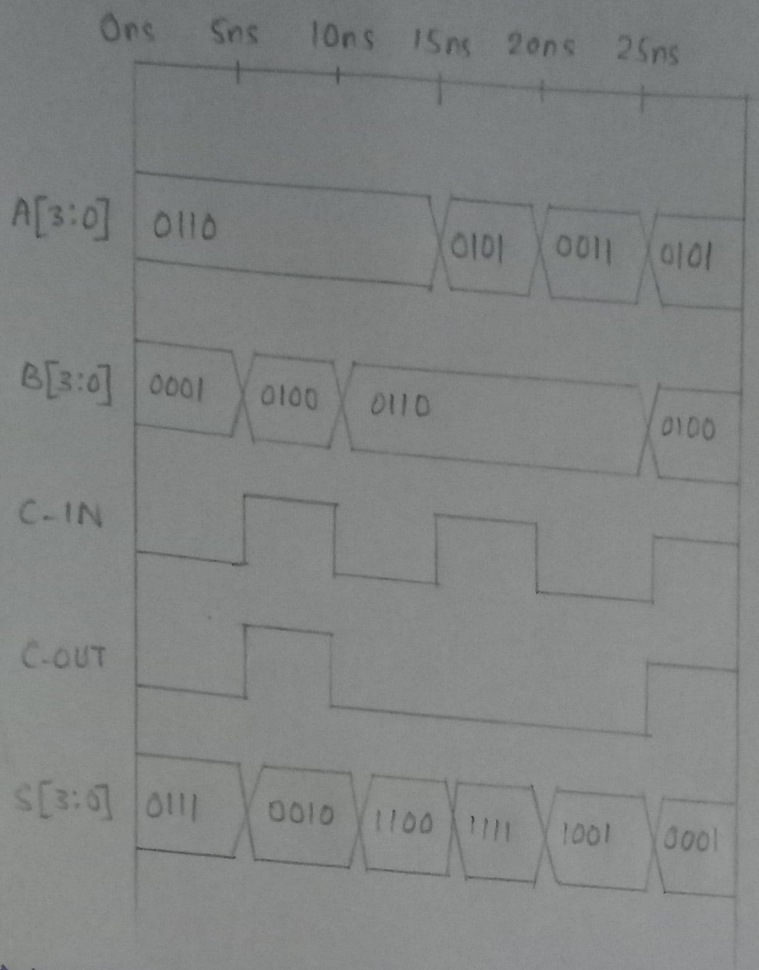
```
module full_adder (input a,b,c , output s, cout);
        assign s = a ^ b ^ c ;
        assign cout = (a & b) | (c & (a ^ b));
endmodule

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

Testbench Code:

```
module adder_subtractor_tb();
    reg [3:0]A ; reg [3:0]B ;
    reg C_IN;
    wire [3:0]S ; wire C_OUT ;
    adder_subtractor uut (.a(A) , .b(B), .c0(C_IN) , .s(S), .cout(C_OUT));
    initial
        begin
        $dumpfile ("dump.vcd"); $dumpvars;
        A = 4'b0110 ; B = 4'b0001 ; C_IN = 0 ; #5 ;
        A = 4'b0110 ; B = 4'b0100 ; C_IN = 1 ; #5 ;
        A = 4'b0110 ; B = 4'b0110 ; C_IN = 0 ; #5 ;
        A = 4'b0101 ; B = 4'b0110 ; C_IN = 1 ; #5 ;
        A = 4'b0011 ; B = 4'b0110 ; C_IN = 0 ; #5 ;
        A = 4'b0101 ; B = 4'b0100 ; C_IN = 1 ; #5 ;
        end
endmodule
```

Timing Diagram:



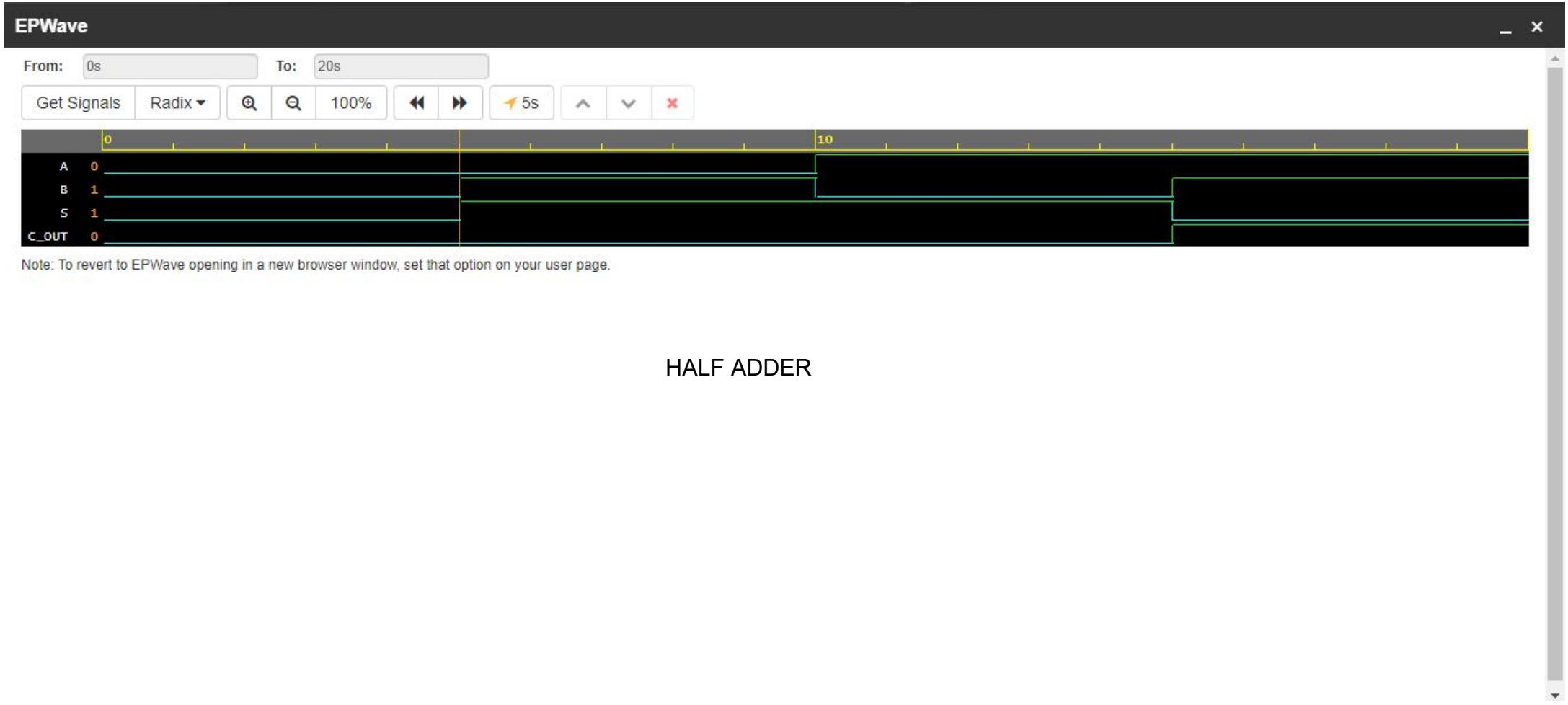
Discussion: In this experiment, we have learned working principals of half adder, full adder and addar subtractor composite unit. We have learned various HDL keywords also.

Questionnaire:

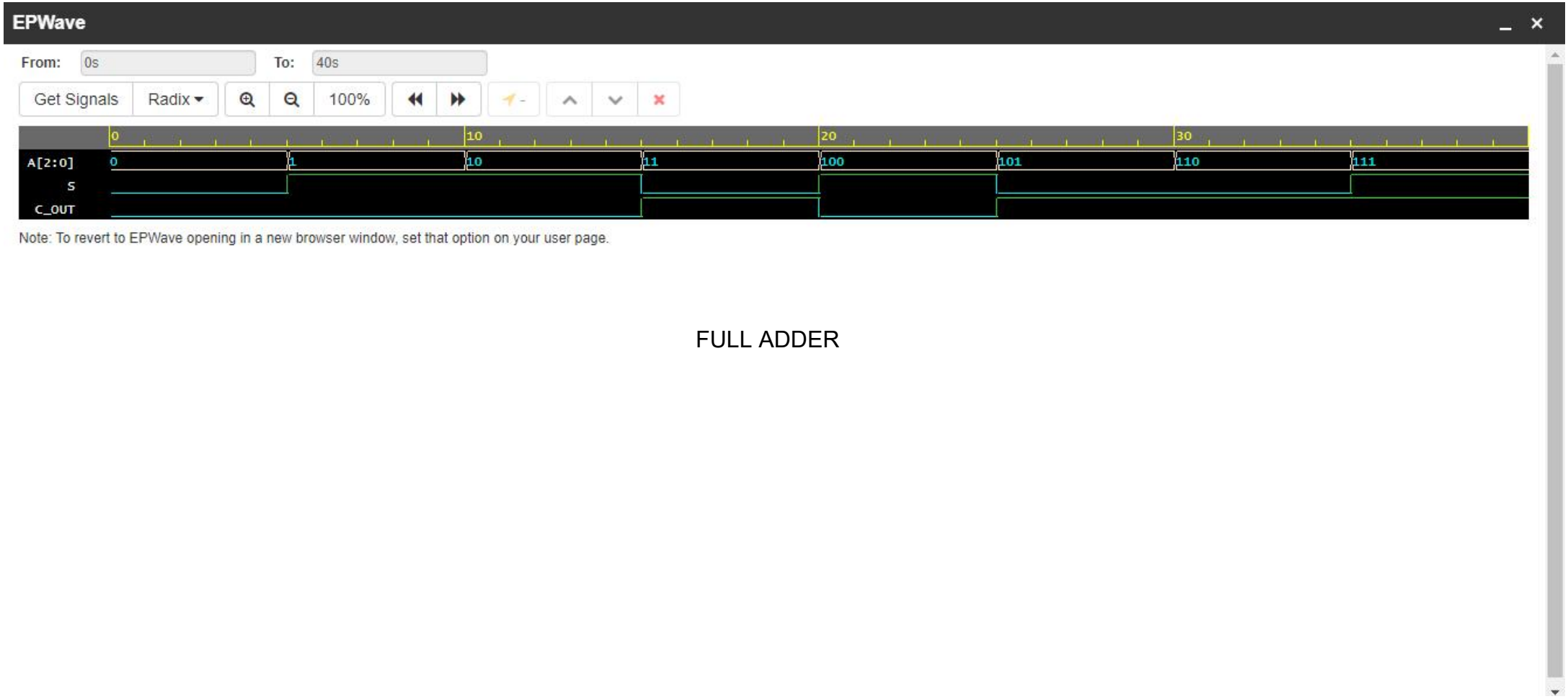
(7.1) x = 8'b 10101101

<< means left shift operator. So, x<= x<<1; shifting the value of x register by one unit. But, here we are using non blocking sequential statement. So, value of x will be 8'b10101101.

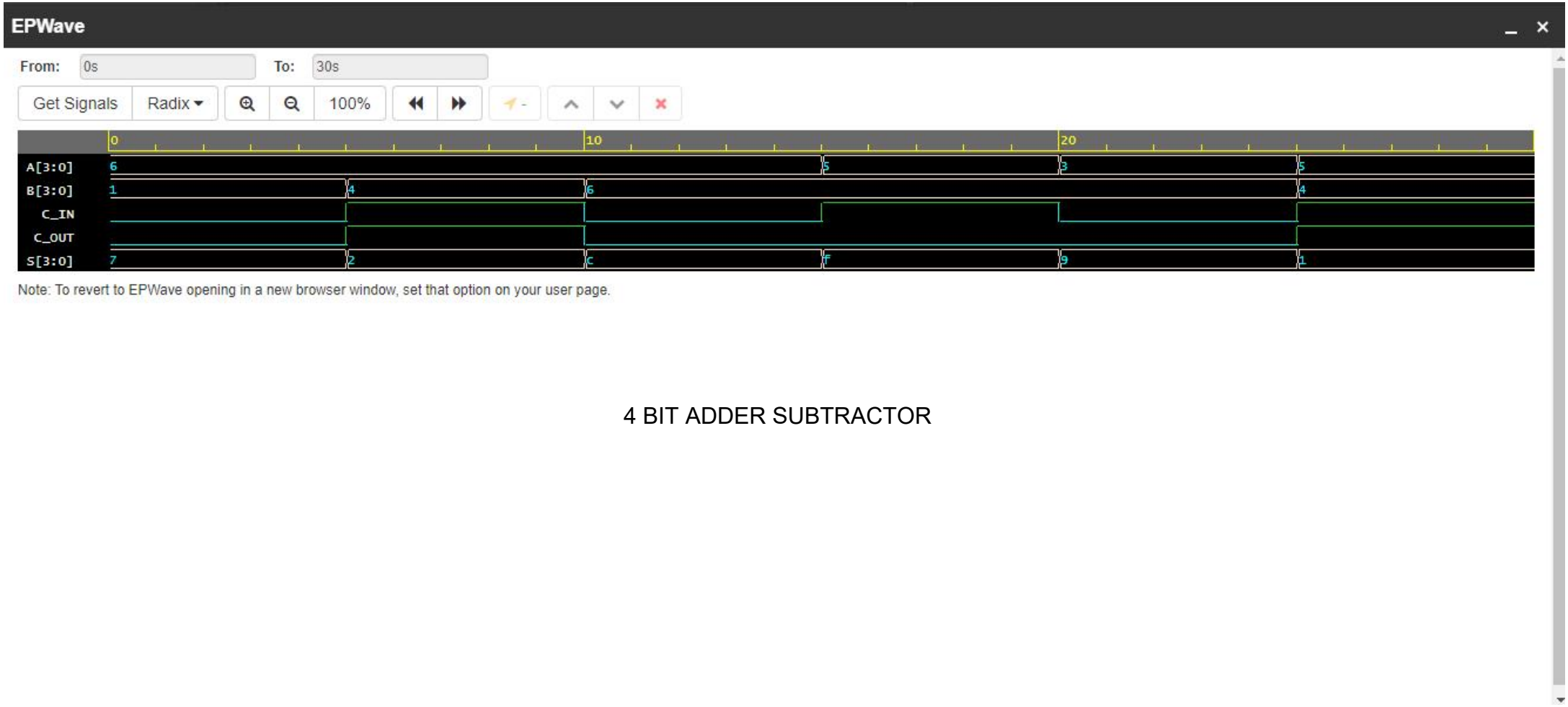
Justification of CO: In this experiment, we have created combinational circuits like adders, subtractors etc. So, CO2 is justified.


EPWave
From:
0s
To:
20s
Get Signals
Radix
100%
5s
0
10
A
0
B
1
S
1
C_OUT
0
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

HALF ADDER


EPWave
From:
0s
To:
40s
Get Signals
Radix
100%
0
10
20
30
A[2:0]
0
1
10
11
100
101
110
111
S
C_OUT
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

FULL ADDER



4 BIT ADDER SUBTRACTOR