Architectural Wizard and IP Catalog

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 are provided by tools so that a designer does not have to "reinvent the wheel." This lab introduces the Architectural Wizard and the IP generator tools available through the IP Catalog. *Please refer to the Vivado tutorial on how to use the Vivado tool for creating projects and verifying digital circuits*.

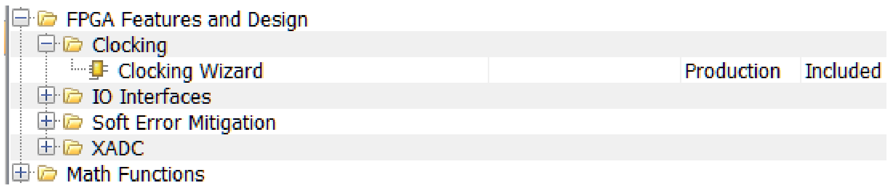
Objectives

After completing this lab, you will be able to:
- Use the Architectural Wizard to configure clocking resource
- Use the IP Catalog 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 Artix-7 family, clocking, SelectIO, soft error mitigation, and XADC resources are supported by the architectural wizard. These resources are accessed under the FPGA Features and Design folder of the IP Catalog tool.

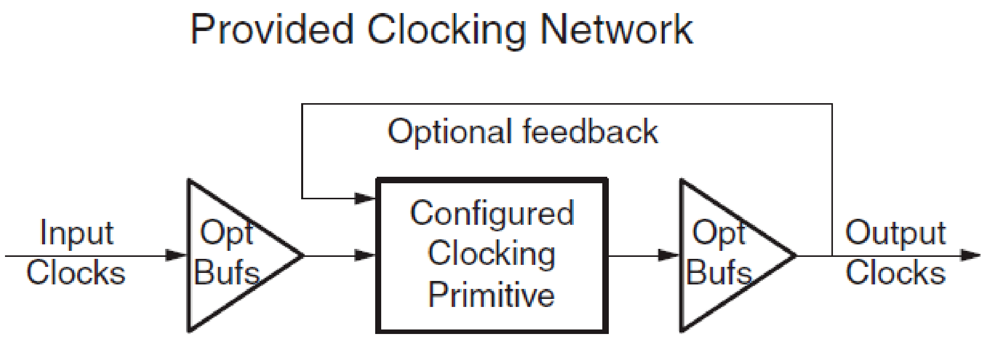
On the Basys3 and the Nexys4 DDR board, a 100 MHz clock source is available which is connected to the W5 (Basys3) and E3 (Nexys4 DDR) pin of the FPGA. This clock source can be used to generate a number of clocks of different frequencies and phase shifts. This is done by using architectural resources called Digital Clock Manager (DCM) and Phase Locked Loop (PLL) of the Artix-7 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 entry. When the wizard opens, you will notice that there are five tabs.

The first tab, titled Clocking Options, has parameters related to input clock and clocking features, the input frequency value and range. Since the actual clock source is 100 MHz, we will keep the value to default.

The second tab, titled Output Clocks, has parameters which are related to the output clocks and desired frequencies. Change the Requested Output Frequency to 1.000 MHz and notice that it shows the frequency in red indicating something is wrong. Move the mouse over the same and you will see a pop-up indicating that the actual frequency range for this device is 4.687 MHz to 800 Mhz. Change it to 5.000 MHz for now. We will unclick the **RESET** from the output clock and create an asynchronous reset.

The third tab, titled MMCM Settings, shows the calculated settings. You can check the box and override the values as long as you know what they will do and how they will affect the design. We like to see when the clock is stable.

The fourth tab, titled Port Renaming, allows you to change the port names. We will use the default names.

The fifth and last tab, titled Summary, shows you the summary.

Click **OK** and then click **Generate** to generate the files used for synthesis, implementation, and simulation.

The files, including the instantiation file, are accessible through the IP Sources tab. Here is an example of the .veo file content.

```
clk_5MHz instance_name
 (// Clock in ports
  .clk_in1(CLK_IN1),          // IN
  // Clock out ports
  .clk_out1(CLK_OUT1),        // OUT
  // Status and control signals
  .reset(RESET),              // IN
  .locked(LOCKED));           // OUT
```

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, Q, the generated one second signal, and LED15 to output the DCM lock signal. Go through the design flow, generate the bitstream, and download it into the Basys3 or the Nexys4 DDR board. Verify the functionality.

Since the 7-segment displays on the Basys3 or the Nexys4 DDR 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.

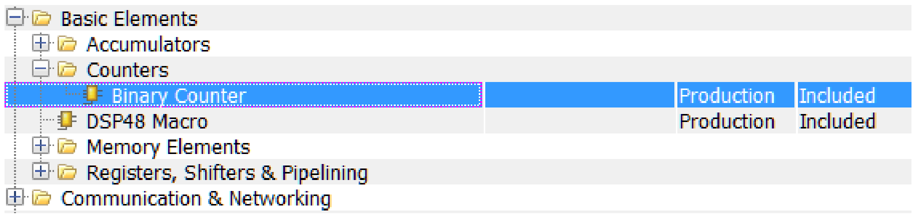
1-2. Modify the design of Lab2_2_1 (binary to BCD converter) to display the 4-bit binary inputs converted to BCD values on two 7-segment displays (instead of one 7-segment and one LED). Use the 100 MHz clock source to generate a 5 MHz clock and the appropriate clock divider circuit to drive the two 7-segment displays with a refresh rate of about 500 Hz. Generate the bitstream and download it into the Basys3 or the Nexys4 DDR board to verify the functionality.

IP Catalog Part 2

The IP Catalog of the Vivado 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 two tabs configuration. The configuration parameters of the core on the first tab, titled Basic, include:

Implement Using: Fabric or DSP48
Output Width
Increment Value
Loadable, Restrict Count, Count Mode (Up, Down, UPDPWN), Threshold

The second tab, titled Control, include

Synchronous Clear, Clock Enable and various other settings.

The designer can select the desired functionality and click on OK to generate the IP.

2-1. Use the IP Catalog 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 Basys3 or the Nexys4 DDR board.

Conclusion

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