# LAB MANUAL

# on

# VLSI DSP

**Contents**

**Guidelines for using Xilinx ISE tool:**

To start ISE, Click: Start → All Programs → Xilinx ISE → Project Navigator

**Create a New Project**

To create a new project:

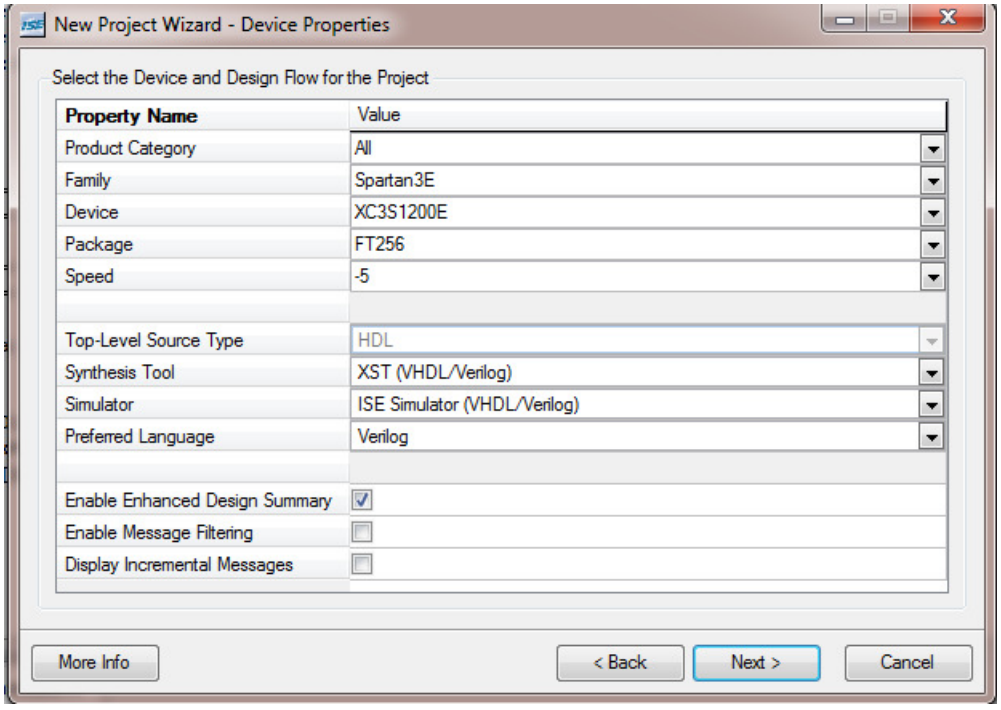
1. Select File > New Project... The New Project Wizard appears.

2. Type any name in the Project Name field.

3. Enter or browse to a location (directory path) for the new project.

4. Click Next to move to the device properties page.

5. Fill in the properties in the table as shown below:




7. Click Next to proceed to the Create New Source window in the New Project Wizard. At the end of the next section, your new project will be complete.

**Creating a VHDL Source**

Create a VHDL source file for the project as follows:

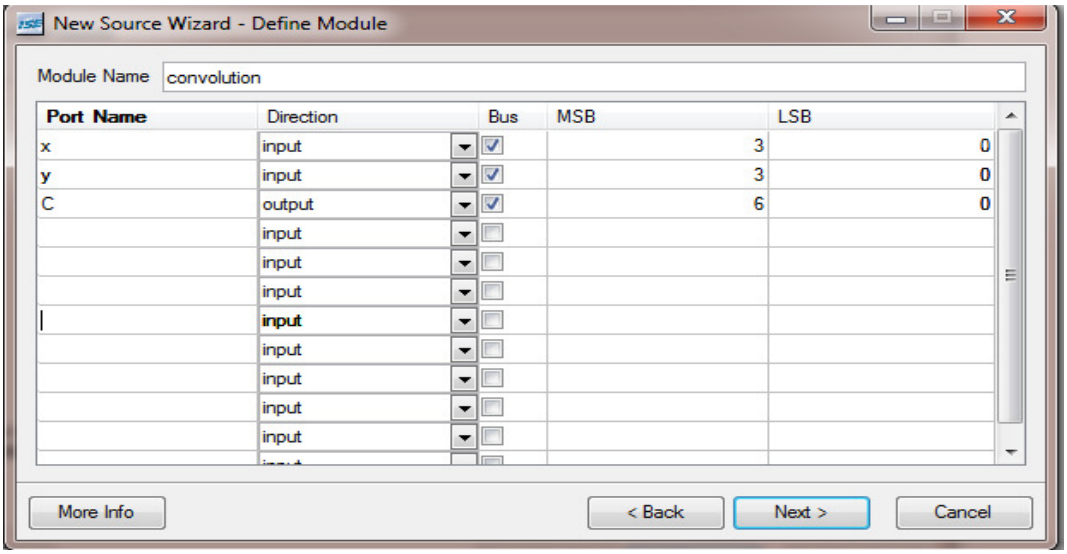
1. Click the New Source button in the New Project Wizard.

2. Select Verilog Module as the source type.

3. Type any file name of your choice (this file name also represents the name of the entity).

4. Click Next.

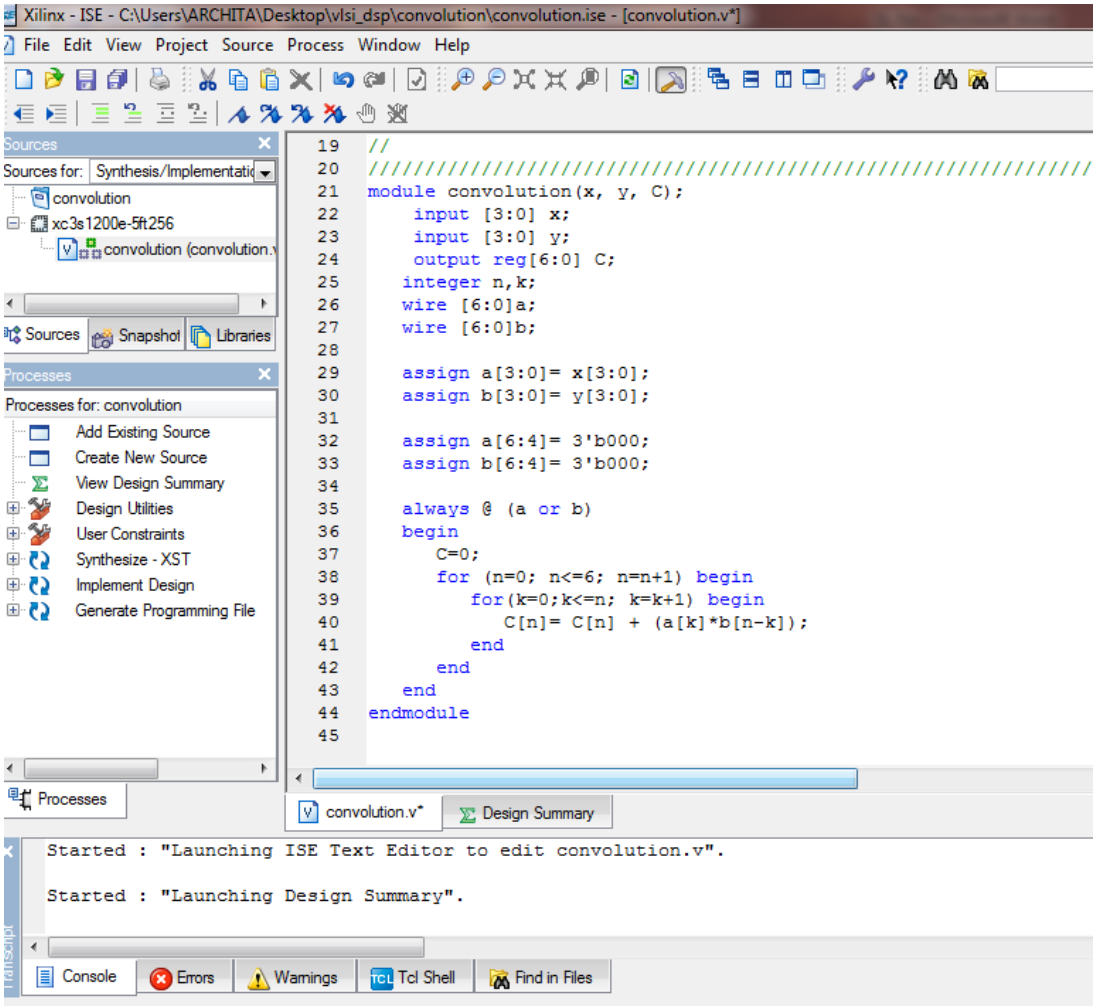
5. Declare the ports for the design by filling in the port information as shown below:




7. Click Next, then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.

8. Click Next, then Next, then Finish.

The source file containing the entity/architecture pair displays in the Workspace, and the design displays in the Source tab, as shown below (write the code after architecture … begin..):




## Checking the Syntax of the New Counter Module

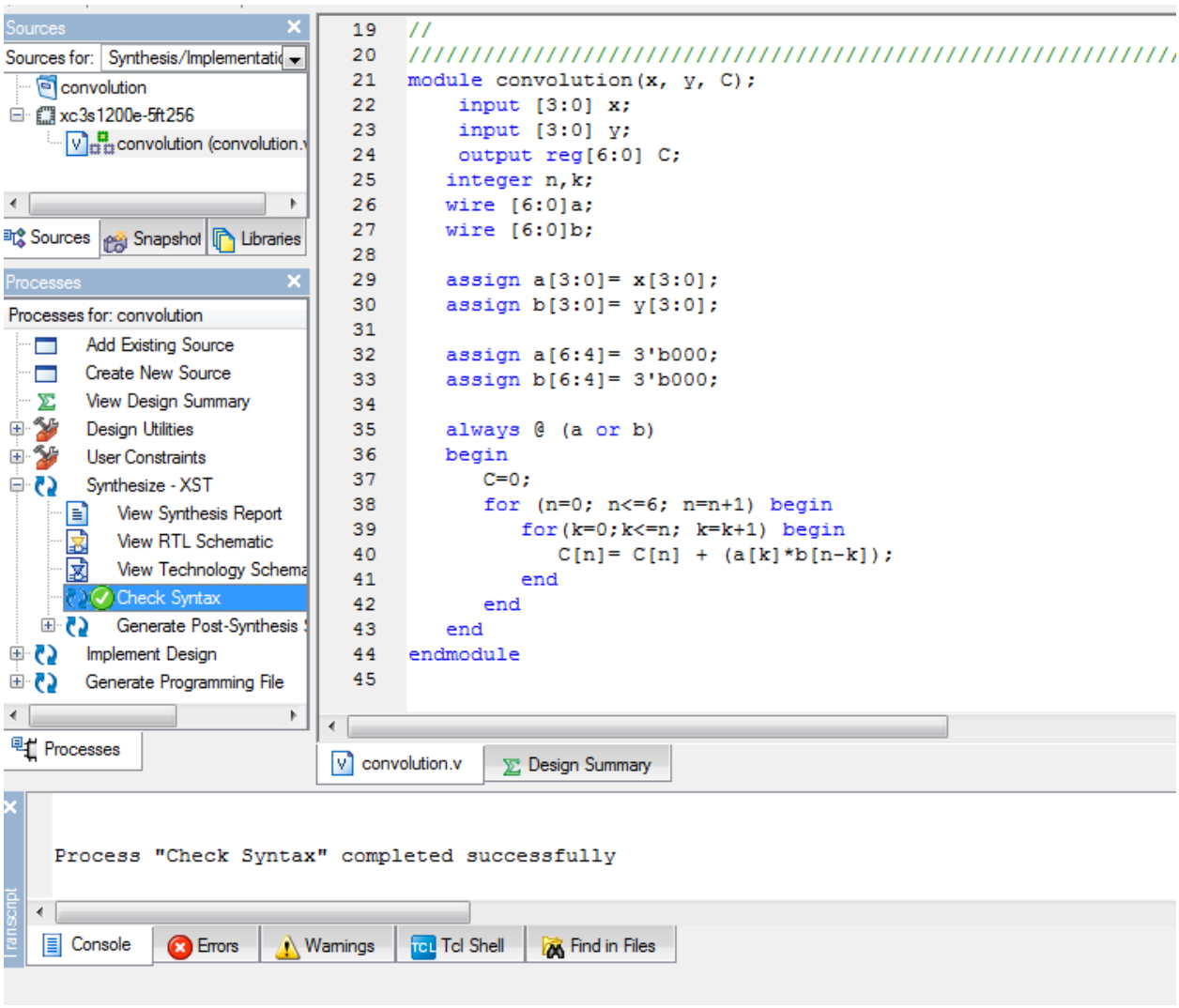
When the source files are complete, check the syntax of the design to find errors and typos.

1. Verify that Implementation is selected from the drop-down list in the Sources window.

2. Select the counter design source in the Sources window to display the related processes in the Processes window.

3. Click the "+" next to the Synthesize-XST process to expand the process group.

4. Double-click the Check Syntax process. Note: You must correct any errors found in your source files. You can check for errors in the Console tab of the Transcript window. If you continue without valid syntax, you will not be able to simulate or synthesize your design.




5. Close the HDL file.

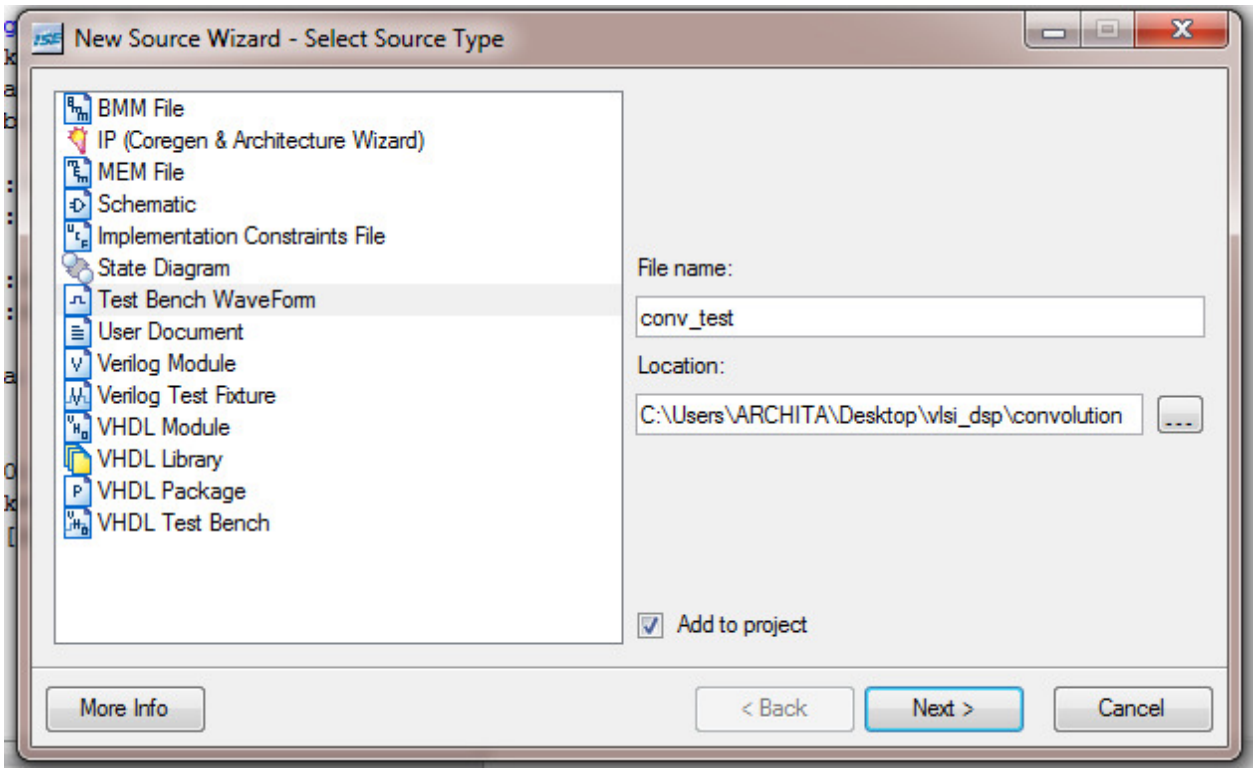
**Simulating Design Functionality**

For Verifying Functionality using Behavioral Simulation, create a test bench waveform program containing input stimulus you can use to verify the functionality of the design. The test bench waveform is a graphical view of a test bench. Create the test bench waveform as follows:

1. Select the designed HDL file in the Sources window.

2. Create a new test bench source by selecting Project → New Source.

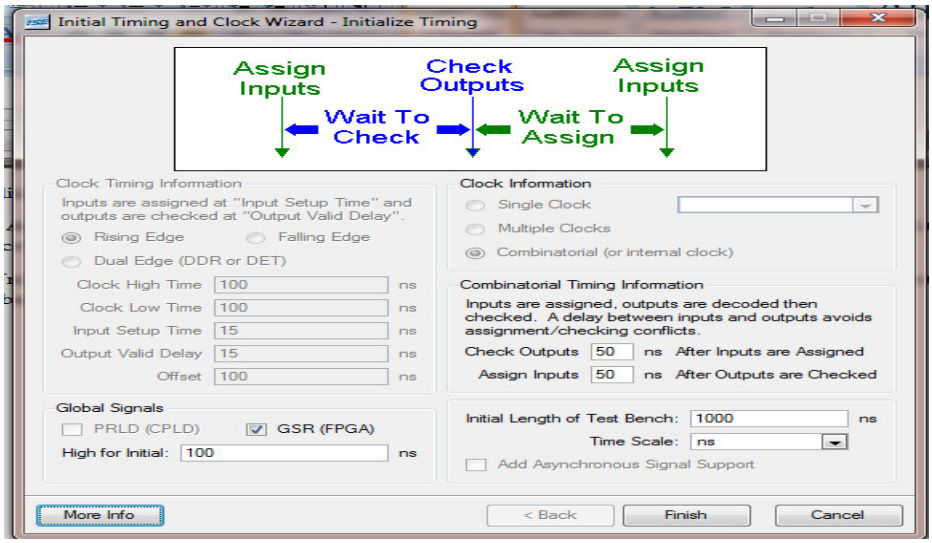
3. In the New Source Wizard, select Test Bench waveform as the source type, and type any name in the File Name field.



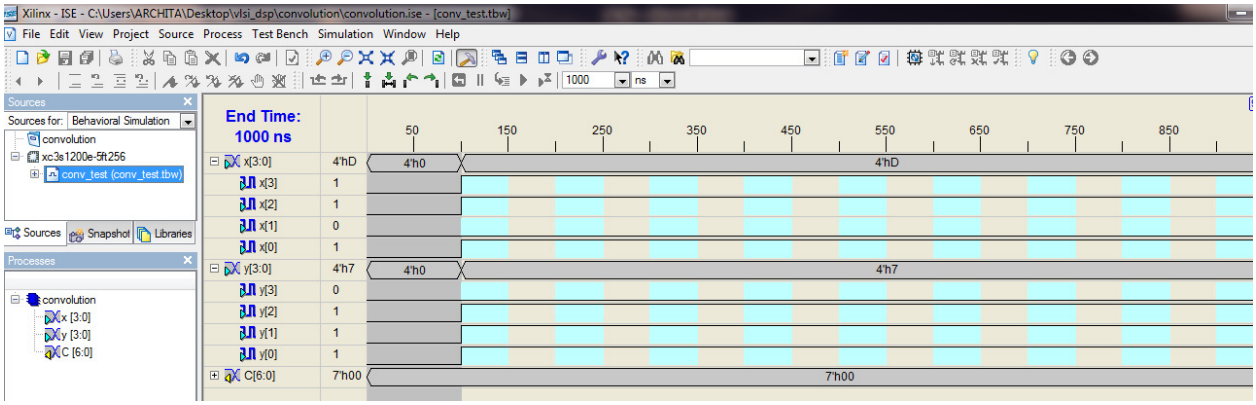

4. Click Next. 5.

The Associated Source page shows that you are associating the test bench waveform with the source file counter. Click Next → Next →Finish.

7. In the Initial timing and clock wizard window → Click Finish



8.

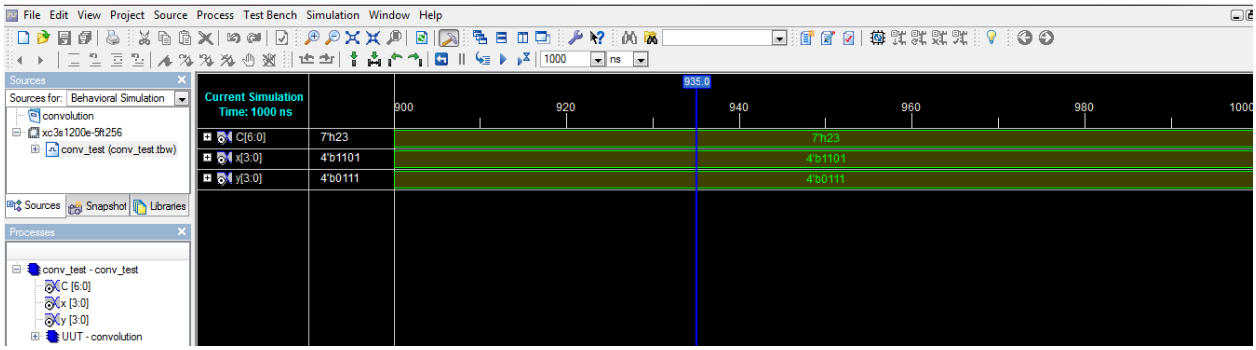



After getting the above window → declare the values for the inputs manually by clicking on the blue tabs → Then save the file

8. In the sources tab, from the pop down window, make it Behavioral simulation → Then click on the file_name.tbw

9. In the processes tab → expand the Xilinx ISE simulator → Double Click on Simulate Behavioral Model.

10. To view your simulation results, select the Simulation tab and zoom in on the transitions.




12. Verify that the design is running as expected.

13. Close the simulation view. If you are prompted with the following message, "You have an active simulation open. Are you sure you want to close it?, click Yes to continue. You have now completed simulation of your design using the ISE Simulator.

**SL No. 1**

**Aim: Write a verilog code to perform convolution of X and Y (both having 8 bits).**

**Theory: 2/3 lines theory. Give the expression also.**

**Program:** Refer to the example below

**Test Bench waveform:**

**Conclusion:** (Give proper justification w.r.t the test bench waveform)

**Viva Question**

1. **What is convolution?**

**Useful links:**

1. **https://en.wikipedia.org/wiki/Convolution**
2. **http://www.dspguide.com/ch6.htm**

**Example** : **Write a verilog code to perform convolution of X and Y (both having 4 bits).**

**Theory:** Convolution is the process by which an input interacts with another system or an input, to produce an output. Convolution between of an input signal x[n] with another I/P y[n] is given as,

$$C[n] = x[n] * y[n] = \sum_{k=0}^{n} x[k] \quad y[n-k]$$

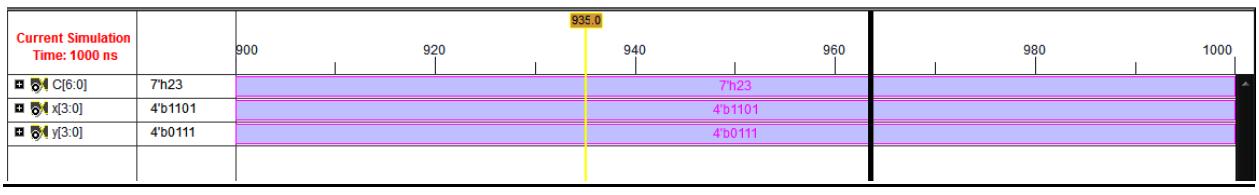
where * denotes the convolution

**Program:**

```
module convolution(x, y, C);
  input [3:0] x;
  input [3:0] y;
  output reg[6:0] C;
```

```
        integer n,k;
        wire [6:0]a;
        wire [6:0]b;
        assign a[3:0]= x[3:0];
        assign b[3:0]= y[3:0];
        assign a[6:4]= 3'b000;
        assign b[6:4]= 3'b000;

        always @ (a or b)
        begin
                C=0;
                for (n=0; n<=6; n=n+1) begin
                        for(k=0;k<=n; k=k+1) begin
                                C[n]= C[n] + (a[k]*b[n-k]);
                        end
                end
        end
endmodule
```

**Test Bench waveform:**




A

Fig: Test Bench waveform

**Conclusion:** In the test bench waveform, observing the line segment A. we have x=4'b1101 and

Y=4'b0111. Therefore, C is obtained as follows :

C[0]=x[0] and y[0]= 1 and 1=1

C[1]=(x[0] and y[1]) xor (x[1] and y[0]) = (1 and 1) xor (1 and 1) = 1 xor 1 = 1

C[2]=(x[0] and y[2]) xor (x[1] and y[1]) xor (x[2] and y[0]) = (1 and 1) xor (0 and 1) xor (1 and 1)

= 1 xor 0 xor 1= 0

C[3]=(x[0] and y[3]) xor (x[1] and y[2]) xor (x[2] and y[1]) xor (x[3] and y[0])

= (1 and 0) xor (0 and 1) xor (1 and 1) xor (1 and 1) = 0 xor 0 xor 1 xor 1= 0

C[4]= (x[1] and y[3]) xor (x[2] and y[2]) xor (x[3] and y[1]) = (0 and 0) xor (1 and 1) xor (1 and 1)

= 0 xor 1 xor 1= 0

C[5]=(x[2] and y[3]) xor (x[3] and y[2]) = (1 and 0) xor (1 and 1) = 0 xor 1 = 1

C[6]=x[3] and y[3]= 1 and 0=0

Therefore, C =7'b0100011=7'h23. Thus, the above waveform is verified.

**SL No. 2**

**Aim:** Write a verilog code to perform correlation between X and Y **(both having 8 bits).**

**Theory: 2/3 lines theory. Give the expression also.**

**Program:**

Steps:

1. Declare the input and output ports specifying the number of bits.
2. Run the loop according to the expression.

**Test Bench waveform:**

**Conclusion:** (Give proper justification w.r.t the test bench waveform)

**Viva Question**

1. **What is correlation?**

**Useful links:**

1. **https://www.surveysystem.com/correlation.htm**

**SL No. 3**

**Aim:** Write a verilog code for convolution encoding**.**

**Theory: 2/3 lines theory. Give the expression also.**

**Program:**

Steps:

1. Declare the input and output ports specifying the number of bits.
2. Run the loop according to the expression.

**Test Bench waveform:**

**Conclusion:** (Give proper justification w.r.t the test bench waveform)

**Viva Question**

1. **What is** convolution encoding**?**

**Useful links:**

1. https://en.wikipedia.org/wiki/Convolutional_code

**SL No. 4**

**Aim:** Write a verilog code on moving average process.

**both having 8 bits).**

**Theory: 2/3 lines theory. Give the expression also.**

**Program:**

Steps:

1. Declare the input and output ports specifying the number of bits.
2. Run the loop according to the expression.

**Test Bench waveform:**

**Conclusion:** (Give proper justification w.r.t the test bench waveform)

**Viva Question**

1. **What is a** moving average process?

**Useful links:**

1. https://en.wikipedia.org/wiki/Moving-average_model

**SL No. 5**

**Aim:** Write a verilog code for FIR filtering.

**Theory: 2/3 lines theory. Give the expression also.**

**Program:**

Steps:

1. Declare the input and output ports specifying the number of bits.
2. Run the loop according to the expression.

**Test Bench waveform:**

**Conclusion:** (Give proper justification w.r.t the test bench waveform)

**Viva Question**

1. What is FIR filtering?
2. Give the differences between FIR and IIR filtering.

**Useful links:**

1. https://en.wikipedia.org/wiki/Finite_impulse_response

**SL No. 6**

**Aim:** Design a Moore FSM using overlapping method.

**Theory: 2/3 lines theory. Give the state table and the state diagram.**

**Program:**

Steps:

1. Declare the input and the output ports.
2. Write the program w.r.t the state diagram.

**Test Bench waveform:**

**Conclusion:** (Give proper justification of the test bench waveform w.r.t the state table)

**Viva Question**

1. What is an FSM?
2. Define Moore FSM.

**Suggested Reading:**

1. https://en.wikipedia.org/wiki/Moore_machine
2. Refer to any good digital electronics book (e.g. **Anand Kumar**)

**SL No. 7**

**Aim:** Design a Mealy FSM using overlapping method.

**Theory: 2/3 lines theory. Give the state table and the state diagram.**

**Program:**

Steps:

1. Declare the input and the output ports.
2. Write the program w.r.t the state diagram.

**Test Bench waveform:**

**Conclusion:** (Give proper justification of the test bench waveform w.r.t the state table)

**Viva Question**

1. What is a Mealy FSM?
2. Difference between Moore and Mealy FSM.

**Suggested Reading:**

1. https://en.wikipedia.org/wiki/Mealy_machine
2. Refer to any good digital electronics book (e.g. **Anand Kumar**)

**SL No. 8**

**Aim:** Design a Mealy FSM using non-overlapping method.

**Theory: 2/3 lines theory. Give the state table and the state diagram.**

**Program:**

Steps:

1. Declare the input and the output ports.
2. Write the program w.r.t the state diagram.

**Test Bench waveform:**

**Conclusion:** (Give proper justification of the test bench waveform w.r.t the state table)

**Viva Question**

1. Difference between overlapping method and non-overlapping method of a FSM.

**Suggested Reading:**

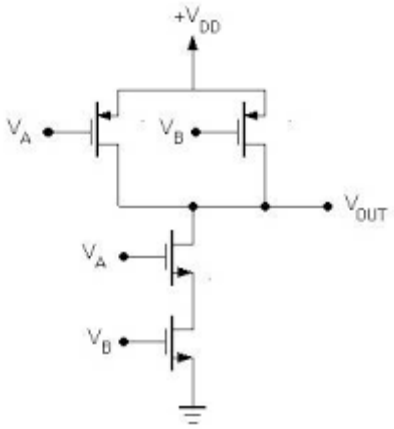
3. https://en.wikipedia.org/wiki/Mealy_machine
4. Refer to any good digital electronics book (e.g. **Anand Kumar**)

**SL No. 9**

**Aim:** Design a NAND gate using CMOS logic.

**Theory: 2/3 lines theory. Give the expression also.**

**Circuit diagram:**




**Truth table**

**Program:** Refer to the example below

**Test Bench waveform:**

**Conclusion:** (Give proper justification w.r.t the truth table)

**Viva Question**

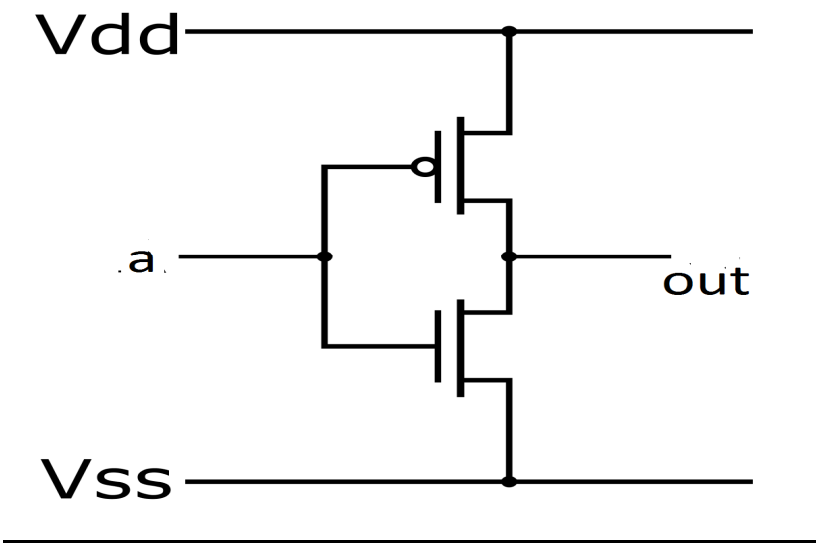
1. What is CMOS logic?

**Suggested Reading:**

1. http://www.async.caltech.edu/~cs181/docs/notes/Chapter2.pdf

**Example: Write a verilog code to design an inverter using switching logic.**

**Logic Diagram**




**Program:**

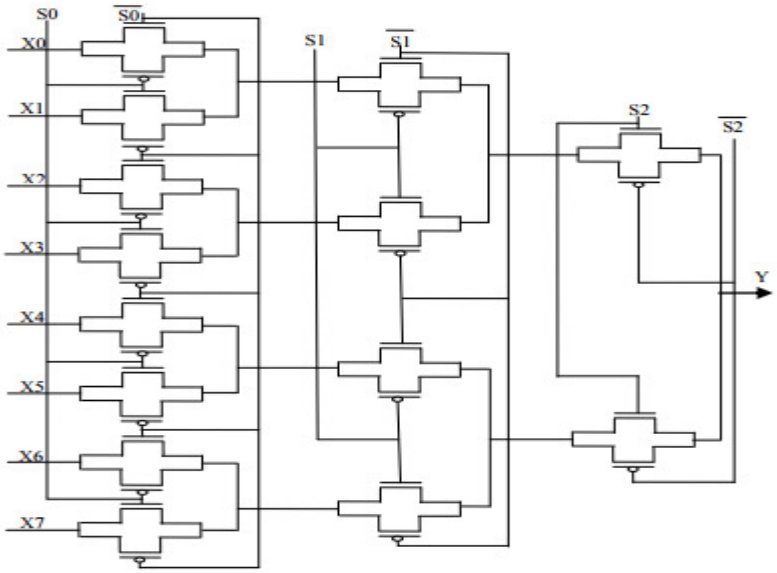
```
module cmos_inv(out,a);
   input a;
   output out;
         supply1 Vdd;
         supply0 Vss;

         pmos a1(out,Vdd,a);
         nmos a2(out,Vss,a);
endmodule
```

**SL No. 10**

**Aim:** Design a 8:1 MUX using switching logic.

**Theory: 2/3 lines theory. Give the expression also.**

**Logic diagram:**




**Truth table:**

**Program:**

Steps:

1. Declare the input and the output ports.
2. Write the program wrt the logic diagram.

**Test Bench waveform:**

**Conclusion:** (Give proper justification w.r.t the truth table)

**Viva Question**

1. Application of a multiplexer.

**Suggested Reading:**

1. https://en.wikipedia.org/wiki/Multiplexer