

SIMULATION AND IMPLEMENTATION OF 2Bit AND 4-BIT 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 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

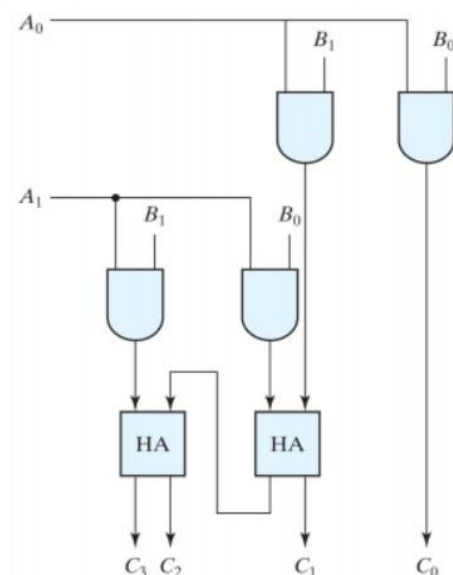
On the board, by giving required input, the LEDs starts to glow light, indicating the output.

Logic Diagram:

2 bit Multiplier

of decimal numbers

		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

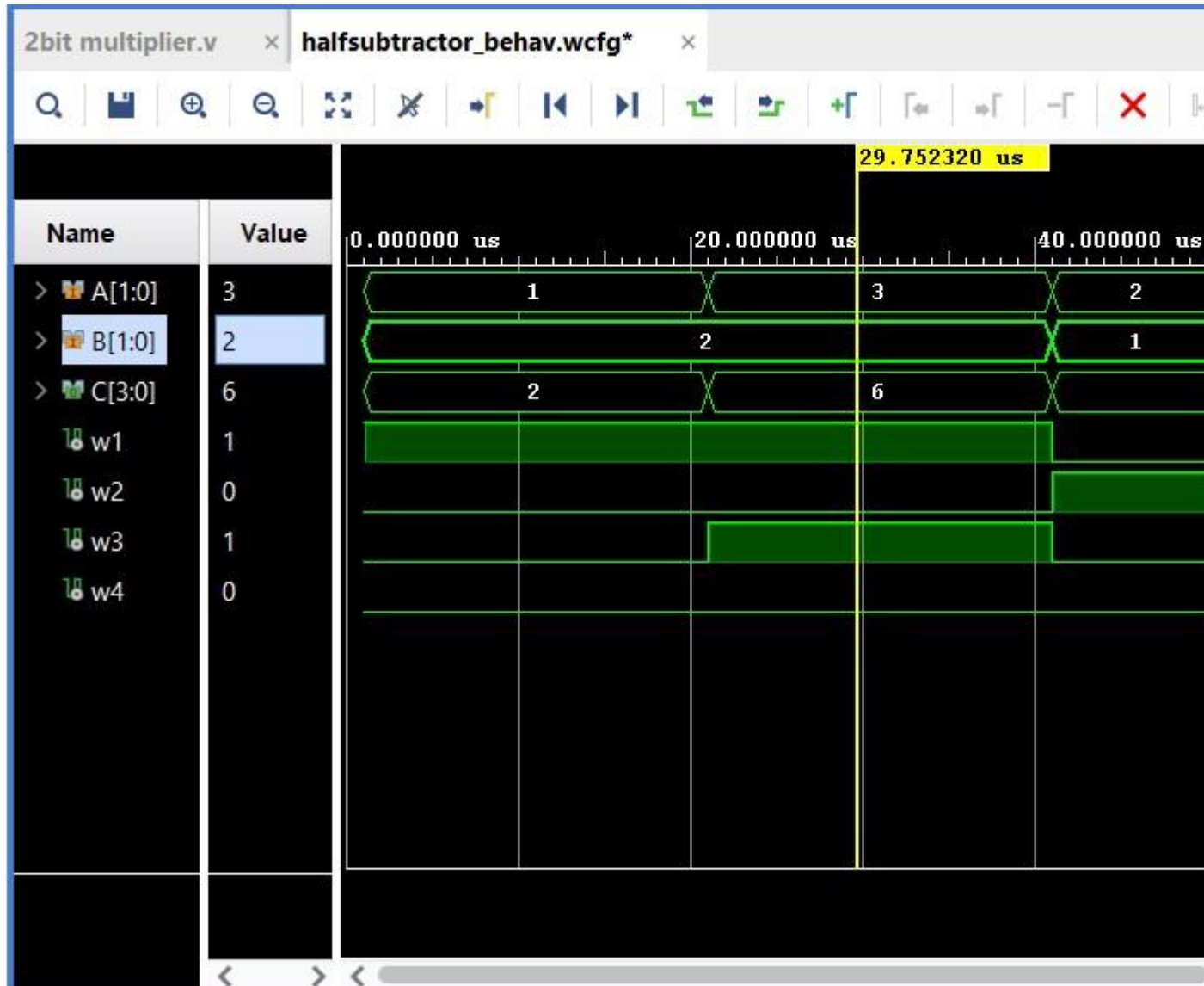


Verilog code:

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

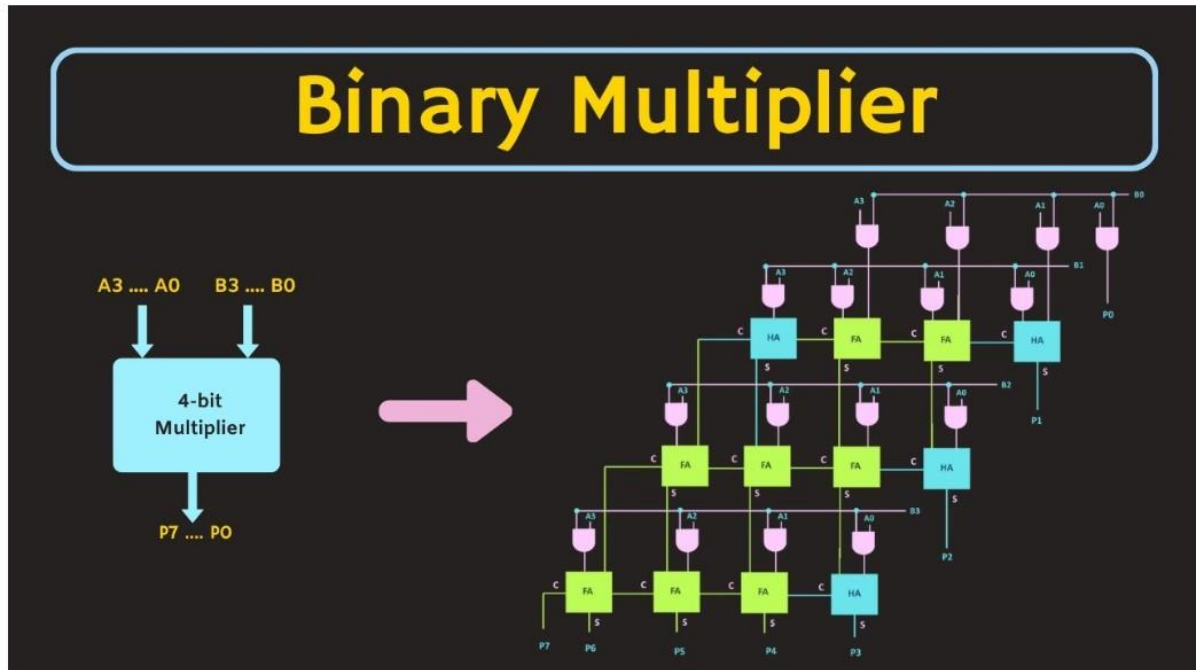
```
module mul_2(A,B,C);  
  
input [1:0]A,B;  
  
output [3:0]C;  
  
wire w1,w2,w3,w4;  
  
and g1(C[0],A[0],B[0]);  
  
and g2(w1,A[0],B[1]);  
  
and g3(w2,A[1],B[0]);  
  
and g4(w3,A[1],B[1]);  
  
ha ha1(w1,w2,C[1],w4);  
  
ha ha2(w4,w3,C[2],C[3]);  
  
endmodule
```

Output Waveform



Logic Diagram:

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;
```

```
xor g1(w1,a,b);
```

```
xor g2(sum,w1,c);
```

```
and g3(w2,a,b);
```

```
and g4(w3,w1,c);
```

```
or g5(carry,w3,w2);
```

```
endmodule
```

```
module mul(x,y,z);
```

```
input [3:0]x,y;
```

```
output [7:0]z;
```

```
wire [17:1]w;
```

```
ha ha1(x[1]&y[0],x[0]&y[1],z[1],w[1]);
```

```
fa fa1(x[2]&y[0],x[1]&y[1],w[1],w[5],w[2]);
```

```
fa fa2(x[3]&y[0],x[2]&y[1],w[2],w[6],w[3]);
```

```
ha ha2(x[3]&y[1],w[3],w[7],w[4]);
```

```
ha ha3(w[5],x[0]&y[2],z[2],w[8]);
```

```
fa fa3(w[6],x[1]&y[2],w[8],w[12],w[9]);
```

```
fa fa4(w[7],x[2]&y[2],w[9],w[13],w[10]);
```

```
fa fa5(w[4],x[3]&y[2],w[10],w[14],w[11]);
```

```

ha ha4(w[12],x[0]&y[3],z[3],w[15]);

fa fa6(w[13],x[1]&y[3],w[15],z[4],w[16]);

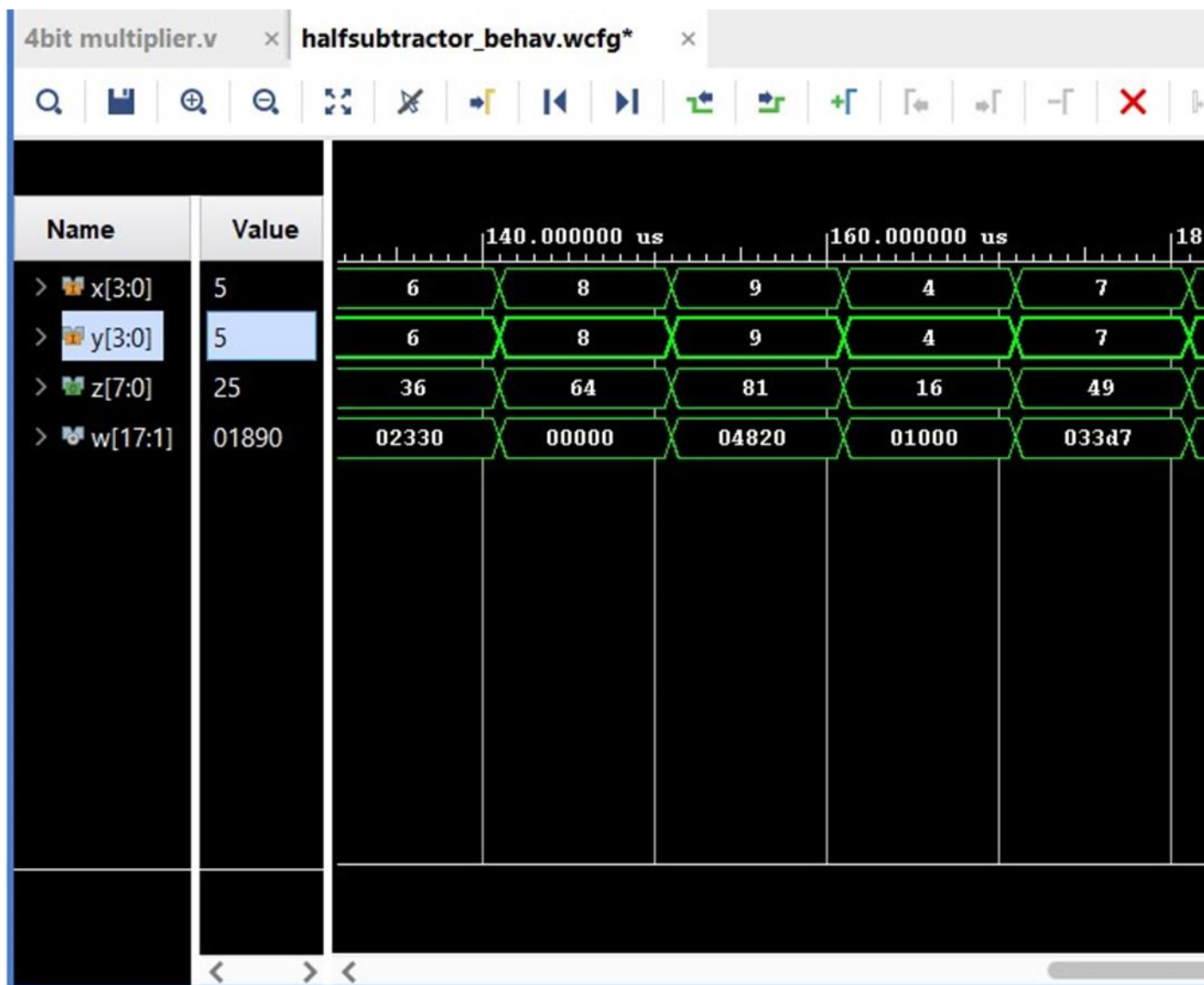
fa fa7(w[14],x[2]&y[3],w[16],z[5],w[17]);

fa fa8(w[11],x[3]&y[3],w[17],z[6],z[7]);

endmodule

```

Output Waveform:



Result

The simulation and synthesis 2Bit-Multiplier and 4Bit-Multiplier by running successfully using Xilinx ISE.