LAB – 2

Exercise:

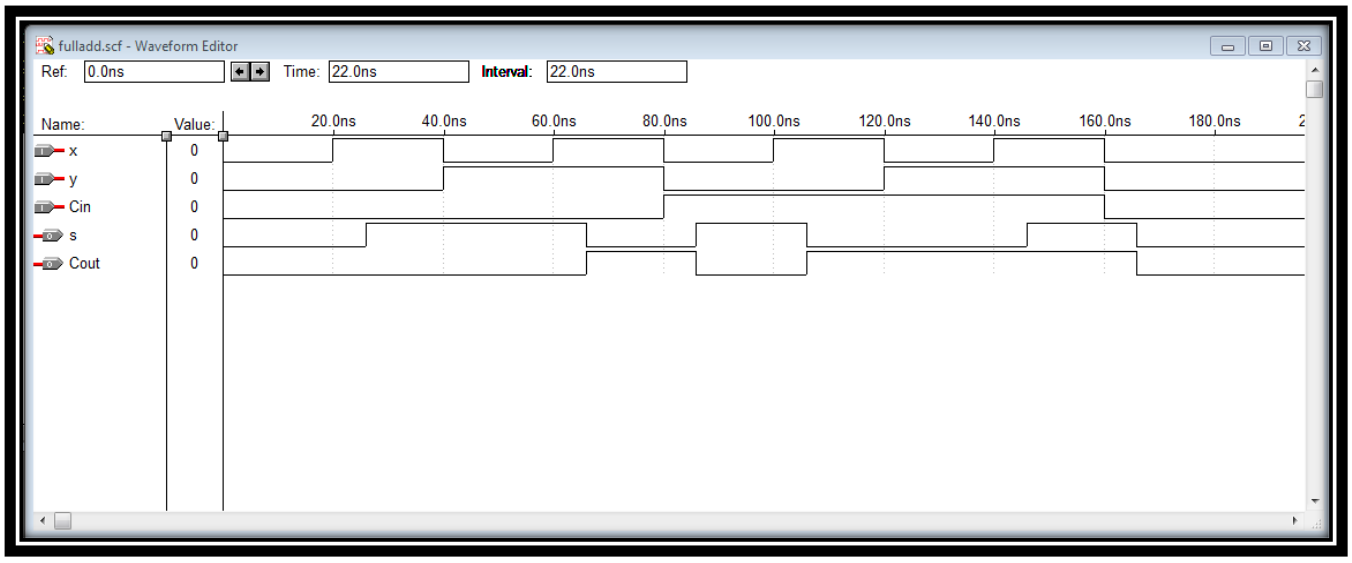
Write behavioral Verilog code to implement the following and simulate

1. Full adder

Verilog code:

```
module fulladd(Cin,x,y,s,Cout);
input Cin,x,y;
output Cout,s;
assign s=x^y^Cin;
assign Cout=(x&y)|(x&Cin)|(y&Cin);
endmodule
```

Output:



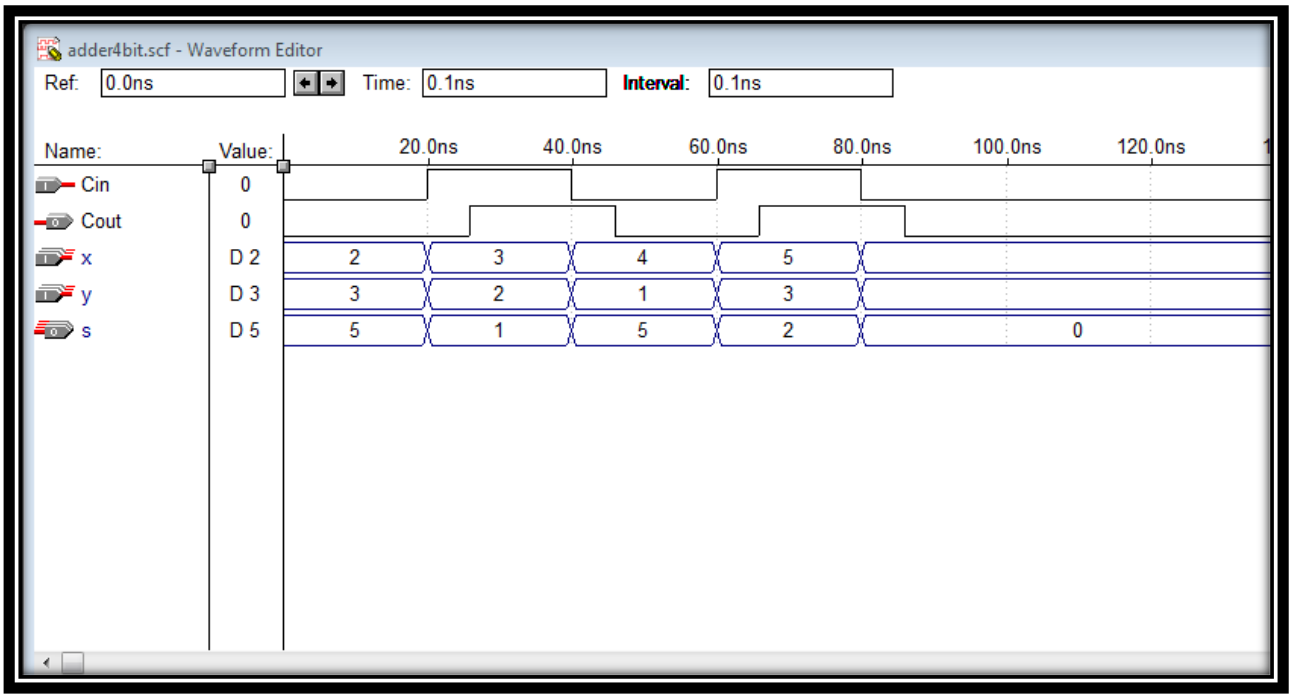
2. Four-bit adder/ subtractor

Verilog code:

```
module fulladd(Cin,x,y,s,Cout);
input Cin,x,y;
output Cout,s;
assign s=x^y^Cin;
assign Cout=(x&y)|(x&Cin)|(y&Cin);
endmodule

module adder4bit(Cin,x,y,s,Cout);
input Cin;
input [3:0]x,y;
output [3:0]s;
output Cout;
wire [3:1]c;
fulladd stage0(Cin,x[0],y[0]^Cin,s[0],c[1]);
fulladd stage1(c[1],x[1],y[1]^Cin,s[1],c[2]);
fulladd stage2(c[2],x[2],y[2]^Cin,s[2],c[3]);
fulladd stage3(c[3],x[3],y[3]^Cin,s[3],Cout);
endmodule
```

Output:

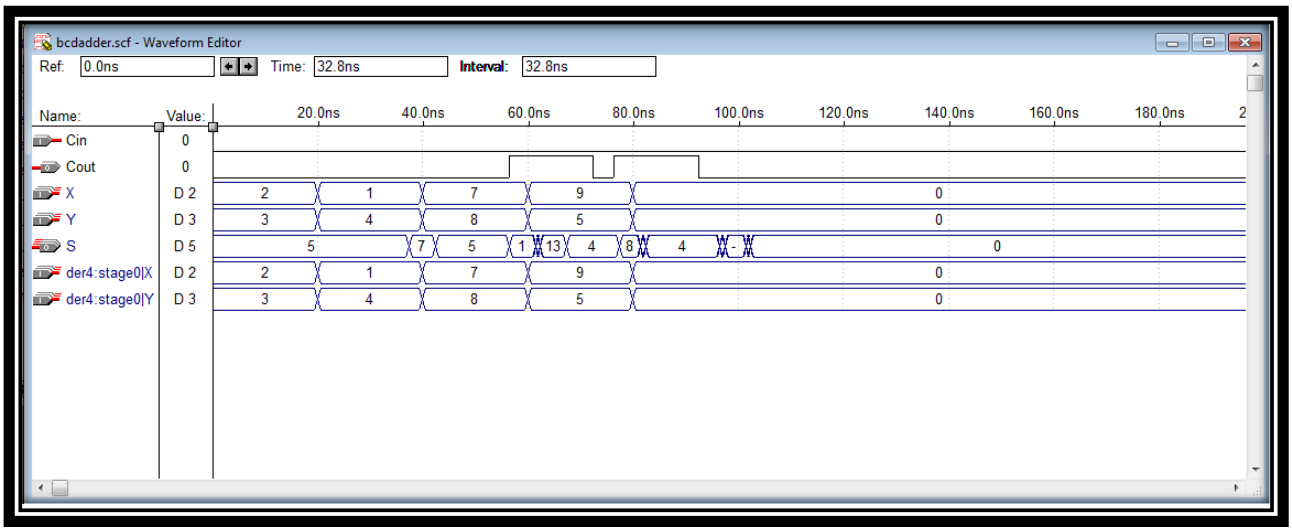


3. Single-digit BCD adder using a four-bit adder(s).

Verilog code:

```
module fulladd(Cin, x, y, s, Cout);
input Cin, x, y;
output s, Cout;
assign s = x ^ y ^ Cin;
assign Cout = (x & y) | (x & Cin) | (y & Cin);
endmodule
module adder4(Cin,X,Y,S,Cout);
input Cin;
input [3:0]X,Y;
output [3:0]S;
output Cout;
wire [3:1]C;
fulladd stage0(Cin,X[0],Y[0]^Cin,S[0],C[1]);
fulladd stage1(C[1],X[1],Y[1]^Cin,S[1],C[2]);
fulladd stage2(C[2],X[2],Y[2]^Cin,S[2],C[3]);
fulladd stage3(C[3],X[3],Y[3]^Cin,S[3],Cout);
endmodule
module bcdadder(Cin,X,Y,S,Cout);
input Cin;
input [3:0]X,Y;
output [3:0]S;
output Cout;
wire m,a,b,c;
wire [3:0]Z;
adder4 stage0(Cin,X,Y,Z,m);
assign a=Z[3]&Z[1];
assign b=Z[3]&Z[2];
assign Cout=a|b|m;
adder4 stage1(Cin,{1'b0,Cout,Cout,1'b0},Z,S,c);
endmodule
```

Output:


bcdadder.scf - Waveform Editor
Ref: 0.0ns
Time: 32.8ns
Interval: 32.8ns
Name:
Value:
20.0ns
40.0ns
60.0ns
80.0ns
100.0ns
120.0ns
140.0ns
160.0ns
180.0ns
Cin
Cout
X
Y
S
der4:stage0|X
der4:stage0|Y

****