SIMULATION AND IMPLEMENTATION OF MULTIPLIER

AIM: To simulate and synthesis multiplier using Xilinx ISE.

APPARATUS REQUIRED: Xilinx 14.7 Spartan6 FPGA

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 Floorplan Area/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.

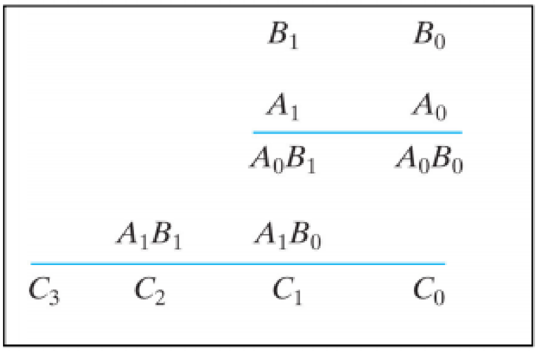
STEP:10 Double click on the Implement Design and double click on the Generate Programming File to create a bitstream of the design.(.v) file is converted into .bit file here.

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

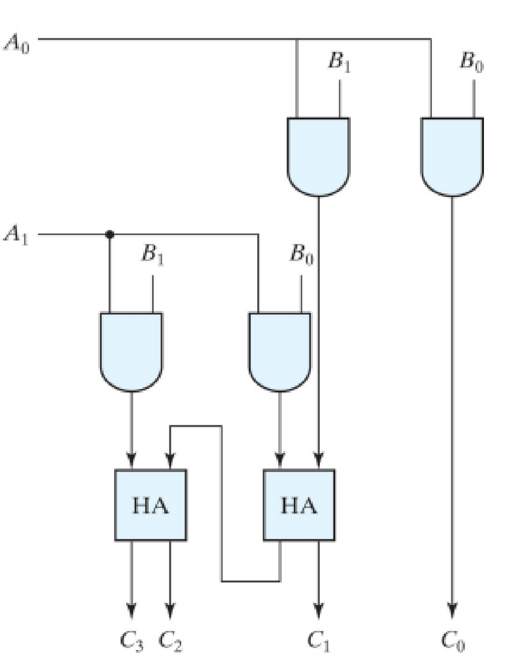
Logic Diagram:

2 bit Multiplier

of decimal numbers



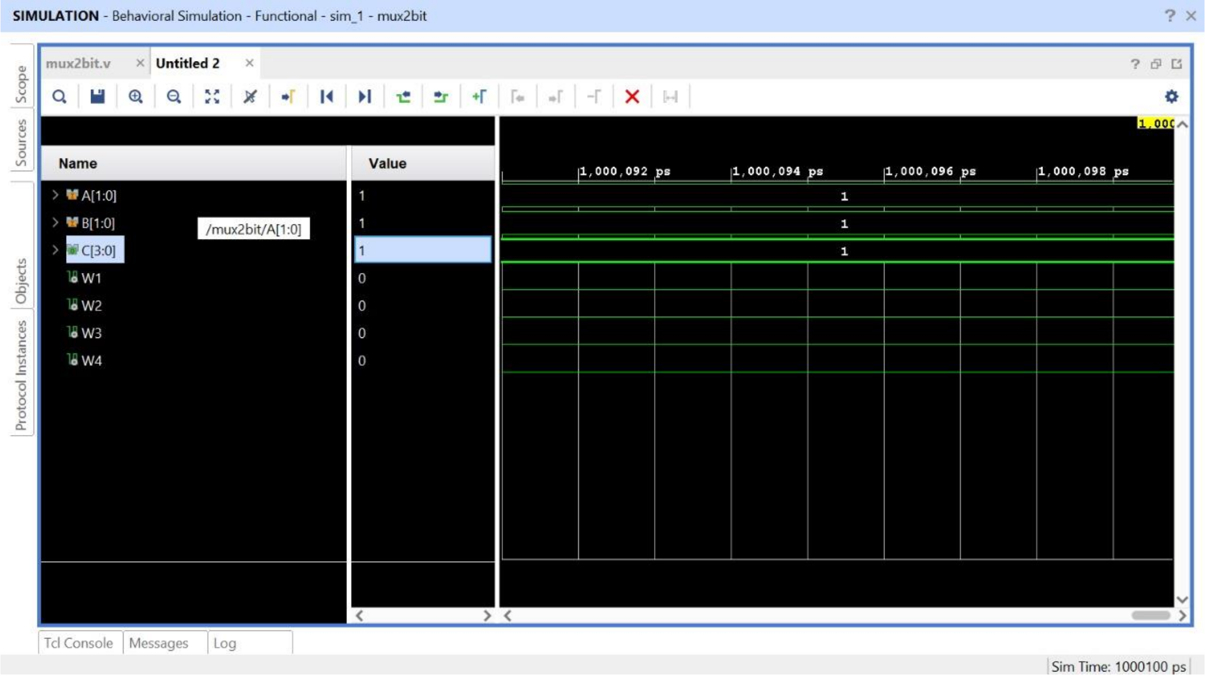
		B_1	B_0
		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0



Verilog code

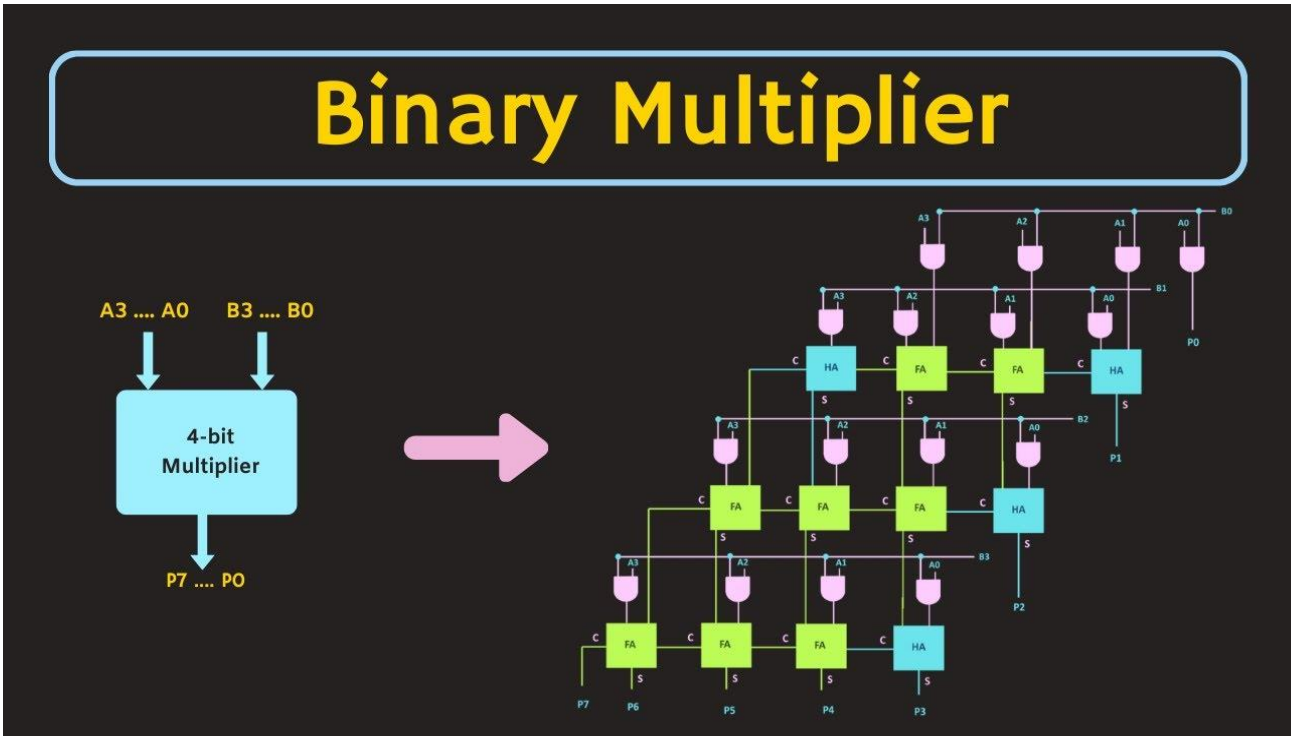
```
module ha(a,b,sum,carry);
input a,b;
output sum,carry;
assign sum=a^b;
assign carry=a&b;
endmodule
module mux2bit(A,B,C);
input [1:0]A,B;
output [3:0]C;
wire W1,W2,W3,W4;
and g1(C[0],A[0],B[0]);
and g2(W1,A[0],B[1]);
and g3(W2,A[1],B[0]);
and g4(W3,A[1],B[1]);
ha ha1(W1,W2,C[1],W4);
ha ha2(W4,W3,C[3],C[2]);
endmodule
```

Output Waveform



Name	Value
A[1:0]	1
B[1:0]	1
C[3:0]	1
W1	0
W2	0
W3	0
W4	0

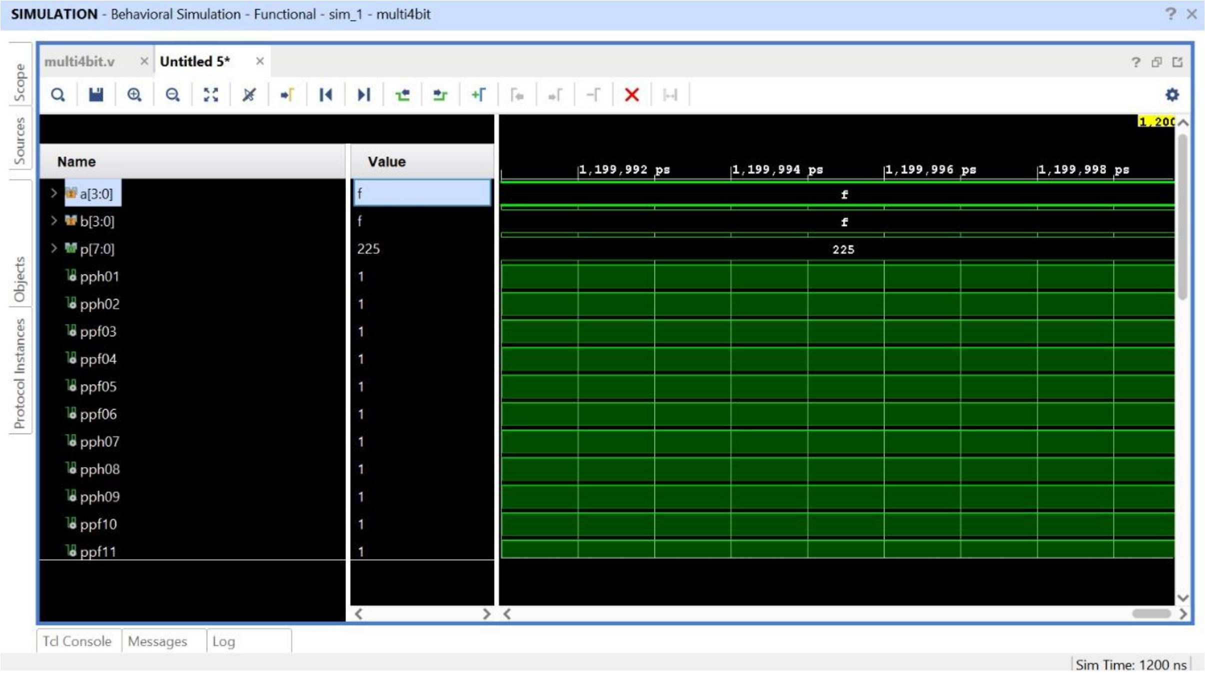
4 Bit Multiplier



Verilog code

```
module ha(a,b,s,c);
input a,b;
output s,c;
xor(s,a,b);
and(c,a,b);
endmodule
module fa(a,b,c,sum,carry);
input a,b,c;
output sum,carry;
wire w1,w2,w3;
ha ha1(a,b,w1,w2);
ha ha2(w1,c,sum,w3);
or g1(carry,w3,w2);
endmodule
module multi4bit(a,b,p);
input [3:0]a,b;
output [7:0]p;
wire
pph01,pph02,ppf03,ppf04,ppf05,ppf06,pph07,pph08,pph09,ppf10,ppf11,ppf12
,ppf13,ppf14,ppf15,pph16,pph17,ppf18,ppf19,ppf20,ppf21,ppf22,ppf23,c1,c
2,c3,c4,c5,c6,c7,c8,c9;
and g1(p[0],a[0],b[0]);
and g2(pph01,a[1],b[0]);
and g3(pph02,a[0],b[1]);
and g4(ppf03,a[2],b[0]);
and g5(ppf04,a[1],b[1]);
and g6(ppf05,a[3],b[0]);
and g7(ppf06,a[2],b[1]);
and g8(pph07,a[3],b[1]);
ha ha1(pph01,pph02,p[1],c1);
fa fa1(ppf03,ppf04,c1,pph09,c2);
fa fa2(ppf06,ppf05,c2,ppf11,c3);
ha ha2(pph07,c3,ppf13,ppf14);
and g9(pph08,a[0],b[2]);
and g10(ppf10,a[1],b[2]);
and g11(ppf12,a[2],b[2]);
and g12(ppf15,a[3],b[2]);
ha ha3(pph08,pph09,p[2],c4);
fa fa3(ppf10,ppf11,c4,pph17,c5);
fa fa4(ppf12,ppf13,c5,ppf19,c6);
fa fa5(ppf14,ppf15,c6,ppf21,ppf22);
and g13(pph16,a[0],b[3]);
and g14(ppf18,a[1],b[3]);
and g15(ppf20,a[2],b[3]);
and g16(ppf23,a[3],b[3]);
ha ha4(pph16,pph17,p[3],c7);
fa fa6(ppf18,ppf19,c7,p[4],c8);
fa fa7(ppf20,ppf21,c8,p[5],c9);
fa fa8(ppf22,ppf23,c9,p[6],p[7]);
endmodule
```

Output Waveform



SIMULATION - Behavioral Simulation - Functional - sim_1 - multi4bit

Name	Value
a[3:0]	f
b[3:0]	f
p[7:0]	225
pph01	1
pph02	1
ppf03	1
ppf04	1
ppf05	1
ppf06	1
pph07	1
pph08	1
pph09	1
ppf10	1
ppf11	1

Sim Time: 1200 ns

RESULT:

Thus the simulation and implementation of multiplier is verified successfully.