SIMULATION AND IMPLEMENTATION OF MULTIPLIERS

AIM:

To simulate and implement the following circuits using Vivado 2023.2.

1) 2 bit multiplier

2) 4 bit multiplier

APPARATUS REQUIRED:

Vivado 2023.2

PROCEDURE:

STEP:1 Launch the Vivado 2023.2 software.

STEP:2 Click on “create project ” from the starting page of vivado .

STEP:3 Choose the design entry method:RTL(verilog/VHDL)

STEP:4 Crete design source and give name to it and click finish.

STEP:5 Write the verilog code and check the syntax.

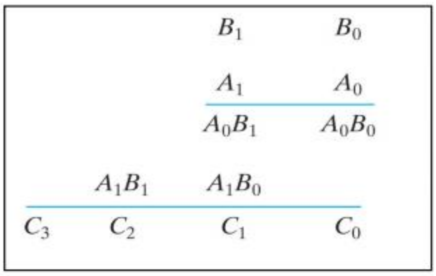
STEP:6 Click “run simulation” in the navigator window and click “Run behavioral simulation”.

STEP:7 Verify the output in the simulation window.

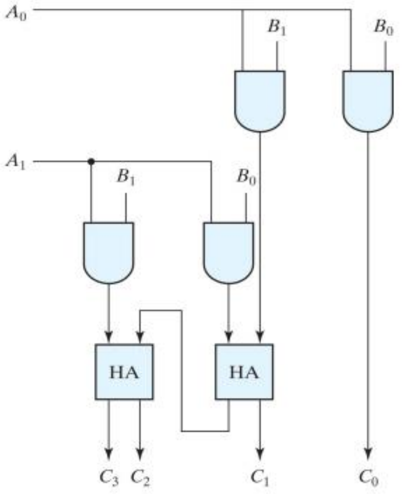
Logic Diagram

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



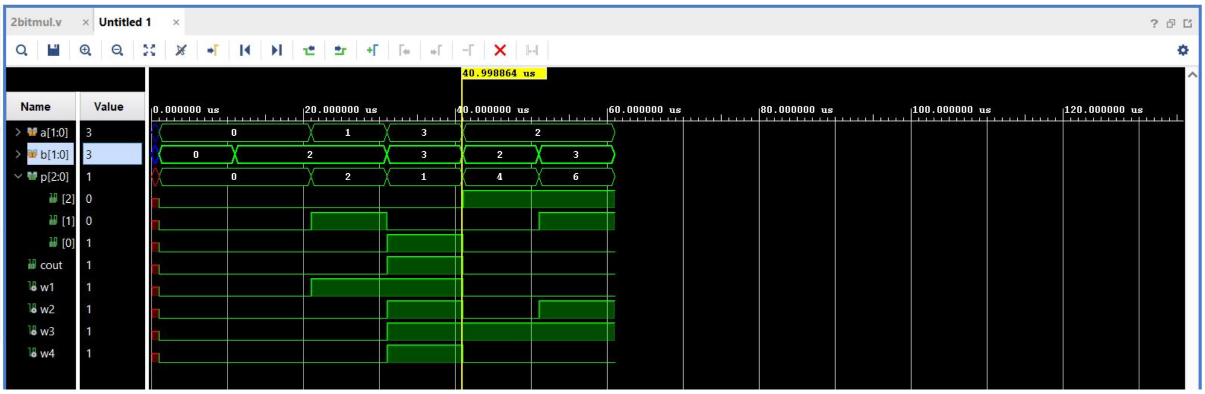
VERILOG CODE

2 bit multiplier

```
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
```

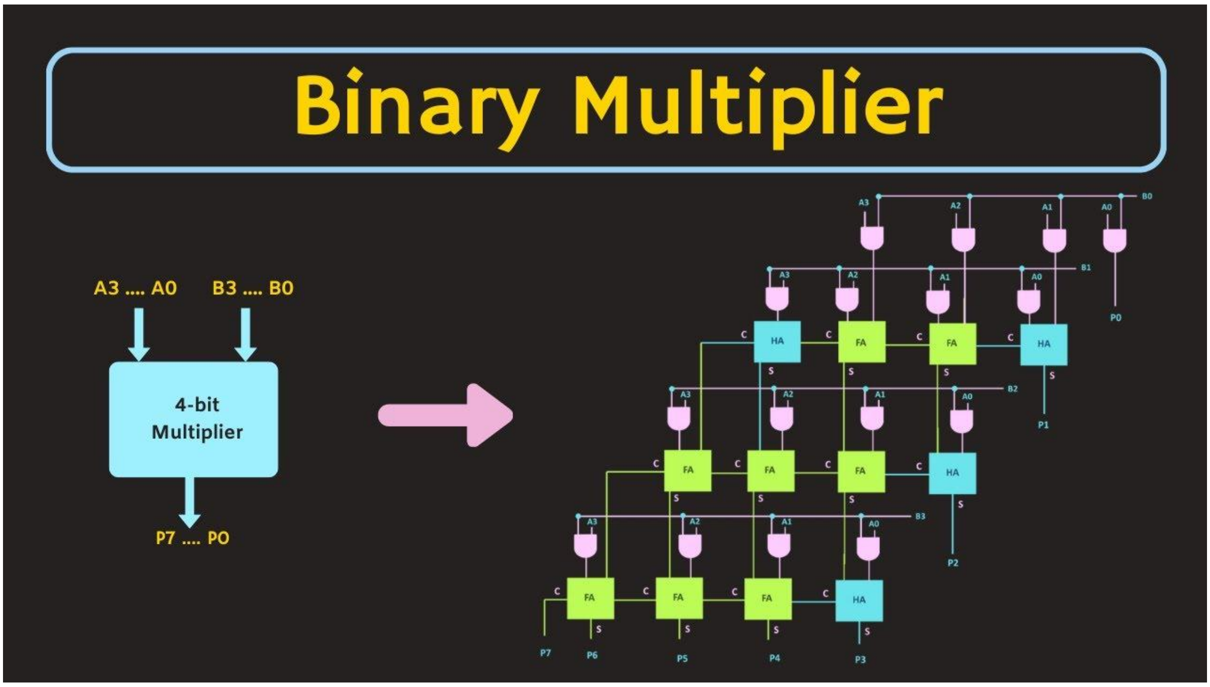
OUTPUT WAVEFORM

2 bit multiplier



LOGIC DIAGRAM

4 Bit Multiplier



VERILOG CODE

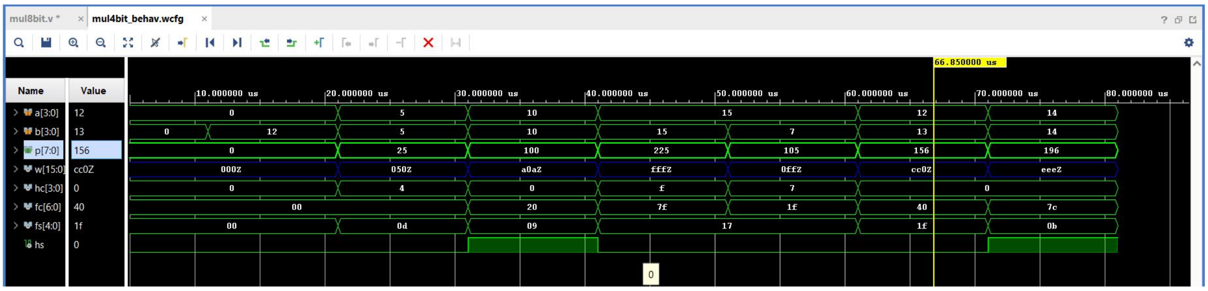
4 bit multiplier

```
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 WAVEFORM

4 bit multiplier



RESULT

Thus the simulation and implementation of multipliers is done and the outputs are verified successfully.