

Architecture Wizard and Pins Assignment Lab

Introduction

In this lab, you will extend the design using the Architecture wizard and Pin Planning feature of PlanAhead tool. The Architecture Wizard feature enables designers to configure and add FPGA resources to the design. The Pin Planning feature of PlanAhead enables designers to add location constraints to a design.

Objectives

After completing this lab, you will be able to:

- Use the Architecture Wizard to configure and instantiate a DCM component
- Use PlanAhead to assign pin locations
- Implement the design and confirm that the pin assignments were used
- Download and test design in hardware

Design Description

This lab will make use of the UART Real-Time clock design and detailed information can be found in the UART_real_time_clock.pdf provided with the PicoBlaze distribution. This section attempts to organize and highlight the feature of the design.

The design implements a real-time clock maintaining time in hours, minutes and seconds together with the ability to set an alarm. The unusual feature of the design is that a UART serial communication is used to set and observe the time/alarm sending simple text commands and messages via a utility such as Hyper Terminal.

The design understands some simple ASCII commands and even supports some editing during their entry using the backspace key on your keyboard. A command is only completed when a carriage return is entered. The design is ready to accept a command when the "KCPSM3>" prompt is displayed.

The "uclock" program provided with the distribution is able to interpret upper and lower case characters by converting commands (see documentation for details) to upper case before analyzing them. Incorrect commands will result in a "syntax error" message and incorrect time values will be indicated by an "Invalid Time" message. Although it is unlikely to occur when using Hyper Terminal, an "overflow error" message will be generated if commands are transmitted faster than the design can process them (ie. UART receiver buffer becomes full).

The design requires a 55 MHz clock. Since the Nexys3 board includes a 100 MHz oscillator, you will use the Architecture Wizard to generate a DCM with a 55 MHz output and instantiate it into the design.

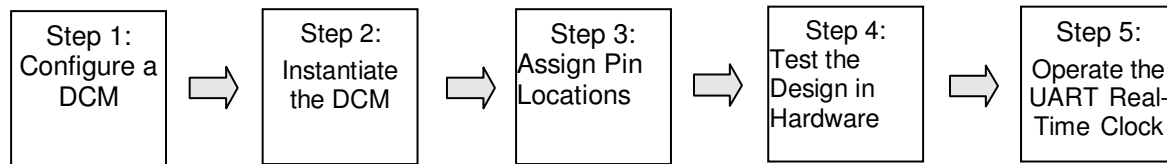
Procedure

This lab is separated into steps that consist of general overview statements that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

This lab comprises 5 primary steps: You will configure a DCM, instantiate the DCM, assign pin locations, test the design in hardware and, finally, operate the UART real-time clock.

Note: If you are unable to complete the lab at this time, you can download the lab files for this module from the Xilinx University Program site at <http://www.xilinx.com/university>

General Flow for this Lab



Configure a DCM


Step 1

1-1. This version of the design is missing a DCM component. Use the Architecture Wizard to configure a DCM component to output a clock at 55 MHz.

1-1-1. Select Start → All Programs → Xilinx ISE Design Suite 13.2 → ISE Design Tools → Project Navigator.

1-1-2. Select File → Open Project or click  if no project is open in the Project Navigator, and open the arwz_pace.xise project from

- Verilog users: `c:\xup\fpgaflow\labs\verilog\lab2`
- VHDL users: `c:\xup\fpgaflow\labs\vhd\lab2`

1-1-3. In the Hierarchy window, right-click and choose Create New Source or click  on the side tool buttons.

If you do not see the Create New Source process, ensure that an HDL source file is selected in the **Sources** in Project window.

1-1-4. In the New Source window, select IP (CORE Generator & Architecture Wizard) and enter my_dcm as the file name and click Next.

1-1-5. In the Select IP window, expand FPGA Features and Design → Clocking and select Clocking Wizard v3.2i (Figure 1).

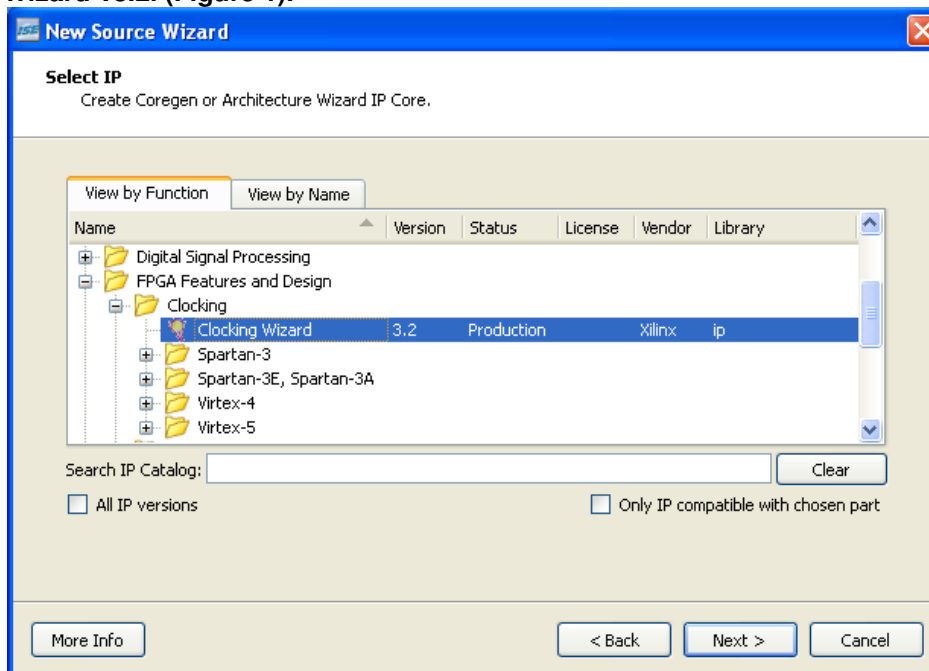


Figure 1. Architecture Wizard Selection Box

1-1-6. Click Next, and click Finish.

1-1-7. In the Clocking Features / Input Clocks window, keep the defaults and click Next.



Clocking Wizard xilinx.com:ip:clk_wiz:3.2

Component name:

Clocking Features / Input Clocks

Clocking Features

- ☒ Frequency synthesis
- ☒ Phase alignment (known phase relationship to input clock)
- ☐ Minimize power
- ☐ Dynamic phase shift
- ☐ Dynamic reconfiguration (in system output freq modification)

Jitter Optimization

- ☒ Balanced
- ☐ Minimize output jitter (low clock jitter filtering)
- ☐ Maximize input jitter filtering (allow larger input jitter)

Clock Manager Type

Mode

- ☒ Auto Selection
(Recommended: Wizard selects primitive)
- ☐ Manual Selection
(User selects primitive)

Input Jitter Unit

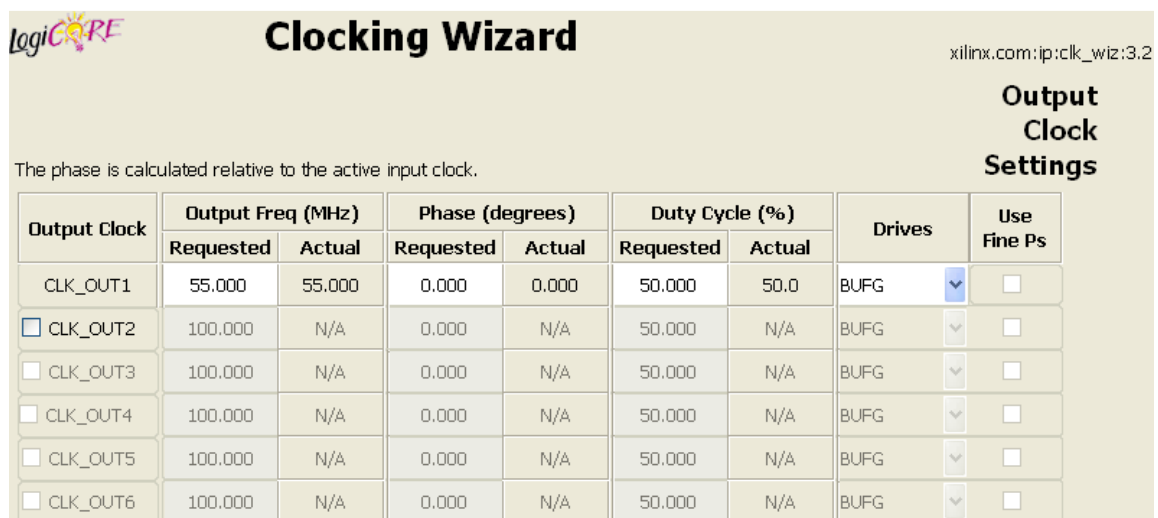
- ☒ UI
- ☐ PS

Input Clock Information

Input Clock	Input Freq (MHz)		Input Jitter	Source
	Value	Valid Range		
primary	100.000	5.000 - 500.000	0.010	Single ended clock capable pin

Figure 2. Clocking Features / Input Clocks Window

- 1-1-8. In the **Output Clock Settings** window (Figure 3), enter **55 MHz** as the output frequency and click **Next**.



Clocking Wizard xilinx.com:ip:clk_wiz:3.2

Output Clock Settings

The phase is calculated relative to the active input clock.

Output Clock	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Use Fine Ps
	Requested	Actual	Requested	Actual	Requested	Actual		
<input checked="" type="checkbox"/> CLK_OUT1	55.000	55.000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>

Figure 3. Output Clock Settings Window

- 1-1-9. In the **I/O and Feedback** dialog, uncheck **RESET** box and click **Next**.

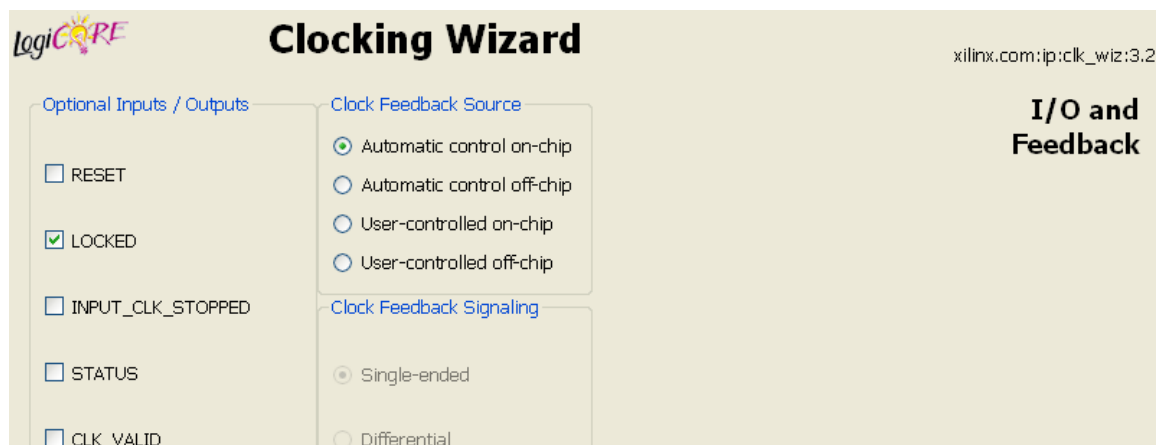


Figure 4. I/O and Feedback Window

1-1-10. Click **Next** three times and then click **Generate**

Notice that a new file (**my_dcm.xco**) is added as a Source in the project (**Figure 5**). This source file will not be included in the design hierarchy until the component has been instantiated into one of the HDL source files. You will do this in the next step.

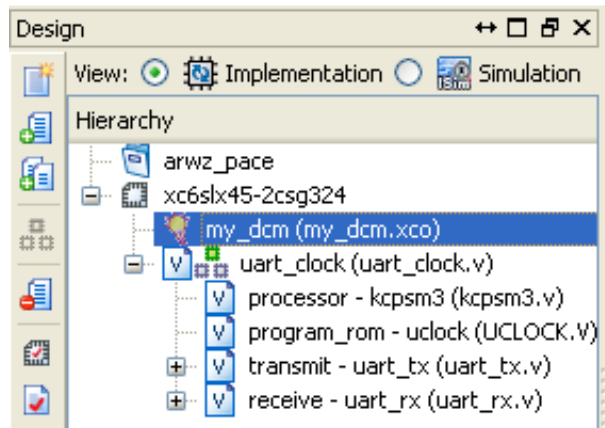


Figure 5. DCM listed in design hierarchy

Instantiate the DCM into a Verilog Design

Step 2a

VHDL users can skip to the next section ... Step 2b

Now that you have created the necessary files, you will instantiate the DCM component into your design. You will copy-and-paste the text from the Instantiation Template into `uart_clock.v` and connect the signals.

2a-1. In the **Hierarchy** in Project window, double-click **uart_clock.v** to open the source code in the text editor.

2a-1-1. Select **my_dcm.xco** in the **Hierarchy** in Project window.

2a-1-2. In the Processes for Source window, double-click **View HDL Instantiation Template** to open the instantiation template in the text editor.

2a-1-3. From the Instantiation Template *my_dcm.veo*, copy the **instantiation module** and paste into *uart_clock.v* under the comment *//Insert DCM component here*.

2a-1-4. Make sure complete the instantiation by filling in the port connections as follows:

```

170 // Insert DCM component here
171 my_dcm instance_name
172 (// Clock in ports
173   .CLK_IN1      (clk),      // IN
174   // Clock out ports
175   .CLK_OUT1     (clk55MHz),  // OUT
176   // Status and control signals
177   .LOCKED       (lock));    // OUT
178

```

Note: the *uart_clock.v* design has been updated so all design modules are clocked using the *clk55MHz* signal.

2a-1-5. Add a signal declaration for the 55 MHz output of the DCM under the comment *// Signals for DCM*, as follows:

```
wire clk55MHz;
```

note: the *uart_clock.v* design has been updated so all design modules are clocked using the *clk55MHz* signal.

2a-1-6. Add an output pin for *lock* in the top-level module as follows:

```

module uart_clock
(
    tx,
    rx,
    alarm,
    clk,
    lock);

output tx;
input rx;
output alarm;
input clk;
output lock;

```

note: this **lock** output pin will drive the *led1* on the Nexys3 board, and is driven by the *lock* signal on the DCM. This will indicate to the user that the DCM has successfully locked onto the 100 MHz clock signal from the on-board oscillator.

2a-1-7. Click **File → Save** to save the file.

Notice that the source file *my_dcm.xco* is now inserted into the correct place in the design hierarchy.

Instantiate the DCM into a VHDL Design

Step 2b

Verilog users can skip to the next section ... Step 3

Now that you have created the necessary files, you can instantiate the DCM component into your design. Copy-and-paste the text from the Instantiation Template into *uart_clock.vhd* and connect the signals.

2b-1. In the **Hierarchy** in *Project* window, double-click **uart_clock.vhd** to open the source code in the text editor.

2b-1-1. Select **my_dcm.xco** in the **Hierarchy** in *Project* window.

2b-1-2. In the **Processes for Source** window, double-click **View HDL Instantiation Template** to open the instantiation template in the text editor.

2b-1-3. From the Instantiation Template *my_dcm.vho*, copy the **component declaration** (begins at *COMPONENT my_dcm* and end after *END COMPONENT;*) and paste into *uart_clock.vhd* under the comment **-- Insert DCM component declaration here.**

2b-1-4. From the HDL Instantiation Template *my_dcm.vho*, copy the **component instantiation** (begin at *your_instance_name: my_dcm* until the end of the file) and paste into *uart_clock.vhd* under the comment **-- Insert DCM component instantiation here.**

2b-1-5. Complete the instantiation by filling in the port connections as follows:

```

193      -- Insert DCM component instantiation here
194      your_instance_name : my_dcm
195      port map
196      (
197          -- Clock in ports
198          CLK_IN1 => clk,
199          -- Clock out ports
200          CLK_OUT1 => clk55MHz,
201          -- Status and control signals
202          LOCKED => lock);

```

2b-1-6. Add a signal declaration for the 55 MHz output of the DCM under the comment **-- Signals for DCM**, as follows:

```
signal clk55MHz : std_logic;
```

2b-1-7. Add an output pin for lock in the entity as follows:

```

entity uart_clock is
  Port (
    tx : out std_logic;
    rx : in std_logic;
    alarm : out std_logic;
    clk : in std_logic;
    lock : out std_logic
  );
end uart_clock;

```

note: this **lock** output pin will drive the led0 on the Nexys3 board, and is driven by the lock signal

on the DCM. This will indicate to the user that the DCM has successfully locked onto the 100 MHz clock signal from the on-board oscillator.

2b-1-8. Click **File** → **Save** to save the file.

Notice that the source file `my_dcm.xco` is now inserted into the correct place in the design hierarchy.

Assign Pin Locations

Step 3

3-1. In this step, you will use **PlanAhead** to assign locations to the pins in the design. You will then verify in the **Pad report** that the pins have actually been assigned after running **place & route**.

3-1-1. In the **Hierarchy** window, select the top-level design file `uart_clock.vhd/.v`

3-1-2. In the **Processes** window, expand **User Constraints** and double-click **I/O Pin Planning (PlanAhead) - Pre-Synthesis** to open PlanAhead.

Click **yes** when asked to add a UCF file to the project.

Click **Close** to close the introductory window if it shows up (You can uncheck the box if you do not want to see this introductory window every time you invoke PlanAhead from ISE).

For VHDL users: At this stage the software may report 8 errors. Ignore them as they are not created by your design.

3-1-3. In the **I/O Ports** window select **tx** signal. In the **I/O Port Properties** window type **N18** in the site field and click **Apply**. Select **configure** tab and select **LVC MOS33** as the I/O standard. Click **Apply** button to assign these properties. Similarly enter the pin constraints for other pins in the design as shown in the **I/O Ports** list window (**Figure 6**).

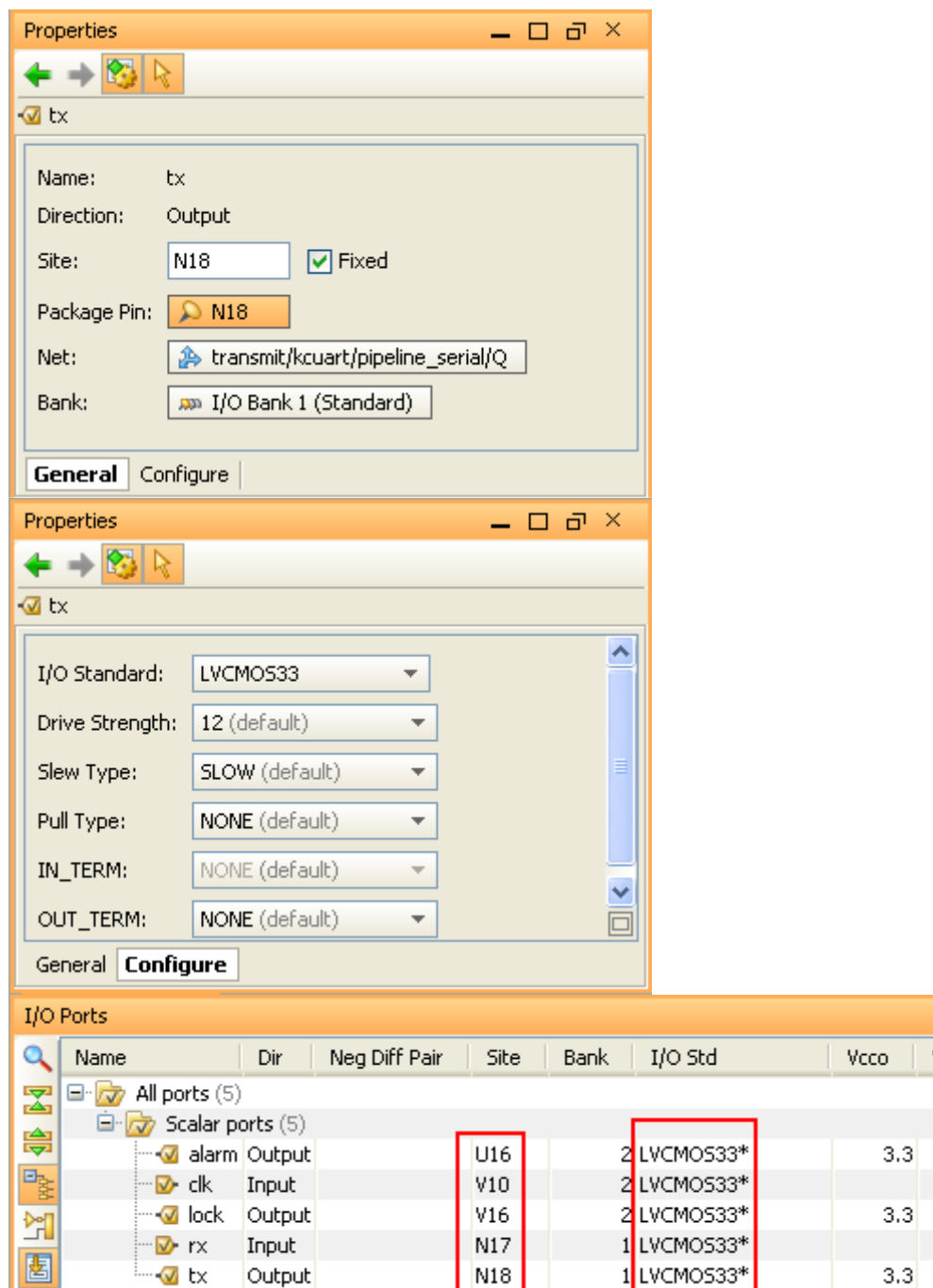



Figure 6. Enter Pin Location Constraints

Listed below are descriptions for the I/O signals that you have just assigned. The complete pinout for the Nexys3 board can be found in the user manual

- clk : connected to 100 MHz oscillator
- alarm : connected to led0
- lock : connected to led1
- rx : connected to pin that receives serial data from XR21V1410L16
- tx : connected to pin that transmits serial data to XR21V1410L16

3-2. View your pin assignments in relation to the internal logic.

- 3-2-1.** In the I/O Ports window select a pin (e.g. rx) and click on Fit Selection button (). Notice that the FloorPlan window shows the selected I/O and the Device Package window shows the (Figure 7).

The colored bar alongside the I/O pins indicates which pins are in the same I/O bank. You can easily see which pins have been assigned to the same bank.

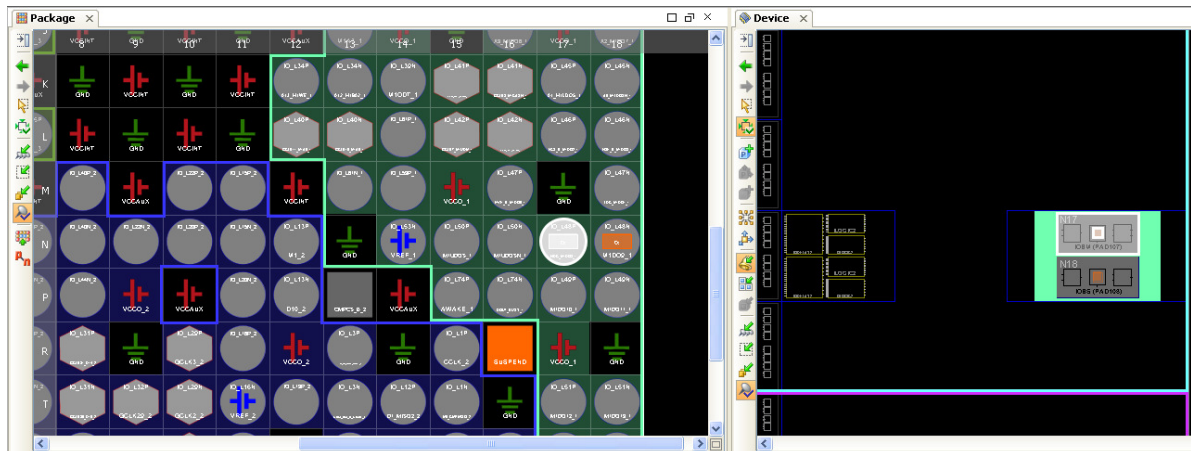


Figure 7. Device and Package Windows

- 3-2-2.** Click each I/O pin in the I/O port and observe the corresponding pin in the Device and Package windows when click on Fit Selection button.
- 3-2-3.** Click **File → Save Design** to save these pin placements.
- 3-2-4.** Click **File → Exit** to close PlanAhead. Click **OK**.
- 3-2-5.** Highlight the UCF file in the Project Navigator, expand **User Constraints** and double-click **Edit Constraints (Text)** to view the constraints created in the `uart_clock.ucf` file through PlanAhead. View the text version of the UCF file to confirm that the constraints were written to the file.

```
# PlanAhead Generated physical constraints

NET "alarm" LOC = U16;
NET "clk" LOC = V10;
NET "lock" LOC = V16;
NET "rx" LOC = N17;
NET "tx" LOC = N18;

# PlanAhead Generated IO constraints

NET "alarm" IOSTANDARD = LVCMOS33;
NET "clk" IOSTANDARD = LVCMOS33;
NET "lock" IOSTANDARD = LVCMOS33;
NET "rx" IOSTANDARD = LVCMOS33;
NET "tx" IOSTANDARD = LVCMOS33;
```

Figure 8. UCF

3-2-6. Select the top-level design file **uart_clock.vhd/.v** in the **Hierarchy** in Project window.

3-2-7. In the **Processes of Source** window, expand the **Implement Design** process, expand **Place & Route**, expand **Back-Annotate Pin Locations** and double-click **View Locked Pin Constraints**

The Project Navigator automatically determines which processes must be run, and it will open the report once the Place & Route process is completed.

3-2-8. Scroll down in the report and confirm that the pin numbers for the I/O signals match the assignments you made.

You can also view the pin assignments by clicking on Pinout Report under Design Overview section of the Design Summary window.

3-2-9. Double-click on **Generate Programming** File to generate bitstream file.

Test the Design in Hardware

Step 4

4-1. Configure and start a Hyper Terminal session. Connect and power the board. Generate the bitstream and configure the FPGA. Verify operation of the real-time UART clock in hardware.

4-1-1. Open a Hyper Terminal session by going to **Start → All Programs → Accessories → Communications → Hyper Terminal**.

4-1-2. Give the session a name, click **OK**, and specify COM port connection (ie. COM1).

4-1-3. Click the Configure button and specify the following parameters for the port settings. Click **OK** when finished.

- Baud rate of 38400, 8 data bits, No parity bits, 1 stop bit, No flow control

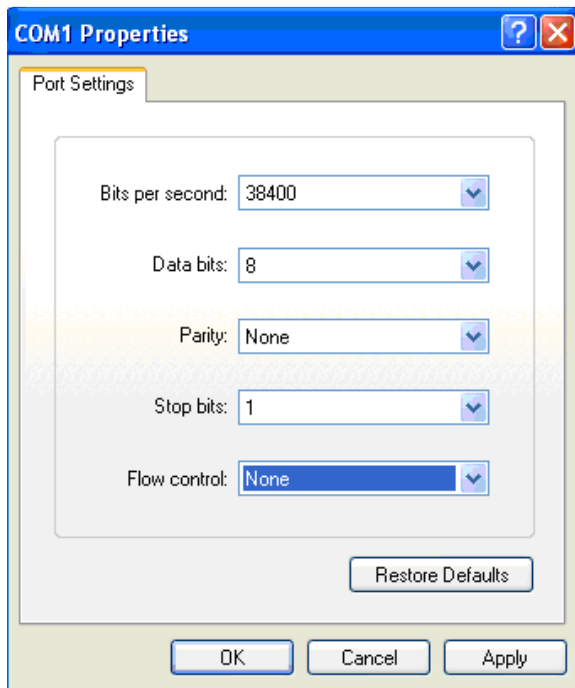


Figure 9. Settings for Serial Port Communications

- 4-1-4. Click the **Settings** tab, the **ASCII Setup** tab and then click so that a check mark appears next to the **Append line feeds to incoming line ends** option, and then click **OK**. Click **OK** again to exit the properties dialogue.

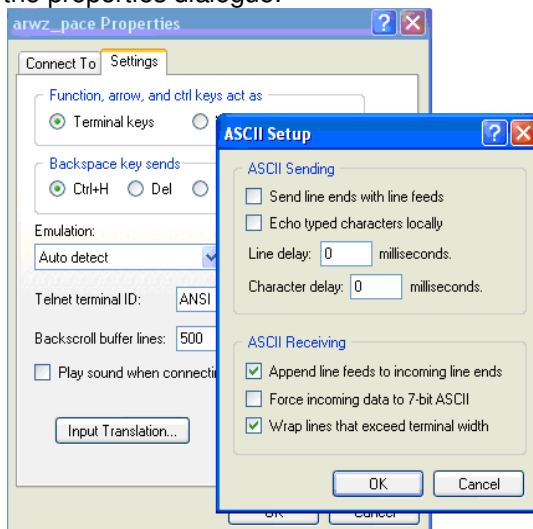



Figure 10. ASCII Settings for Serial Port Connection

- 4-1-5. Connect the cables (power, USB, and rs232) and power the board.
- 4-1-6. Select **uart_clock.vhd/.v**, expand **Configure Target Device**, and double-click on **Manage Configuration Project (iMPACT)**.

- 4-1-7. When the **iMPACT** opens, double-click on **Boundary Scan** ( **Boundary Scan**) in **iMPACT Flow** window.
- 4-1-8. Click **Initialize Chain** button. Click **YES** ,browse to the project directory and select **uart_clock.bit** file for the xc6SLX16. Next, click **NO** as we do not want to attach the PROM files. And then click **OK**.

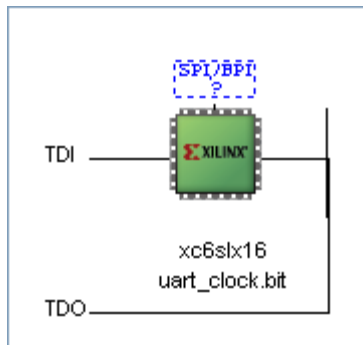


Figure 12. JTAG Chain with assigned configuration files

- 4-1-9. Right-click on the xc6SLX16 in the iMPACT window, select **Program** and click **OK**.

Note: You should now see the KCPSM3> prompt in the Hyper Terminal window. If not, then hit Enter key.

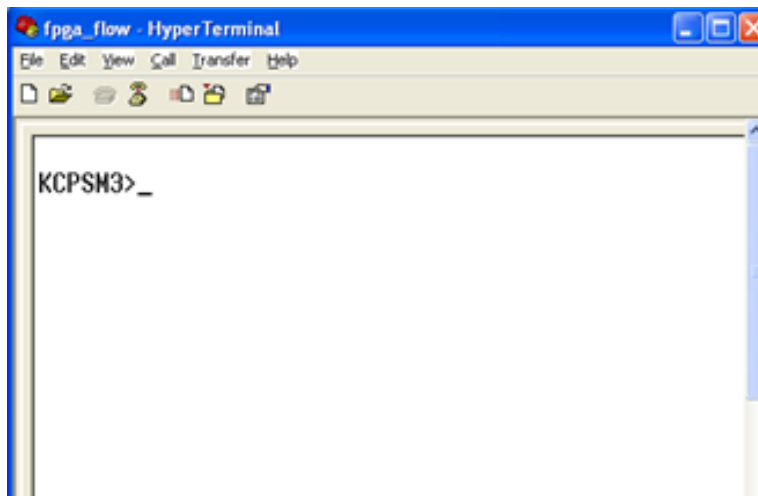


Figure 13. Serial Communication with PicoBlaze

Operate the UART Real-Time Clock

Step 5

- 5-1. You will issue commands to operate the UART real-time clock, as specified in the **UART_real_time_clock.pdf** file.

- 5-1-1. Enter the command “time” at the command prompt to display the current time in the form of hh:mm:ss.

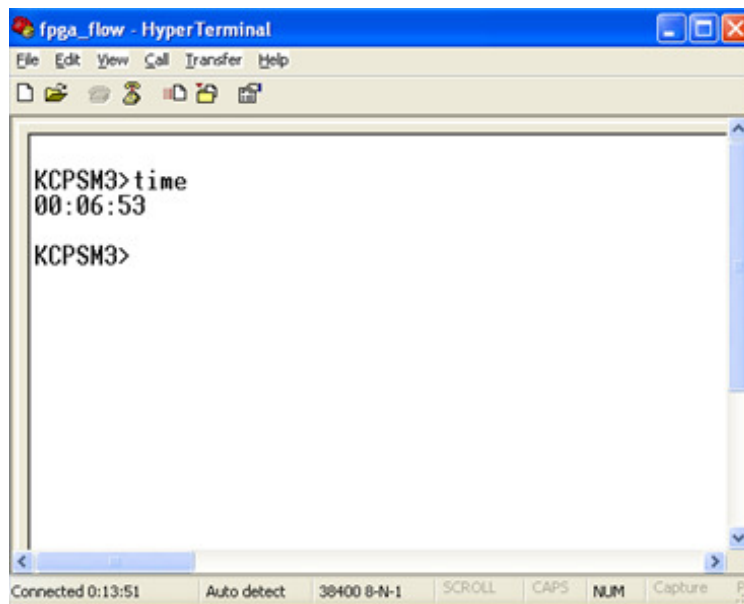


Figure 14. Display of Current Time

- 5-1-2. Enter the command “alarm” at the command prompt to display the current alarm time in the form of hh:mm:ss.

Note: the alarm is inactive.

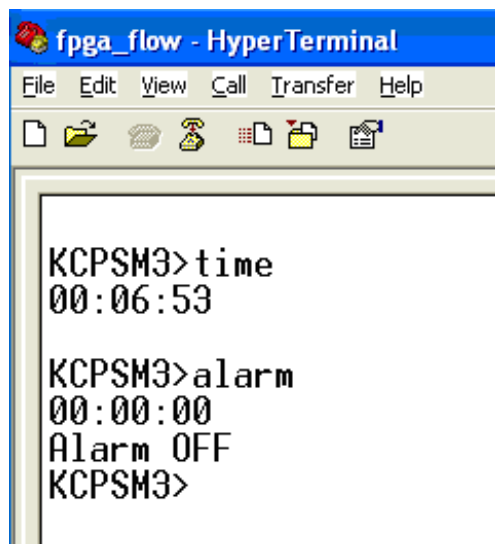


Figure 15. Display of alarm time and status

- 5-1-3. Enter the command “alarm on” for the alarm to become active.
- 5-1-4. Enter the command “alarm 00:00:30” to set the alarm to 30 seconds.
- 5-1-5. Enter the command “time 00:00:00” to set the time.

Note: You should notice that led0 on the Nexys3 board will light up once the alarm goes off.

5-1-6. Enter the command “alarm off” to shut off the alarm.

You should notice that the led0 turns off, indicating that the alarm has been disarmed.

5-1-7. Close iMPACT program without saving the project and close ISE.

Conclusion

In this lab, you used the Architecture Wizard to configure a DCM component and PlanAhead tool to assign top-level signals to specific pins. You completed the design by instantiating the component into the design. Finally, you tested the design in hardware by downloading the bitstream using iMPACT and entering alarm clock commands via Hyper Terminal.