

Input/Output Specification and Verilog Code:

- Input ranges from -128 to +127
- Negative numbers are represented using 2's-complement notation.
- Again, if the output is negative, then it is obtained in 2's-complement form.

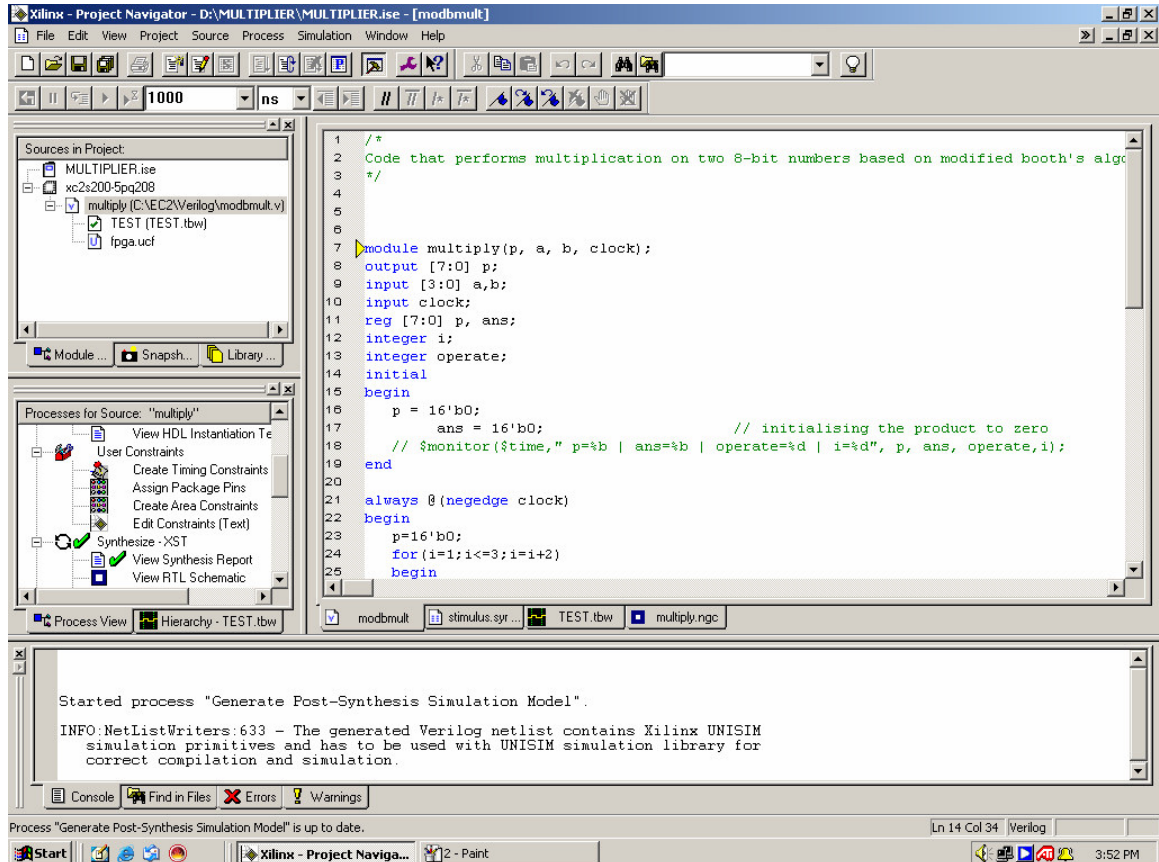
```
module multiply(p, a, b, clock);
output [15:0] p;
input [7:0] a,b;
input clock;
reg [15:0] p, ans;
integer i;
integer operate;
initial
begin
    p = 16'b0;           // initialising the product to zero
    ans = 16'b0;
end
always @(negedge clock)
begin
    p=16'b0;
    for(i=1;i<=7;i=i+2) //loop for multiplication
    begin
        if(i==1)
            operate = b[0]-b[1]-b[1];           //b[-1] = 0 (assumed)
        else
            operate = b[i-1] + b[i-2] - b[i] - b[i];
        case (operate)
            1:           //multiplication by 1
            begin
                ans = a;
                ans = ans<<(i-1);
                p = p + ans;
            end
            2:
            begin
                ans = a<<1;           //multiplication by 2
                ans = ans<<(i-1);
                p = p + ans;
            end
            -1:           //multiplication by -1
            begin
                ans = ~a + 1;
                ans = ans<<(i-1);
                p = p + ans;
            end
            -2:           //multiplication by -2
            begin
                ans = a<<1;
                ans = ~ans + 1;
                ans = ans<<(i-1);
                p = p + ans;
            end
        endcase
    end
end
endmodule
```

Simulation and Implementation in FPGA:

ModelSim is used for simulation of the code and Xilinx is used for the implementation of the code in FPGA.

Here are some screenshots at various stages of the implementation procedure:

The Code:



The screenshot displays the Xilinx Project Navigator interface. The main window shows the Verilog code for a multiplier module named 'multiply'. The code is as follows:

```
1 /*
2 Code that performs multiplication on two 8-bit numbers based on modified booth's algo
3 */
4
5
6
7 module multiply(p, a, b, clock);
8 output [7:0] p;
9 input [3:0] a,b;
10 input clock;
11 reg [7:0] p, ans;
12 integer i;
13 integer operate;
14 initial
15 begin
16     p = 16'b0;
17     ans = 16'b0; // initialising the product to zero
18     // $monitor($time," p=%b | ans=%b | operate=%d | i=%d", p, ans, operate,i);
19 end
20
21 always @(negedge clock)
22 begin
23     p=16'b0;
24     for(i=1;i<=3;i=i+2)
25     begin
```

The interface also shows the 'Sources in Project' pane on the left, listing files like 'MULTIPLIER.isc', 'xc2s200-5pq208', 'multiply (C:\EC2\Verilog\modbmult.v)', 'TEST (TEST.tbw)', and 'fpga.ucf'. Below that is the 'Processes for Source: "multiply"' pane, which lists various tasks such as 'View HDL Instantiation Te...', 'User Constraints', 'Create Timing Constraints', 'Assign Package Pins', 'Create Area Constraints', 'Edit Constraints (Text)', 'Synthesize -XST', 'View Synthesis Report', and 'View RTL Schematic'. At the bottom, the console window shows the message: 'Started process "Generate Post-Synthesis Simulation Model". INFO:NetListWriters:633 - The generated Verilog netlist contains Xilinx UNISIM simulation primitives and has to be used with UNISIM simulation library for correct compilation and simulation.'

Input Assignment for Simulation:

The screenshot shows the Xilinx Project Navigator interface. The main window displays a timing diagram for a simulation. The simulation is set to 1000 ns. The input signals are:

- clock: A square wave signal.
- a[3:0]: A 4-bit bus signal that starts at 0, then changes to 8, and then to 7.
- b[3:0]: A 4-bit bus signal that starts at 0, then changes to 1, and then to 5.
- p[7:0]: A 8-bit bus signal that remains at 0.

The console window at the bottom displays the following message:

```
INFO:NetListWriters:633 - The generated Verilog netlist contains Xilinx UNISIM simulation primitives and has to be used with UNISIM simulation library for correct compilation and simulation.
```

Output Waveform (simulated using ModelSim):

The screenshot shows the ModelSim waveform viewer. The waveform displays the output signals for the simulation. The signals are:

- /TEST/p: A 4-bit bus signal that starts at 00100011, then changes to 00001000, and then to 00100011.
- /TEST/a: A 4-bit bus signal that starts at 0000, then changes to 1000, and then to 0111.
- /TEST/b: A 4-bit bus signal that starts at 0101, then changes to 0000, and then to 0001.
- /TEST/clock: A square wave signal.
- /TEST/TX_FILE: A signal that starts at 2.
- /TEST/TX_ERROR: A signal that starts at 0.

Synthesized Circuit:

Technology Schema – Hardware Implementation

The screenshot displays the Xilinx Project Navigator interface for a project named 'MULTIPLIER'. The main window shows a technology schema for the synthesized circuit, which is a multiplier. The circuit is composed of several 6-input LUTs (Look-Up Tables) and other logic elements, interconnected to perform multiplication. The LUTs are labeled with names like 'LUT6_0', 'LUT6_1', 'LUT6_2', 'LUT6_3', 'LUT6_4', 'LUT6_5', 'LUT6_6', 'LUT6_7', 'LUT6_8', 'LUT6_9', 'LUT6_10', 'LUT6_11', 'LUT6_12', 'LUT6_13', 'LUT6_14', 'LUT6_15', 'LUT6_16', 'LUT6_17', 'LUT6_18', 'LUT6_19', 'LUT6_20', 'LUT6_21', 'LUT6_22', 'LUT6_23', 'LUT6_24', 'LUT6_25', 'LUT6_26', 'LUT6_27', 'LUT6_28', 'LUT6_29', 'LUT6_30', 'LUT6_31', 'LUT6_32', 'LUT6_33', 'LUT6_34', 'LUT6_35', 'LUT6_36', 'LUT6_37', 'LUT6_38', 'LUT6_39', 'LUT6_40', 'LUT6_41', 'LUT6_42', 'LUT6_43', 'LUT6_44', 'LUT6_45', 'LUT6_46', 'LUT6_47', 'LUT6_48', 'LUT6_49', 'LUT6_50', 'LUT6_51', 'LUT6_52', 'LUT6_53', 'LUT6_54', 'LUT6_55', 'LUT6_56', 'LUT6_57', 'LUT6_58', 'LUT6_59', 'LUT6_60', 'LUT6_61', 'LUT6_62', 'LUT6_63', 'LUT6_64', 'LUT6_65', 'LUT6_66', 'LUT6_67', 'LUT6_68', 'LUT6_69', 'LUT6_70', 'LUT6_71', 'LUT6_72', 'LUT6_73', 'LUT6_74', 'LUT6_75', 'LUT6_76', 'LUT6_77', 'LUT6_78', 'LUT6_79', 'LUT6_80', 'LUT6_81', 'LUT6_82', 'LUT6_83', 'LUT6_84', 'LUT6_85', 'LUT6_86', 'LUT6_87', 'LUT6_88', 'LUT6_89', 'LUT6_90', 'LUT6_91', 'LUT6_92', 'LUT6_93', 'LUT6_94', 'LUT6_95', 'LUT6_96', 'LUT6_97', 'LUT6_98', 'LUT6_99'. The circuit is connected to a 6-bit bus. The Instance Contents pane on the left lists the components used in the circuit, including 'multiply_n0009(5)cy', 'multiply_n0009(5)lut', 'multiply_n0009(5)_xor', and 'multiply_n0009(6)lut_SW0'. The Console pane at the bottom displays the following message: 'INFO:NetListWriters:633 - The generated Verilog netlist contains Xilinx UNISIM simulation primitives and has to be used with UNISIM simulation library for correct compilation and simulation.'

RTL Implementation:

The screenshot displays the Xilinx Project Navigator interface for a Verilog project named 'multiply.ngr'. The main window shows a detailed RTL implementation of a multiplier circuit, with a vertical list of components on the left side of the diagram area. The circuit consists of several logic gates, including AND gates, OR gates, and a full adder, interconnected to perform multiplication. The implementation is shown in a hierarchical view, with the circuit components and their connections visible.

The Instance Contents panel on the left shows the following structure:

- Instance Contents
 - Pins
 - Nets
 - Instances

The Select the Modules or Snapshot tab panel shows the following text:

Select the Modules or Snapshot tab
(No Processes Available)

The Console window at the bottom displays the following timing analysis results:

```
Minimum input arrival time before clock: 30.269ns  
Maximum output required time after clock: 7.999ns  
Maximum combinational path delay: No path found  
-----
```

The status bar at the bottom indicates the project is 'Ready' and shows the device ID [3564,2482]. The taskbar at the bottom shows the Start button, 2 Windows, Document1, Microsoft PowerPoint, Xilinx - Project Navigator, VeriLogger Pro, Mozilla Firefox, and the system tray with the time 5:28 PM.

Gate level implementation of one of the blocks:

The screenshot displays the Xilinx Project Navigator interface. The main window shows the 'LUT Dialog' for 'LUT 4' with 'INIT = CCC8'. The 'Schematic' tab is active, showing a logic diagram with four inputs (D) and one output (D). The circuit consists of three 2-input AND gates (AR2) and two 2-input OR gates (OR2). The top-left AND gate (AR2) has inputs from the top two D inputs. The top-right AND gate (AR2) has inputs from the top two D inputs. The bottom-left AND gate (AR2) has inputs from the bottom two D inputs. The top-right OR gate (OR2) has inputs from the outputs of the top-left and top-right AND gates. The bottom-right OR gate (OR2) has inputs from the outputs of the top-right and bottom-left AND gates. The output of the bottom-right OR gate is connected to the output D. A 'Close' button is visible at the bottom of the dialog.

Instance Contents:

- _old_p_7(2)_G
- _old_p_7(3)
- _old_p_7(3)_F
- _old_p_7(3)_G
- _old_p_7(4)I
- _old_p_7(5)I

Select the Modules or Snapshot tab (No Processes Available)

INFO:NetListWriter: simulation pri: correct compil.

Console Find in Files Errors Warnings

[2752,429]

Start Xilinx - Project Naviga... 6 - Paint Xilinx: PACE - D:\MULTIPLI... 3:55 PM

Karnaugh map for the above block:

LUT 4
INIT = CCC8

Schematic | Truth Table | Karnaugh Map

	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	1	1
10	0	0	0	0

INFO:NetListWriter:
simulation pri:
correct compil.

Close

Console Find in Files Errors Warnings

[2752,429]

Start | Xilinx - Project Naviga... | 7 - Paint | Xilinx: PACE - D:\MULTIPLI... | 3:55 PM

Layout of FPGA and I/O port Assignments:

The screenshot displays the Xilinx ISE Project Navigator interface for a project named 'MULTIPLIER'. The main window shows the 'Device Architecture for xc2s200-5-pq208' in 'Architecture View'. On the left, the 'Design Object List - I/O Pins' table provides a detailed list of pin assignments.

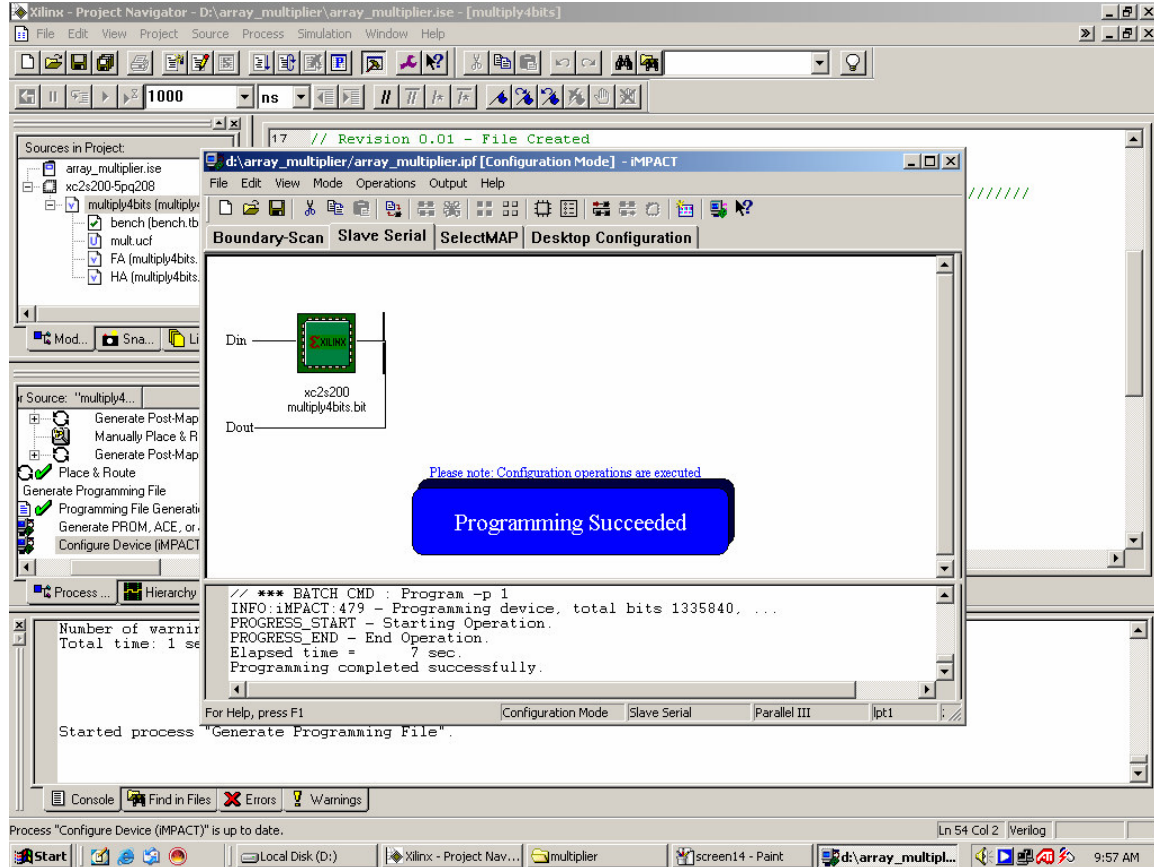
I/O Name	I/O Direction	Loc	Bank	I/O Std.	V _{ref}	V _{cco}	Drive Str.
a<0>	Input	p49	BANK6				
a<1>	Input	p48	BANK6				
a<2>	Input	p57	BANK5				
a<3>	Input	p58	BANK5				
b<0>	Input	p59	BANK5				
b<1>	Input	p60	BANK5				
b<2>	Input	p61	BANK5				
b<3>	Input	p62	BANK5				
clock	Input	p80					
p<0>	Output	p89	BANK4				

Below the main table, a smaller table shows a summary of pin groups:

#	Group	I/O Direction	Loc	I/O Std.	V _{ref}	V _{cco}	Drive Str.
8	p	Output					
4	b	Input					
4	a	Input					

The Architecture View shows a grid representing the device's pinout. Colored dashed lines (blue, green, red, yellow) outline the physical locations of the pins as defined in the Design Object List. A timing diagram on the right shows a signal with a 900 ns period.

Programming the FPGA:



Conclusion:

The main advantage of using FPGA is that it gives high level of flexibility to the user to rapidly construct and test any hardware. FPGAs consist of a lot of gates out of which those that are needed get programmed by the application of suitable field. For example, the Spartan-2 FPGAs consists of 200,000 gates. Thus the number of gates used depends upon the hardware to be implemented and in most of the cases all the gates are not used. Therefore FPGAs are used for testing purposes only.

Hardware Description Languages (HDL) are useful to simulate a digital design before implementing it in the hardware. These languages provide the flexibility to describe the circuit in terms of its behaviour (behavioural modeling). The Compiler implements the code into the hardware.