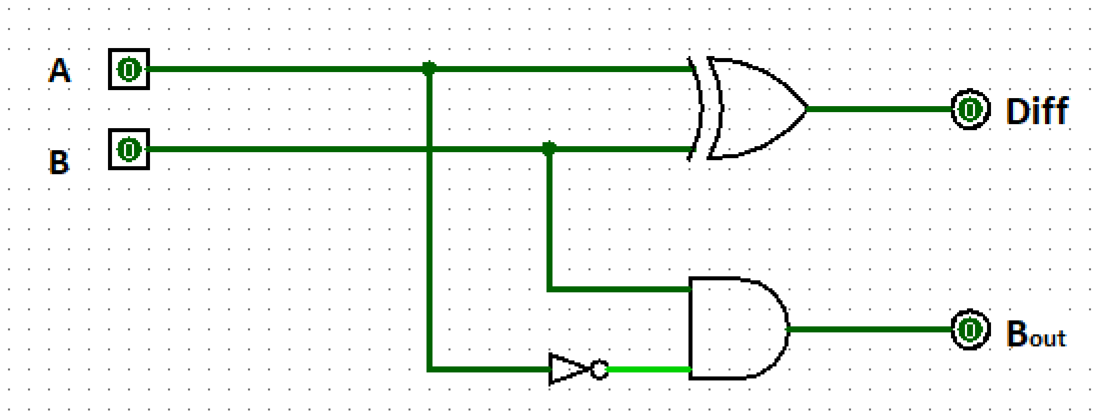
Date: 16-09-2022

Assignment-5

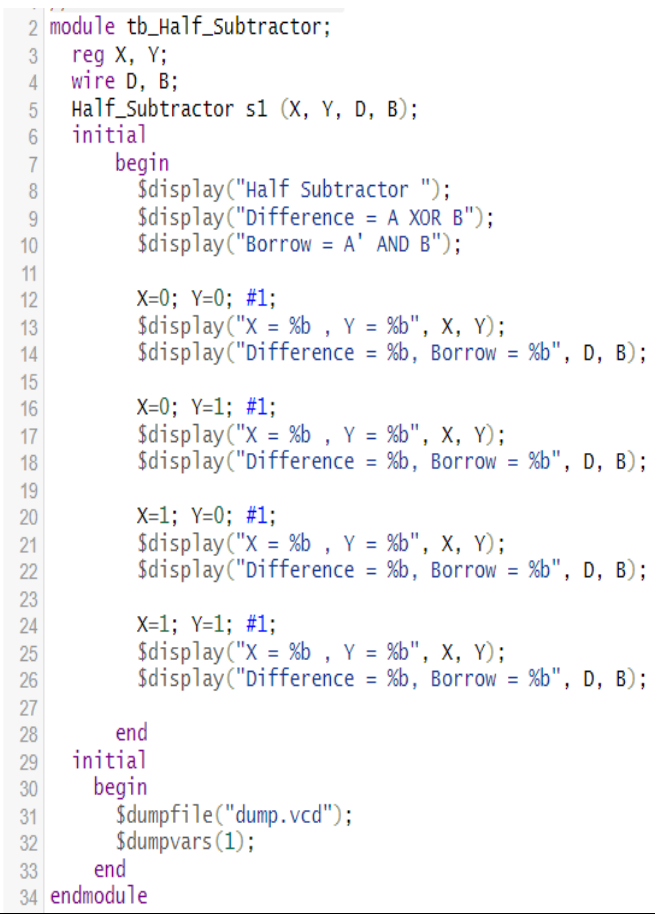
Aim: Implementation of Subtractor circuits

Question-1: Write a Verilog code to design a Half subtractor and test it using the Waveform.



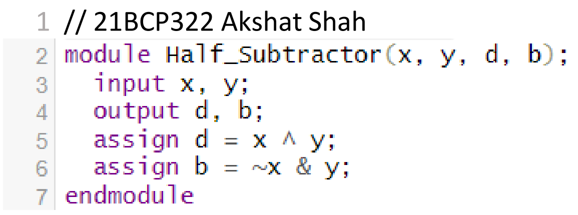
1.1 Circuit diagram of half subtractor

Test bench Code



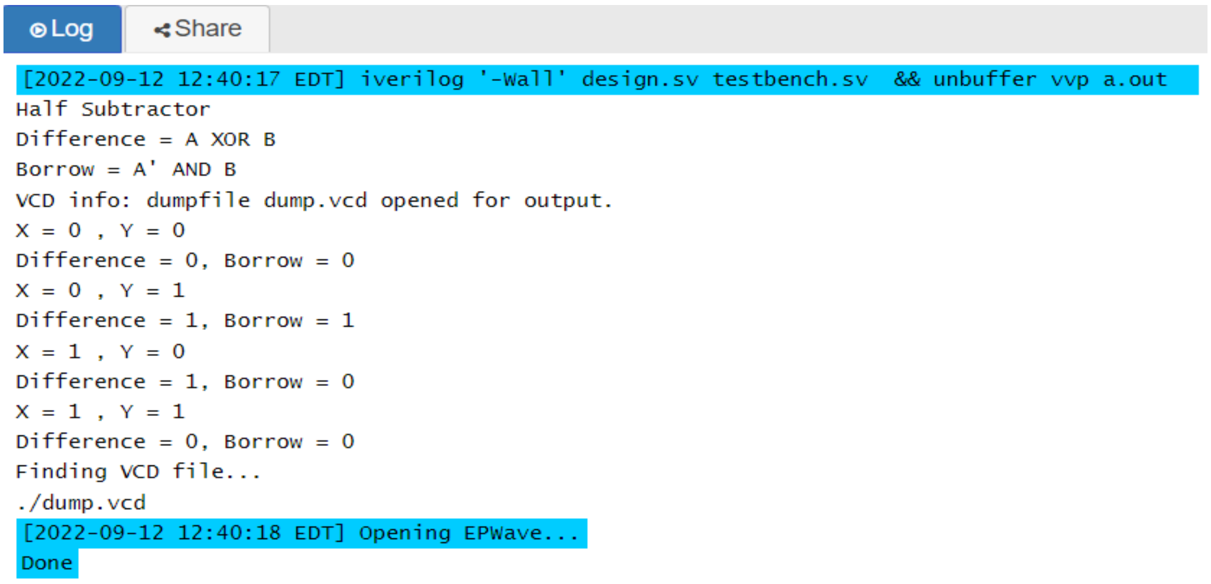
```verilog
module tb_Half_Subtractor;
  reg X, Y;
  wire D, B;
  Half_Subtractor s1 (X, Y, D, B);
  initial
      begin
        $display("Half Subtractor ");
        $display("Difference = A XOR B");
        $display("Borrow = A' AND B");

        X=0; Y=0; #1;
        $display("X = %b , Y = %b", X, Y);
        $display("Difference = %b, Borrow = %b", D, B);

        X=0; Y=1; #1;
        $display("X = %b , Y = %b", X, Y);
        $display("Difference = %b, Borrow = %b", D, B);

        X=1; Y=0; #1;
        $display("X = %b , Y = %b", X, Y);
        $display("Difference = %b, Borrow = %b", D, B);

        X=1; Y=1; #1;
        $display("X = %b , Y = %b", X, Y);
        $display("Difference = %b, Borrow = %b", D, B);

      end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

Design Code



```verilog
// 21BCP322 Akshat Shah
module Half_Subtractor(x, y, d, b);
  input x, y;
  output d, b;
  assign d = x ^ y;
  assign b = ~x & y;
endmodule
```

Output:



```
[2022-09-12 12:40:17 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
Half Subtractor
Difference = A XOR B
Borrow = A' AND B
VCD info: dumpfile dump.vcd opened for output.
X = 0 , Y = 0
Difference = 0, Borrow = 0
X = 0 , Y = 1
Difference = 1, Borrow = 1
X = 1 , Y = 0
Difference = 1, Borrow = 0
X = 1 , Y = 1
Difference = 0, Borrow = 0
Finding VCD file...
./dump.vcd
[2022-09-12 12:40:18 EDT] Opening EPWave...
Done
```

Figure 1.1: Output for Half Subtractor

Waveform:

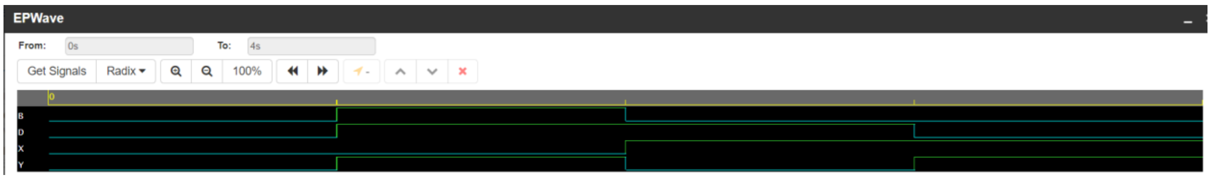
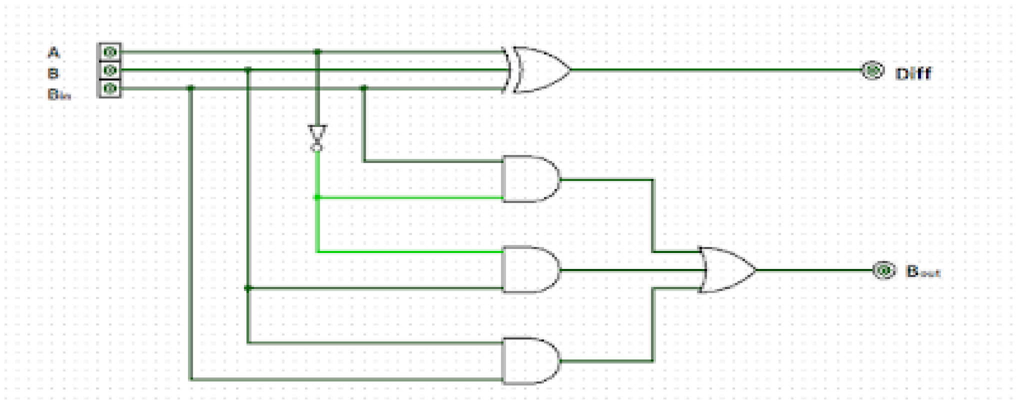


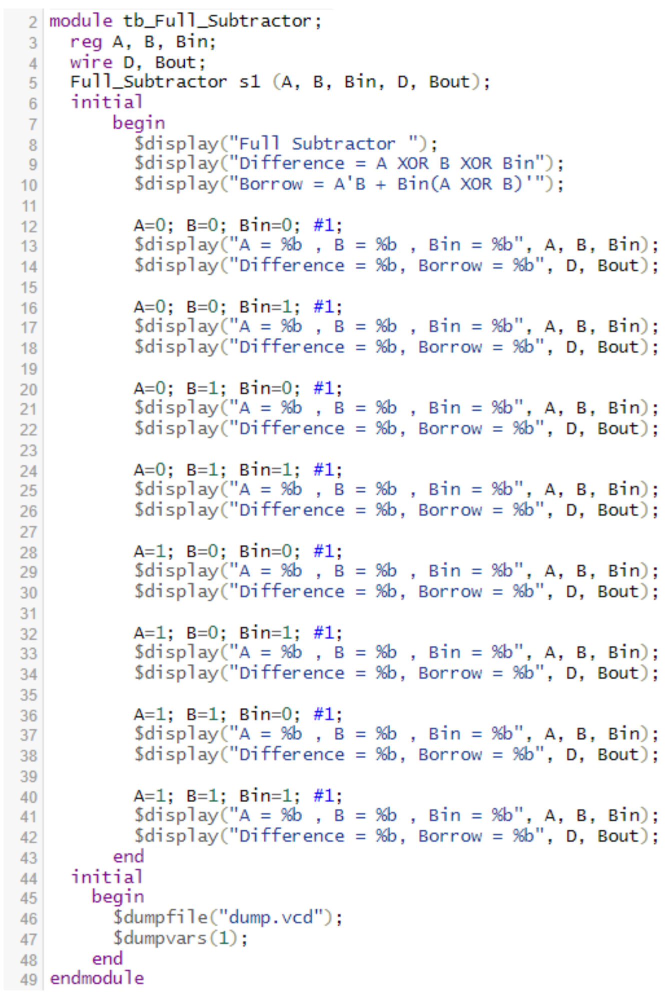
Fig EP Waveform for Half Subtractor

Question 2 :Write a Verilog code to design a Full subtractor using Half subtractors and test it using the Waveform.



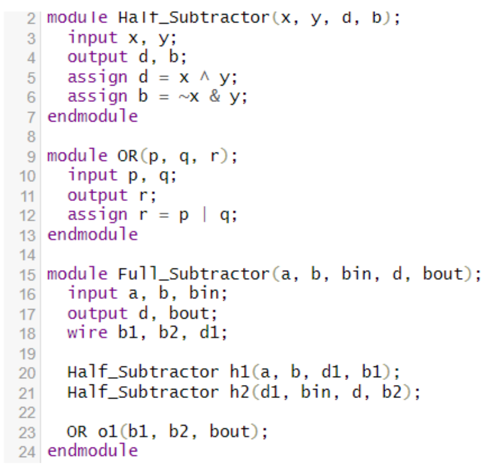
Circuit diagram of full subtractor

Test bench Code



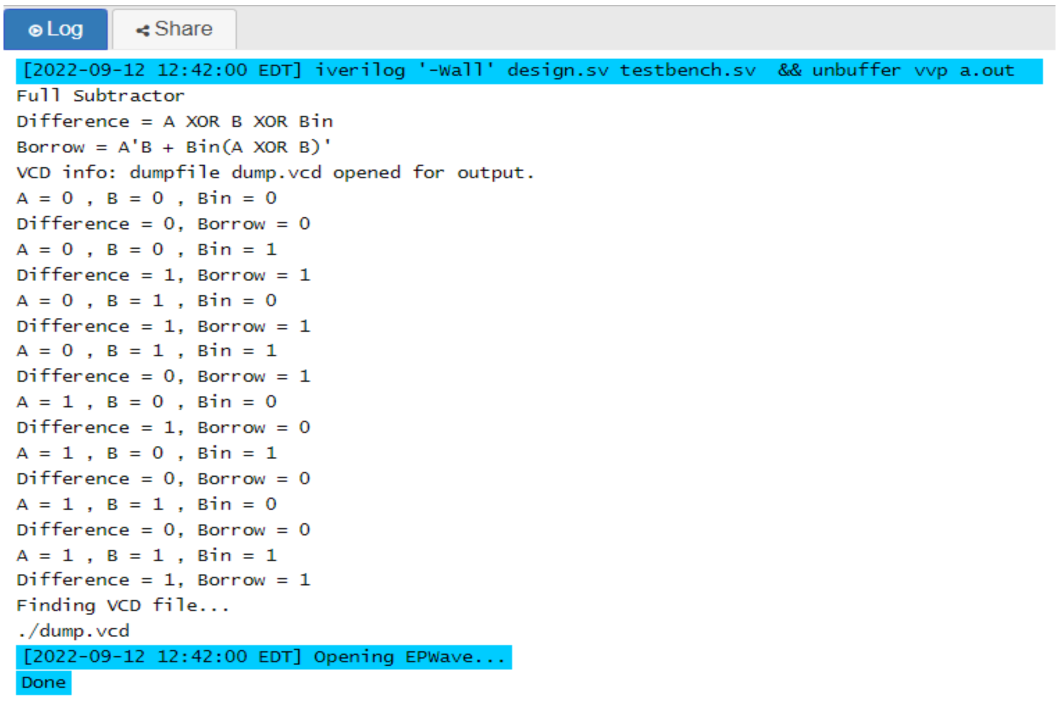
```
module tb_Full_Subtractor;
  reg A, B, Bin;
  wire D, Bout;
  Full_Subtractor s1 (A, B, Bin, D, Bout);
  initial
      begin
        $display("Full Subtractor ");
        $display("Difference = A XOR B XOR Bin");
        $display("Borrow = A'B + Bin(A XOR B)'");

        A=0; B=0; Bin=0; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);

        A=0; B=0; Bin=1; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);

        A=0; B=1; Bin=0; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);

        A=0; B=1; Bin=1; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);

        A=1; B=0; Bin=0; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);

        A=1; B=0; Bin=1; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);

        A=1; B=1; Bin=0; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);

        A=1; B=1; Bin=1; #1;
        $display("A = %b , B = %b , Bin = %b", A, B, Bin);
        $display("Difference = %b, Borrow = %b", D, Bout);
      end
  initial
     begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
     end
endmodule
```

Design Code



```
module Half_Subtractor(x, y, d, b);
  input x, y;
  output d, b;
  assign d = x ^ y;
  assign b = ~x & y;
endmodule

module OR(p, q, r);
  input p, q;
  output r;
  assign r = p | q;
endmodule

module Full_Subtractor(a, b, bin, d, bout);
  input a, b, bin;
  output d, bout;
  wire b1, b2, d1;

  Half_Subtractor h1(a, b, d1, b1);
  Half_Subtractor h2(d1, bin, d, b2);

  OR o1(b1, b2, bout);
endmodule
```

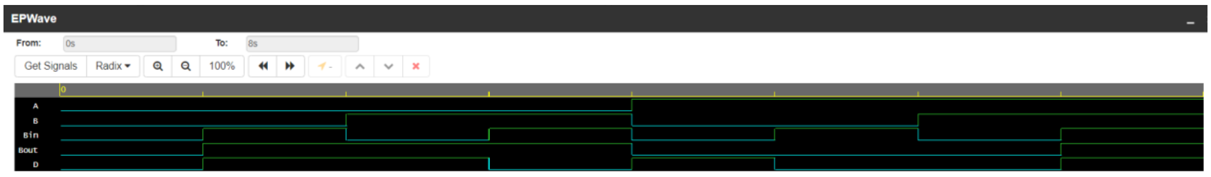
Output:



```
[2022-09-12 12:42:00 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
Full Subtractor
Difference = A XOR B XOR Bin
Borrow = A'B + Bin(A XOR B)'
VCD info: dumpfile dump.vcd opened for output.
A = 0 , B = 0 , Bin = 0
Difference = 0, Borrow = 0
A = 0 , B = 0 , Bin = 1
Difference = 1, Borrow = 1
A = 0 , B = 1 , Bin = 0
Difference = 1, Borrow = 1
A = 0 , B = 1 , Bin = 1
Difference = 0, Borrow = 1
A = 1 , B = 0 , Bin = 0
Difference = 1, Borrow = 0
A = 1 , B = 0 , Bin = 1
Difference = 0, Borrow = 0
A = 1 , B = 1 , Bin = 0
Difference = 0, Borrow = 0
A = 1 , B = 1 , Bin = 1
Difference = 1, Borrow = 1
Finding VCD file...
./dump.vcd
[2022-09-12 12:42:00 EDT] Opening EPWave...
Done
```

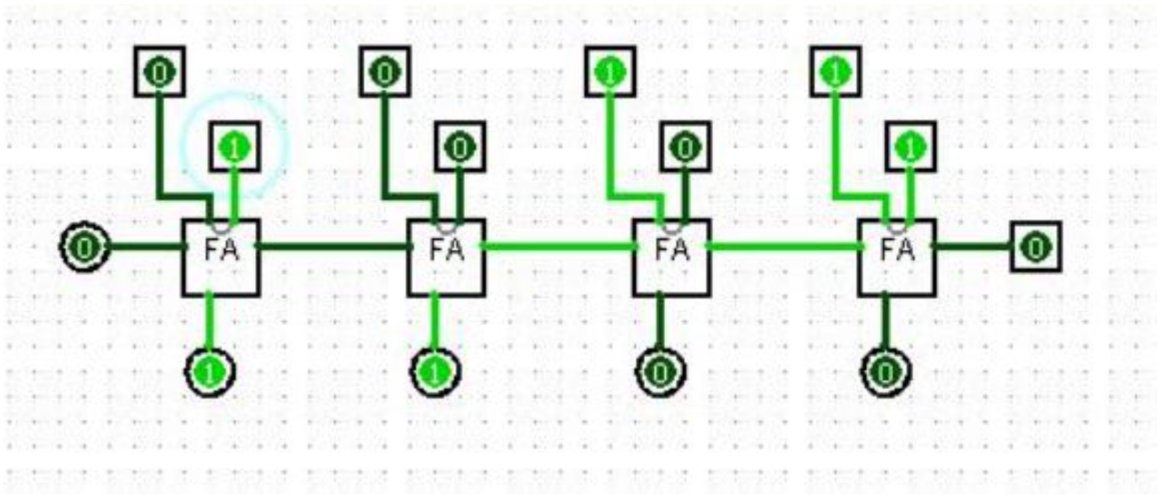
Output for Full Subtractor

Waveform:

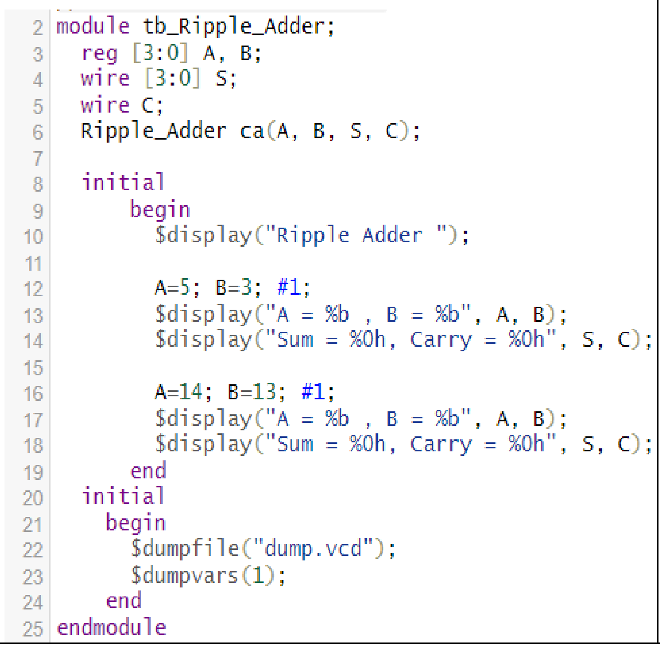
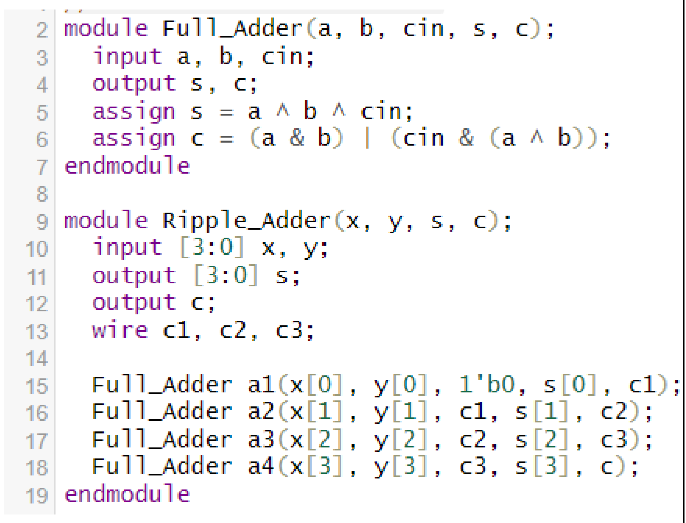


Waveform for Full Subtractor

Question 3: Write the Verilog code (either structural or behavioural) for a 4-bit ripple carry adder. The block diagram of a 4-bit ripple carry adder has been given in Figure 3.1 for your reference.



Circuit diagram of 4-bit ripple carry adder

Test bench Code	Design code
	

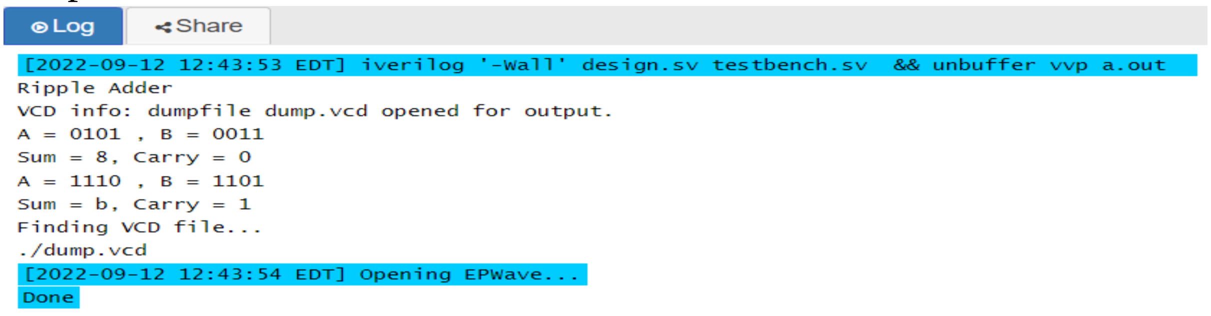
Test bench Code:

```
module tb_Ripple_Adder;
  reg [3:0] A, B;
  wire [3:0] S;
  wire C;
  Ripple_Adder ca(A, B, S, C);

  initial
      begin
        $display("Ripple Adder ");

        A=5; B=3; #1;
        $display("A = %b , B = %b", A, B);
        $display("Sum = %0h, Carry = %0h", S, C);

        A=14; B=13; #1;
        $display("A = %b , B = %b", A, B);
        $display("Sum = %0h, Carry = %0h", S, C);
      end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

Design code:

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

module Ripple_Adder(x, y, s, c);
  input [3:0] x, y;
  output [3:0] s;
  output c;
  wire c1, c2, c3;

  Full_Adder a1(x[0], y[0], 1'b0, s[0], c1);
  Full_Adder a2(x[1], y[1], c1, s[1], c2);
  Full_Adder a3(x[2], y[2], c2, s[2], c3);
  Full_Adder a4(x[3], y[3], c3, s[3], c);
endmodule
```

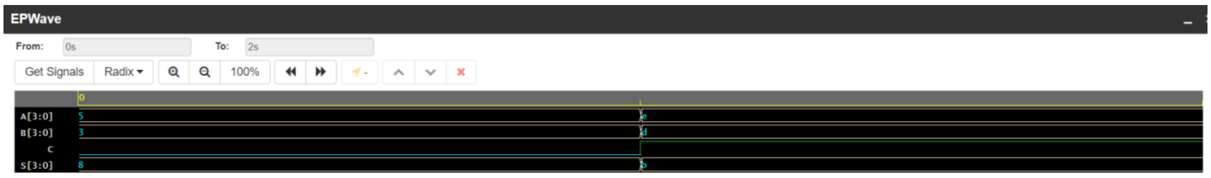
Output:



```
[2022-09-12 12:43:53 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
Ripple Adder
VCD info: dumpfile dump.vcd opened for output.
A = 0101 , B = 0011
Sum = 8, Carry = 0
A = 1110 , B = 1101
Sum = b, Carry = 1
Finding VCD file...
./dump.vcd
[2022-09-12 12:43:54 EDT] Opening EPWave...
Done
```

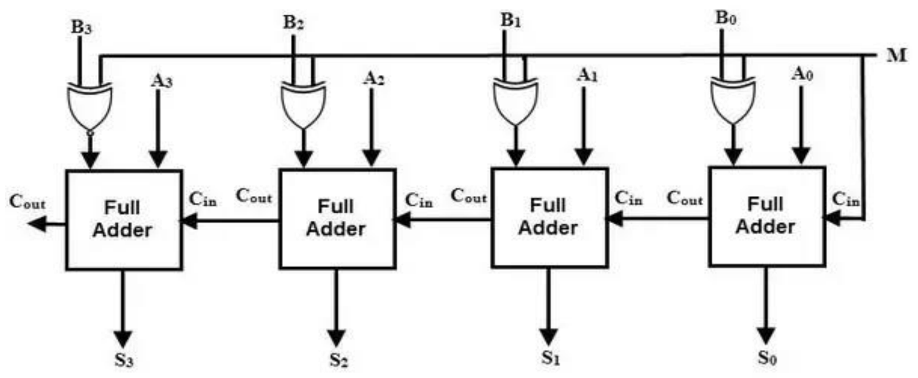
: Output for four-bit Ripple Carry Adder

Waveform:



Waveform for four-bit Ripple Carry Adder

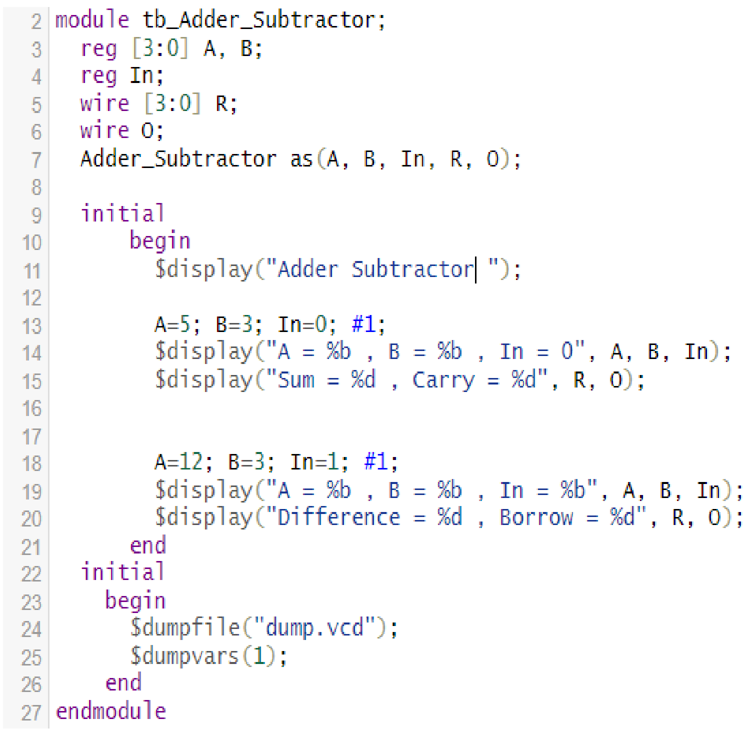
Question 4: Write the Verilog code to construct a 4-bit adder-subtractor using. The logic diagram of a 4-bit adder-subtractor using a ripple carry adder has been given in Figure 4.1 for your reference.



Circuit diagram of 4-bit adder-subtractor

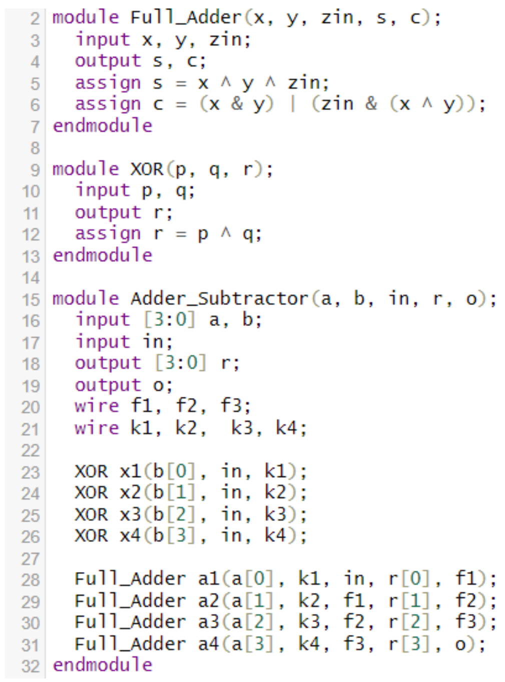
Test Bench code	Design Code
see below	see below

Test Bench code



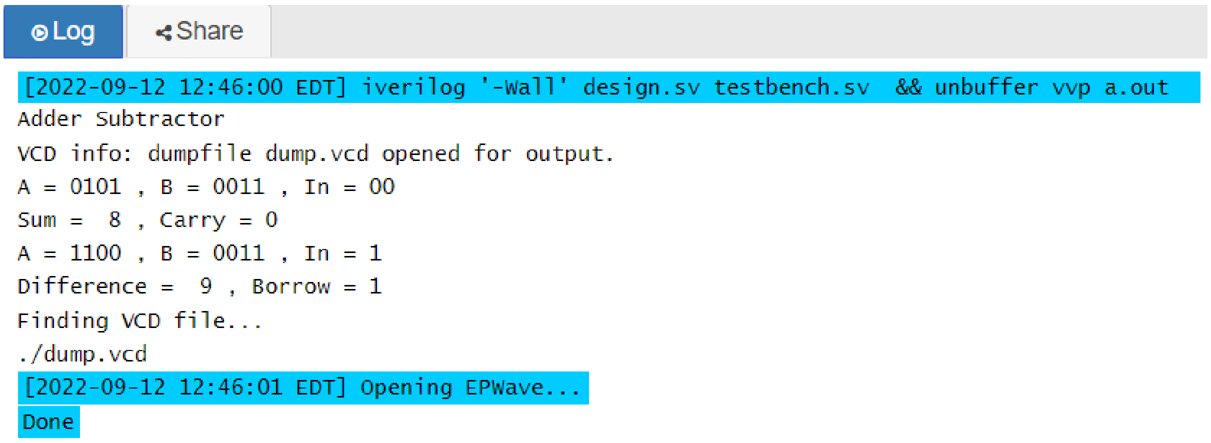
```verilog
module tb_Adder_Subtractor;
  reg [3:0] A, B;
  reg In;
  wire [3:0] R;
  wire O;
  Adder_Subtractor as(A, B, In, R, O);

  initial
      begin
        $display("Adder Subtractor ");

        A=5; B=3; In=0; #1;
        $display("A = %b , B = %b , In = 0", A, B, In);
        $display("Sum = %d , Carry = %d", R, O);


        A=12; B=3; In=1; #1;
        $display("A = %b , B = %b , In = %b", A, B, In);
        $display("Difference = %d , Borrow = %d", R, O);
      end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

Design Code



```verilog
module Full_Adder(x, y, zin, s, c);
  input x, y, zin;
  output s, c;
  assign s = x ^ y ^ zin;
  assign c = (x & y) | (zin & (x ^ y));
endmodule

module XOR(p, q, r);
  input p, q;
  output r;
  assign r = p ^ q;
endmodule

module Adder_Subtractor(a, b, in, r, o);
  input [3:0] a, b;
  input in;
  output [3:0] r;
  output o;
  wire f1, f2, f3;
  wire k1, k2,  k3, k4;

  XOR x1(b[0], in, k1);
  XOR x2(b[1], in, k2);
  XOR x3(b[2], in, k3);
  XOR x4(b[3], in, k4);

  Full_Adder a1(a[0], k1, in, r[0], f1);
  Full_Adder a2(a[1], k2, f1, r[1], f2);
  Full_Adder a3(a[2], k3, f2, r[2], f3);
  Full_Adder a4(a[3], k4, f3, r[3], o);
endmodule
```

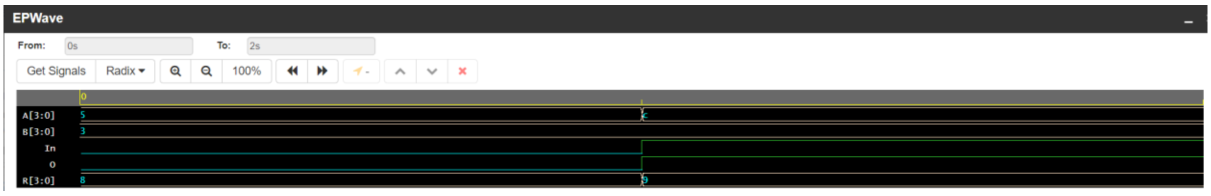
Output:



```
[2022-09-12 12:46:00 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
Adder Subtractor
VCD info: dumpfile dump.vcd opened for output.
A = 0101 , B = 0011 , In = 00
Sum =  8 , Carry = 0
A = 1100 , B = 0011 , In = 1
Difference =  9 , Borrow = 1
Finding VCD file...
./dump.vcd
[2022-09-12 12:46:01 EDT] Opening EPWave...
Done
```

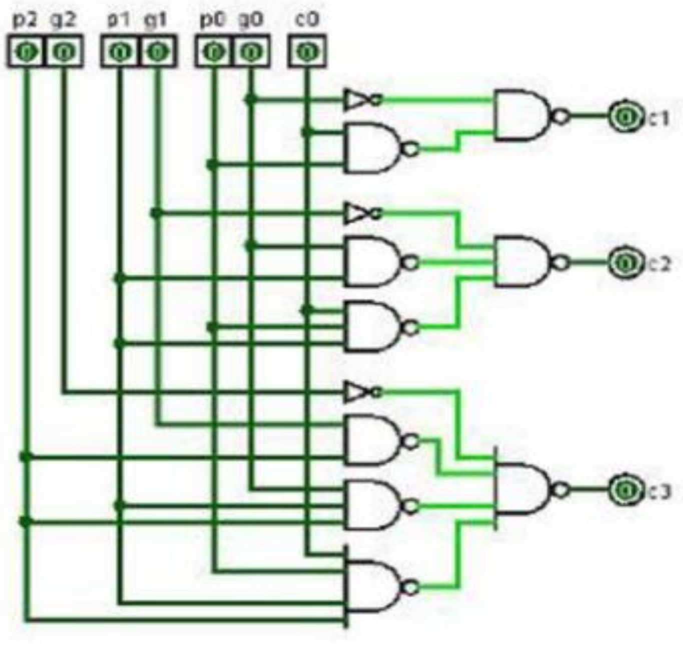
Output for four-bit Adder-cum-Subtractor

Waveform:

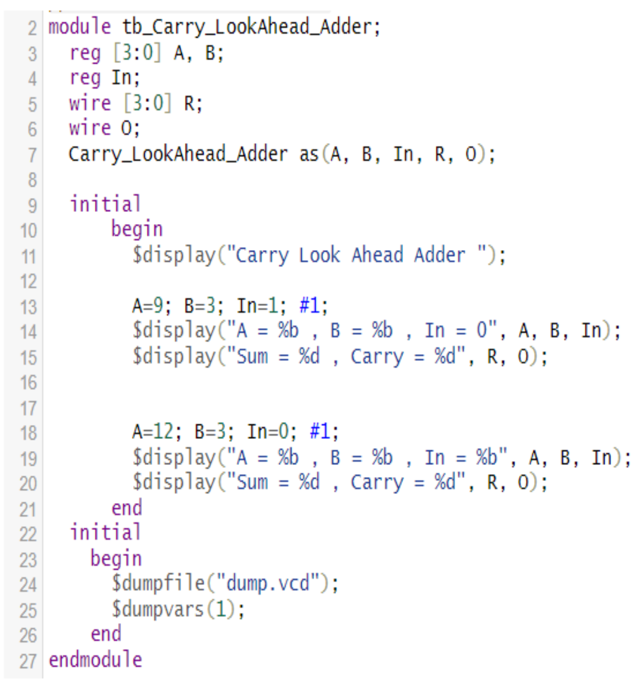
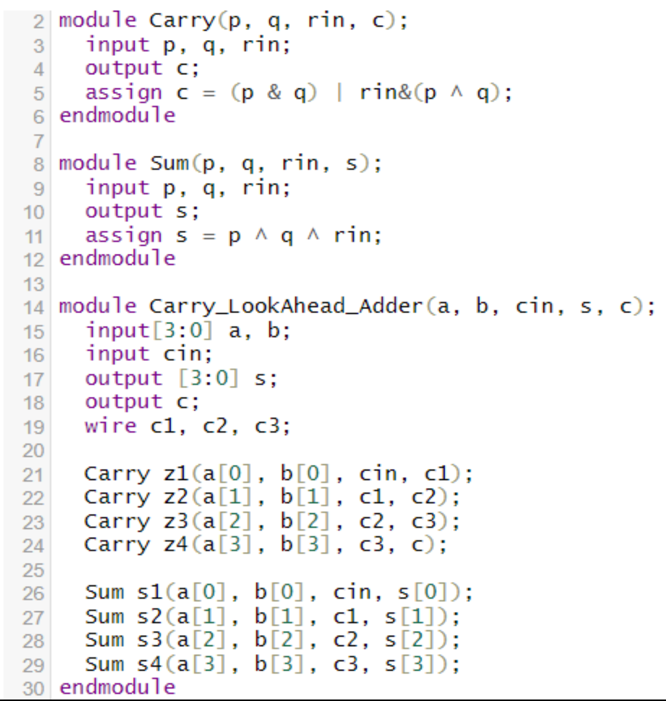


Waveform for four-bit Adder-cum-Subtractor

Question 5:Write the Verilog code for a 4-bit carry look-ahead adder (CLA). The block diagram of the same has been given in Figure 3.



Circuit diagram of 4 bit CLA

Test bench code	Design Code
	

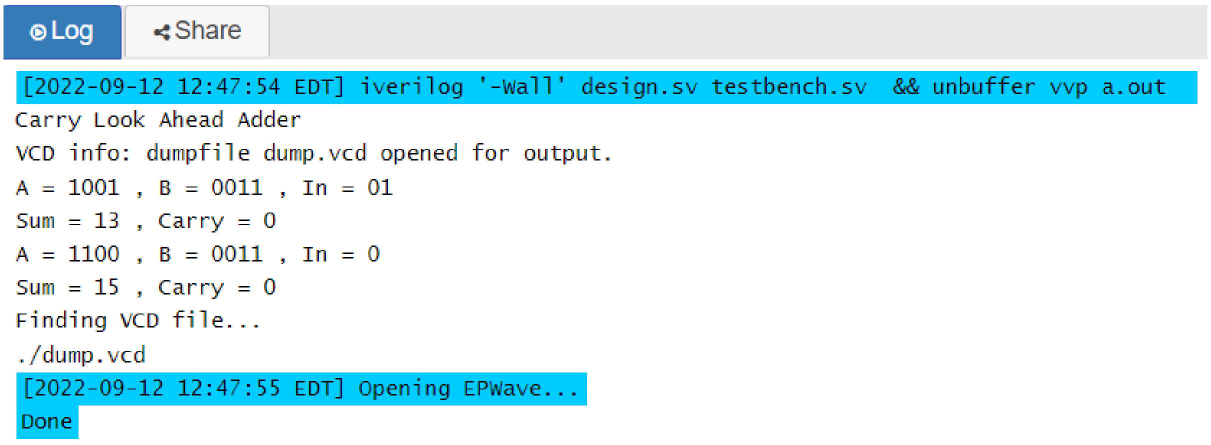
Test bench code

```verilog
module tb_Carry_LookAhead_Adder;
  reg [3:0] A, B;
  reg In;
  wire [3:0] R;
  wire O;
  Carry_LookAhead_Adder as(A, B, In, R, O);

  initial
      begin
        $display("Carry Look Ahead Adder ");

        A=9; B=3; In=1; #1;
        $display("A = %b , B = %b , In = 0", A, B, In);
        $display("Sum = %d , Carry = %d", R, O);


        A=12; B=3; In=0; #1;
        $display("A = %b , B = %b , In = %b", A, B, In);
        $display("Sum = %d , Carry = %d", R, O);
      end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

Design Code

```verilog
module Carry(p, q, rin, c);
  input p, q, rin;
  output c;
  assign c = (p & q) | rin&(p ^ q);
endmodule

module Sum(p, q, rin, s);
  input p, q, rin;
  output s;
  assign s = p ^ q ^ rin;
endmodule

module Carry_LookAhead_Adder(a, b, cin, s, c);
  input[3:0] a, b;
  input cin;
  output [3:0] s;
  output c;
  wire c1, c2, c3;

  Carry z1(a[0], b[0], cin, c1);
  Carry z2(a[1], b[1], c1, c2);
  Carry z3(a[2], b[2], c2, c3);
  Carry z4(a[3], b[3], c3, c);

  Sum s1(a[0], b[0], cin, s[0]);
  Sum s2(a[1], b[1], c1, s[1]);
  Sum s3(a[2], b[2], c2, s[2]);
  Sum s4(a[3], b[3], c3, s[3]);
endmodule
```

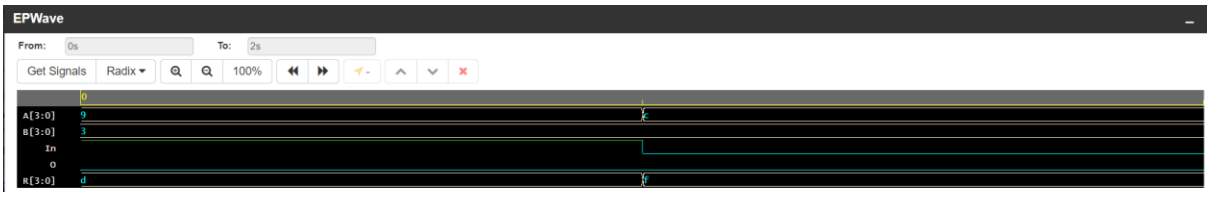
Output:



```
[2022-09-12 12:47:54 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
Carry Look Ahead Adder
VCD info: dumpfile dump.vcd opened for output.
A = 1001 , B = 0011 , In = 01
Sum = 13 , Carry = 0
A = 1100 , B = 0011 , In = 0
Sum = 15 , Carry = 0
Finding VCD file...
./dump.vcd
[2022-09-12 12:47:55 EDT] Opening EPWave...
Done
```

Output for four-bit Carry Look-Ahead Adder (CLA)

Waveform:



Waveform for four-bit Carry Look-Ahead Adder (CLA)