VHDL LAB MANUAL

VI SEMESTER B.E (E&C)

(For private circulation only)

VISHVESHWARAIAH TECHNOLOGICAL UNIVERSITY

NAME.....

REG NO.....

BATCH.....

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY

MARLUR, TUMKUR-572105

CONTENTS

XILINX-ISE TUTORIALS

ISE Quick Start Tutorial

Getting Started

Starting the ISE Software

For Windows users, start ISE from the Start menu by selecting:

Start _ Programs _ Xilinx ISE 7 _ Project Navigator

The ISE Project Navigator opens. The Project Navigator lets you manage the sources and processes in your ISE project. All of the tasks in the Quick Start Tutorial are managed from within Project Navigator.

Stopping and Restarting a Session

At any point during this tutorial you can stop your session and continue at a later time.
To stop the session:

- Save all source files you have opened in other applications.
- Exit the software (ISE and other applications).

The current status of the ISE project is maintained when exiting the software.
To restart your session, start the ISE software again. ISE displays the contents and state of your project with the last saved changes.

Accessing Help

At any time during the tutorial, you can access online help for additional information about a variety of topics and procedures in the ISE software as well as related tools.
To open Help you may do either of the following:

- Press **F1** to view Help for the specific tool or function that you have selected or highlighted.
- Launch the **ISE Help Contents** from the Help menu. It contains information about creating and maintaining your complete design flow in ISE.

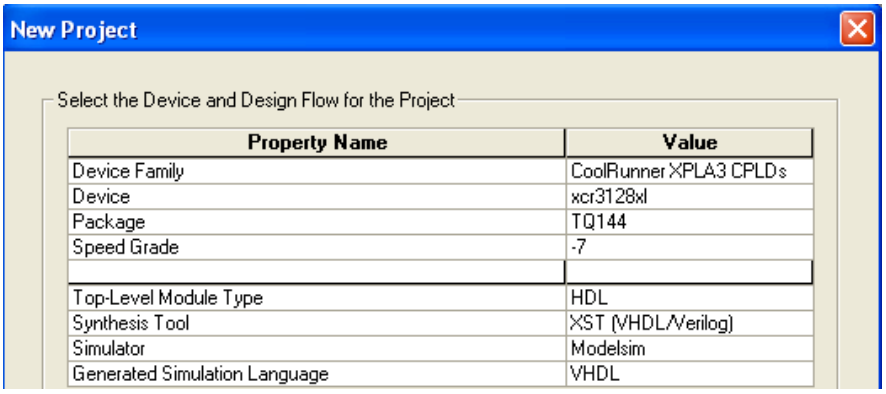
Creating a New Project in ISE

In this section, you will create a new ISE project. A project is a collection of all files necessary to create and to download a design to a selected FPGA or CPLD device.

To create a new project for this tutorial:

1. Select **File > New Project**. The New Project Wizard appears.
2. First, enter a location (directory path) for the new project.
3. Type **tutorial** in the Project Name field. When you type **tutorial** in the Project Name field, a tutorial subdirectory is created automatically in the directory path you selected.
4. Select **HDL** from the Top-Level Module Type list, indicating that the top-level file in your project will be HDL, rather than Schematic or EDIF.
5. Click **Next** to move to the project properties page.
6. Fill in the properties in the table as shown below
   Device Family: **CoolRunner XPLA3 CPLDs**
   Device: **xcr3128xl**
   Package: **TQ144**
   Speed Grade: **-7**
   Top-Level Module Type: **HDL**
   Synthesis Tool: **XST (VHDL/Verilog)**
   Simulator: **ModelSim**
   Generated Simulation Language: **VHDL** or **Verilog**, depending on the language you want to use when running behavioral simulation.

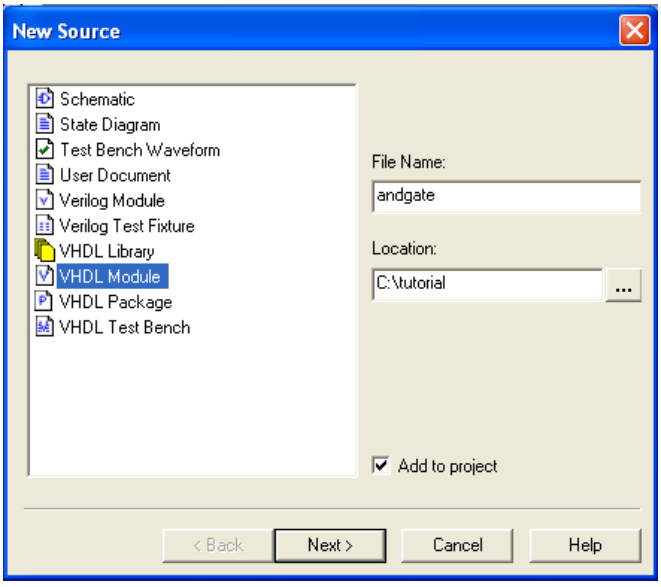
When the table is complete, your project properties should look like the following:



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 created.

Creating an HDL Source

In this section, you will create a top-level HDL file for your design. Determine the language that you wish to use for the tutorial. Then, continue either to the "Creating a VHDL Source" section below.

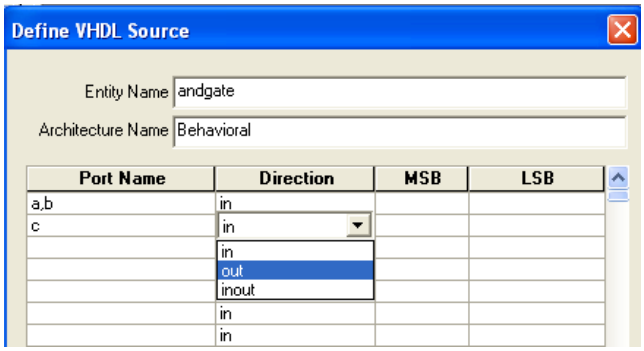


This simple AND Gate design has two inputs: A and B. This design has one output called C

1. Click **New Source** in the New Project Wizard to add one new source to your project.
2. Select **VHDL Module** as the source type in the New Source dialog box.
3. Type in the file name **andgate**.
4. Verify that the **Add to project** checkbox is selected.
5. Click **Next**.
6. Define the ports for your VHDL source.

   In the Port Name column, type the port names on three separate rows: **A**, **B** and **C**.

   In the Direction column, indicate whether each port is an input, output, or inout. For A and B, select **in** from the list. For C, select **out** from the list.



7. Click **Next** in the Define VHDL Source dialog box.
8. Click **Finish** in the New Source Information dialog box to complete the new source file template.
9. Click **Next** in the New Project Wizard.
10. Click **Next** again.
11. Click **Finish** in the New Project Information dialog box.

    ISE creates and displays the new project in the Sources in Project window and adds the andgate.vhd file to the project.
12. Double-click on the **andgate.vhd** file in the Sources in Project window to open the VHDL file in the ISE Text Editor.

    The andgate.vhd file contains:

    - Header information.
    - Library declaration and use statements.
    - Entity declaration for the counter and an empty architecture statement.
13. In the header section, fill in the following fields:

    Design Name: **andgate.vhd**

    Project Name: **andgate**

    Target Device: **xcr3128xl- TQ144**

    Description: **This is the top level HDL file for an up/down counter.**

    Dependencies: **None**

    Note: It is good design practice to fill in the header section in all source files.
14. Below the `end process` statement, enter the following line:

```
C <= A and B;
```

15. Save the file by selecting **File > Save**.

Checking the Syntax of the New Counter Module

When the source files are complete, the next step is to check the syntax of the design. Syntax errors and typos can be found using this step.

1. Select the **counter** design source in the ISE Sources window to display the related processes in the Processes for Source window.
2. Click the "**+**" next to the Synthesize-XST process to expand the hierarchy.
3. Double-click the **Check Syntax** process.

When an ISE process completes, you will see a status indicator next to the process name.

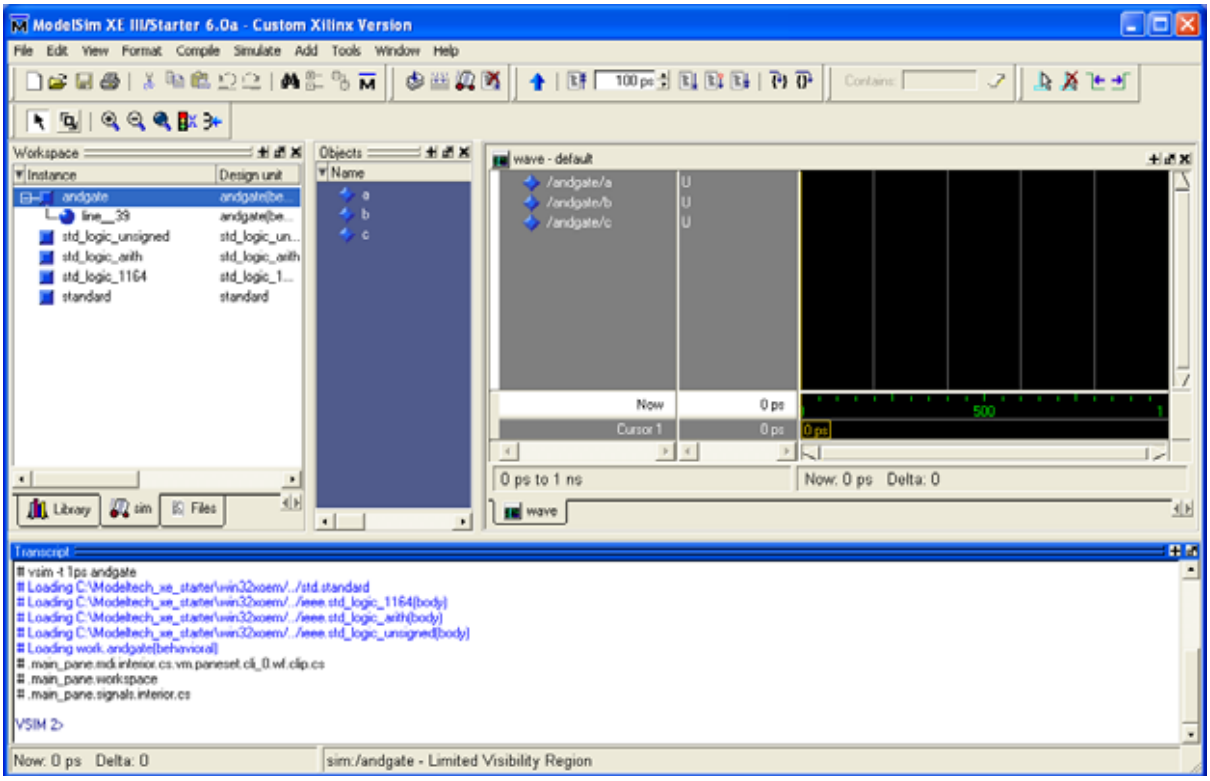
- If the process completed successfully, a green check mark appears.
- If there were errors and the process failed, a red X appears.
- A yellow exclamation point means that the process completed successfully, but some warnings occurred.
- An orange question mark means the process is out of date and should be run again.

4. Look in the **Console** tab of the Transcript window and read the output and status messages produced by any process that you run.

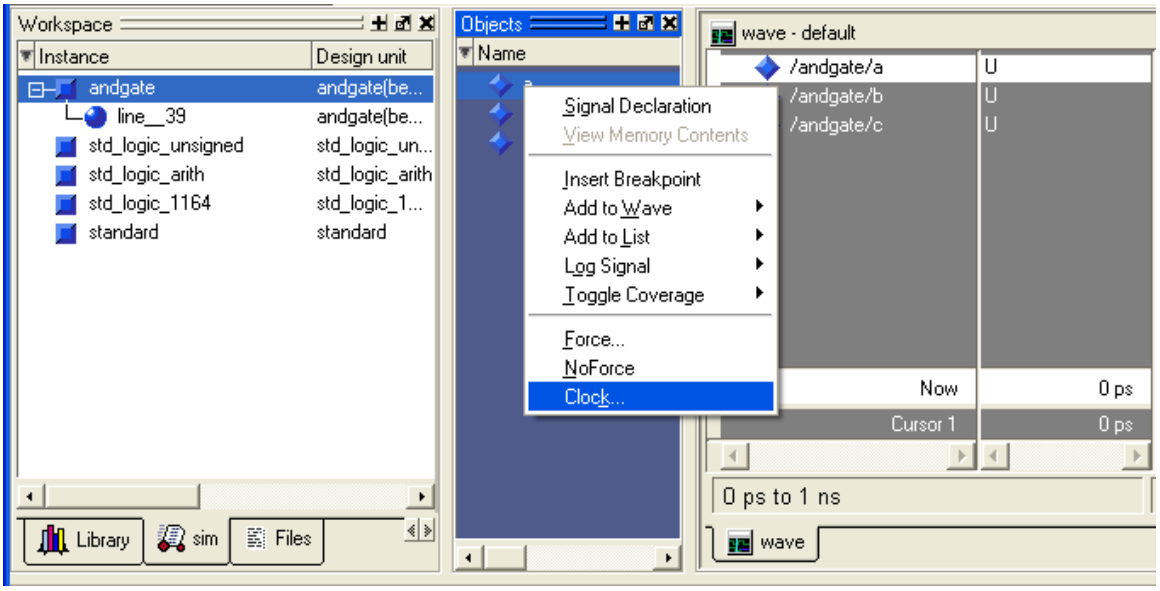
Caution! You must correct any errors found in your source files. If you continue without valid syntax, you will not be able to simulate or synthesize your design.

Simulation

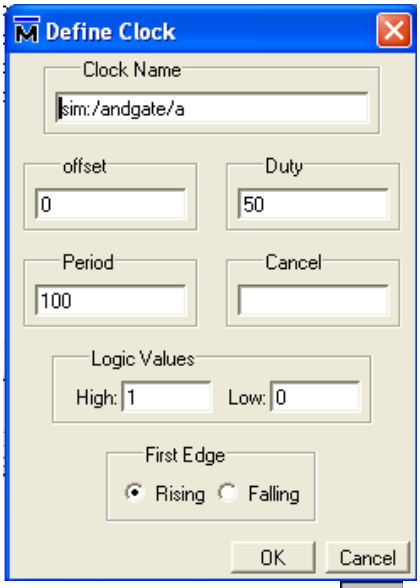
1. Double click Launch ModelSim Simulator in the Process View window.



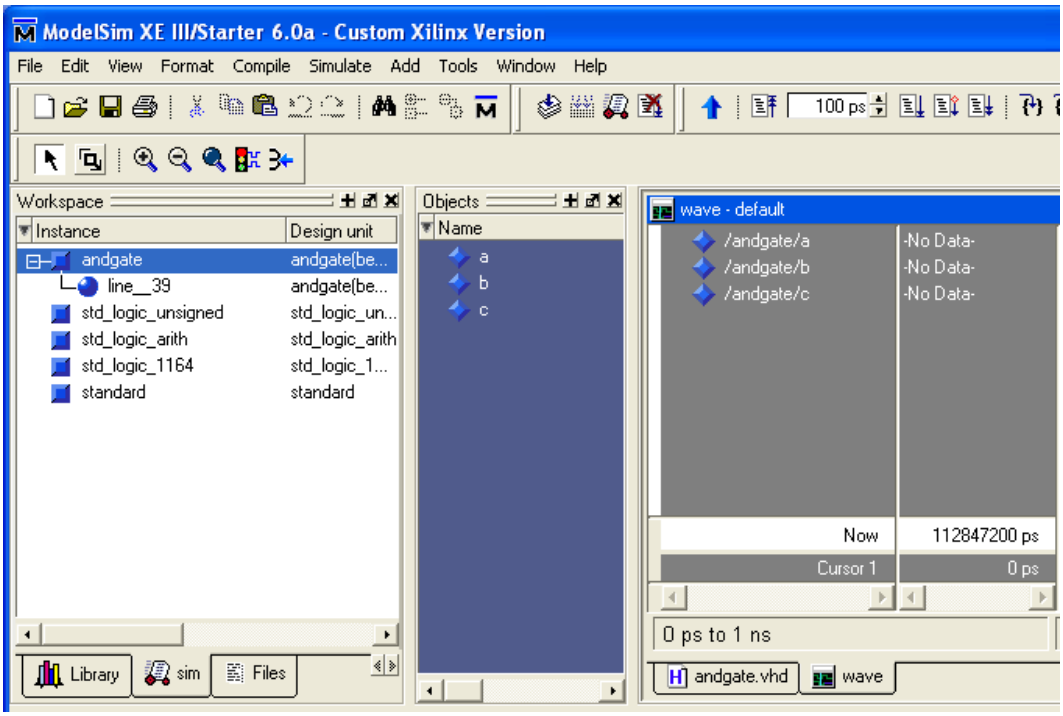
2. Right Click '*a*' to open a context menu.
3. Select Force or Clock to add the signal.



4. Define the Clock or Force signal to load appropriate signal

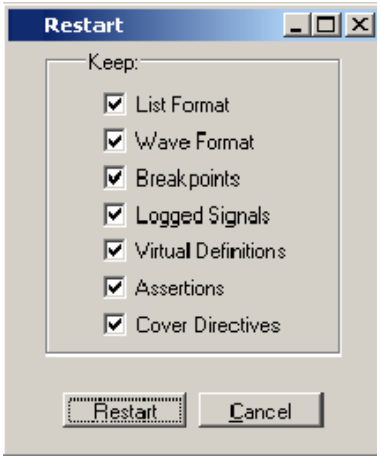


5. Run the simulation by clicking the Run icon in the Main or Wave window toolbar. 

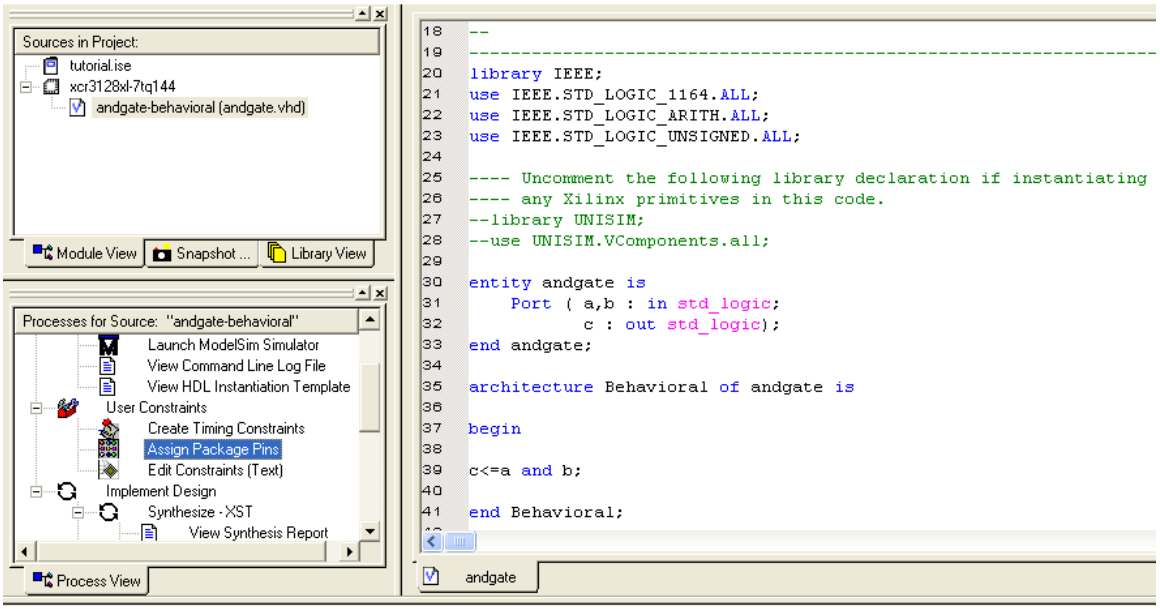


6. Waveform can be observed in the wave window

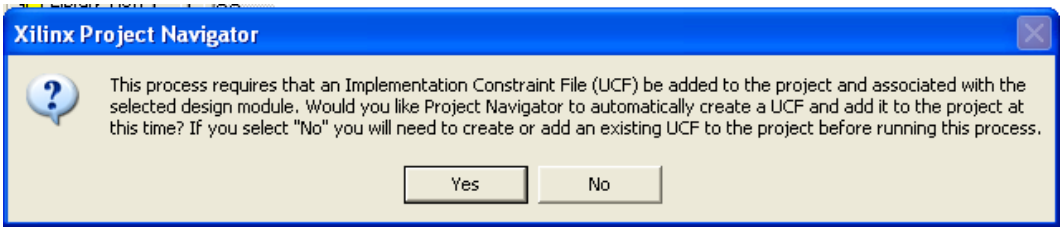
7. Click the Run -All icon on the Main or Wave window toolbar. The simulation continues running until you execute a break command. 
8. Click the Break icon. The simulation stops running. 
9. To restart the simulation, click the Restart icon to reload the design elements and reset the simulation time to zero. The Restart dialog that appears gives you options on what to retain during the restart.  Click the Restart button in the Restart dialog.



Assigning Pin Location



1. Double-click the **Assign Package Pins** process found in the User Constraints process group. ISE runs the Synthesis and Translate steps and automatically creates a User Constraints File (UCF). You will be prompted with the following message:



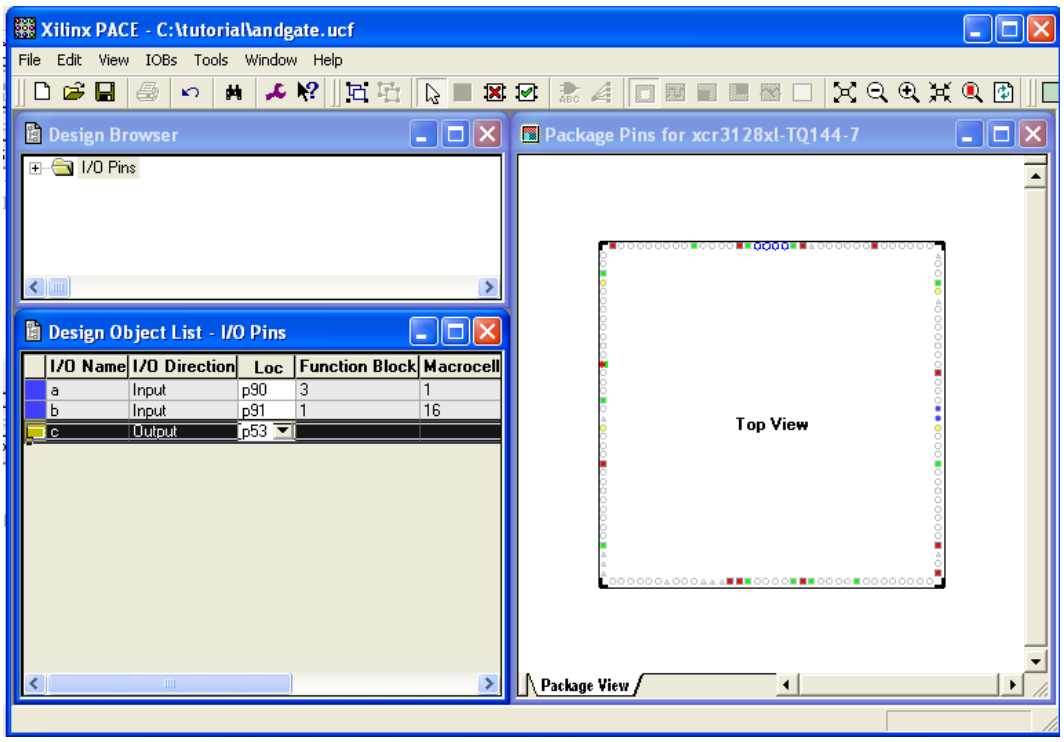
2. Click **Yes** to add the UCF file to your project. The `counter.ucf` file is added to your project and is visible in the Sources in Project window. The Xilinx Constraints Editor opens automatically.
3. Now the Xilinx Pinout and Area Constraints Editor (PACE) opens.
4. You can see your I/O Pins listed in the Design Object List window. Enter a pin location for each pin in the Loc column as specified below:

   A: p90

   B: p91

   C: p53

5. Click on the **Package View** tab at the bottom of the window to see the pins you just added. Put your mouse over grid number to verify the pin assignment.



5. Select **File _ Save**. You are prompted to select the bus delimiter type based on the synthesis tool you are using. Select **XST Default <>** and click **OK**.
6. Close PACE.

Creating Configuration Data

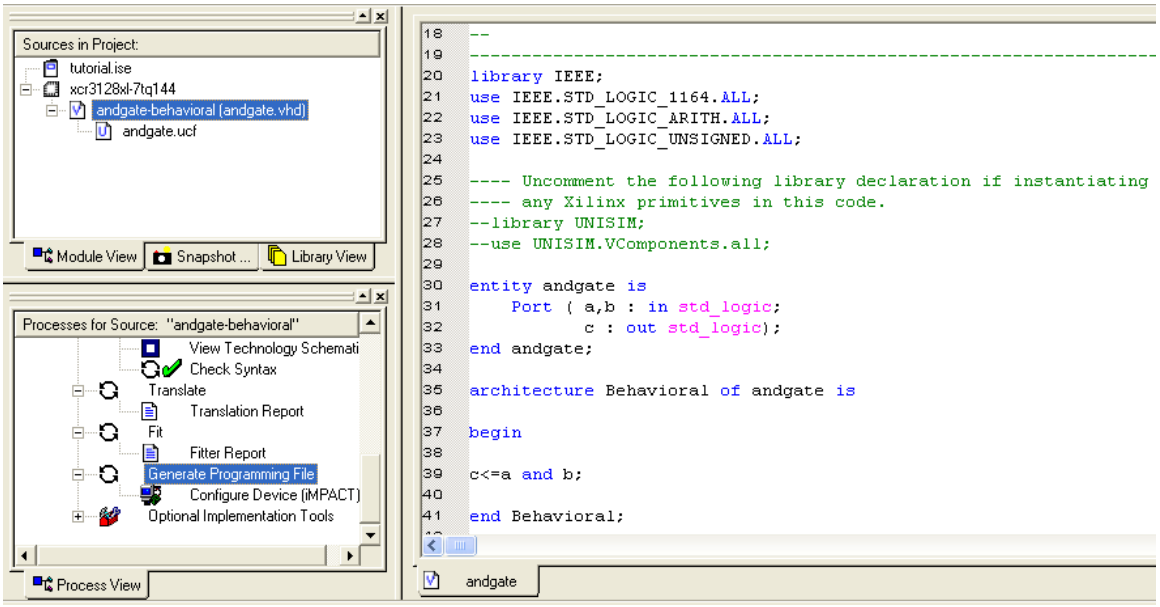
The final phase in the software flow is to generate a program file and configure the device.

Generating a Program File

The Program File is a encoded file that is the equivalent of the design in a form that can be downloaded into the CPLD device.

1. Double Click the **Generate Programming File** process located near the bottom of the Processes for Source window.

   The Program File is created. It is written into a file called `andgate.jed`. This is the actual configuration data.



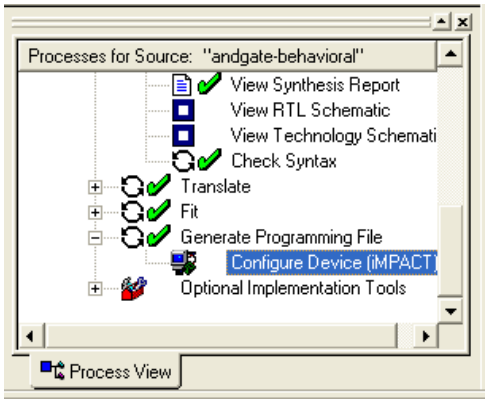
Configuring the Device

iMPACT is used to configure your FPGA or CPLD device. This is the last step in the design process. This section provides simple instructions for configuring a Spartan-3 xc3s200 device connected to your PC.

Note: Your board must be connected to your PC before proceeding. If the device on your board does not match the device assigned to the project, you will get errors. Please refer to the iMPACT Help for more information. To access the help, select **Help > Help Topics**.

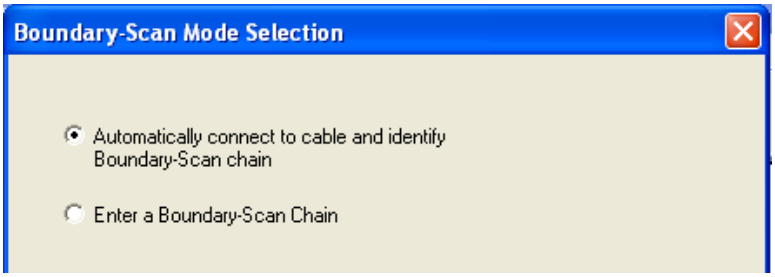
To configure the device:

1. Click the "**+**" sign to expand the **Generate Programming File** processes.



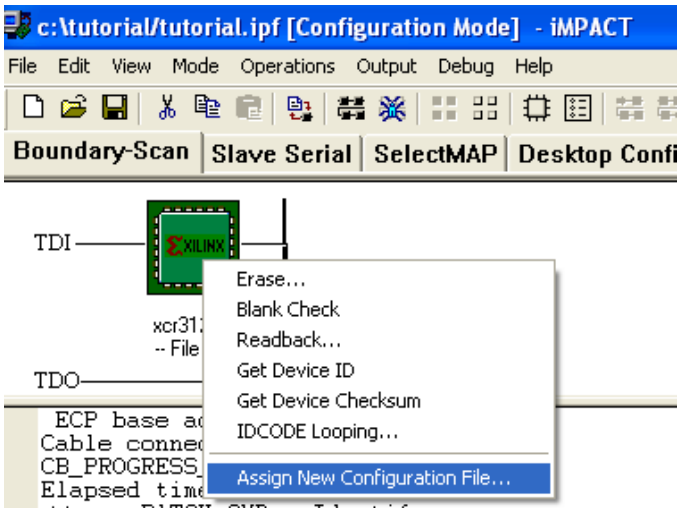
2. Double-click the **Configure Device (iMPACT)** process. iMPACT opens and the Configure Devices dialog box is displayed.

3. In the Configure Devices dialog box, verify that **Boundary-Scan Mode** is selected and click **Next**.
4. Verify that **Automatically connect to cable and identify Boundary-Scan chain** is selected and click **Finish**.



5. If you get a message saying that there was one device found, click **OK** to continue.

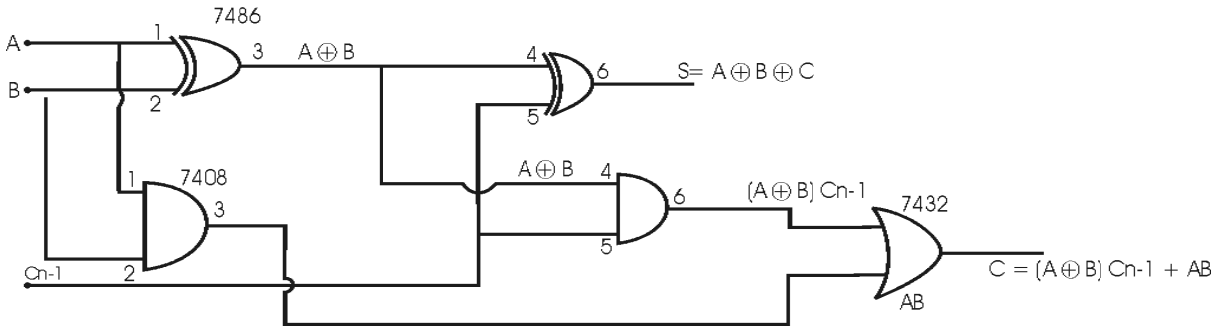
6. The iMPACT will now show the detected device, right click the device and select **New Configuration File.**



7. The **Assign New Configuration File** dialog box appears. Assign a configuration file to each device in the JTAG chain. Select the `andgate.jed` file and click **Open**.
8. Right-click on the counter device image, and select **Program...** to open the **Program Options** dialog box.
9. Click **OK** to program the device. ISE programs the device and displays Programming Succeeded if the operation was successful.
10. Close iMPACT without saving.

PROGRAMS

1. VHDL CODE FOR FULL ADDER DATA FLOW:



Full Adder				
A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

EXPRESSIONS:

$S = A \oplus B \oplus Ci$

$CO = (A \oplus B)Ci + AB$

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity fa1 is
          Port ( a,b,ci : in  STD_LOGIC; s,co : out  STD_LOGIC);
        end fa1;
architecture Behavioral of fa1 is
begin
        s<=a xor b xor ci;
        co<=(a and b)or (b and ci)or (ci and a);
end Behavioral;
```

VHDL CODE FOR FULL ADDER BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity fa1 is
          Port ( a,b,ci : in  STD_LOGIC; s,co : out  STD_LOGIC);
        end fa1;
architecture Behavioral of fa1 is
begin
        process(a,b,ci)
begin
        s<=a xor b xor ci;
        co<=(a and b)or (b and ci)or (ci and a);
end process;
end Behavioral;
```

2. VHDL CODE FOR FULL ADDER STRUCTURAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity fa1 is
          Port ( a,b,cin : in  STD_LOGIC;
               s,cout : out  STD_LOGIC);
        end fa1;
architecture struct of fa1 is
        component and21
        port(a,b:in std_logic;          ---components, entity and architecture
        c:out std_logic);               --- must be declared separately
        end component;
                component xor21
                port(a,b:in std_logic;  ---components, entity and architecture
                c:out std_logic);       --- must be declared separately
                end component;
        component or31
        port(a,b:in std_logic;          ---components, entity and architecture
        d:out std_logic);               --- must be declared separately
        end component;
signal s1,s2,s3:std_logic;
begin
u1:xor21 port map(a,b,s1);
u2:xor21 port map(s1,cin,s);
u3:and21 port map(a,b,s2);
u4:and21 port map(s1,cin,s3);
u6:or31 port map(s2,s3,cout);
end struct;
```

3. VHDL CODE FOR MULTIPLEXER (8:1):

INPUTS								SELECT LINES			O/P
d(7)	d(6)	d(5)	d(4)	d(3)	d(2)	d(1)	d(0)	s(2)	s(1)	s(0)	f
X	X	X	X	X	X	X	0	0	0	0	0
X	X	X	X	X	X	X	1	0	0	0	1
X	X	X	X	X	X	0	X	0	0	1	0
X	X	X	X	X	X	1	X	0	0	1	1
X	X	X	X	X	0	X	X	0	1	0	0
X	X	X	X	X	1	X	X	0	1	0	1
X	X	X	X	0	X	X	X	0	1	1	0
X	X	X	X	1	X	X	X	0	1	1	1
X	X	X	0	X	X	X	X	1	0	0	0
X	X	X	1	X	X	X	X	1	0	0	1
X	X	0	X	X	X	X	X	1	0	1	0
X	X	1	X	X	X	X	X	1	0	1	1
X	0	X	X	X	X	X	X	1	1	0	0
X	1	X	X	X	X	X	X	1	1	0	1
0	X	X	X	X	X	X	X	1	1	1	0
1	X	X	X	X	X	X	X	1	1	1	1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity mux1 is
           Port ( d : in  STD_LOGIC_VECTOR (7 downto 0);
                s : in  STD_LOGIC_VECTOR (2 downto 0);
                f : out  STD_LOGIC);
        end mux1;
architecture Behavioral of mux1 is
begin
        f<= d(0) when s="000" else
        d(1) when s="001" else
        d(2) when s="010" else
        d(3) when s="011" else
        d(4) when s="100" else
        d(5) when s="101" else
        d(6) when s="110" else
        d(7) when s="111";
end Behavioral;
```

4. VHDL CODE FOR Demultiplexer (1:8):

SELECT LINES			I/P	OUTPUT							
s(2)	s(1)	s(0)	f	d(7)	d(6)	d(5)	d(4)	d(3)	d(2)	d(1)	d(0)
0	0	0	0	X	X	X	X	X	X	X	0
0	0	0	1	X	X	X	X	X	X	X	1
0	0	1	0	X	X	X	X	X	X	0	X
0	0	1	1	X	X	X	X	X	X	1	X
0	1	0	0	X	X	X	X	X	0	X	X
0	1	0	1	X	X	X	X	X	1	X	X
0	1	1	0	X	X	X	X	0	X	X	X
0	1	1	1	X	X	X	X	1	X	X	X
1	0	0	0	X	X	X	0	X	X	X	X
1	0	0	1	X	X	X	1	X	X	X	X
1	0	1	0	X	X	0	X	X	X	X	X
1	0	1	1	X	X	1	X	X	X	X	X
1	1	0	0	X	0	X	X	X	X	X	X
1	1	0	1	X	1	X	X	X	X	X	X
1	1	1	0	0	X	X	X	X	X	X	X
1	1	1	1	1	X	X	X	X	X	X	X

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

        entity dmux is
        port(f:in std_logic;
        s:in std_logic_vector(2 downto 0);
        y:out std_logic_vector(7 downto 0));
        end demux;

architectural behavioral of dmux is
begin
        y(0)<=f when s="000"else'0';
        y(1)<=f when s="001"else'0';
        y(2)<=f when s="010"else'0';
        y(3)<=f when s="011"else'0';
        y(4)<=f when s="100"else'0';
        y(5)<=f when s="101"else'0';
        y(6)<=f when s="110"else'0';
        y(7)<=f when s="111"else'0';
end behavioral;
```

5. VHDL CODE FOR ENCODER WITHOUT PRIORITY (8:3):

SELECT LINES			OUTPUT							
s(2)	s(1)	s(0)	y(7)	y(6)	y(5)	y(4)	y(3)	Y(2)	y(1)	y(0)
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	X	X	X	X	X	X	X	X

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity enco is
           Port ( i : in  STD_LOGIC_VECTOR (7 downto 0);
                y : out  STD_LOGIC_VECTOR (2 downto 0));
        end enco;
architecture Behavioral of enco is
begin
        with i select
        y<="000" when "00000001",
        "001" when "00000010",
        "010" when "00000100",
        "011" when "00001000",
        "100" when "00010000",
        "101" when "00100000",
        "110" when "01000000",
        "111" when others;
end Behavioral;
```

6. VHDL CODE FOR ENCODER WITH PRIORITY (8:3):

SELECT LINES			OUTPUT							
s(2)	s(1)	s(0)	y(7)	y(6)	y(5)	y(4)	y(3)	y(2)	y(1)	y(0)
0	0	0	0	0	0	0	0	1	1	1
0	0	1	0	0	0	0	0	1	1	0
0	1	0	0	0	0	0	0	1	0	1
0	1	1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	1	0
1	0	1	0-	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	1	0	0
1	1	1	X	X	X	X	X	X	X	X

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity enco1 is
           Port ( i : in  STD_LOGIC_VECTOR (7 downto 0);
                y : out  STD_LOGIC_VECTOR (2 downto 0));
        end enco1;
architecture Behavioral of enco1 is
begin
        with i select
        y<="000" when "00000111",
        "001" when "00000110",
        "010" when "00000101",
        "011" when "00000100",
        "100" when "00000011",
        "101" when "00000010",
        "110" when "00000001",
        "111" when others;
end Behavioral;
```

7. VHDL CODE FOR 3:8 DECODER:

SELECT LINES			OUTPUT							
s(2)	s(1)	s(0)	y(7)	y(6)	y(5)	y(4)	y(3)	y(2)	y(1)	y(0)
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0
X	X	X	0	0	0	0	0	0	0	0

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity dec1 is
           Port ( s : in  STD_LOGIC_VECTOR (2 downto 0);
                y : out  STD_LOGIC_VECTOR (7 downto 0));
        end dec1;
architecture Behavioral of dec1 is
begin
        with sel select
        y<="00000001" when "000",
        "00000010" when "001",
        "00000100" when "010",
        "00001000" when "011",
        "00010000" when "100",
        "00100000" when "101",
        "01000000" when "110",
        "10000000" when "111",
        "00000000" when others;
end Behavioral;
```

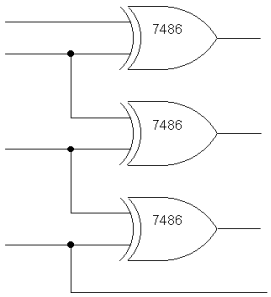
8. VHDL CODE FOR 2-bit comparator:

A1	A0	B1	B0	Y1 (A > B)	Y2 (A = B)	Y3 (A < B)
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity comp is
           Port ( a,b : in  STD_LOGIC_VECTOR (1 downto 0);
                y : out  STD_LOGIC_VECTOR (2 downto 0));
        end comp;
architecture Behavioral of comp is
begin
        y<= "100" when a>b else
        "001" when a<b else
        "010" when a=b;
end Behavioral;
```

9. VHDL CODE FOR Binary to gray (USING EXOR GATES):

CIRCUIT DIAGRAM:



Using EX-OR gates

EXPRESSIONS:

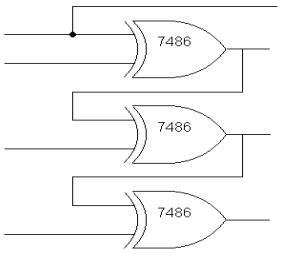
G(0)=B(0) ⊕ B(1)
G(1)=B(1) ⊕ B(2)
G(2)=B(2) ⊕ B(3)
G(3)=B(3)

Inputs				Outputs			
B (3)	B (2)	B (1)	B (0)	G (3)	G (2)	G (1)	G (0)
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity Binary_Gray is
        port(  b: in std_logic_vector(3 downto 0);          --Binary Input
             g: out std_logic_vector(3 downto 0));          --Gray Output
        end  binary_gray;
architecture behavioral of  Binary_gray is
begin
        b(3)<= g(3);
        b(2)<= g(3) xor g(2);
        b(1)<= g(2) xor g(1);
        b(0)<= g(1) xor g(0);
end behavioral;
```

10. VHDL CODE FOR GRAY TO BINARY:

CIRCUIT DIAGRAM:



Using EX-OR gates

EXPRESSIONS:

$B(0)=G(0) \oplus G(1) \oplus G(2) \oplus G(3)$
$B(1)=G(1) \oplus G(2) \oplus G(3)$
$B(2)=G(2) \oplus G(3)$
$B(1)=G(3)$

INPUT				OUTPUT			
G(3)	G(2)	G(1)	G(0)	B (3)	B (2)	B (1)	B (0)
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity gb1 is
           Port ( g : in  STD_LOGIC_VECTOR (3 downto 0);
                b : out  STD_LOGIC_VECTOR (3 downto 0));
        end gb1;
architecture Behavioral of gb1 is
begin
        b(3)<= g(3);
        b(2)<= g(3) xor g(2);
        b(1)<= g(3) xor g(2) xor g(1);
        b(0)<= g(3) xor g(2) xor g(1) xor g(0);
end behavioral;
```

11. VHDL CODE FOR JK FLIP FLOP WITH ASYNCHRONOUS RESET: (Master Slave JK Flip-Flop)

Reset	J	K	Clock	Qn+1	$\overline{Qn+1}$	Status
1	X	X	X	0	1	Reset
0	0	0	⎍	Qn	$\overline{Qn}$	No Change
0	0	1	⎍	0	1	Reset
0	1	0	⎍	1	0	Set
0	1	1	⎍	$\overline{Qn}$	Qn	Toggle

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity jkff1 is
          Port ( j,k,clk,reset : in  STD_LOGIC;
               Q : inout  STD_LOGIC);
        end jkff1;
architecture Behavioral of jkff1 is
        signal div:std_logic_vector(22 downto 0);
        signal clkd:std_logic;
begin
process(clk)
        begin
        if rising_edge(clk)then
        div<= div+1;
end if;
end process;
        clkd<=div(22);
process(clkd,reset)
begin
        if(reset='1')then
        Q<= '0';
        elsif(clkd'event and clkd='1')then
        if(j='0' and k='0')then
        Q<= Q;
        elsif(j='0' and k='1')then
        Q<= '0';
        elsif(j='1' and k='0')then
        Q<= '1';
        elsif(j='1' and k='1')then
        Q<= not Q;
end if;
end if;
end process;
end Behavioral;
```

12. VHDL CODE FOR T FLIP FLOP:

Clear	T	Clock	Qn+1	$\overline{Qn+1}$
1	0	0	0	$\overline{Qn}$
0	1	⎍	$\overline{Qn}$	Qn

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity tff is
           Port ( t,clk,rst : in  STD_LOGIC;
                q : inout  STD_LOGIC);
        end tff;
architecture Behavioral of tff is
        signal div:std_logic_vector(22 downto 0);
        signal clkd:std_logic;
begin
process(clk)
begin
        if rising_edge(clk)then
        div<= div+1;
end if;
end process;
        clkd<=div(20);
process(clkd,rst)
begin
        if(rst='1')then
        q<='0';
        elsif (clkd'event and clkd='1' and t='1') then
        q<= not q;
        else q<=q;
end if;
end process;
end Behavioral;
```

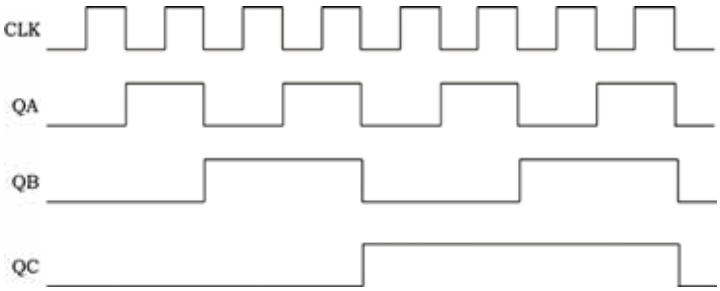
13. VHDL CODE FOR D FLIP FLOP:

Clear	D	Clock	Qn+1	$\overline{Qn+1}$
1	0	0	0	1
0	1	⎍	1	0

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity dff is
           Port ( d,res,clk : in  STD_LOGIC;
                q : out  STD_LOGIC);
        end dff;
architecture Behavioral of dff is
begin
process(clk)
begin
        if (res ='1')then q<='0';
        elsif clk'event and clk='1'
        then q<=d;
        end if;
end process;
end Behavioral;
```

14. ASYNCHRONOUS BINARY UP COUNTER:

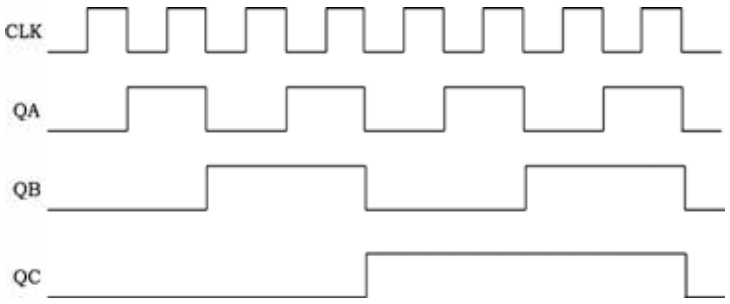
3-bit Asynchronous up counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0
9	0	0	1




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity asybin1 is
           Port ( rs,clk : in  STD_LOGIC;
                q : inout  STD_LOGIC_VECTOR (3 downto 0));
        end asybin1;
architecture Behavioral of asybin1 is
        signal div:std_logic_vector(22 downto 0);
        signal temp:STD_LOGIC_VECTOR (3 downto 0);
        signal clkd:std_logic;
begin
process(clk)
begin
        if rising_edge(clk)then
        div<= div+1;
        end if;
end process;
        clkd<=div(22);
process(clkd,rs)
begin
        if(rs='1')then temp<=(others=>'0');      --for down counter temp=>"1111";
        elsif(clkd='1' and clkd'event) then
        temp<=temp+1;                            --for down counter temp<= temp-1;
        q<= temp;
        end if;
end process;
end Behavioral;
```

15. SYNCHRONOUS BINARY UP COUNTER:

3-bit Asynchronous up counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0
9	0	0	1




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity synbicount is
           Port ( rs,clk : in  STD_LOGIC;
                q : inout  STD_LOGIC_VECTOR (3 downto 0));
        end synbicount;
architecture Behavioral of synbicount is
        signal div:std_logic_vector(22 downto 0);
        signal temp:STD_LOGIC_VECTOR (3 downto 0);
        signal clkd:std_logic;
begin
process(clk)
begin
        if rising_edge(clk)then
        div<= div+1;
        end if;
end process;
        clkd<=div(22);
process(clkd)
begin
        if(clkd='1' and clkd'event) then
        if(rs='1')then temp<=(others=>'0');  ---for down counter temp<="1111"
        else temp<=temp+1;                    ---for down counter temp<= temp-1;
        end if;
        q<= temp;
        end if;
end process;
end Behavioral;
```

16. BCD UP COUNTER:

Clock	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity bcdupcount is
        Port ( clk,rst : in  STD_LOGIC;
        q : inout  STD_LOGIC_VECTOR (3 downto 0));
        end bcdupcount;
architecture Behavioral of bcdupcount is
        signal div:std_logic_vector(22 downto 0);
        signal clkd:std_logic;
begin
process(clkd)
begin
        if rising_edge(clk)then
        div<= div+1;
        end if;
end process;
        clkd<=div(22);
process(clkd,rst)
begin
        if rst='0' or q="1010" then
        q<="0000";
        elsif clkd'event and clkd='1' then
        q<=q+1;
        end if;
end process;
         q<=q;
 end Behavioral;
```

17. BCD DOWN COUNTER:

Clock	QD	QC	QB	QA
0	1	0	0	1
1	1	0	0	0
2	0	1	1	1
3	0	1	1	0
4	0	1	0	1
5	0	1	0	0
6	0	0	1	1
7	0	0	1	0
8	0	0	0	1
9	0	0	0	0

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity bcddowncounter1 is
        Port ( clk,rst : in  STD_LOGIC;
        q : inout  STD_LOGIC_VECTOR (3 downto 0));
        end bcddowncounter1;
architecture Behavioral of bcddowncounter1 is
        signal div:std_logic_vector(22 downto 0);
        signal clkd:std_logic;
begin
process(clkd)
begin
        if rising_edge(clk)then
        div<= div+1;
        end if;
end process;
        clkd<=div(22);
process(clkd,rst)
begin
        if rst='0' or q="1111" then
        q<="1001";
        elsif clkd'event and clkd='1' then
        q<=q-1;
        end if;
end process;
        q<=q;
end Behavioral;
```

18. VHDL CODE FOR DC MOTOR INTERFACE

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;
        Entity dcmotor is
        Port (start,dir,clk:in std_logic;
                pwm_out:out std_logic;
                out_dc:out std_logic_vector(1 downto 0));
        end dcmotor;
architecture dcmotor_a of dcmotor is
        signal clk1:std_logic;
        signal div:std_logic_vector (24 downto 0);
begin
process (clk,start)
begin
        if(start='0') then
        div<="00000000000000000000000000";
        elsif(clk' event and clk='1')then
        div<=div+1;
        end if;
        clk1<=div(19);
end process;
process(clk1)
begin
        if(clk1'event and clk='1')then
        if start='0' then
        out_dc<="00";
        elsif start='1' and dir='1' then
        out_dc<="10";
        pwm_out<='1';
        elsif start='1' and dir='0' then
        out_dc<="01";
        pwm_out<='1';
        end if;
        end if;
end process;
end dcmotor_a;
```

PIN ASSIGNMENT

XC3128TQ144		
Connector	Device pin	Property
	128	Clk
P 18/6	6	Dir
P 18/13	14	Out_dc(0)
P 18/14	15	Out_dc(1)
P 18/11	11	Pwm_out
P 18/5	5	Start

XC2S100TQ144-5		
Connector	Device pin	Property
	18	Clk
P 18/6	44	Dir
P 18/13	54	Out_dc(0)
P 18/14	56	Out_dc(1)
P 18/11	50	Pwm_out
P 18/5	43	Start

19. VHDL CODE FOR STEPPER MOTOR INTERFACE

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;
        entity stm_st is
        port (clk:in std_logic; rst:in std_logic;
    out_stm:out std_logic_vector(3 downto 0));
        end stm_st;
        architecture stm_st_a of stm_st is
        type state_type is (s0,s1,s2,s3);
        signal state:state_type;
        signal div:st_logic_vector(20 downto 0);
        signal lk,clkwise,start:std_logic;
begin
process(clk,rst)
begin
        if (rst='0') then
        div<=(others=>'0');
        elsif(clk'event and clk='1') then
        div<=div+1;
        end if;
end process;
        lk<=dic(15);
process(lk,rst,clkwise)
begin
        if(rst='1')then
        state<=s0;
        elsif lk'event and lk='1' then
        if clkwise='0' then
        case state is
        when s0=>state<=s1;
        when s1=>state<=s2;
        when s2=>state<=s3;
        when s3=>state<=s0;
        when others=>null;
        end case
        end if;
        end if;
end process;
        with state select
        out_stm<="0110" when s0,
        "1010" when s1, "1001" when s2,
        "0101" when s3;
End stm_st_a;
```

PIN ASSIGNMENT

XC3128TQ144		
Connector	Device pin	Property
	128	Clk
P 14/1	88	Dir
P 15/1	132	Out_stm(0)
P 15/2	131	Out_stm(1)
P 15/3	121	Out_stm(2)
P 15/4	120	Out_stm(3)
	125	Rst

XC2S100TQ144-5		
Connector	Device pin	Property
	18	Clk
P 18/5	43	Dir
P 18/21	62	Out_stm(0)
P 18/22	65	Out_stm(1)
P 18/19	60	Out_stm(2)
P 18/20	64	Out_stm(3)
	86	Rst

20. VHDL CODE FOR SEVEN SEGMENT DISPLAY INTERFACE

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;
        Entity mux_disp is
        Port (clk:in std_logic                                  ---4mhz
        Rst: in std_logic;
        seg: out std_logic_vector(6 downto 0)
        base: out std_logic_vector(3 downto 0));
        end mux_disp;
architecture mux_disp_arch of mux_disp is
        signal count : std_logic_vector(25 downto 0);
        signal base_cnt: std_logic_vector(1 downto 0);
        signal seg_cnt: std_logic_vector(3 downto 0);
begin
process(clk,rst)
begin
        if rst = '1' then
        count<=(others=>'0');
        elsif
        clk='1' and clk'event then
        count<= + '1';
end if;
end process;
        base_cnt<=count(12 downto11);
        seg_cnt<=count(25 downto 22);
Base<= "1110" when base_cnt="00" else
        "1101" when base_cnt="01" else
        "1011" when base_cnt="10" else
        "0111" when base_cnt="11" else
        "1111";
Seg<=   "0111111" when seg_cnt="0000" else          ---0
        "0000110" when seg_cnt="0001" else          ---1
        "1011011" when seg_cnt="0010" else          ---2
        "1001111" when seg_cnt="0011" else          ---3
        "1100110" when seg_cnt="0100" else          ---4
        "1101101" when seg_cnt="0101" else          ---5
        "1111101" when seg_cnt="0110" else          ---6
        "0000111" when seg_cnt="0111" else          ---7
        "1111111" when seg_cnt="1000" else          ---8
        "1100111" when seg_cnt="1001" else          ---9
        "1110111" when seg_cnt="1010" else          ---A
        "1111100" when seg_cnt="1011" else          ---B
        "0111001" when seg_cnt="1100" else          ---C
        "1011110" when seg_cnt="1101" else          ---D
        "1111001" when seg_cnt="1110" else          ---E
        "1110001" when seg_cnt="0000" else          ---F
        "0000000";
        End mux_disp_arch;
```

21. VHDL CODE FOR RELAY INTERFACE

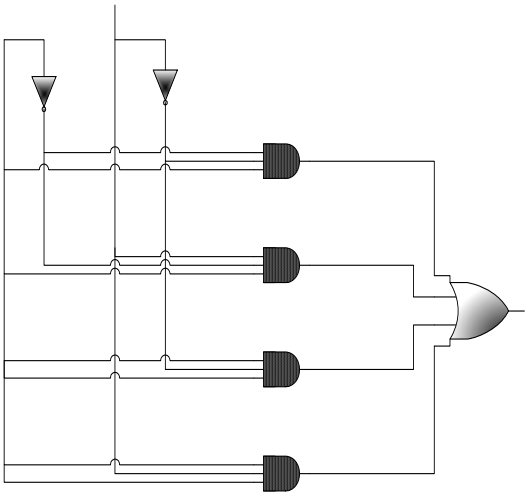
```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;
        Entity relay is
        Port (sw : in std_logic;
                Rl1,led : out std_logic);
        End realy;
Architecture behavioral of relay is
Begin
        Rl1 <= ws;
        Led <= sw;
End behavioral;
```

PIN ASSIGNMENT

XC3128TQ144		
Connector	Device pin	Property
P 18/3	41	Sw
P 18/5	43	Rl1
P 18/21	62	Led

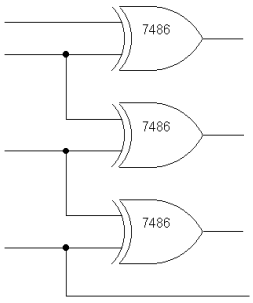
XC2S100TQ144-5		
Connector	Device pin	Property
P 18/3	1	Sw
P 18/5	5	Rl1
P 18/21	23	Led

22. VHDL CODE FOR MULTIPLEXER (4:1)(STRUCTURAL):



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity mux is
        Port ( i0,i1,i2,i3,s0,s1 : in  STD_LOGIC;
        y : out  STD_LOGIC);
        end mux;
architecture struct of mux is
        component and1                          ----components, entity and architecture
        port (l,m,u:in std_logic;n:out std_logic);   --- must be declared separately
        end component;
        component or1                           ----components, entity and architecture
        port (o,p,x,y:in std_logic;q:out std_logic);  --- must be declared separately
        end component;
        component not1                          ----components, entity and architecture
        port (r:in std_logic;s:out std_logic);      --- must be declared separately
        end component;
        signal s2,s3,s4,s5,s6,s7:std_logic;
begin
        u1:and1 port map(i0,s2,s3,s4);
        u2:and1 port map(i1,s2,s1,s5);
        u3:and1 port map(i2,s0,s3,s6);
        u4:and1 port map(i3,s0,s1,s7);
        u5:not1 port map(s0,s2);
        u6:not1 port map(s1,s3);
        u7:or1 port map(s4,s5,s6,s7,y);
end struct;
```

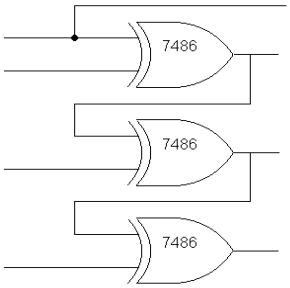
23. VHDL CODE FOR BINARY TO GRAY (Structural):



Using EX-OR gates

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity bg is
        Port ( b : in  STD_LOGIC_VECTOR (3 downto 0);
        g : out  STD_LOGIC_VECTOR (3 downto 0));
        end bg;
architecture struct of bg is
        component xor1 port(a,b:in std_logic;   ----components, entity and architecture
        c:out std_logic);                       --- must be declared separately
        end component;
        component and1 port(d,e:in std_logic;   ----components, entity and architecture
        f:out std_logic);                       --- must be declared separately
        end component;
begin
        u1:and1 port map(b(3),b(3),g(3));
        u2:xor1 port map(b(3),b(2),g(2));
        u3:xor1 port map(b(2),b(1),g(1));
        u4:xor1 port map(b(1),b(0),g(0));
end struct;
```

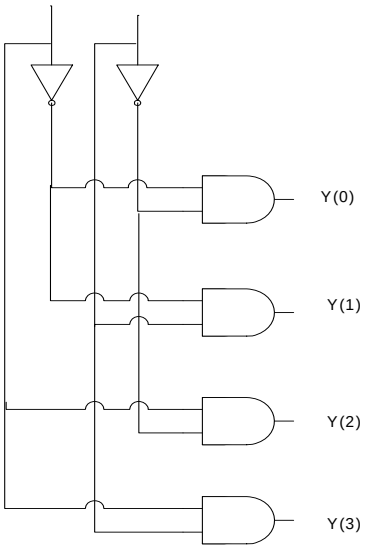
24. VHDL CODE FOR GRAY TO BINARY (Structural):



Using EX-OR gates

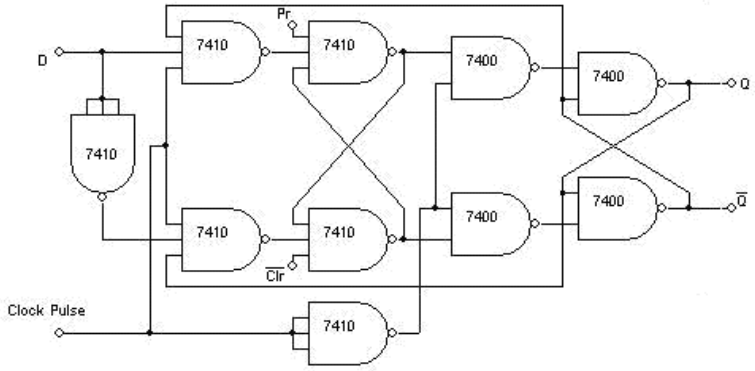
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity gb is
        Port ( g : in  STD_LOGIC_VECTOR (3 downto 0);
        b : out  STD_LOGIC_VECTOR (3 downto 0));
        end gb;
architecture struct of gb is
        component xor1 port(a,b:in std_logic;    ----components, entity and architecture
        c:out std_logic);                        --- must be declared separately
        end component;
        component and1 port(d,e:in std_logic;    ----components, entity and architecture
        f:out std_logic);                        --- must be declared separately
        end component;
        component xor12 port(g,h,i:in std_logic; ----components, entity and architecture
        k:out std_logic);                        --- must be declared separately
        end component;
        component xor13 port(l,m,n,o:in std_logic; ----components, entity and architecture
        p:out std_logic);                        --- must be declared separately
        end component;
begin
u1:and1 port map(g(3),g(3),b(3));
u2:xor1 port map(g(3),g(2),b(2));
u3:xor12 port map(g(3),g(2),g(1),b(1));
u4:xor13 port map(g(3),g(2),g(1),g(0),b(0));
end struct;
```

25. VHDL CODE FOR DECODER (Structural):



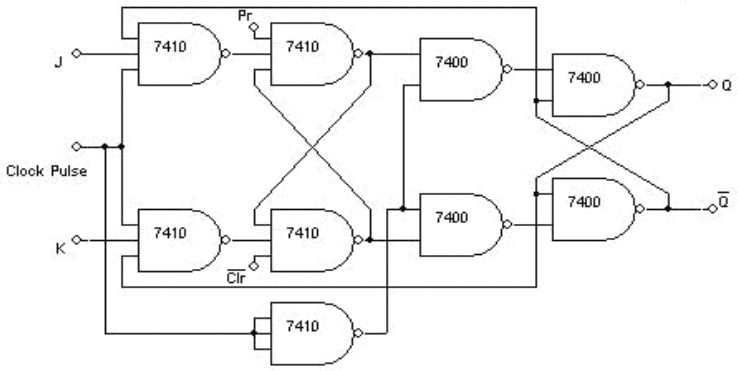

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity dec is
        Port ( a,b : in  STD_LOGIC;
        y: out  STD_LOGIC_VECTOR (3 downto 0));
        end dec;
architecture Behavioral of dec is
        component and1 is                       ----components, entity and architecture
        port(p,q:in std_logic;r:out std_logic);  --- must be declared separately
        end component;
        component not1 is                        ----components, entity and architecture
        port (d:in std_logic;e:out std_logic);   --- must be declared separately
        end component;
        signal s1,s2:std_logic;
begin
        u1:and1 port map(s1,s2,y(0));
        u2:and1 port map(s1,b,y(1));
        u3:and1 port map(a,s2,y(2));
        u4:and1 port map (a,b,y(3));
        u5:not1 port map(a,s1);
        u6:not1 port map(b,s2);
end Behavioral;
```

26. VHDL CODE FOR D FLIP FLOP (Structural):



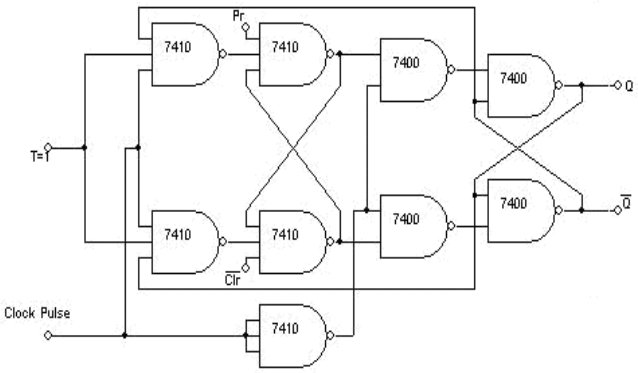

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity dff1 is
        Port ( d,clk,pr,clr : in  STD_LOGIC;
        q,qn : inout  STD_LOGIC);
        end dff1;
architecture struct of dff1 is
        component clkdiv is                     ----components, entity and architecture
        port(clk:in std_logic;clk_d:out std_logic);   --- must be declared separately
        end component;
        component nand1 is
        port(a,b,c:in std_logic;                ----components, entity and architecture
        d:out std_logic);                       --- must be declared separately
        end component;
        component nand12 is
        port(x,y:in std_logic;                  ----components, entity and architecture
        z:out std_logic);                       --- must be declared separately
        end component;
        component nand13 is
        port(e:in std_logic;                    ----components, entity and architecture
        f:out std_logic);                       --- must be declared separately
        end component;
        signal s1,s2,s3,s4,s5,s6,s7,s8,s9:std_logic;
begin
        u10:clkdiv port map(clk,s7);
        u1:nand1 port map(d,s7,qn,s1);
        u2:nand1 port map(s9,s7,q,s2);
        u3:nand1 port map(pr,s1,s4,s3);
        u4:nand1 port map(s2,clr,s3,s4);
        u5:nand12 port map(s3,s8,s5);
        u6:nand12 port map(s8,s4,s6);
        u7:nand12 port map(s5,qn,q);
        u8:nand12 port map(s6,q,qn);
        u9:nand13 port map(s7,s8);
        u11:nand13 port map(d,s9);
end struct;
```

27. VHDL CODE FOR JK FLIP FLOP (Structural):



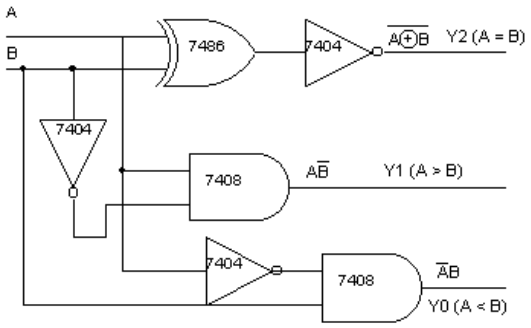

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity jkff is
        Port ( j,k,clk,pr,clr : in  STD_LOGIC;
        q,qn : inout  STD_LOGIC);
        end jkff;
architecture struct of jkff is
        component clkdiv is                     ----components, entity and architecture
        port(clk:in std_logic;clk_d:out std_logic); --- must be declared separately
        end component;
        component nand1 is                      ----components, entity and architecture
        port(a,b,c:in std_logic;                --- must be declared separately
        d:out std_logic);
        end component;
        component nand12 is                     ----components, entity and architecture
        port(x,y:in std_logic;                   --- must be declared separately
        z:out std_logic);
        end component;
        component nand13 is
        port(e:in std_logic;                    ----components, entity and architecture
        f:out std_logic);                       --- must be declared separately
        end component;
        signal s1,s2,s3,s4,s5,s6,s7,s8:std_logic;
begin
        u10:clkdiv port map(clk,s7);
        u1:nand1 port map(j,qn,s7,s1);
        u2:nand1 port map(k,s7,q,s2);
        u3:nand1 port map(pr,s1,s4,s3);
        u4:nand1 port map(s2,clr,s3,s4);
        u5:nand12 port map(s3,s8,s5);
        u6:nand12 port map(s8,s4,s6);
        u7:nand12 port map(s5,qn,q);
        u8:nand12 port map(s6,q,qn);
        u9:nand13 port map(s7,s8);
end struct;
```

28. VHDL CODE FOR T FLIP FLOP (Structural):




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity tff1 is
        Port ( t,clk,pr,clr : in  STD_LOGIC;
        q,qn : inout  STD_LOGIC);
        end tff1;
architecture struct of tff1 is
        component clkdiv is                     ----components, entity and architecture
        port(clk:in std_logic;clk_d:out std_logic); --- must be declared separately
        end component;
        component nand1 is
        port(a,b,c:in std_logic;                ----components, entity and architecture
        d:out std_logic);                       --- must be declared separately
        end component;
        component nand12 is
        port(x,y:in std_logic;                  ----components, entity and architecture
        z:out std_logic);                       --- must be declared separately
        end component;
        component nand13 is
        port(e:in std_logic;                    ----components, entity and architecture
        f:out std_logic);                       --- must be declared separately
        end component;
        signal s1,s2,s3,s4,s5,s6,s7,s8:std_logic;
begin
        u10:clkdiv port map(clk,s7);
        u1:nand1 port map(t,qn,s7,s1);
        u2:nand1 port map(t,s7,q,s2);
        u3:nand1 port map(pr,s1,s4,s3);
        u4:nand1 port map(s2,clr,s3,s4);
        u5:nand12 port map(s3,s8,s5);
        u6:nand12 port map(s8,s4,s6);
        u7:nand12 port map(s5,qn,q);
        u8:nand12 port map(s6,q,qn);
        u9:nand13 port map(s7,s8);
end struct;
```

29. <u>1 BIT COMPARATOR (structural):</u>

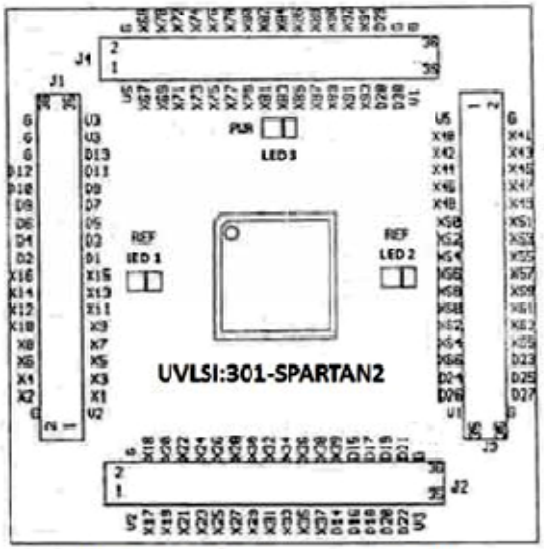


A	B	Y1 (A>B)	Y2 (A = B)	Y3 (A < B)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
        entity comp is
        Port ( a,b : in  STD_LOGIC; y : out  STD_LOGIC_VECTOR (2 downto 0));
        end comp;
architecture struct of comp is
        component and1
        port(l,m:in std_logic;              ----components, entity and architecture
        n:out std_logic);                    --- must be declared separately
        end component;
        component xnor1 is
        port(p,q:in std_logic;              ----components, entity and architecture
        r:out std_logic);                   --- must be declared separately
        end component;
        component notgate1 is
        port(s:in std_logic;                ----components, entity and architecture
        t:out std_logic);                   --- must be declared separately
        end component;
        signal s1,s2:std_logic;
begin
        u1:and1 port map(a,s2,y(0));
        u2:and1 port map(s1,b,y(1));
        u3:xnor1 port map(a,b,y(2));
        u4:notgate1 port map(a,s1);
        u5:notgate1 port map(b,s2);
end struct;
```

PIN Details of FPGA & CPLD

SPARTAN 2 XC2S30TQ144



UVLSI-SPARTAN2 SILK TOP

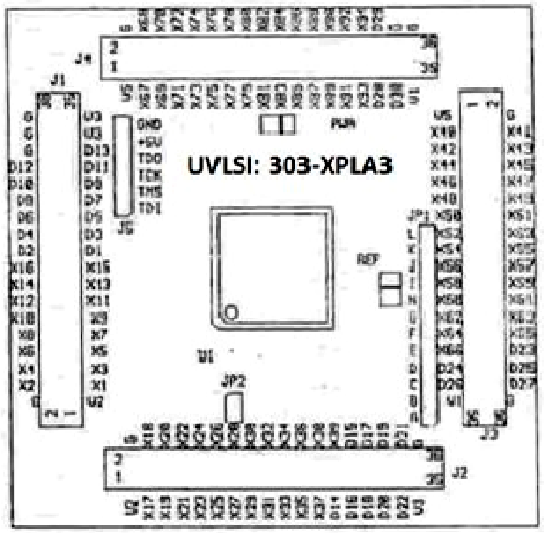
SPARTAN II DEDICATED PINS			
SWITCH ON UVLSI 201	DEVICE PIN NO	PROPERTY	FPGA SIGNAL
RESET 1	88	I_GCK0	CLRN
RESET 2	86	I/O	RESET_N
	91	I_GCK1	RESET
REFERENCE LED	87	I/O	
EXTERNAL CLOCK INPUT	15	I_GCK3	CLK2
ON BOARD CLOCK INPUT	18	I_GCK2	CLK1

CONNECTOR	DEVICE PIN	PROPERTY	SIGNAL	SHARED PINS
P 17/10	131	INPUT 7	SW1 digital I/P |7	
P 17/11	130	INPUT 6	SW1 digital I/P |6	
P 17/12	129	INPUT 5	SW1 digital I/P |5	
P 17/13	126	INPUT 4	SW1 digital I/P |4	
P 17/14	124	INPUT 3	SW1 digital I/P |3	
P 17/15	123	INPUT 2	SW1 digital I/P |2	
P 17/16	122	INPUT 1	SW1 digital I/P |1	
P 17/17	102	OUTPUT 16	O/P LED O 16	
P 17/18	101	OUTPUT 15	O/P LED O 15	
P 17/19	100	OUTPUT 14	O/P LED O 14	
P 17/20	99	OUTPUT 13	O/P LED O 13	
P 17/21	96	OUTPUT 12	O/P LED O 12	
P 17/22	95	OUTPUT 11	O/P LED O 11	
P 17/23	94	OUTPUT 10	O/P LED O 10	
P 17/24	93	OUTPUT 9	O/P LED O 9	
P 17/25	85	OUTPUT 8	O/P LED O 8	
P 17/26	84	OUTPUT 7	O/P LED O 7	
P 17/27	83	OUTPUT 6	O/P LED O 6	
P 17/28	80	OUTPUT 5	O/P LED O 5	SHARED
P 17/29	79	OUTPUT 4	O/P LED O 4	SHARED
P 17/30	78	OUTPUT 3	O/P LED O 3	SHARED
P 17/31	77	OUTPUT 2	O/P LED O 2	SHARED
P 17/32	76	OUTPUT 1	O/P LED O 1	SHARED
P 17/33	80	SEG A	7 Segment O/P |=> Segment 'a'	
P 17/34	79	SEG B	7 Segment O/P |=> Segment 'b'	
P 17/35	78	SEG C	7 Segment O/P |=> Segment 'c'	
P 17/36	77	SEG D	7 Segment O/P |=> Segment 'd'	
P 17/37	76	SEG E	7 Segment O/P |=> Segment 'e'	
P 17/38	75	SEG F	7 Segment O/P |=> Segment 'f'	
P 17/39	74	SEG G	7 Segment O/P |=> Segment 'g'	
P 17/40	67	SEG DP	7 Segment O/P |=> Segment 'dp'	
P 17/41	31	DISP 1	Digit 0 select o/p (base drive 0)	SHARED
P 17/42	30	DISP 2	Digit 1 select o/p (base drive 1)	SHARED
P 17/43	29	DISP 3	Digit 2 select o/p (base drive 2)	SHARED
P 17/44	28	DISP 4	Digit 3 select o/p (base drive 3)	SHARED
P 17/45	112	KEY1	Key k1	

CONNECTOR	DEVICE PIN	PROPERTY	SIGNAL	SHARED PINS
P 14/1	121	I/O	EXT I/P 1	
P 14/2	120	I/O	EXT I/P 2	
P 14/3	118	I/O	EXT I/P 3	
P 14/4	117	I/O	EXT I/P 4	
P 14/5	116	I/O	EXT I/P 5	
P 14/6	115	I/O	EXT I/P 6	
P 14/7	114	I/O	EXT I/P 7	
P 14/8	113	I/O	EXT I/P 8	
P 15/1	21	Dedicated O	Buffered O/P 9	LCD-DB 4
P 15/2	20	Dedicated O	Buffered O/P 10	LCD-DB 5
P 15/3	19	Dedicated O	Buffered O/P 11	LCD-DB 6
P 15/4	13	Dedicated O	Buffered O/P 12	LCD-DB 7
P 15/5	12	Dedicated O	Buffered O/P 13	LCD-RS
P 15/6	11	Dedicated O	Buffered O/P 14	LCD-E
P 15/7	10	Dedicated O	Buffered O/P 15	
P 15/8	7	Dedicated O	Buffered O/P 16	
P 16/1	31	Dedicated O	Buffered O/P 1	
P 16/2	30	Dedicated O	Buffered O/P 2	
P 16/3	29	Dedicated O	Buffered O/P 3	
P 16/4	28	Dedicated O	Buffered O/P 4	
P 16/5	27	Dedicated O	Buffered O/P 5	
P 16/6	26	Dedicated O	Buffered O/P 6	
P 16/7	23	Dedicated O	Buffered O/P 7	
P 16/8	22	Dedicated O	Buffered O/P 8	
P 17/1	141	INPUT 16	SW1 digital I/P |16	
P 17/2	140	INPUT 15	SW1 digital I/P |15	
P 17/3	139	INPUT 14	SW1 digital I/P |14	
P 17/4	138	INPUT 13	SW1 digital I/P |13	
P 17/5	137	INPUT 12	SW1 digital I/P |12	
P 17/6	136	INPUT 11	SW1 digital I/P |11	
P 17/7	134	INPUT 10	SW1 digital I/P |10	
P 17/8	133	INPUT 9	SW1 digital I/P |9	
P 17/9	132	INPUT 8	SW1 digital I/P |8	

CONNECTOR	DEVICE PIN	PROPERTY	SIGNAL	SHARED PINS
P 17/46	105	KEY2	Key k2	
P 17/47	104	KEY3	Key k3	
P 17/48	103	KEY4	Key k4	
P 17/49	5V	VCC	VCC	
P 17/50	GND	Ground	Ground	
P 18/1	38	DAC 0	DAC data0 o/p from FPGA	MSB
P 18/2	40	DAC 1	DAC data1 o/p from FPGA	
P 18/3	41	DAC 2	DAC data2 o/p from FPGA	
P 18/4	42	DAC 3	DAC data3 o/p from FPGA	
P 18/5	43	DAC 4	DAC data4 o/p from FPGA	
P 18/6	44	DAC 5	DAC data5 o/p from FPGA	
P 18/7	46	DAC 6	DAC data6 o/p from FPGA	
P 18/8	47	DAC 7	DAC data7 o/p from FPGA	LSB
P 18/9	48	ADC_D0	Data 0 from ADC 0808	
P 18/10	49	ADC_D1	Data 1 from ADC 0808	
P 18/11	50	ADC_D2	Data 2 from ADC 0808	
P 18/12	51	ADC_D3	Data 3 from ADC 0808	
P 18/13	54	ADC_D4	Data 4 from ADC 0808	
P 18/14	56	ADC_D5	Data 5 from ADC 0808	
P 18/15	57	ADC_D6	Data 6 from ADC 0808	
P 18/16	58	ADC_D7	Data 7 from ADC 0808	
P 18/17	59	ADC_A0	ADC CHANNEL SELECT BIT	
P 18/18	63	ADC_START	Write for ADC0808	
P 18/91	60	ADC_A1	ADC CHANNEL SELECT BIT	
P 18/20	64	ADC_ALE	ADC ALE Signal	
P 18/21	62	ADC_A2	ADC CHANNEL SELECT BIT	
P 18/22	65	ADC_EOC	Interrupt signal from ADC0808	
			(End OF Conversion)	
P 18/23			555 FREQ o/p to FPGA	
P 18/24	66	ADC_OE	ADC OUTPUT ENABLE (OE)	
P 18/25	5V	VCC	VCC	
P 18/26	GND	GROUND	GND	

COOL RUNNER XCR3128XL TQ144



UVLSI-COOL RUNNER SILK TOP

COOL RUNNER DEDICATED PINS			
SWITCH ON UVLSI 201	DEVICE PIN NO	PROPERTY	FPGA SIGNAL
RESET 1	125	CLK3	CLRN
RESET 2	29	I/O	RESET_N
	126	CLK2	RESET
REFERENCE LED	53	I/O	
EXTERNAL CLOCK INPUT	127	CLK1	CLK2
ON BOARD CLOCK INPUT	128	CLK0	CLK1

CONNECTOR	DEVICE PIN	PROPERTY	SIGNAL	SHARED PINS
P 17/6	101	INPUT 11	SW1 digital I/P |11	
P 17/7	100	INPUT 10	SW1 digital I/P |10	
P 17/8	99	INPUT 9	SW1 digital I/P |9	
P 17/9	98	INPUT 8	SW1 digital I/P |8	
P 17/10	97	INPUT 7	SW1 digital I/P |7	
P 17/11	96	INPUT 6	SW1 digital I/P |6	
P 17/12	94	INPUT 5	SW1 digital I/P |5	
P 17/13	93	INPUT 4	SW1 digital I/P |4	
P 17/14	92	INPUT 3	SW1 digital I/P |3	
P 17/15	91	INPUT 2	SW1 digital I/P |2	
P 17/16	90	INPUT 1	SW1 digital I/P |1	
P 17/17	72	OUTPUT 16	O/P LED O 16	
P 17/18	71	OUTPUT 15	O/P LED O 15	
P 17/19	70	OUTPUT 14	O/P LED O 14	
P 17/20	69	OUTPUT 13	O/P LED O 13	
P 17/21	68	OUTPUT 12	O/P LED O 12	
P 17/22	67	OUTPUT 11	O/P LED O 11	
P 17/23	66	OUTPUT 10	O/P LED O 10	
P 17/24	65	OUTPUT 9	O/P LED O 9	
P 17/25	63	OUTPUT 8	O/P LED O 8	
P 17/26	62	OUTPUT 7	O/P LED O 7	
P 17/27	61	OUTPUT 6	O/P LED O 6	
P 17/28	60	OUTPUT 5	O/P LED O 5	
P 17/29	56	OUTPUT 4	O/P LED O 4	
P 17/30	55	OUTPUT 3	O/P LED O 3	
P 17/31	54	OUTPUT 2	O/P LED O 2	
P 17/32	53	OUTPUT 1	O/P LED O 1	
P 17/33	46	SEG A	7 Segment O/P |=> Segment 'a'	
P 17/34	45	SEG B	7 Segment O/P |=> Segment 'b'	
P 17/35	44	SEG C	7 Segment O/P |=> Segment 'c'	
P 17/36	42	SEG D	7 Segment O/P |=> Segment 'd'	
P 17/37	41	SEG E	7 Segment O/P |=> Segment 'e'	
P 17/38	40	SEG F	7 Segment O/P |=> Segment 'f'	
P 17/39	39	SEG G	7 Segment O/P |=> Segment 'g'	
P 17/40	38	SEG DP	7 Segment O/P |=> Segment 'dp'	
P 17/41	37	DISP 1	Digit 0 select o/p (base drive 0)	

CONNECTOR	DEVICE PIN	PROPERTY	SIGNAL	SHARED PINS
P 14/1	88	I/O	EXT I/O 1	
P 14/2	87	I/O	EXT I/O 2	
P 14/3	86	I/O	EXT I/O 3	
P 14/4	84	I/O	EXT I/O 4	
P 14/5	83	I/O	EXT I/O 5	
P 14/6	82	I/O	EXT I/O 6	
P 14/7	81	I/O	EXT I/O 7	
P 14/8	80	I/O	EXT I/O 8	
P 14/9	5V	VCC	VCC	
P 14/10	GND	GND	GND	
P 15/1	132	Dedicated O	Buffered O/P 9	LCD-DB 4
P 15/2	131	Dedicated O	Buffered O/P 10	LCD-DB 5
P 15/3	121	Dedicated O	Buffered O/P 11	LCD-DB 6
P 15/4	120	Dedicated O	Buffered O/P 12	LCD-DB 7
P 15/5	119	Dedicated O	Buffered O/P 13	LCD-RS
P 15/6	118	Dedicated O	Buffered O/P 14	LCD-E
P 15/7	117	Dedicated O	Buffered O/P 15	
P 15/8	116	Dedicated O	Buffered O/P 16	
P 15/9	5V	VCC	VCC	
P 15/10	GND	GND	GND	
P 16/1	141	Dedicated O	Buffered O/P 1	
P 16/2	140	Dedicated O	Buffered O/P 2	
P 16/3	139	Dedicated O	Buffered O/P 3	
P 16/4	138	Dedicated O	Buffered O/P 4	
P 16/5	137	Dedicated O	Buffered O/P 5	
P 16/6	136	Dedicated O	Buffered O/P 6	
P 16/7	134	Dedicated O	Buffered O/P 7	
P 16/8	133	Dedicated O	Buffered O/P 8	
P 16/9	5V	VCC	VCC	
P 16/10	GND	GND	GND	
SIGNALS FROM GPIO BOARD				
P 17/1	110	INPUT 16	SW1 digital I/P |16	
P 17/2	109	INPUT 15	SW1 digital I/P |15	
P 17/3	107	INPUT 14	SW1 digital I/P |14	
P 17/4	106	INPUT 13	SW1 digital I/P |13	
P 17/5	102	INPUT 12	SW1 digital I/P |12	

CONNECTOR	DEVICE PIN	PROPERTY	SIGNAL	SHARED PINS
P 17/42	32	DISP 2	Digit 1 select o/p (base drive 1)	
P 17/43	31	DISP 3	Digit 2 select o/p (base drive 2)	
P 17/44	30	DISP 4	Digit 3 select o/p (base drive 3)	
P 17/45	79	KEY1	Key k1	
P 17/46	78	KEY2	Key k2	
P 17/47	77	KEY3	Key k3	
P 17/48	74	KEY4	Key k4	
P 17/49	5V	VCC	VCC	
P 17/50	GND	Ground	Ground	
P 18/1	142	DAC 0	DAC data0 o/p from FPGA	MSB
P 18/2	143	DAC 1	DAC data1 o/p from FPGA	
P 18/3	1	DAC 2	DAC data2 o/p from FPGA	
P 18/4	2	DAC 3	DAC data3 o/p from FPGA	
P 18/5	5	DAC 4	DAC data4 o/p from FPGA	
P 18/6	6	DAC 5	DAC data5 o/p from FPGA	
P 18/7	7	DAC 6	DAC data6 o/p from FPGA	
P 18/8	8	DAC 7	DAC data7 o/p from FPGA	LSB
P 18/9	9	ADC_D0	Data 0 from ADC 0808	
P 18/10	10	ADC_D1	Data 1 from ADC 0808	
P 18/11	11	ADC_D2	Data 2 from ADC 0808	
P 18/12	12	ADC_D3	Data 3 from ADC 0808	
P 18/13	14	ADC_D4	Data 4 from ADC 0808	
P 18/14	15	ADC_D5	Data 5 from ADC 0808	
P 18/15	16	ADC_D6	Data 6 from ADC 0808	
P 18/16	18	ADC_D7	Data 7 from ADC 0808	
P 18/17	21	ADC_A0	ADC CHANNEL SELECT BIT	
P 18/18	25	ADC_START	Write for ADC0808	
P 18/91	22	ADC_A1	ADC CHANNEL SELECT BIT	
P 18/20	26	ADC_ALE	ADC ALE Signal	
P 18/21	23	ADC_A2	ADC CHANNEL SELECT BIT	
P 18/22	28	ADC_EOC	Interrupt signal from ADC0808	
			(End OF Conversion)	
P 18/23			555 FREQ o/p to FPGA	
P 18/24	27	ADC_OE	ADC OUTPUT ENABLE (OE)	
P 18/25		VCC	VCC	
P 18/26		GROUND	GND	