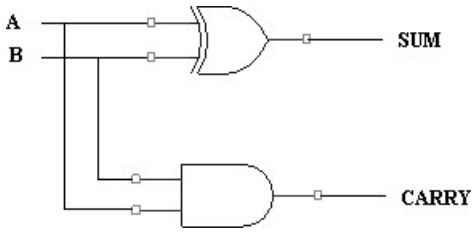
ADDERS AND SUBTRACTORS

AIM:
To develop the source code for adders and subtractors by using VERILOG and obtain the simulation.

ALGORITM:
Step1: Define the specifications and initialize the design.
Step2 Write the source code in VERILOG
Step3: Check the syntax and debug the errors if found, obtain the synthesis report.
Step4: Verify the output by simulating the source code

BASIC ADDERS & SUBTRACTORS:

HALF ADDER:
LOGIC DIAGRAM:



TRUTH TABLE:

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

VERILOG SOURCE CODE:

Dataflow Modeling:

```
module ha_dataflow(a, b, s, ca);
   input a;
   input b;
   output s;
   output ca;
          assign#2 s=a^b;
          assign#2 ca=a&b;
endmodule
```

Behavioral Modeling:

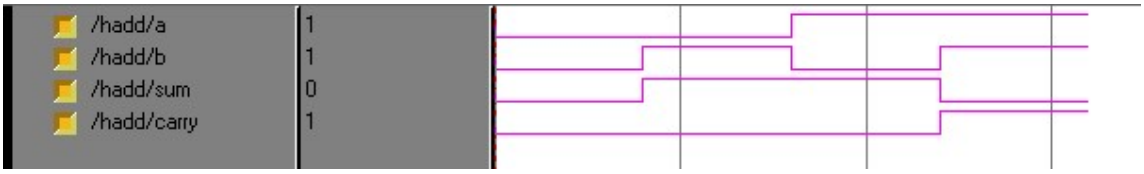
```
module ha_behv(a, b, s, ca);
   input a;
   input b;
   output s;
   output ca;
          reg s,ca;
          always @ (a or b) begin
          s=a^b;
          ca=a&b;
          end
```

```
endmodule
```

Structural Modeling:

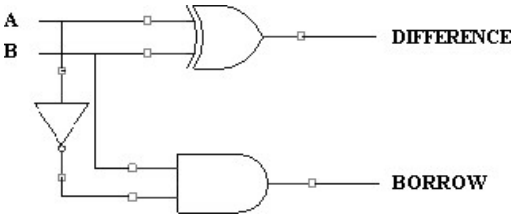
```
module ha_struct(a, b, s, ca);
   input a;
   input b;
   output s;
   output ca;
          xor
          x1(s,a,b);
          and
          a1(ca,a,b);
endmodule
```

Simulation output:



HALF SUBSTRACTOR:

LOGIC DIAGRAM:



TRUTH TABLE

A	B	DIFFERENCE	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

VERILOG SOURCE CODE:
Dataflow Modeling:

```
module hs_dataflow(a, b, dif, bor);
   input a;
   input b;
   output dif;
   output bor;
          wire abar;
          assign#3 abar=~a;
          assign#3 dif=a^b;
          assign#3 bor=b&abar;
endmodule
```

Behavioral Modeling:

```
module hs_behv(a, b, dif, bor);
```

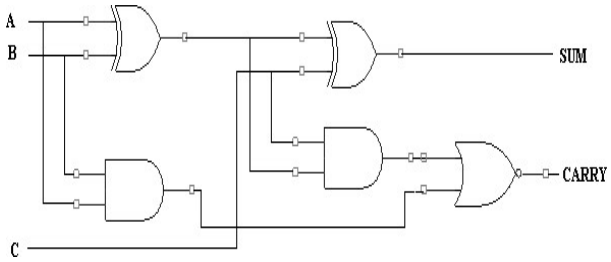
```
    input a;
    input b;
    output dif;
    output bor;
            reg dif,bor;
            reg abar;
            always@(a or b) begin
            abar=~a;
            dif=a^b;
            bor=b&abar;
            end
endmodule
```

Structural Modeling:

```
module hs_struct(a, b, dif, bor);
    input a;
    input b;
    output dif;
    output bor;
            wire abar;
            xor
            x1(dif,a,b);
            not
            n1(abar,a);
            and
            a1(bor,abar,b);
endmodule
```

FULL ADDER:

LOGIC DIAGRAM:



TRUTH TABLE:

A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VERILOG SOURCE CODE:

Dataflow Modeling:

```
module fulladddataflow(a, b, cin, sum, carry);
```

```
    input a;
    input b;
    input cin;
    output sum;
    output carry;
 assign sum=a^b^cin;
 assign carry=(a & b) | (b & cin) | (cin & a);
 endmodule
```

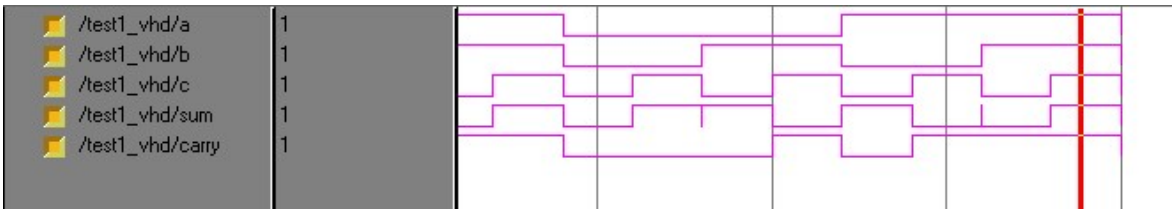
Behavioral Modeling:

```
module fuladbehavioral(a, b, c, sum, carry);
    input a;
    input b;
    input c;
    output sum;
    output carry;
            reg sum,carry;
            reg t1,t2,t3;
            always @ (a or b or c) begin
            sum = (a^b)^c;
            t1=a & b;
            t2=b & c;
            t3=a & c;
            carry=(t1 | t2) | t3;
            end
endmodule
```

Structural Modeling:

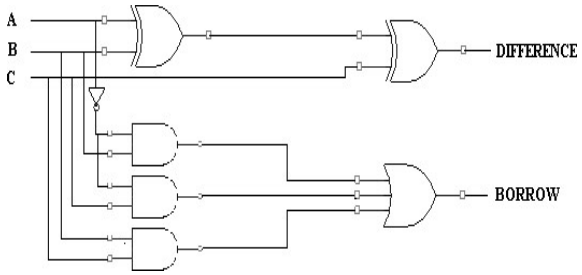
```
module fa_struct(a, b, c, sum, carry);
    input a;
    input b;
    input c;
    output sum;
    output carry;
            wire p,q,r,s;
            xor
            x1(p,a,b),
            x2(sum,p,c);
            and
            a1(q,a,b),
            a2(r,b,c),
            a3(s,a,c);
            or
            o1(carry,q,r,s);
endmodule
```

Simulation output:



FULL SUBSTRACTOR:

LOGIC DIAGRAM:



TRUTH TABLE:

A	B	C	DIFFERENCE	BORROW
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

VERILOG SOURCE CODE:

Dataflow Modeling:

```
module fulsubdataflow(a, b, cin, diff, borrow);
   input a;
   input b;
   input cin;
   output diff;
   output borrow;
            wire abar;
            assign abar= ~ a;
            assign diff=a^b^cin;
            assign borrow=(abar & b) | (b & cin) |(cin & abar);

endmodule
```

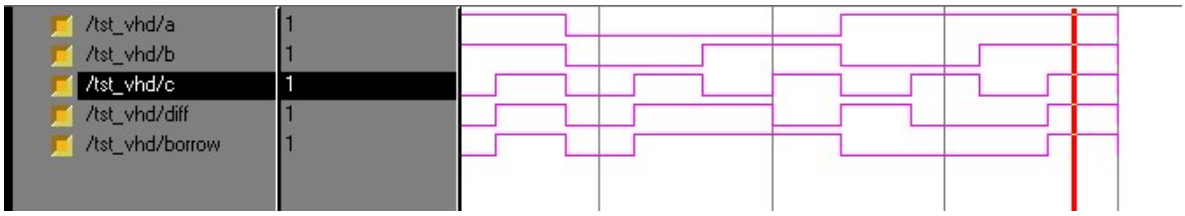
Behavioral Modeling:

```
module fulsubbehavioral(a, b, cin, diff, borrow);
   input a;
   input b;
   input cin;
   output diff;
   output borrow;
            reg t1,t2,t3;
            reg diff,borrow;
            reg abar;
            always @ (a or b or cin) begin
            abar= ~ a;
            diff = (a^b)^cin;
            t1=abar & b;
            t2=b & cin;
            t3=cin & abar;
            borrow=(t1 | t2) | t3;
            end
           endmodule
```

Structural Modeling:

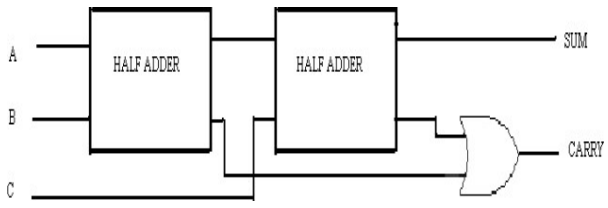
```
module fs_struct(a, b, c, diff, borrow);
   input a;
   input b;
   input c;
   output diff;
   output borrow;
            wire abar,p,q,r,s;
            not
            n1(abar,a);
            xor
            x1(p,a,b),
            x2(diff,p,c);
            and
            a1(q,abar,b),
            a2(r,abar,c),
            a3(s,a,c);
            or
            o1(borrow,q,r,s);
endmodule
```

Simulation output:



FULL ADDER USING TWO HALF ADDERS:

LOGIC DIAGRAM:



TRUTH TABLE:

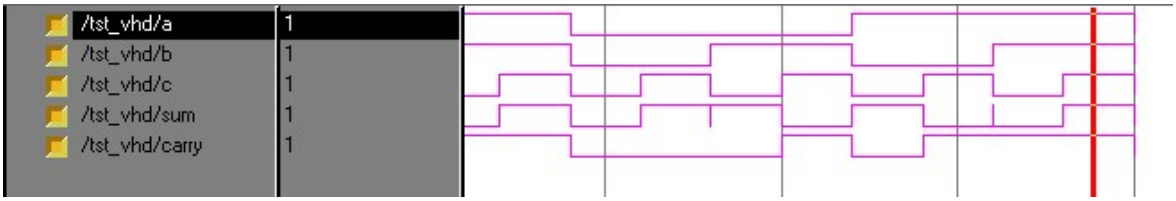
A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VERILOG SOURCE CODE:
Structural Modeling:

```
module fa_2ha(a, b, c, sum, carry);
   input a;
   input b;
   input c;
   output sum;
```

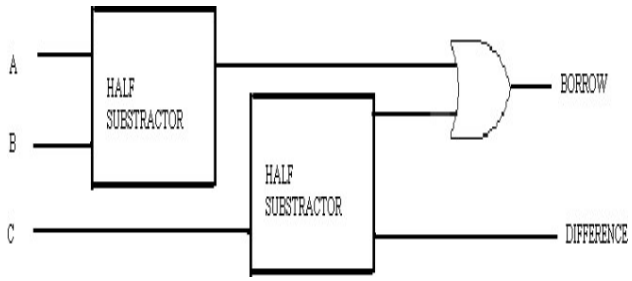
```
    output carry;
            wire p,q,r;
            ha_dataflow
            h1(a,b,p,q),
            h2(p,c,sum,r);
            or
            o1(carry,q,r);
            endmodule
module ha_dataflow(a, b, s, ca);
    input a;
    input b;
    output s;
    output ca;
            assign#2 s=a^b;
            assign#2 ca=a&b;
endmodule
```

Simulation output:



FULL SUBTRACTOR USING TWO HALF SUBTRACTORS:

LOGIC DIAGRAM:



TRUTH TABLE:

A	B	C	DIFFERENCE	BORROW
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

VERILOG SOURCE CODE:

Structural Modeling:

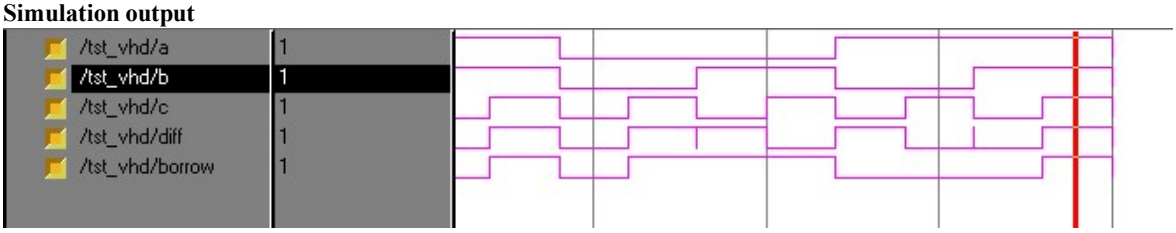
```
module fs_2hs(a, b, c, diff, borrow);
    input a;
    input b;
    input c;
    output diff;


    output borrow;
    wire p,q,r;
```

```
        hs_dataflow
        h1(a,b,p,q),
        h2(p,c,diff,r);
        or
        o1(borrow,q,r);
endmodule

module hs_dataflow(a, b, dif, bor);
   input a;
   input b;
   output dif;
   output bor;
        wire abar;
        assign#3 abar=~a;
        assign#3 dif=a^b;
        assign#3 bor=b&abar;
endmodule
```

Simulation output



Result: