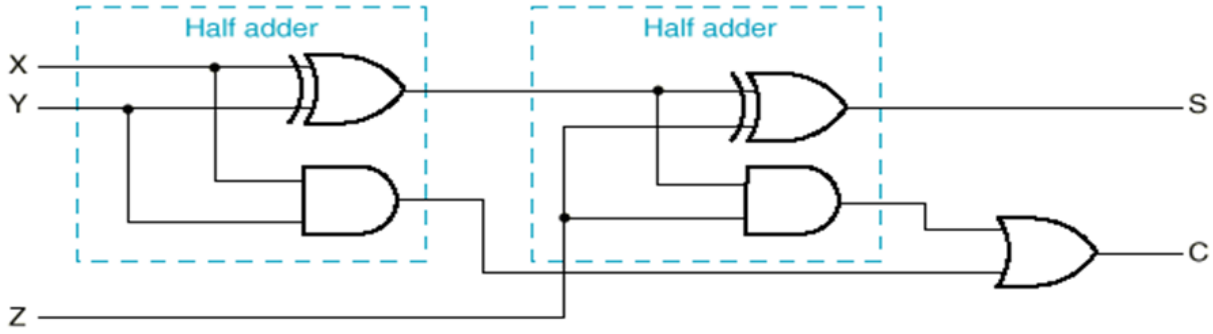
گزارش کار شماره 7 آزمایشگاه منطقی

سروین نامی 9931103
آبتین زندی 9931071

آزمایش اول: در طراحی fullAdder مطابق کد زیر ما دور خروجی Sum و Cout که برای حاصل جمع دو بایت ورودی ما هستند را داریم که برای جمع کردن از دو گیت xor و and برای نمایش حاصل جمع و بیت ناقل استفاده می کنیم مطابق مدار زیر :

fullAdder circuit:



در کد زیر برای گیت های and و 5 or نانو ثانیه و برای گیت xor 10 نانو ثانیه تاخیر در نظر گرفتیم:

fullAdder verilog code :

```
21  module fullAdder(input A,input B,input Cin,output Sum,output Cout);
22
23      wire tempWire;
24      wire tempWireAnd1;
25      wire tempWireAnd2;
26
27      xor #(10) (tempWire,A,B);
28      xor #(10) (Sum,tempWire,Cin);
29      and #(5)  (tempWireAnd1,tempWire,Cin);
30      and #(5)  (tempWireAnd2,A,B);
31      or  #(5)  (Cout,tempWireAnd1,tempWireAnd2);
32
33  endmodule
```

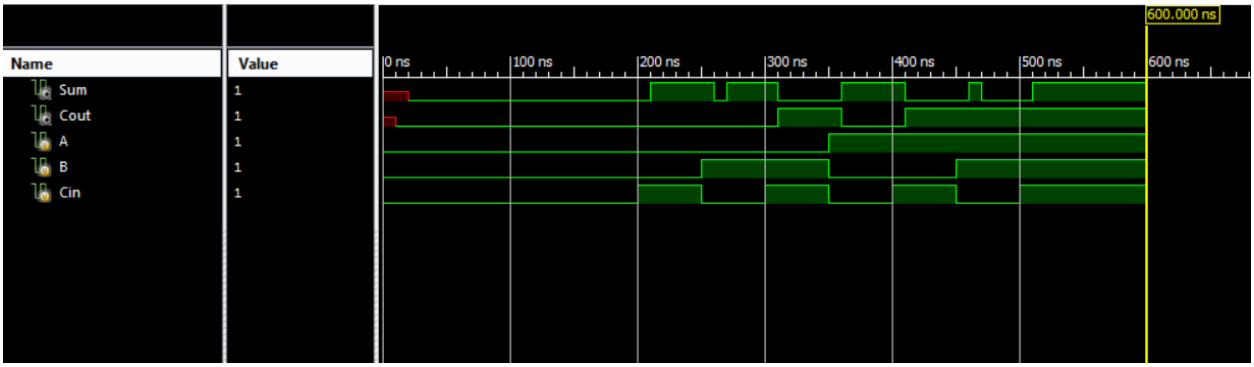
fullAdder Truth Table:

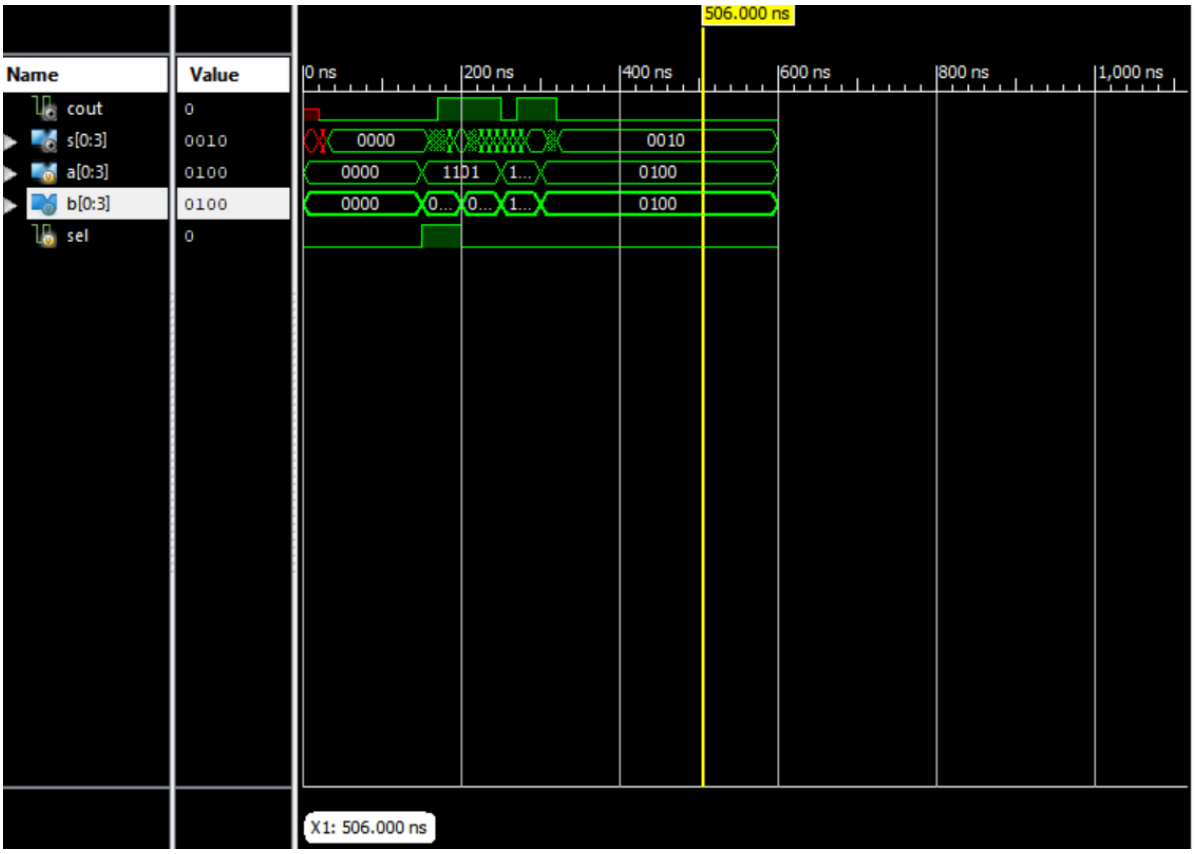
Full Adder Truth Table				
A	B	Cin	Cout	Sum
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

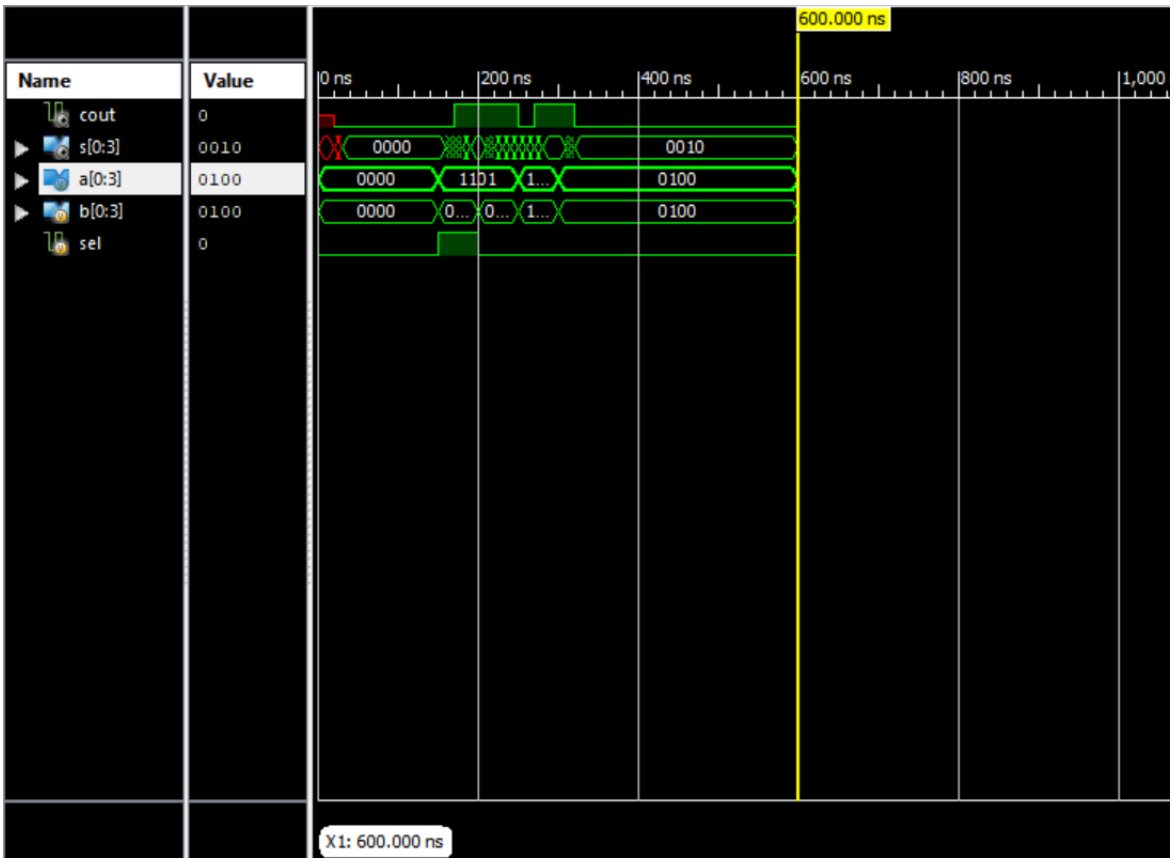
fullAdder Verilog code TestBench:

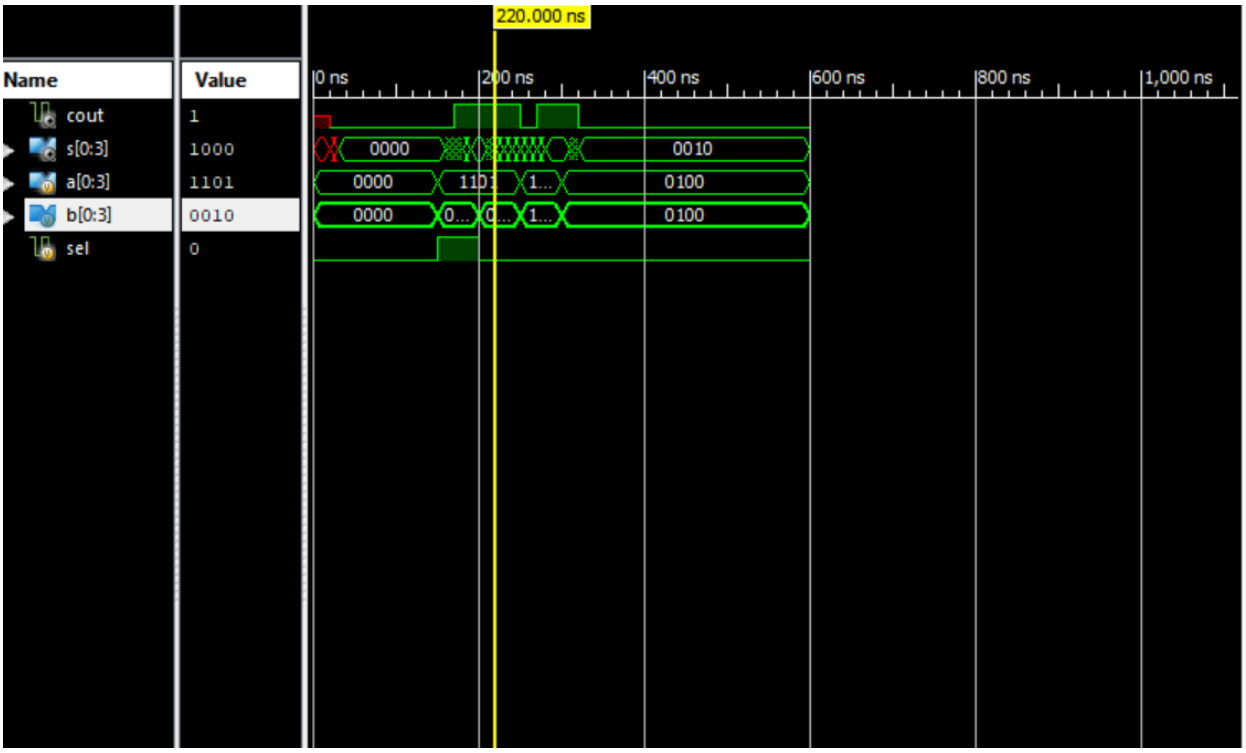
```
#100;

#50 A=0; B = 0; Cin = 0;
#50 A=0; B = 0; Cin = 1;
#50 A=0; B = 1; Cin = 0;
#50 A=0; B = 1; Cin = 1;
#50 A=1; B = 0; Cin = 0;
#50 A=1; B = 0; Cin = 1;
#50 A=1; B = 1; Cin = 0;
#50 A=1; B = 1; Cin = 1;
#100;
$finish;
```

fullAdder Verilog code TimeLine:


600.000 ns
Name
Value
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
Sum
1
Cout
1
A
1
B
1
Cin
1


506.000 ns
Name
Value
cout
0
s[0:3]
0010
a[0:3]
0100
b[0:3]
0100
sel
0
0 ns
200 ns
400 ns
600 ns
800 ns
1,000 ns
0000
0010
0000
1101
1...
0100
0000
0...
0...
1...
0100
X1: 506.000 ns


600.000 ns
Name
Value
cout
0
s[0:3]
0010
a[0:3]
0100
b[0:3]
0100
sel
0
0 ns
200 ns
400 ns
600 ns
800 ns
1,000
0000
0010
0000
1101
1...
0100
0000
0...
0...
1...
0100
X1: 600.000 ns




```
module adderSubtractor(
    input a,
    input b,
    input s,
    input cin,
    output cout,
    output sum
    );

   wire e;
    xor #(10) (e, s, b);
    fullAdder f(a, e, cin, cout, sum);

endmodule
```

```
module adderSubtractor4bit(
    input [0:3] a,
    input [0:3] b,
    input sel,
    output cout,
    output [0:3] s
    );

    wire c1, c2, c3;

    adderSubtractor a1(.a(a[0]), .b(b[0]), .s(sel), .cin(sel), .sum(s[0]), .cout(c1));
    adderSubtractor a2(.a(a[1]), .b(b[1]), .s(sel), .cin(c1), .sum(s[1]), .cout(c2));
    adderSubtractor a3(.a(a[2]), .b(b[2]), .s(sel), .cin(c2), .sum(s[2]), .cout(c3));
    adderSubtractor a4(.a(a[3]), .b(b[3]), .s(sel), .cin(c3), .sum(s[3]), .cout(cout));

endmodule
```

```
        .cout(cout),
        .s(s)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        sel = 0;

        // Wait 100 ns for global reset to finish
        #100;
        #50 a = 1101;b = 0100;sel = 1;
        #50 a = 1101;b = 1010;sel = 0;
        #50 a = 0011;b = 0011;sel = 0;
        #50 a = 0100;b = 0100;sel = 0;
        // Add stimulus here
        #300;
        $finish;

    end

endmodule
```

```
module fullAdder(
    input x,
    input y,
    input cin,
    output cout,
    output s
    );

    wire e1, e2, e3;

   xor #(10) (s, x, y, cin);

   and #(5) (e1, x, y);
   and #(5) (e2, x, cin);
   and #(5) (e3, y, cin);
   or #(5) (cout, e1, e2, e3);

endmodule
```