# Architectural Wizard and CORE Generator

## Introduction

Today's Xilinx FPGAs contain many more resources than basic, LUT, CLB, IOB, and routing. The FPGAs are now being used to implement much more complex digital circuits compared to glue logic when they were invented. Some complex architectural resources, such as clocking, must be configured and instantiated instead of inferred. There are also commonly used complex circuits, such as the Reed-Solomon decoder and tools so that a designer does not have to "reinvent the wheel" and develop the basic features on their own. This lab introduces architectural wizard and CORE Generator tools available through the IP Catalog. *Please refer to the PlanAhead tutorial on how to use the PlanAhead tool for creating projects and verifying digital circuits*.

## Objectives

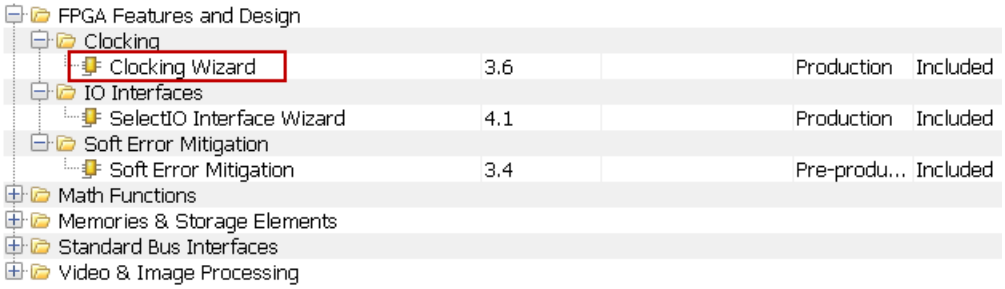
After completing this lab, you will be able to:

- Use the Architectural Wizard to configure clocking resource
- Use the CORE Generator tool to configure and use counters and memories

## Architectural Wizard Part 1

Some specialized and advanced architectural resources can efficiently be utilized when configured and instantiated properly instead of inferring them. Depending on the FPGA family being used, the number and types of such resources vary. In the Spartan-6LX family, clocking, SelectIO, and soft error mitigation (SEM) resources are supported by the architectural wizard. In the Spartan-6LXT family, an additional architectural resource, GTP Transceiver, is available. These resources are accessed under the IP Catalog capability of the PlanAhead tool.

On the Nexys3 board, a 100 MHz clock source is available which is connected to the V10 pin of the FPGA. This clock source can be used to generate a number of clocks of different frequencies and phase shifts. This is done by using architectural resources called Digital Clock Manager (DCM) and Phase Locked Loop (PLL) of the Spartan-6LX family FPGA. The clocking source generator can be invoked by double-clicking the Clocking Wizard entry under the Clocking sub-folder of the FPGA Feature and Design folder of the IP Catalog.

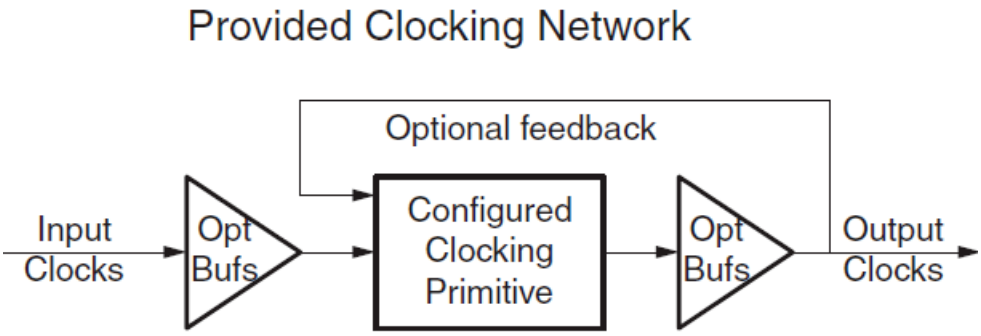



The wizard makes it easy to create HDL source code wrappers for clock circuits customized to your clocking requirements. The wizard guides you in setting the appropriate attributes for your clocking primitive, and also allows you to override any wizard-calculated parameter. In addition to providing an HDL wrapper for implementing the desired clocking circuit, the Clocking Wizard also delivers a timing parameter summary generated by the Xilinx timing tools for the circuit. The main features of the wizard include:

- Accepts up to two input clocks and up to seven output clocks per clock network
- Automatically chooses correct clocking primitive for a selected device

- Automatically configures clocking primitive based on user-selected clocking features
- Automatically implements overall configuration that supports phase shift and duty cycle requirements
- Optionally buffers clock signals

The functionality of the generated core can be viewed as:




Suppose we want to generate a 5MHz clock which is in phase with 100 MHz input clock. Follow the steps below to achieve that:

Double-click on the Clocking Wizard. When the wizard opens, you will notice that there are six configurable pages (steps):

The first page has parameters related to input clock and clocking features. Uncheck the phase shift page, the input frequency value and range are given. Since the actual clock source is 100 MHz, we will keep the value to default.

Click **Next** to see the Page 2 parameters which are related to the output clocks and desired frequencies. Change the Requested Output Frequency to 1.000 MHz and notice that the actual frequency shown is 3.125 MHz as that is the slowest clock frequency that can be generated using the clocking resources. Change it to 5.000 MHz for now.

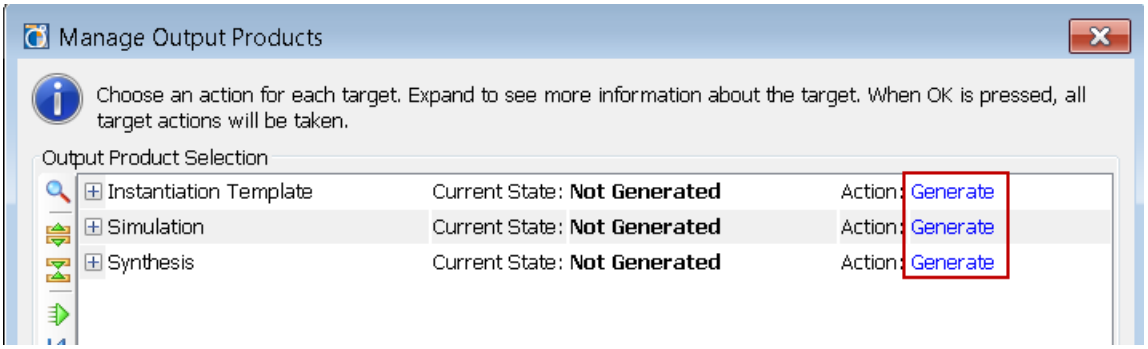
Click **Next** to see the Optional Inputs/Outputs configuration page. It shows RESET and LOCKED as the selected port. You can uncheck/check any of the ports and observe the block diagram changes on the left side of the wizard. We will keep the default as our circuits will have RESET input and we also like to see when the clock is stable.

Click **Next** to see Page 4. It will show that DCM_SP clock resource is used and the calculated parameters value. Keep everything as default.

Click **Next** to see Page 5 which shows Clock Summary and Port Naming. Keep everything as default.

Click **Next** and Page 6 will appear showing the files which will be generated. The important files are .veo (instantiation template file), .v (the source file), and .ucf (core constraints file). Click **Generate** and the core file (.xci) will be generated and added to the project.

Back in the PlanAhead GUI, for non-architectural cores, the above mentioned files are automatically generated. However, for the architectural sources the files have to be explicitly generated. Select the .xci entry in the Sources tab, right-click, and select **Generate Output Products…** option. A form will appear which show what is generated and what can be generated as shown below.




If the Generate drop down menu option is selected for the Instantiation Template, after OK is clicked, the menu will generate the .VHO and related files. Pay attention to the Output product location at the bottom of the menu. These files are accessible through the IP Sources tab. The .VHO file is the VHDL instantiation template for the generated IP. Here is an example of the .VHO file content:

```
-- The following code must appear in the VHDL architecture header:
component clk_wiz_v3_6_0
port
 (-- Clock in ports
  CLK_IN1           : in     std_logic;
  -- Clock out ports
  CLK_OUT1          : out    std_logic;
  -- Status and control signals
  RESET             : in     std_logic;
  LOCKED            : out    std_logic
 );
end component;

…

-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.
your_instance_name : clk_wiz_v3_6_0
  port map
   (-- Clock in ports
    CLK_IN1 => CLK_IN1,
    -- Clock out ports
    CLK_OUT1 => CLK_OUT1,
    -- Status and control signals
    RESET  => RESET,
    LOCKED => LOCKED);
```

**1-1. Design a one-second pulse generator. Use the clocking wizard to generate 5 MHz clock, dividing it further by a clock divider (written in behavioral modeling) to generate one second period signal. The steps on using the Clocking Wizard described above (and the resultant instantiation template) can be used for this exercise. Use the on-board clock source of 100 MHz, the BTNU button to reset the circuit, SW0 as enable, LED0 to output the generated one second signal, and LED7 to output the DCM lock signal. Go through the design flow, generate the bitstream, and download it into the Nexys3 board. Verify the functionality. *Make sure you specify the language of the project to VHDL to generate the appropriate output files.***

Since the 7-segment displays on the Nexys3 board use common cathodes and a particular display is illuminated by asserting the corresponding anode pin, a scanning circuit is required to display information (digits) on more than one display. This circuit should drive the anode signals and corresponding cathode patterns of each digit in a repeating, continuous succession, at an update rate that is faster than the human eye can detect. In order for each of the digits to appear bright and continuously illuminated, all desired digits should be driven once every 1 to 16ms, for a refresh frequency of 1 KHz to 60 Hz. If the update or "refresh" rate is slowed to around 45 Hz, most people will begin to see the display flicker.

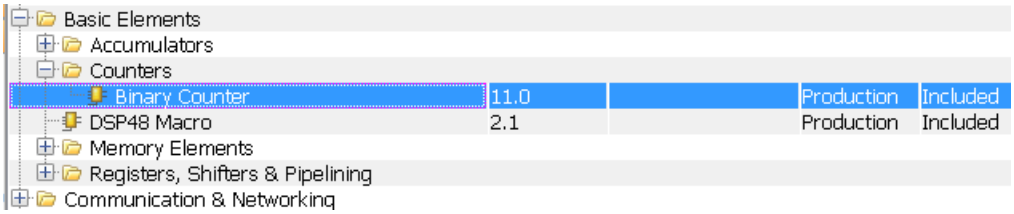
**1-2. Modify the design of Lab2_2_1 (binary to BCD converter) to display the 4-bit binary inputs converted to BCD values on two 7-segment displays (instead of one 7-segment and one LED). Use the 100 MHz clock source to generate a 5 MHz clock and the appropriate clock divider circuit to drive the two 7-segment displays with a refresh rate of about 500 Hz. Generate the bitstream and download it into the Nexys3 board to verify the functionality.**

## CORE Generator System Part 2

The CORE Generator system, available in the IP Catalog of the PlanAhead tool allows you to configure and generate various functional cores. In IP Catalog, the cores are grouped according to functionality which varies from simple basic cores such as an adder to quite complex cores such as the MicroBlaze processor. It also covers cores of various application areas ranging from Automotive to Video and Image Processing.

The process of configuring and generating the cores is similar to the Architectural Wizard. The cores will use various resources including LUT, CLB, DSP48, BRAM etc. as needed. Let us look at how to configure and generate a counter core.

The Binary Counter core generation can be started by double-clicking the Binary Counter entry under the Counters sub-folder located under the Basic Elements branch of the IP catalog.




When invoked, you will see only one page of the configuration. The configuration parameters of the core include:

Implement Using: Fabric or DSP48
Output Width
Increment Value
Loadable, Restrict Count, Count Mode (Up, Down, UPDPWN), Synchronous Clear, Clock Enable and various other settings.

The designer can select the desired functionality and click on the Generate button. Note that unlike the architectural wizard, which requires an explicit instantiation template generation step, the non-architectural cores automatically generate the instantiation template file. The synthesized output and simulation files still need to be explicitly generated.

**2-1. Use CORE Generator system to generate a simple 4-bit counter core which counts up from 0 to 9 (Hint: Use Threshold output when configuring the counter core). Instantiate it two times to create a two digit BCD counter which counts up every one second. Use Architectural Wizard to generate a 5 MHz clock and then use behavioral modeling to generate 1 Hz precise signal to drive the counters. Display the result on the two 7-segment displays. The design input will be a 100 MHz clock source, a reset signal using the BTNU button, and an enable signal using SW0. Verify the design functionality in hardware using the Nexys3 board.**

## Conclusion

In this lab, you learned about the architectural wizard and the CORE Generator system available in the IP Catalog of the PlanAhead tool. You used the architectural wizard to generate a 5 MHz clock and the CORE Generator to generate a counter. The CORE Generator system is a powerful tool providing various functional blocks enabling higher productivity.