Building a Clock Divider Circuit Using Architectural Resources Lab

Overview:
In this lab you will learn that some of the architectural resources of FPGA can not be inferred and configured during instantiation. You will learn how to use architectural wizard of ISE/CORE Generator to configure some complex resources.

Outcome:
You will understand how to use CORE Generator's architectural wizard to configure and instantiate DCM. You will learn how to create a model using ISE create project wizard. You will instantiate lower-level models to create a complex model. You will use ISE simulator to simulate the design. You will add user constraint file (ucf) to assign pins so the design can be targeted to National Instruments (NI) Digital Electronics FPGA Board. You will implement the design and create a bitstream file using ISE's implementation tools. Once bitstream is created, you will download using ISE's iMPACT program and verify the design functionality.

References:
1. National Instruments' Digital Electronics FPGA Board user manual
2. Verilog HDL books

Problem Statement:
Design a blinking circuit that blinks at 1 Hz. You will configure Digital Clock Manager (DCM), instantiate it in your design, and further use clock-divider to display a blinking LED.

Implementation:
Since the input clock on the board is 50 MHz it must be slowed down to 1 Hz as per requirement. One way to do that is to use big counter to model a clock divider. Another way is to use architectural resource, DCM, to slow down. Since slowest speed DCM can be run is at 5 MHz and provide DLL functionality, the DCM must be configured to slow the input down. On DCM, there is CLKDV output which can be used if the output clock is of some predefined value. If the output clock is not a predefined ratio of the input clock then CLKFX port can be used. In our case, we use CLKDV port as the ratio 0f 10 is one of the choices. We then use another clock divider to bring it down to 2 Hz. The 2 Hz clock is then used to blink LED ON and OFF, yielding 1 Hz rate.

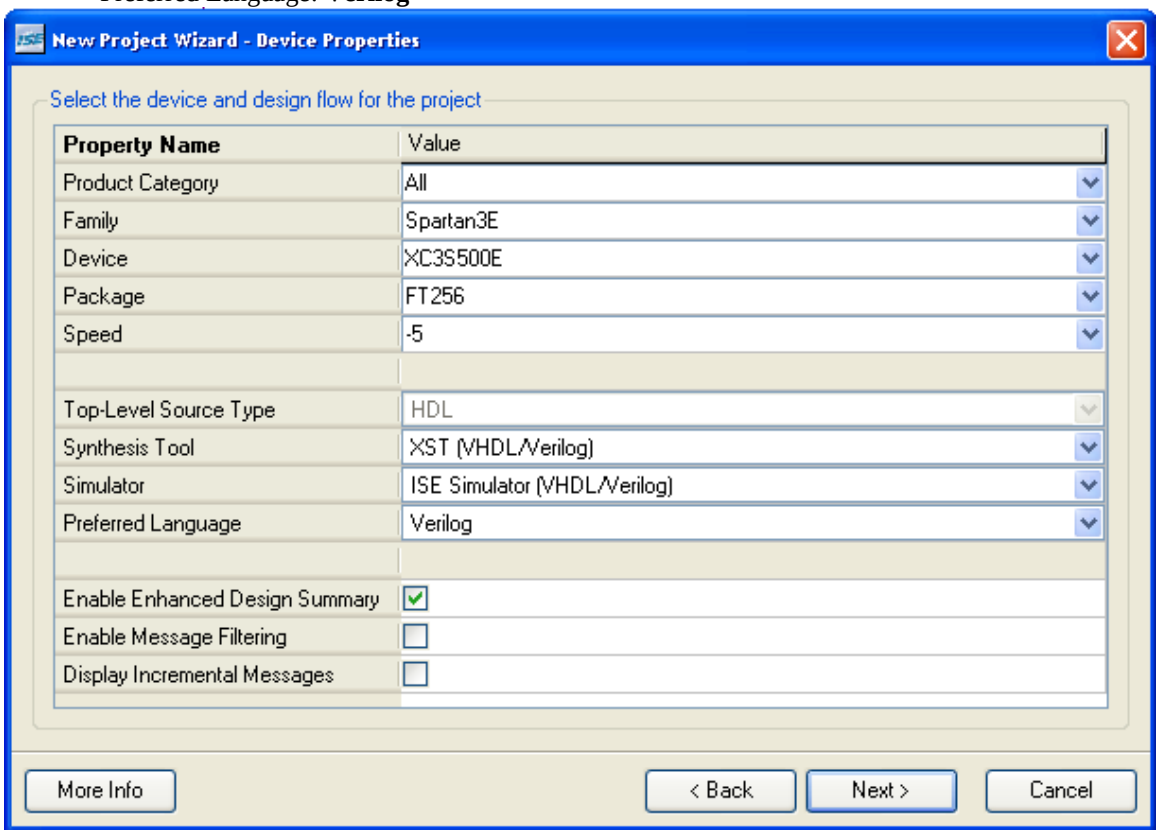
Procedure:

Extract resources.zip file in c:\NI\Verilog_Labs folder

1. Create a ISE project

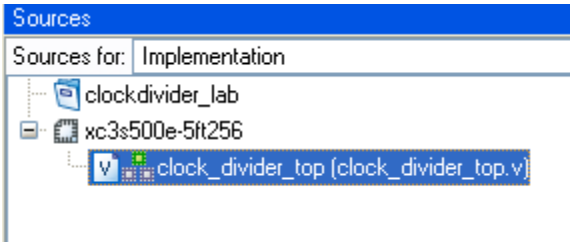
- Launch ISE: Select **Start → Programs → Xilinx ISE Design Suite 10.1 → ISE → Project Navigator**
- In the Project Navigator, select **File → New Project.** The New Project Wizard opens
- For Project Location, use the "…" button to browse to C:\NI\Verilog_labs, and then click **OK**
- For Project Name, type ***clockdivider_lab***
- Click **Next**
- Select the following options and click **Next**

  - ✓ Device Family: **Spartan3E**
  - ✓ Device: **xc3s500E**
  - ✓ Package: **ft256**
  - ✓ Speed Grade: **–5**
  - ✓ Synthesis Tool: **XST (VHDL/Verilog)**
  - ✓ Simulator: **ISE Simulator (VHDL/Verilog)**
  - ✓ Preferred Language: **Verilog**

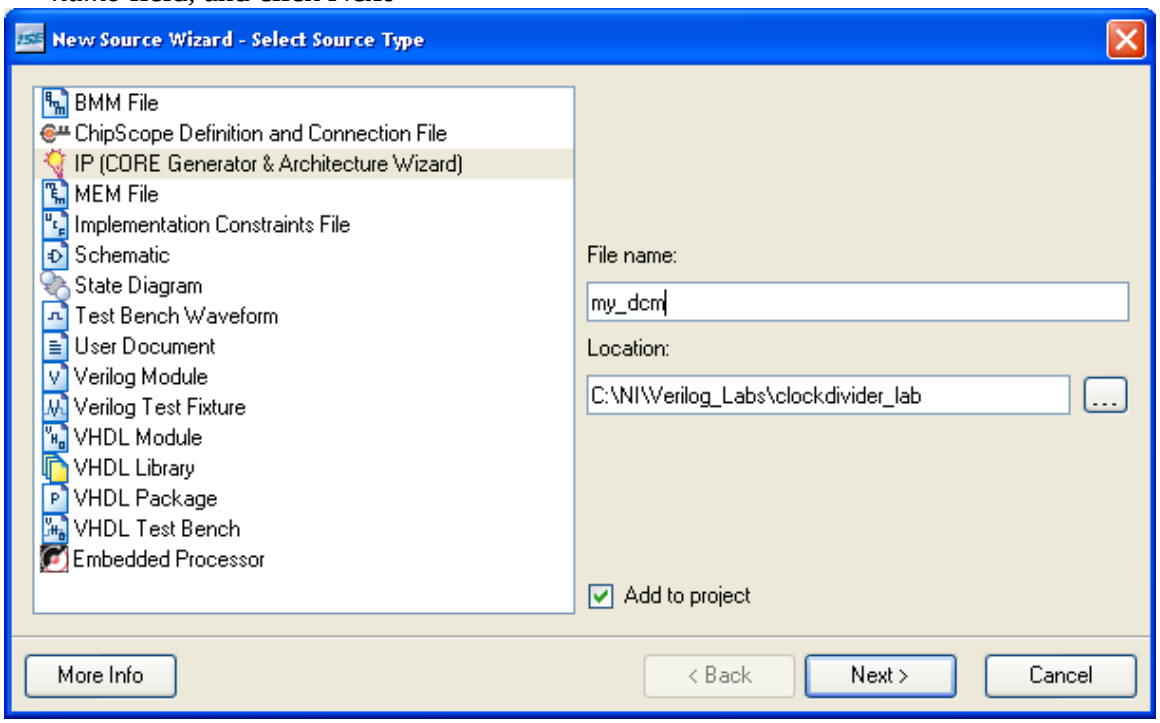


- The **Create New Source** dialog will appear. Click **New Source** and *New Source Wizard* form will appear
- Select *Verilog Module* in the Source Type window (left) and enter **clock_divider_top** in the filename field

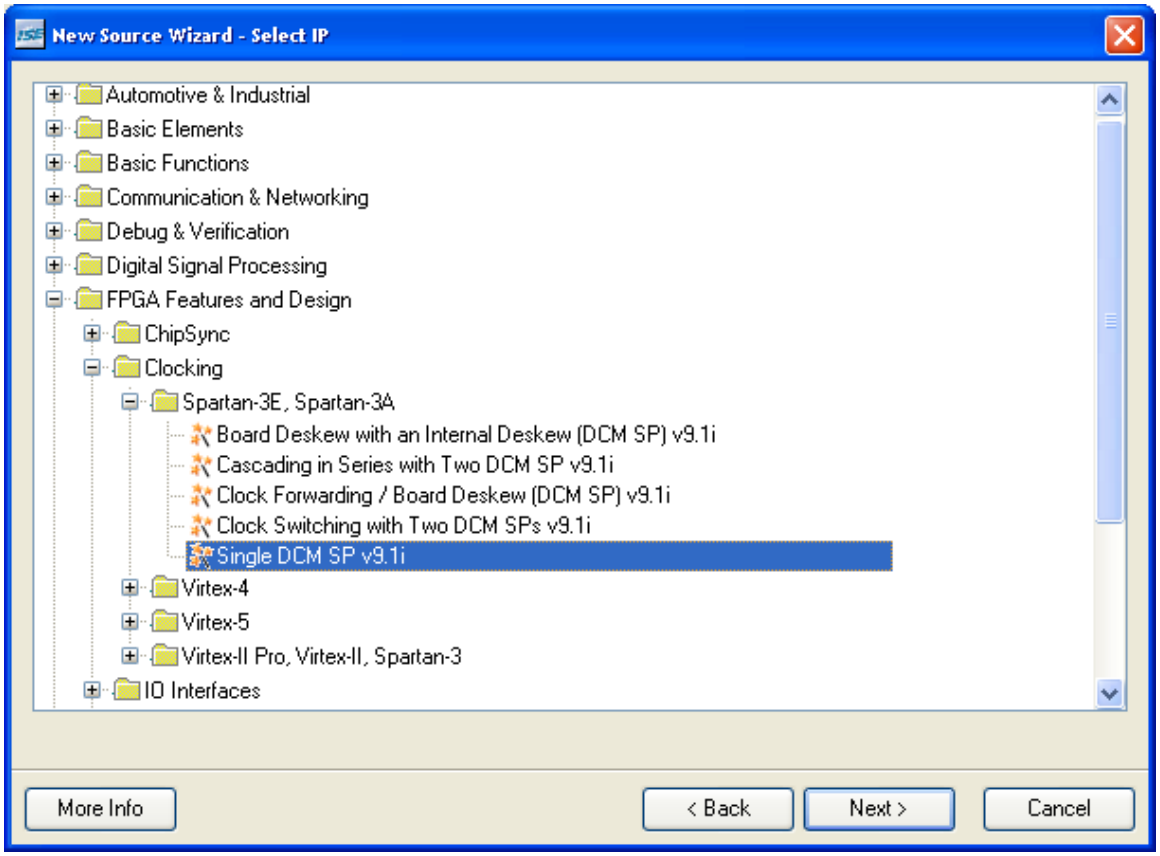
- Click **Next** and I/O Port window will be displayed
- Enter **clk**, **rst** as input port and **lock, led** as output ports and then click **Next**
- Click **Finish** and then **Yes** to create the module file
- Click **Next**, again **Next**, and **Finish** buttons to close the wizard
- A project will be created, clock_divider_top.v model will be added in the *Sources* hierarchy window, and the same file will be opened in a editor window



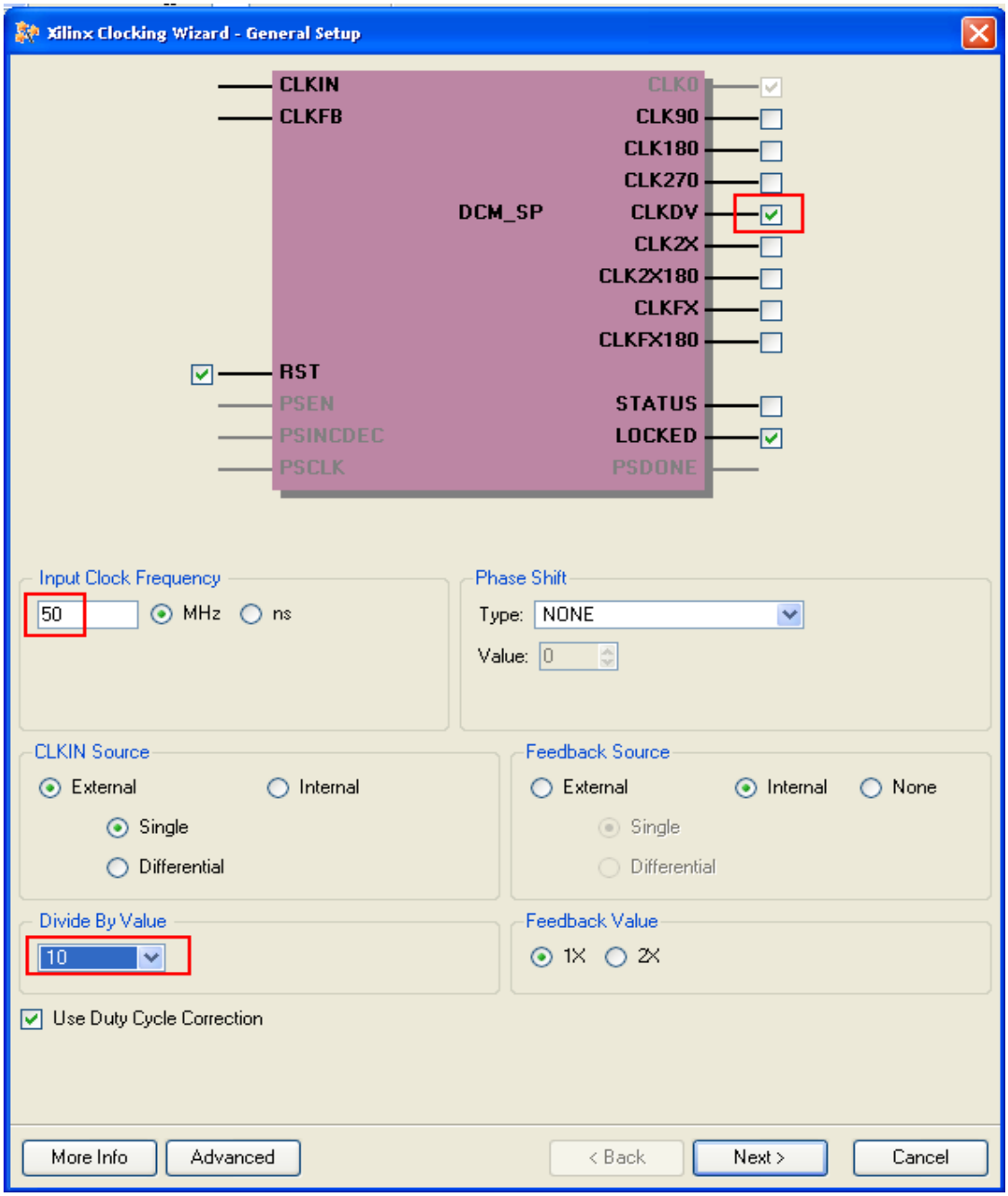
- Select **clock_divider_top** in the *Sources* window, and select **New Source**
- Select *IP (Core Generator & Architecture Wizard)* and enter **my_dcm** in *File name* field, and click **Next**



- Expand *FPGA Feature and Design;* expand *Clocking*; expand *Spartan-3E, Spartan-3A* folders and then select **Single DCM SP v9.1i**



- Click **Next** and **Finish**
- Click on check-box of **CLKDV** output, enter **50** in *Input Clock Frequency* field, and select **10** in the *Divide By Value* drop-down box



- Click **Next**, **Next**, and **Finish** to close the wizard
- Instantiate *my_dcm* by entering the following code and declare **clk_dv** as the wire

```
28  wire clk_dv;
29
30  // Instantiate the module
31  my_dcm D1 (
32      .CLKIN_IN(clk),
33      .RST_IN(rst),
34      .CLKDV_OUT(clk_dv),
35      .CLKIN_IBUFG_OUT(),
36      .CLK0_OUT(),
37      .LOCKED_OUT(lock)
38      );
39
```

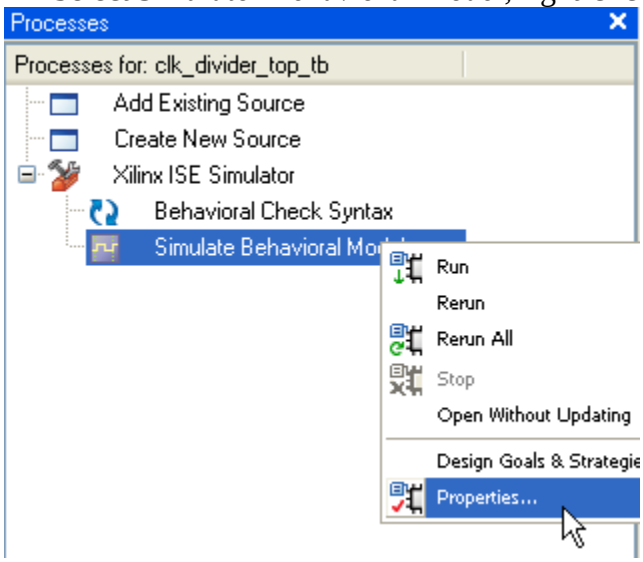
- In order to see the *led* output, the **clk_dv** can be used to toggle the *led*
- Enter the following code to toggle the *led*

```
always @ (posedge clk_dv or posedge rst)
if (rst)
   led <= 0;
else
   led <= ~led;
```

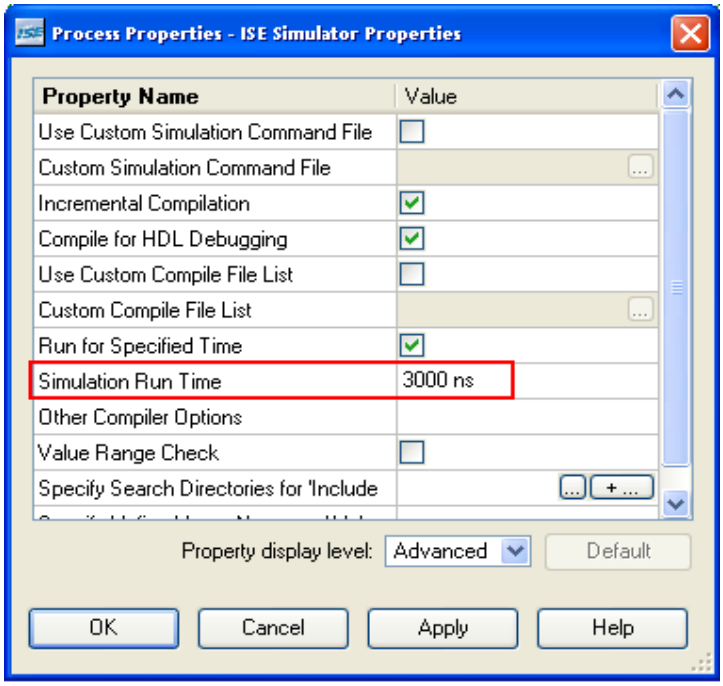
- Since **led** is updated in an always block it must be defined as a *reg* variable
- Modify *led* output port line to read as **output reg led**
- Save and close the file

2. *Simulate the design using ISE*

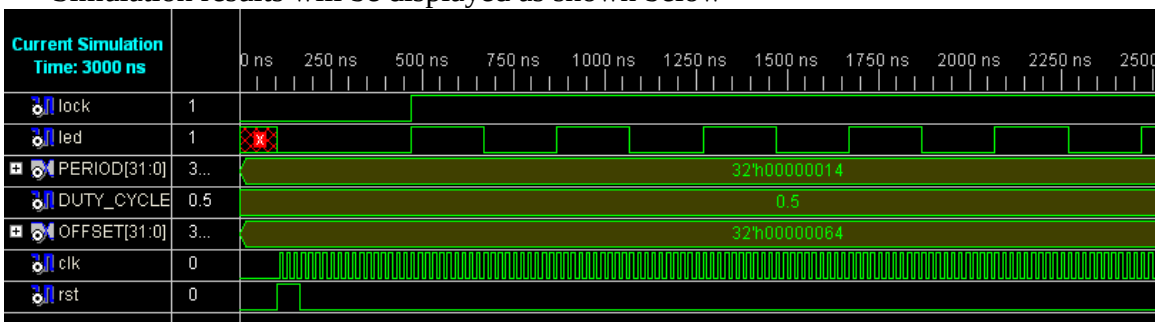
- Right-click on the **clock_divider_top** entry in *Sources* window, right-click and select **Add Copy of Source…**
- Browse to C:\NI\Verilog_labs\resouces\clockdivider_lab and select **clock_divider_top_tb.v**. Notice that the Sources for window changes to Behavioral Simulation from Implementation
- Select **clock_divider_top_tb** in the *Sources* window, expand the **Xilinx ISE Simulator** process in *Processes* window
- Select *Simulator Behavioral Model*, right-click and select **properties**



- Change the *simulation run time* from 1000 to **3000** ns and then click **OK**



- Double-click **Simulator Behavioral Model** to run simulator
- The model will be compiled and the simulator will be run
- Simulation results will be displayed as shown below



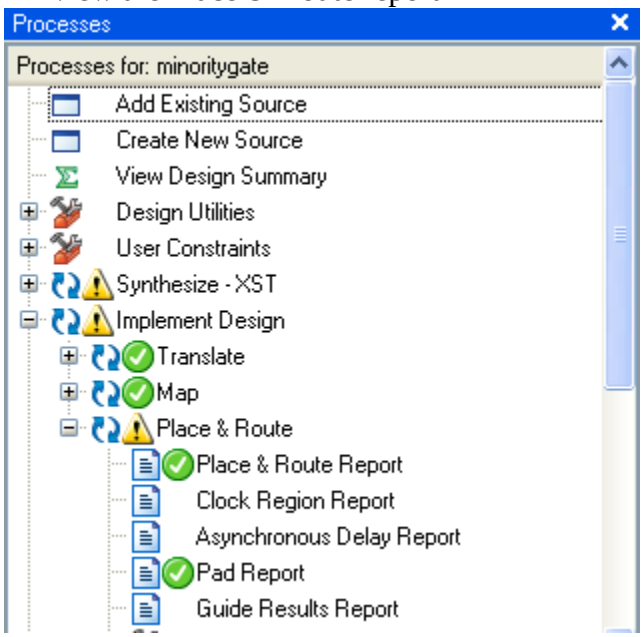
- You can see that it takes about sixteen clock cycles for DCM to get locked after the reset is de-asserted
- You can also see that the led is toggling at 1/20 th rate (clk_dv is 1/10 th of clk input and then the clk_dv is divided by 2 to toggle led
- Close the simulator

3. Implement the design

- Select **implementation** in *Sources for* window
- In order to see the LED toggling in hardware, the clock needs to further slow down to seconds period. A model is provided which can be used to do the same
- Select **clock_divider_top** module in *Sources* window, right-click, and select **Add Copy of Source…**
- Browse to *c:\NI\verilog_labs\resources\clockdivider_lab* and select **clk_divider.v** and click **open**
- Add the following instantiation to include the clk_divider

```
clk_divider C1(
    .clk(clk_dv),
    .rst(rst),
    .slow_clk(slow_clk)
    );
```

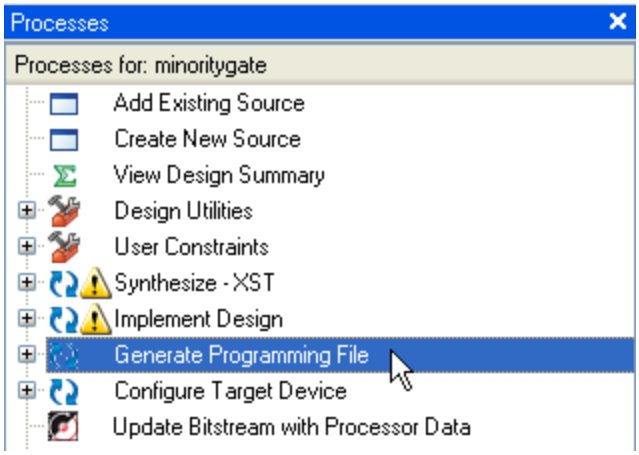
- Note that the *slow_clk* port is connected to wire **slow_clk**. So define it as a wire. Also make change to the *always* block which drives *led* with **slow_clk** instead of *clk_dv*
- Save the changes
- The UCF either must be created or added
- Add the provided ucf file (clock_divider_top.ucf) from *c:\NI\verilog_labs\resources\clockdivider_lab* folder
- Select **clock_divider_top** module in *Sources* window and double-click on **Implement Design** process in *Processes* window. This will go through Synthesis, and Implementation stages
- When the implementation is completed, expand Implement Design process to view the Place & Route report



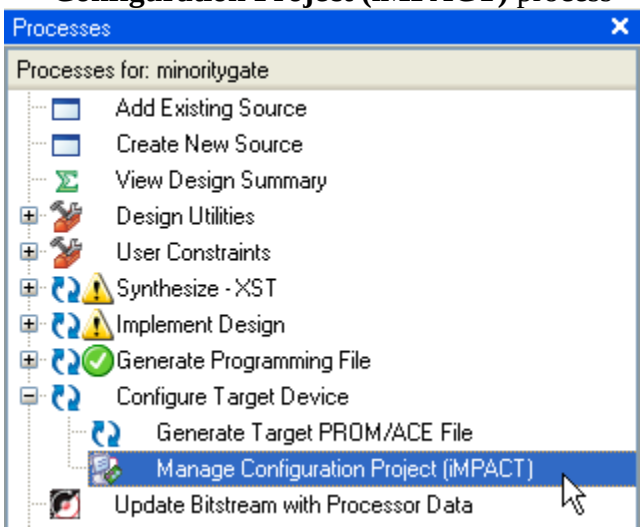
- Double-click on the Place & Route report to view the report. Look at the resource utilization and note that **29** slices, **2** BUFGMUX, and **1** DCM are being used. The 2 BUFGMUX are for clk input and clk_dv signals. The 29 slices are mainly due to the counter that is being used in clk_divider model which models 24-bit wide counter
- You can see similar information by clicking on Design Summary tab and looking at the various information

4. Verify the design in hardware

- Select **clock_divider_top** in *Sources* window and double-click on **Generate Programming File** process to generate the bit file for the design



- Expand **Configure Target Device** process and double-click on **Manage Configuration Project (iMPACT)** process



- Connect the board with the USB-JTAG cable
- Power ON the board
- Click Finish to use the JTAG chain
- Select *clock_divider_top.bit* file to be assigned to xc3s500e device and click **Open**
- Click **Bypass** button for *xcf04s* and then **OK** to use FPGA device programming
- Right-click on the FPGA and select **Program**
- This will program the FPGA and DONE light will lit on the board
- Once programmed successfully, verify that the LED **LD1** is toggling at every second. The LED **LD0** is also lit indicating DCM is locked
- You can also press BTN1 to reset the design. When pressed, notice that both LD0 and LD1 are off as DCM is in reset state. When the button is let go, you will notice that LD0 turns ON first indicating DCM is locked and the LD1 starts blinking as the DCM outputs stable clock
- Once confirmed the functionality, power down the board and close ISE saving project changes

Conclusion:

In this lab exercise you learned how to use architectural wizard of ISE to configure and instantiate architectural resource like DCM. You slowed down the clock from 50 MHz to 5 MHz and then used provided clock divider to further slow down the clock to observe the LED blink. You were able to simulate the design and then verify the complete design in hardware board.