SIMULATION AND IMPLEMENTATION OF MULTIPLIER

AIM

To simulate and synthesis multiplier using Xilinx ISE.

PROCEDURE

STEP:1 Start the Xilinx navigator, Select and Name the New project.

STEP:2 Select the device family, device, package and speed.

STEP:3 Select new source in the New Project and select Verilog Module as the Source type.:

STEP 4 Type the File Name and Click Next and then finish button. Type the code and save it.

STEP:5 Select the Behavioral Simulation in the Source Window and click the check syntax.

STEP:6 Click the simulation to simulate the program and give the inputs and verify the outputs as per the truth table.

STEP:7 Select the Implementation in the Sources Window and select the required file in the Processes Window.

STEP:8 Select Check Syntax from the Synthesize XST Process. Double Click in the FloorplanArea/IO/Logic-Post Synthesis process in the User Constraints process group. UCF(User constraint File) is obtained.

STEP:9 In the Design Object List Window, enter the pin location for each pin in the Loc column Select save from the File menu.

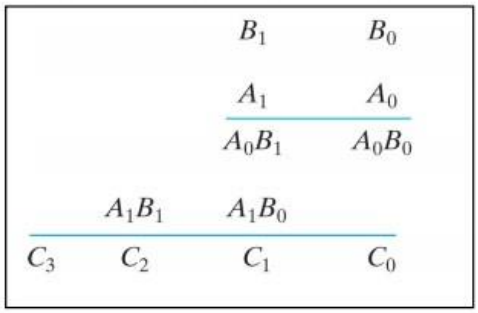
STEP:10 Double click on the Implement Design and double click on the Generate Programming File to create a bitstream of the design.(.v) file is converted into .bit file here.

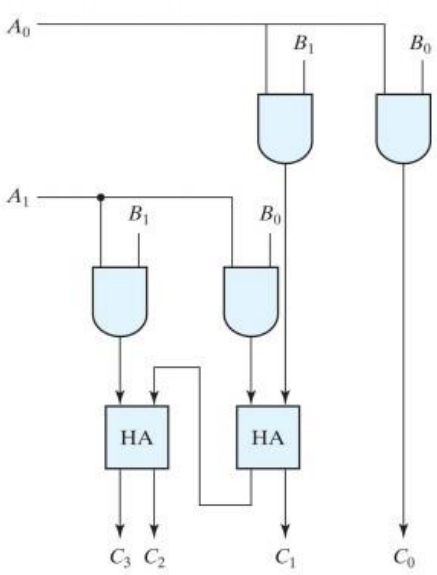
STEP:11 On the board, by giving required input, the LEDs starts to glow light, indicating the output.

LOGIC DIAGRAM

\\2 bit Multiplier\\

of decimal numbers



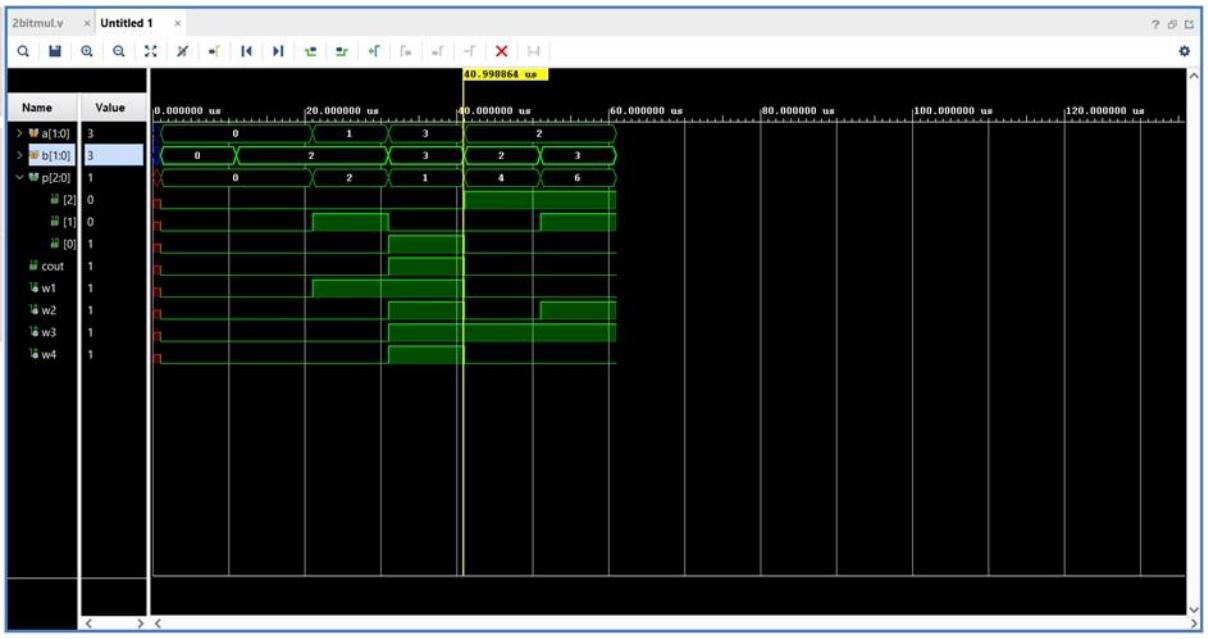


VERILOG CODE

```
module ha(a,b,sum,carry);

input a,b;

output sum,carry;

xor g1(sum,a,b);

and g2(carry,a,b);

endmodule

module bitmul(a,b,p,cout);

input [1:0]a,b;

output [2:0]p;
```

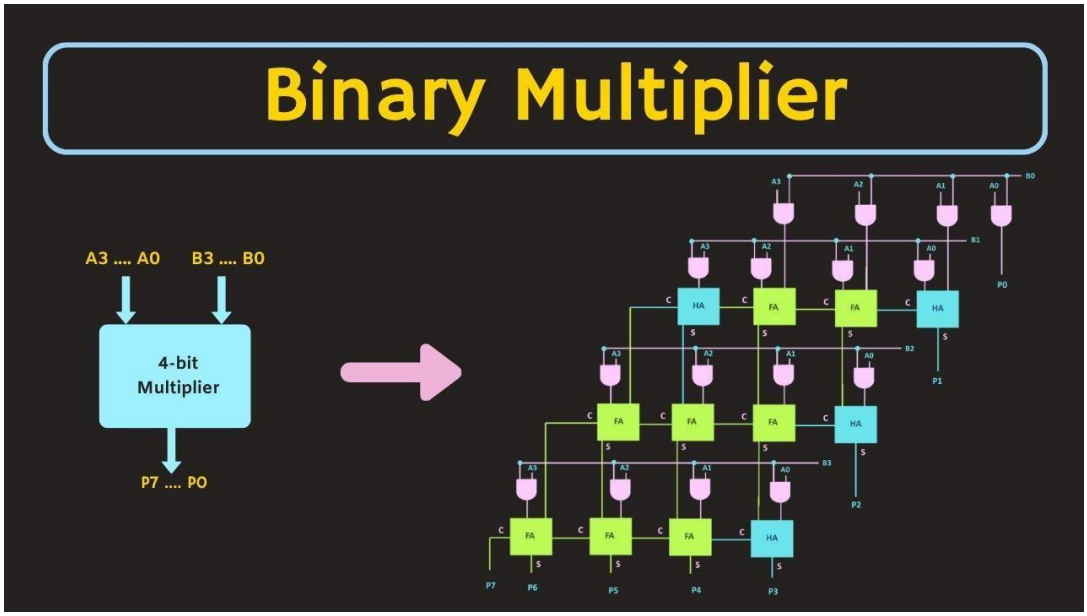
```
output cout;
wire w1,w2,w3,w4;
and (p[0],a[0],b[0]);
and (w1,a[0],b[1]);
and (w2,a[1],b[0]);
and (w3,a[1],b[1]);
ha adder1(w1,w2,p[1],w4);
ha adder2(w3,w4,p[2],cout);
endmodule
```

OUTPUT



LOGIC DIAGRAM

\\ 4 bit Multiplier\\



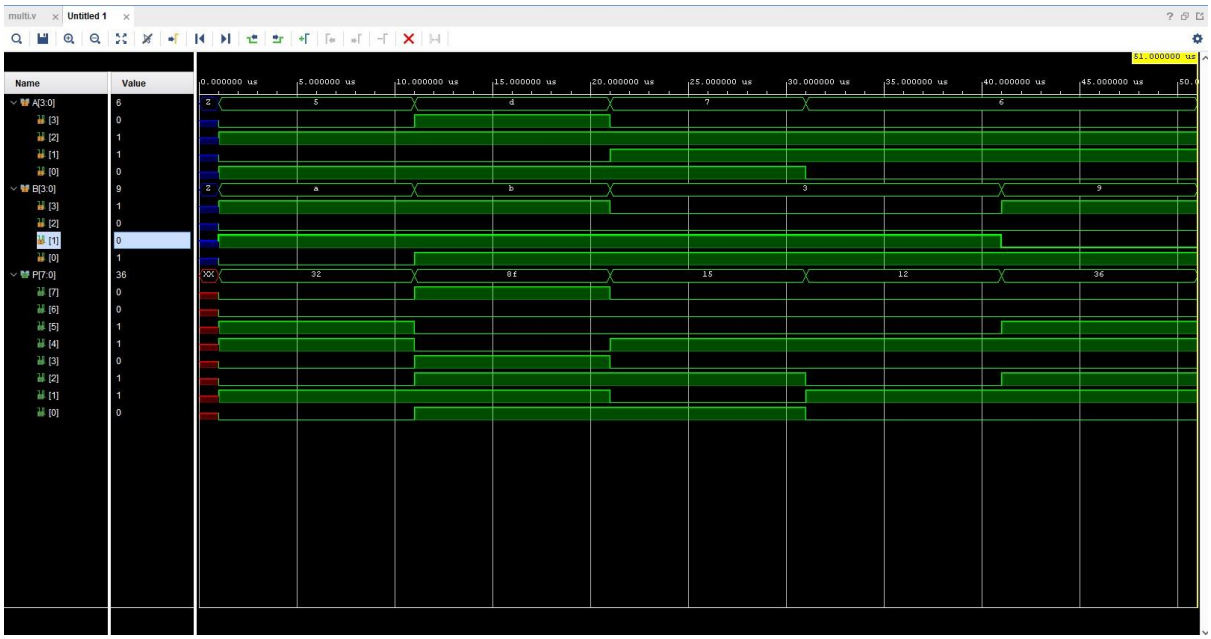
VERILOG CODE

```
module ha(a,b,sum,carry);
input a,b;
output sum,carry;
xor g1(sum,a,b);
and g2(carry,a,b);
endmodule
module fa(a,b,c,sum,carry);
input a,b,c;
```

```
output sum,carry;
wire w1,w2,w3;
xor(w1,a,b);
xor(sum,w1,c);
and(w2,w1,c);
and(w3,a,b);
or(carry,w2,w3);
endmodule
module mul4bit(a,b,p);
input [3:0]a,b;
output [7:0]p;
wire [15:0]w;
wire [3:0]hc;
wire [6:0]fc;
wire [4:0]fs;
wire hs;
and r11(p[0],a[0],b[0]);
and r12(w[1],a[1],b[0]);
and r13(w[2],a[2],b[0]);
and r14(w[3],a[3],b[0]);
and r21(w[4],a[0],b[1]);
and r22(w[5],a[1],b[1]);
and r23(w[6],a[2],b[1]);
and r24(w[7],a[3],b[1]);
ha r31(w[1],w[4],p[1],hc[0]);
```

```
fa r32(w[2],w[5],hc[0],fs[0],fc[0]);
fa r33(w[3],w[6],fc[0],fs[1],fc[1]);
ha r34(w[7],fc[1],hs,hc[1]);
and r41(w[8],a[0],b[2]);
and r42(w[9],a[1],b[2]);
and r43(w[10],a[2],b[2]);
and r44(w[11],a[3],b[2]);
ha r51(w[8],fs[0],p[2],hc[2]);
fa r52(w[9],fs[1],hc[2],fs[2],fc[2]);
fa r53(w[10],hs,fc[2],fs[3],fc[3]);
fa r54(w[11],hc[1],fc[3],fs[4],fc[4]);
and r61(w[12],a[0],b[3]);
and r62(w[13],a[1],b[3]);
and r63(w[14],a[2],b[3]);
and r64(w[15],a[3],b[3]);
ha r71(w[12],fs[2],p[3],hc[3]);
fa r72(w[13],fs[3],hc[3],p[4],fc[5]);
fa r73(w[14],fs[4],fc[5],p[5],fc[6]);
fa r74(w[15],fc[4],fc[6],p[6],p[7]);
endmodule
```

OUTPUT



RESULT

Thus The Simulate And Synthesis Multiplier Using Xilinx Ise Are Verified Successfully