

LogiCORE IP Clocking Wizard v3.2

Getting Started Guide

UG521 June 22, 2011



The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available “AS IS” and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

© Copyright 2008-2011. Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/17/08	1.1.1	Xilinx Confidential DRAFT. Approved for external release under NDS only.
04/24/09	1.1	Initial major release. Supports core version 1.1 and Xilinx tools 11.1.
06/24/09	1.2	Supports core version 1.2 and Xilinx tools 11.2.
09/16/09	1.3	Supports core version 1.3 and Xilinx tools 11.3.
12/02/09	1.4	Supports core version 1.4 and Xilinx tools 11.4.
04/19/10	1.5	Supports core version 1.5 and Xilinx tools 12.1.
07/23/10	1.6	Supports core version 1.6 and Xilinx tools 12.2.
09/21/10	1.7	Supports core version 1.7 and Xilinx tools 12.3
12/14/10	1.8	Supports core version 1.8 and Xilinx tools 12.4
03/01/11	3.1	Supports core version 3.1 and Xilinx tools 13.1
06/22/11	3.2	Supports core version 3.2 and Xilinx tools 13.2. Added new 7 series architectures.

Table of Contents

Revision History	2
Preface: About This Guide	
Additional Resources	5
List of Acronyms	5
Chapter 1: Introduction	
System Requirements	7
About the Core	7
Recommended Design Experience	7
Additional Core Resources	7
Technical Support	8
Feedback	8
Clocking Wizard	8
Document	8
Chapter 2: Core Architecture	
Input Clocks	9
Primitive Instantiation	9
Feedback	9
Output Clocks	10
I/O Signals	10
Chapter 3: Generating the Core	
Clocking Features	15
Selecting Clocking Features	16
Clock Manager Type (Primitive Selection)	16
Configuring Input Clocks	17
Output Clock Settings	18
Configuring Output Clocks	18
Feedback and I/O	19
Selecting Optional Ports	20
Choosing Feedback	20
Primitive Overrides	20
Overriding Calculated Parameters	21
User Provided Comment	21
Clock Summary (First Page)	22
Input Clocking Summary	22
Output Clocking Summary	22
Port Names	22

Core Summary (Second Page)	23
Resource Estimate Summary	23
XPower Estimator Summary	23
File Report Summary	23

Chapter 4: Detailed Example Design

Directory and File Contents	26
<project directory>	26
<project directory>/<component name>	26
<component name>/example design	27
<component name>/doc	27
<component name>/implement	27
implement/results	27
<component name>/simulation	28
simulation/functional	28
simulation/timing	28
Implementation Scripts	28
Simulation Scripts	29
Functional Simulation	29
Timing Simulation	29
Example Design	30
Customizing the Example Design	30
Demonstration Test Bench	30

Appendix A: Migration Guide

Differences between the Clocking Wizard and the Legacy DCM and PLL Wizards	33
Primitive Selection	33
Symbol Pin Activation	33
Parameter Override	33
Port Display Conventions	34
Visibility of Clock Ports	34
GUI Information Gathering Order	34

About This Guide

The *Clocking Wizard Getting Started Guide* provides information about the Xilinx® LogiCORE™ IP Clocking Wizard. This guide walks you through using the graphical user interface (GUI) and explains how to use the sample design and test bench that are provided.

Additional Resources

To find additional documentation, see the Xilinx website at:

www.xilinx.com/support/documentation/index.htm.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

www.xilinx.com/support

List of Acronyms

Acronym	Spelled Out
DCM	Digital Clock Manager
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IES	Incisive Enterprise Simulator
IOB	Input/Output Block
IP	Intellectual Property
ISE®	Integrated Software Environment
ISim	ISE Simulator
MMCM	Mixed-Mode Clock Manager
NGD	Native Generic Database
PLL	Phase-Locked Loop
RTL	Register Transfer Level
VCO	Voltage Controlled Oscillator
VCS	Verilog Compiled Simulator (Synopsys)

Acronym	Spelled Out
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
XST	Xilinx Synthesis Technology

Introduction

This chapter introduces the Clocking Wizard core and provides related information, including recommended design experience, additional resources, technical support, and ways of submitting feedback to Xilinx. The Clocking Wizard core generates source Register Transfer Level (RTL) code to implement a clocking network matched to your requirements. Both Verilog and VHDL design environments are supported.

System Requirements

For a list of System Requirements, see the [ISE Design Suite 13: Release Notes Guide](#).

About the Core

The Clocking Wizard is an ISE® CORE Generator™ IP core, included with the latest ISE Design Suite release in the Xilinx® Download Center. The core is licensed under the terms of the Xilinx End User License and no FLEX license key is required.

Recommended Design Experience

The Clocking Wizard is designed for users with any level of experience. Using the wizard automates the process of creating your clocking network and is highly recommended. The wizard guides users to the proper primitive configuration and allows advanced users to override and manually set any attribute. Although the Clocking Wizard provides a fully verified clocking network, understanding the Xilinx clocking primitives will aid users in making design trade-off decisions.

Additional Core Resources

For details about the Clocking Wizard, perform these steps:

1. Go to the Clocking Wizard section of the Architecture Wizards page: (www.xilinx.com/products/design_resources/conn_central/solution_kits/wizards/index.htm#clocking).
2. In the Clocking Wizards table, select **Documents** link in the column labeled Documents. The Clocking Wizard documents are listed.

Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team with expertise using the Clocking Wizard core.

Xilinx provides technical support for use of this product as described in the *Clocking Wizard Getting Started Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

Feedback

Xilinx welcomes comments and suggestions about the Clocking Wizard and the accompanying documentation.

Clocking Wizard

For comments or suggestions about the Clocking Wizard, submit a WebCase from www.xilinx.com/support/clearexpress/websupport.htm. Be sure to include the following information:

- Product name
- Core version number
- Explanation of your comments

Document

For comments or suggestions about the Clocking Wizard core, submit a WebCase from www.xilinx.com/support/clearexpress/websupport.htm. Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

Core Architecture

The Clocking Wizard generates source code HDL to implement a clocking network. The generated clocking network typically consists of a clocking primitive (DCM_SP, DCM_CLKGEN, PLL_BASE, or MMCM_ADV) plus some additional circuitry which typically includes buffers and clock pins. The network is divided into segments as illustrated in [Figure 2-1](#). Details of these segments are described in the following sections.

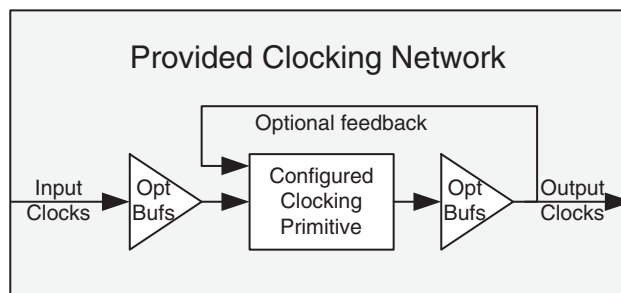


Figure 2-1: **Provided Clocking Network**

Input Clocks

Up to two input clocks are available for the clocking network. Buffers are optionally inserted on the input clock paths based on the buffer type that is selected.

Primitive Instantiation

The primitive, either user or wizard selected, is instantiated into the network. Parameters on primitives are set by the wizard, and can be overridden by you. Unused input ports are tied to the appropriate values. Unused output ports are labeled as such.

Feedback

If phase alignment is not selected, the feedback output port on the primitive is automatically tied to the feedback input port. If phase alignment with automatic feedback is selected, the connection is made, but the path delay is matched to that of CLK_OUT1. If user-controlled feedback is selected, the feedback ports are exposed.

Output Clocks

Buffers that are user-selected are added to the output clock path, and these clocks are provided to the user.

I/O Signals

[Table 2-1](#) describes the input and output ports provided from the clocking network. All ports are optional, with the exception that at least one input and one output clock are required. Availability of ports is controlled by user-selected parameters. For example, when Dynamic Reconfiguration is selected, only those ports related to Dynamic Reconfiguration are exposed to the user. Any port that is not exposed is either tied off or connected to a signal labeled *unused* in the delivered source code. Not all ports are available for all devices or primitives; for example, Dynamic Reconfiguration is a feature only available in Virtex®-6 FPGA Mixed-Mode Clock Manager (MMCM), 7 series FPGA MMCME2, PLLE2 or Spartan®-6 FPGA DCM_CLKGEN primitives.

For details about specific clocking primitive features, see the user guide for the associated FPGA device.

Table 2-1: Clock Network Input and Output Port Descriptions

Port ⁽¹⁰⁾	I/O	Description
Input Clock Ports ⁽¹⁾		
CLK_IN1	Input	Clock in 1: Single-ended primary input clock port. Available when single-ended primary clock source is selected.
CLK_IN1_P	Input	Clock in 1, Positive and Negative: Differential primary input clock port pair. Available when a differential primary clock source is selected.
CLK_IN1_N		
CLK_IN2	Input	Clock in 2: Single-ended secondary input clock port. Available when a single-ended secondary clock source is selected.
CLK_IN2_P	Input	Clock in 2, Positive and Negative: Differential secondary input clock port pair. Available when a differential secondary clock source is selected.
CLK_IN2_N		
CLK_IN_SEL	Input	Clock in Select: When 1, selects the primary input clock. When 0, the secondary input clock is selected. Available when two input clocks are specified.
CLKFB_IN	Input	Clock Feedback in: Single-ended feedback in port of the clocking primitive. Available when user-controlled on-chip, user controller-off chip, or automatic control off-chip feedback option is selected.
CLKFB_IN_P	Input	Clock Feedback in: Positive and Negative: Differential feedback in port of the clocking primitive. Available when the automatic control off-chip feedback and differential feedback option is selected.
CLKFB_IN_N	Input	

Table 2-1: Clock Network Input and Output Port Descriptions (Cont'd)

Port (10)	I/O	Description
Output Clock Ports		
CLK_OUT1	Output	Clock Out 1: Output clock of the clocking network. CLK_OUT1 is not optional.
CLK_OUT1_CE	Input	Clock Enable: Chip enable pin of the output buffer. Available when BUFGCE or BUFHCE or BUFR buffers are used as output clock drivers.
CLK_OUT1_CLR	Input	Counter reset for divided clock output: Available when BUFR buffer is used as output clock driver.
CLK_OUT2-n	Output	Clock Out 2 - n: Optional output clocks of the clocking network that are user-specified. For an MMCM/MMCME2, up to seven are available. For a PLL, PLLE2 or DCM, up to six are available. For a DCM_CLKGEN, up to three are available.
CLK_OUT[2-n]_CE	Input	Clock Enable: Chip enable pin of the output buffer. Available when BUFGCE or BUFHCE or BUFR buffers are used as output clock drivers.
CLK_OUT[2-n]_CLR	Input	Counter reset for divided clock output: Available when BUFR buffer is used as output clock driver.
CLKFB_OUT	Output	Clock Feedback Out: Single-ended feedback port of the clocking primitive. Available when the user-controlled feedback or automatic control off chip with single-ended feedback option is selected.
CLKFB_OUT_P	Output	Clock Feedback Out: Positive and Negative: Differential feedback output port of the clocking primitive. Available when the user-controlled off-chip feedback and differential feedback option is selected.
CLKFB_OUT_N	Output	
Dynamic Reconfiguration Ports for MMCM		
DADDR[6:0]	Input	Dynamic Reconfiguration Address: Address port for use in dynamic reconfiguration; active when DEN is asserted.
DCLK	Input	Dynamic Reconfiguration Clock: Clock port for use in dynamic reconfiguration.
DEN	Input	Dynamic Reconfiguration Enable: Starts a dynamic reconfiguration transaction.
DI[15:0]	Input	Dynamic Reconfiguration Data in: Input data for a dynamic reconfiguration write transaction; active when DEN is asserted.
DO[15:0]	Output	Dynamic Reconfiguration Data Out: Output data for a dynamic reconfiguration read transaction; active when DRDY is asserted.

Table 2-1: Clock Network Input and Output Port Descriptions (Cont'd)

Port ⁽¹⁰⁾	I/O	Description
DRDY	Output	Dynamic Reconfiguration Ready: Completes a dynamic reconfiguration transaction.
DWE	Input	Dynamic Reconfiguration Write Enable: When asserted, indicates that the dynamic reconfiguration transaction is a write; active when DEN is asserted.
Dynamic Reconfiguration Ports for DCM_CLKGEN⁽²⁾		
PROGCLK	Input	Program Clock: Clock port for use in dynamic reconfiguration.
PROGEN	Input	Program Enable: Starts a dynamic reconfiguration transaction.
PROGDATA	Input	Program Data: Serial data stream to reprogram primitive settings.
PROGDONE	Output	Program Done: When asserted, indicates that the reconfiguration transaction is complete.
Dynamic Phase Shift Clock Ports⁽³⁾		
PSCLK	Input	Dynamic Phase Shift Clock: Clock for use in dynamic phase shifting.
PSEN	Input	Dynamic Phase Shift Enable: Starts a dynamic phase shift transaction.
PSINCDEC	Input	Dynamic Phase Shift increment/decrement: When 1, increments the phase shift of the output clock, when 0, decrements the phase shift.
PSDONE	Output	Dynamic Phase Shift Done: Completes a dynamic phase shift transaction.
Status and Control Ports⁽⁴⁾		
RESET	Input	Reset: When asserted, asynchronously clears the internal state of the primitive, and causes the primitive to re-initiate the locking sequence when released.
POWER_DOWN ⁽⁵⁾	Input	Power Down: When asserted, places the clocking primitive into low power state, which stops the output clocks.
STATUS[2:0] ⁽⁶⁾	Output	Status: Contains information about the state of the primitive. See the user guide for specific bit descriptions.
FREEZE ⁽⁷⁾	Input	Freeze: When asserted, prevents tap adjustment in the event that the input clock is lost.
INPUT_CLK_STOPPED ⁽⁸⁾	Output	Input Clock Stopped: When asserted, indicates that the selected input clock is no longer toggling.

Table 2-1: Clock Network Input and Output Port Descriptions (Cont'd)

Port ⁽¹⁰⁾	I/O	Description
LOCKED	Output	Locked: When asserted, indicates that the output clocks are stable and usable by downstream circuitry.
CLK_VALID ⁽⁹⁾	Output	Clock Output Valid: The CLK_VALID output is a logical combination of the DCM status ports which give the best indication that the input clock has been locked to, the DCM is functioning properly, and the output clocks are valid. When asserted, indicates the output clocks are valid.

Notes:

- At least one input clock is required; all designs have at least a CLK_IN1 or a CLK_IN1_P/CLK_IN1_N port. A secondary input clock is supported for Virtex-6 FPGAs only.
- Dynamic reconfiguration ports are available for 7 series FPGA MMCME2, PLLE2, Virtex-6 FPGA MMCM or Spartan-6 FPGA DCM_CLKGEN primitives.
- Dynamic phase shift ports are available for 7 series FPGA MMCME2, Virtex-6 FPGA MMCM or Spartan-6 FPGA DCM primitives.
- Exposure of every status and control port is individually selectable.
- The power-down port is available for the 7 series FPGA MMCME2, PLLE2, Virtex-6 FPGA MMCM primitive.
- The status port is available for the Spartan-6 FPGA DCM and DCM_CLKGEN primitives.
- The freeze port is available for the Spartan-6 FPGA DCM_CLKGEN primitive.
- The input clock stopped port is available for the 7 series FPGA MMCME2, PLLE2, Virtex-6 FPGA MMCM and Spartan-6 FPGA DCM and DCM_CLKGEN primitives.
- clk_valid port is available for DCM and DCM_CLKGEN.
- This version of clocking wizard supports naming of ports as per requirements. The list mentioned in the Table 1 is the default port list.

Table 2-2: Optional Attributes in HDL

Attribute	Description
CLK_OUT[1-7]_BUFR_DIV	The BUFR_DIVIDE attribute of clock outputs which have BUFR as output drivers.

Generating the Core

This chapter describes the GUI and follows the same flow required to set up the clocking network requirements. Tool tips are available in the GUI for most features; place your mouse over the relevant text, and additional information is provided in a pop-up dialog.

Clocking Features

The first page of the GUI allows you to identify the required features of the clocking network and configure the input clocks.

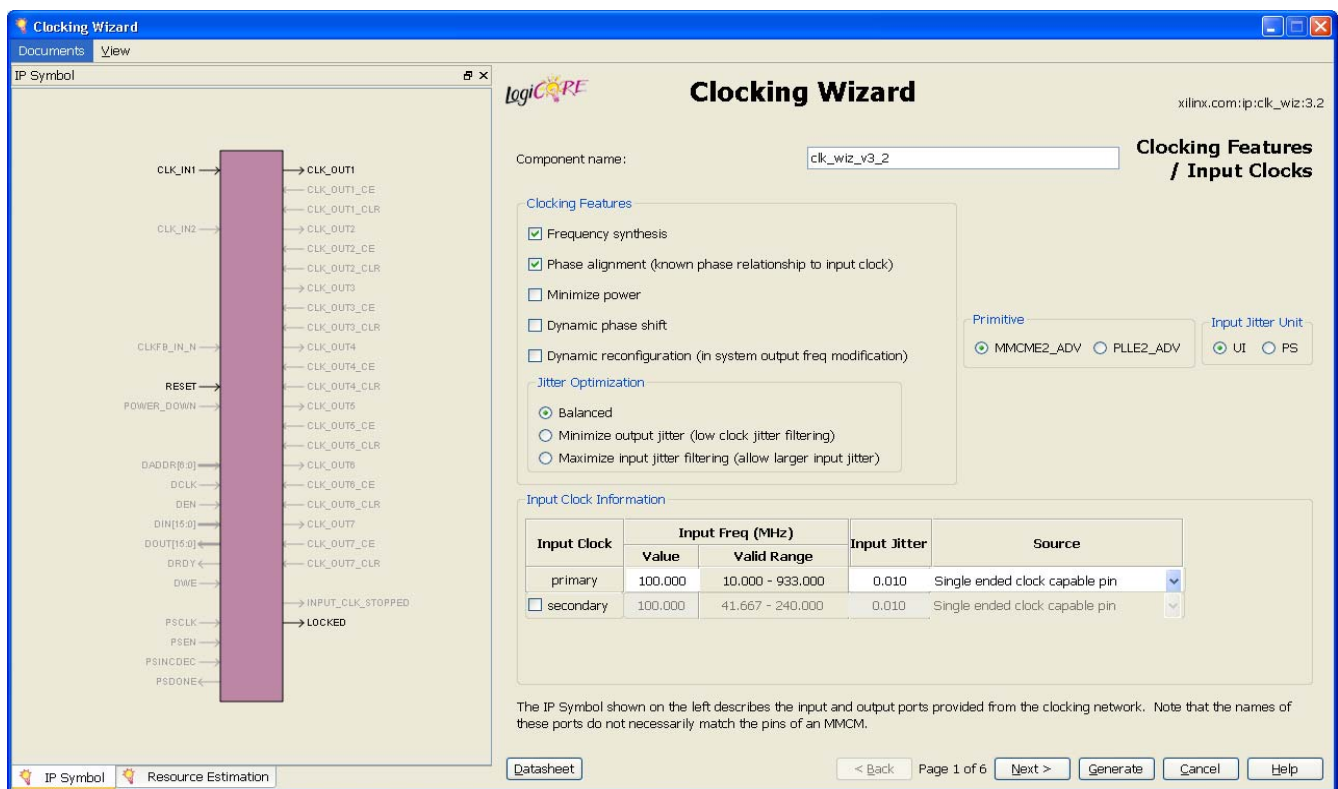


Figure 3-1: Main Screen - Clocking Features (7 Series FPGA Version Depicted)

Selecting Clocking Features

The available clocking features are shown for the selected target device. You can select as many features as desired; however, some features consume additional resources, and some can result in increased power consumption. Additionally, certain combinations of features are not allowed. The GUI will either dim out or hide features which are unavailable.

For example, the Dynamic Reconfiguration Port checkbox is not available for selection in the case of DCM and PLL_BASE as both of these do not support this feature.

Clocking features include:

- **Frequency synthesis.** This feature allows output clocks to have different frequencies than the active input clock.
- **Phase alignment.** This feature allows the output clock to be phase locked to a reference, such as the input clock pin for a device.
- **Minimize power.** This feature minimizes the amount of power needed for the primitive at the possible expense of frequency, phase offset, or duty cycle accuracy.
- **Dynamic phase shift.** This feature allows you to change the phase relationship on the output clocks.
- **Dynamic reconfiguration.** This feature allows you to change the programming of the primitive after device configuration. When this option is chosen, the clocking wizard uses only integer values for M, D and CLKOUT[0:6]_DIVIDE.
- **Balanced.** Selecting Balanced results in the software choosing the correct BANDWIDTH for jitter optimization.
- **Minimize output jitter.** This feature minimizes the jitter on the output clocks, but at the expense of power and possibly output clock phase error. This feature is not available with 'Maximize input jitter filtering'.
- **Maximize input jitter filtering.** This feature allows for larger input jitter on the input clocks, but can negatively impact the jitter on the output clocks. This feature is not available with 'Minimize output jitter'.

Clock Manager Type (Primitive Selection)

7 series devices contain MMCME2 and PLLE2 primitives for the clocking needs. The user has the option to configure either of these by selecting the primitive. Virtex®-6 devices contain MMCM primitives, which encapsulate all clocking needs. Therefore, the Clock Manager Type option is not available for a Virtex-6 FPGA based project. For a Spartan®-6 FPGA based project, the wizard automatically picks the appropriate primitive (DCM_SP, DCM_CLKGEN, or PLL_BASE), based on the other clocking features you select. You can force the use of a specific primitive by selecting "Manual Selection" and choosing the desired primitive. Functionality not available for that primitive, such as specific clocking features, is shown as unavailable.

For Spartan-6 devices, the wizard supports DCM to PLL and PLL to DCM cascades. To have the cascading implemented, the user has to manually select the "DCM_SP" primitive and then choose the type of cascading needed. The Wizard allows only one clock to be configured when DCM to PLL cascade is chosen. The user can configure up to six clocks when PLL to DCM cascade is chosen. In both the cases, the PLL is used only to reduce the clock jitter. PLL does not do any frequency synthesis.

Configuring Input Clocks

For Virtex-6 and 7 series FPGAs, you can enable an additional input clock. Select the box next to the secondary input clock to enable the additional input clock. Depending on the frequency of the secondary input clock, this can cause a less ideal network to be created than might be possible if just the primary input clock was present (more output jitter, higher power, etc.)

Enter the frequency and peak-to-peak period (cycle) jitter for the input clocks. The wizard then uses this information to create the clocking network. Additionally, a UCF (user constraint file) is created using the values entered. For the best calculated clocking parameters, it is best to fully specify the values. For example, for a clock requirement of 33 1/3 MHz, enter 33.333 MHz rather than 33 MHz.

You can select which buffer type drives your input clock, and this is then instantiated in the provided source code. If your input buffers are located externally, selecting “No buffer” leaves the connection blank. If Phase Alignment is selected, you do not have access to pins that are not dedicated clock pins, because the skew introduced by a non-clock pin is not matched by the primitive.

You can choose the units for input clock jitter by selecting either the UI or PS radio button. The input jitter box accepts the values based on this selection.

For Spartan-6 FPGA designs, the ISE® tool chain infers BUFIO2 for input clock routing which is not part of the generated HDL.

Output Clock Settings

The second page of the GUI (Figure 3-2) configures requirements for the output clocks. Each selected output clock can be configured on this screen.

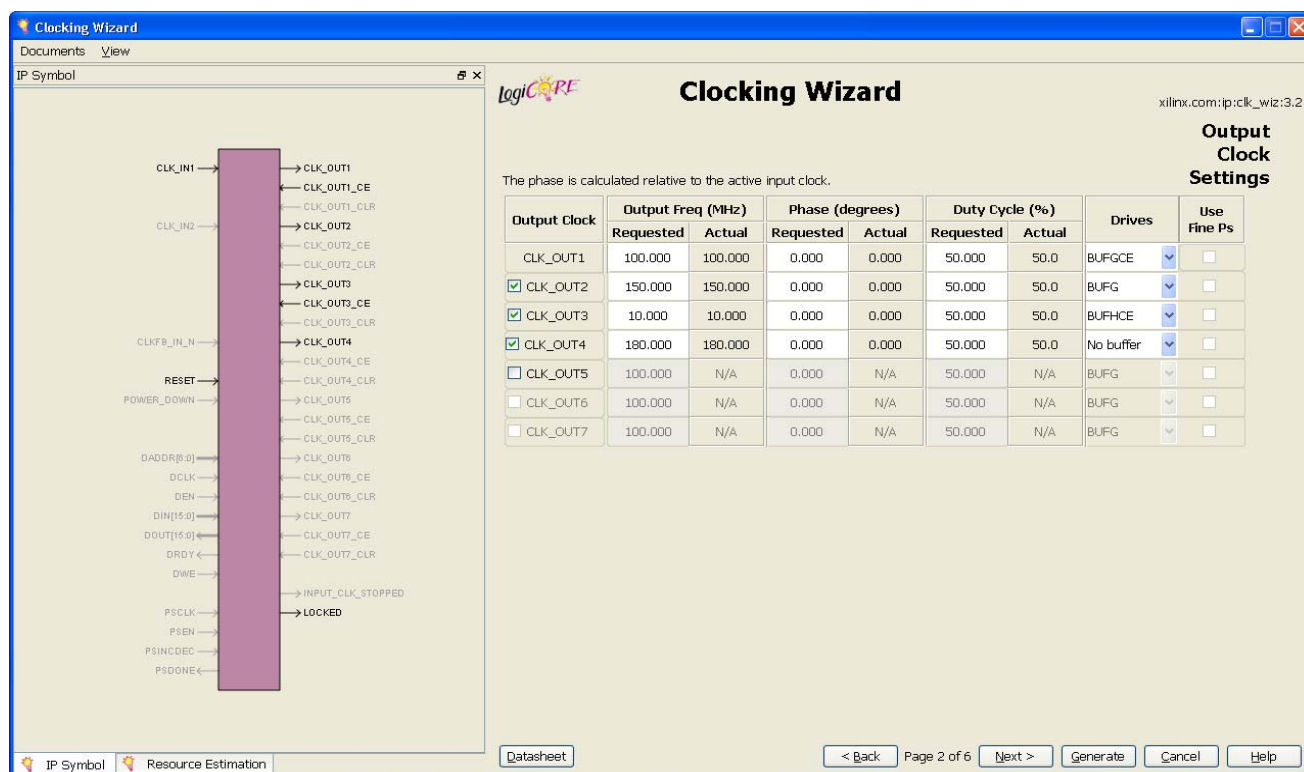


Figure 3-2: Output Clock Settings Screen (7 Series FPGA Version Depicted)

Configuring Output Clocks

To enable an output clock, click on the box located next to it. Output clocks must be enabled sequentially.

You can specify values for the output clock frequency, phase shift, and duty cycle assuming that the primary input clock is the active input clock. The clocking wizard attempts to derive a clocking network that meets your criteria exactly. In the event that a solution cannot be found, best attempt values are provided and are shown in the actual value column. Achieving the specified output frequency takes precedence over implementing the specified phase, and phase in turn takes higher precedence in the clock network derivation process than duty cycle. The precedence of deriving the circuits for the CLK_OUT signals is CLK_OUT1 > CLKOUT2 > CLKOUT3, and so on. Therefore, finding a solution for CLK_OUT1 frequency has a higher priority. Values are recalculated every time an input changes. Because of this, it is best to enter the requirements from top to bottom and left to right. This helps to pinpoint requested values that cannot be supported exactly.

If phase alignment is selected, the phase shift is with respect to the active input clock. If phase alignment is not selected, phase shift is with respect to CLK_OUT1.

Not all primitives allow duty-cycle specification. For example, a DCM_SP is restricted to a 50/50 duty cycle. In the event that duty cycle cannot be specified, the requested column is dimmed.

You can choose which type of buffer is instantiated to drive the output clocks, or “No buffer” if the buffer is already available in external code. The buffers available depend on your device family. For all outputs that have BUFR as the output driver, the “BUFR_DIVIDE” attribute is available as a generic parameter in the HDL. The user can change the divide value of the BUFR while instantiating the design.

If you choose the **Dynamic phase shift** clocking feature in a 7 series and Virtex-6 FPGA design, the ‘Use Fine Ps’ check boxes are available. ‘Use Fine Ps’ allows you to enable the Variable Fine Phase Shift on MMCM and MMCME2. Select the appropriate check box for any clock that requires dynamic phase shift. The wizard resets the requested phase field to “0.000” when ‘Use Fine Ps’ is selected. See the [Virtex-6 FPGA Clocking Resources User Guide](#) for more details.

Feedback and I/O

The third GUI screen ([Figure 3-3](#)) provides information to configure the rest of the clocking network.

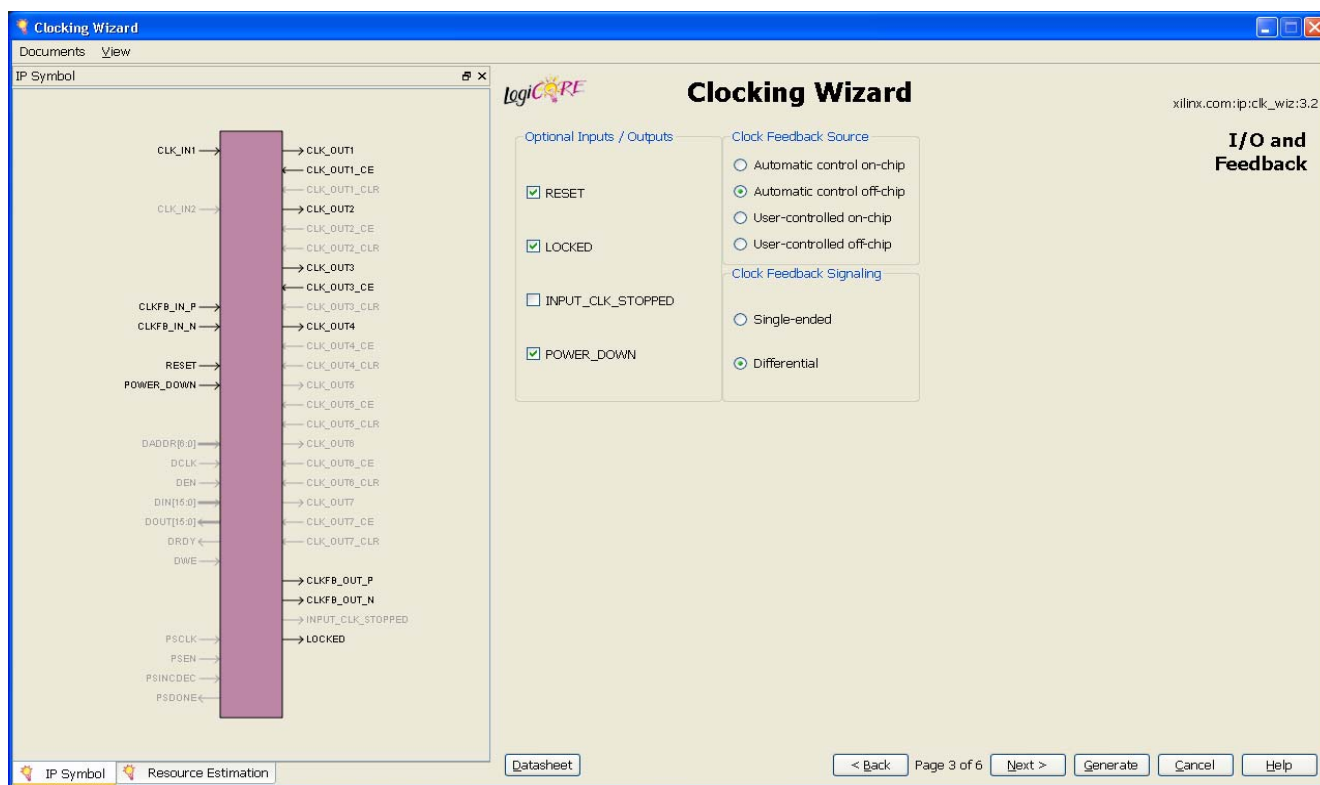


Figure 3-3: Feedback, and I/O Configuration Screen (7 Series FPGA Version Depicted)

Selecting Optional Ports

All other optional ports that are not handled by selection of specific clocking features are listed under Optional Inputs/Outputs. Click to select the ports that you wish to make visible; inputs that are unused are tied off appropriately, and outputs that are unused are labeled as such in the provided source code.

Choosing Feedback

Feedback selection is only available when phase alignment is selected. When phase alignment is not selected, the output feedback is directly connected to the input feedback. For designs with phase alignment, choose automatic control on-chip if you want the feedback path to match the insertion delay for CLK_OUT1. You can also select user-controlled feedback if the feedback is in external code. If the path is completely on the FPGA, select on-chip; otherwise, select off-chip.

For designs that require external feedback and related I/O logic, choose automatic control off-chip feedback. You can choose either single-ended or differential feedback in this mode. The wizard generates the core logic and logic required to route the feedback signals to the I/O.

Primitive Overrides

One or more pages of device and primitive specific parameter overrides are displayed, as shown in [Figure 3-4](#).

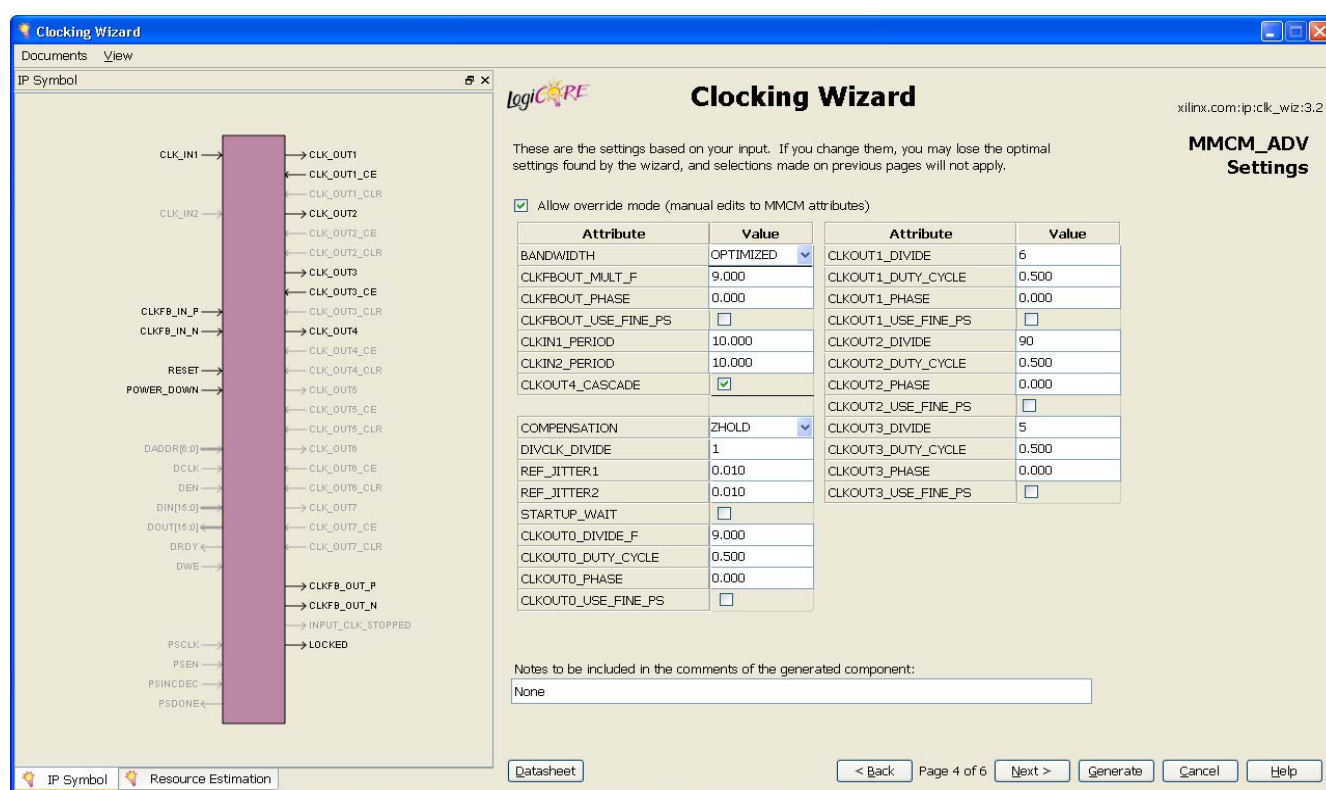


Figure 3-4: Primitive Override Screen (7 Series FPGA Version Depicted)

Overriding Calculated Parameters

The clocking wizard selects optimal settings for the parameters of the clocking primitive. You can override any of these calculated parameters if you wish. By selecting “Allow override mode,” the overridden values are used rather than the calculated values as primitive parameters. The wizard uses the settings as shown on this screen for any timing calculations, and any settings changed here are reflected in the summary pages. It is important to verify that the values you are choosing to override are correct because the wizard implements what you have chosen even if it causes issues with the generated network. Parameters listed are relevant for the physical clocks on the primitive, rather than the logical clocks created in the source code. For example, to modify the settings calculated for the highest priority `CLK_OUT1`, you actually need to modify `CLKOUT0 *` parameters, and not the `CLKOUT1 *` parameters for a MMCM or PLL.

Note: For Virtex-6 FPGA designs, certain combinations of `CLKIN` frequency and `DIVCLK_DIVIDE` are not allowed when the `BANDWIDTH` chosen is `HIGH`. In such cases, the wizard automatically sets the `BANDWIDTH` to `OPTIMIZED`.

User Provided Comment

The generated source code contains the input and output clock summaries shown in the next summary page (Figure 3-5). You can manually add an additional comment in the box at the bottom of the page. Use underscores in place of spaces.

Clock Summary (First Page)

The first summary page (Figure 3-5) displays summary information about the input and output clocks. This information is also provided as comments in the generated source code, and in the provided UCF.

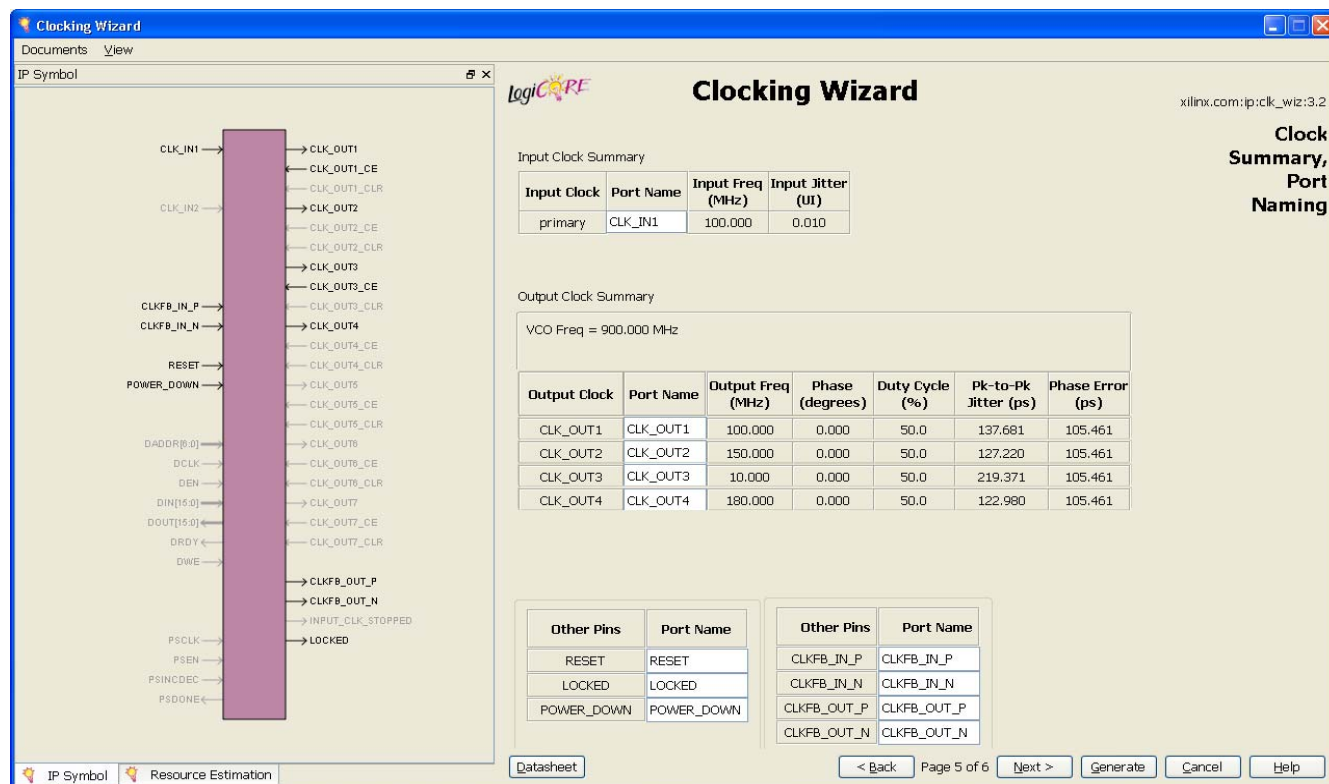


Figure 3-5: Clock Summary Screen

Input Clocking Summary

Information entered on the first page of the GUI (Figure 3-1) is shown for the input clocks.

Output Clocking Summary

Derived timing information for the output clocks is shown. If the chosen primitive has an oscillator, the VCO frequency is provided as reference. If you have a secondary input clock enabled, you can choose which clock is used to calculate the derived values.

Port Names

The Wizard allows users to name the ports according to their needs. If users want to name the HDL port for primary clock input, they should simply type in the necessary port name in the adjacent text box. The text boxes have the default names in them.

In the case of Primary clock input the default name is CLK_IN1.

Note: Be careful. Changing the port names could result in syntax errors if the port name entered is any reserved word of VHDL or Verilog or if that signal is already declared in the module.

Core Summary (Second Page)

The second summary page (Figure 3-6) contains general summary information.

Clocking Wizard

Documents View

IP Symbol

logiCORE

Clocking Wizard

xilinx.com:ip:clk_wiz:3.2

Core Summary

XPower Estimator input parameters

Core Name	Input Clock (MHz)	Phase Shift
clk_wiz_v3_2	100.000	None

Divide Counter	Mult Counter	CLKOUT0 Divider	CLKOUT1 Divider	CLKOUT2 Divider	CLKOUT3 Divider	CLKOUT4 Divider	CLKOUT5 Divider	CLKOUT6 Divider
1	9.000	9.000	6	90	5	off	off	off

Generated files

File Name	Description
clk_wiz_v3_2. (v vhd)	Verilog or VHDL clocking network source
clk_wiz_v3_2. (veo vho)	Verilog or VHDL instantiation template
clk_wiz_v3_2/clk_wiz_v3_2.ucf	Example constraints file
clk_wiz_v3_2/clk_wiz_readme.txt	README file for the core
clk_wiz_v3_2.xco	CORE Generator file used to recreate core
clk_wiz_v3_2_list.txt	Synthesis tools integration file for the core
clk_wiz_v3_2_xmdf.tcl	Project Navigator integration file for the core
clk_wiz_v3_2/doc	Directory for documentation delivered with the core
clk_wiz_v3_2/example_design	Directory for synthesizable example design
clk_wiz_v3_2/implementation	Directory for files to implement the example design
clk_wiz_v3_2/simulation	Directory tree for simulatable test bench and control

IP Symbol Resource Estimation Datasheet < Back Page 6 of 6 Next > Generate Cancel Help

Figure 3-6: Core Summary Screen

Resource Estimate Summary

A resource estimate is provided based on the chosen clocking features.

XPower Estimator Summary










Input parameters to the Xpower tool are provided.

File Report Summary

A summary of created output files is provided. See [Chapter 4, Detailed Example Design](#).

Detailed Example Design

This chapter provides detailed information about the example design, including a description of files and the directory structure generated by the Xilinx® CORE Generator™ software, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

-  [`<project directory>`](#)
Top-level project directory; name is user-defined
 -  [`<project directory>/<component name>`](#)
Core release notes file
 -  [`<component name>/doc`](#)
Product documentation
 -  [`<component name>/example design`](#)
Verilog and VHDL design files
 -  [`<component name>/implement`](#)
Implementation script files
 -  [`implement/results`](#)
Results directory, created after implementation scripts are run, and contains implement script results
 -  [`<component name>/simulation`](#)
Simulation scripts
 -  [`simulation/functional`](#)
Functional simulation files
 -  [`simulation/timing`](#)
Timing simulation files

Directory and File Contents

The clocking wizard core directories and their associated files are defined in the following sections.

<project directory>

The <project directory> contains all the CORE Generator software project files.

Table 4-1: Project Directory

Name	Description
<project_dir>	
<component_name>.v[hd]	Verilog or VHDL source code.
<component_name>.xco	CORE Generator software project-specific option file; can be used as an input to the CORE Generator software.
<component_name>_flist.txt	List of files delivered with the core.
<component_name>.{veo vho}	VHDL or Verilog instantiation template.
<component_name>.ise	Files used to incorporate the core into an ISE® software project.

[Back to Top](#)

<project directory>/<component name>

The <component name> directory contains the readme file provided with the core, which can include last-minute changes and updates.

Table 4-2: Component Name Directory

Name	Description
<project_dir>/<component_name>	
clocking_wizard_v3_2_readme.txt	Clocking Wizard v3.2 readme file.
<component_name>.ucf	Constraint files containing the timing parameters for the created clock network. The output clock constraints are commented out, but can be cut and pasted into constraint files for use in downstream modules.

[Back to Top](#)

<component name>/example design

The example design directory contains the example design files provided with the core.

Table 4-3: Example Design Directory

Name	Description
<project_dir>/<component_name>/example_design	
<component_name>_exdes.v[hd]	Implementable Verilog or VHDL example design, with all created clocks driving a counter.

[Back to Top](#)

<component name>/doc

The doc directory contains the PDF documentation provided with the core.

Table 4-4: Doc Directory

Name	Description
<project_dir>/<component_name>/doc	
clk_wiz_ds709.pdf	Clocking Wizard v3.2 Data Sheet
clk_wiz_gsg521.pdf	Clocking Wizard v3.2 Getting Started Guide

[Back to Top](#)

<component name>/implement

The implement directory contains the core implementation script files for ISE as well as PlanAhead™ software.

Table 4-5: Implement Directory

Name	Description
<project_dir>/<component_name>/implement	
Scripts and projects to implement the example design	

[Back to Top](#)

implement/results

The results directory is created by the implement script, after which the implement script results are placed in the results directory.

Table 4-6: Results Directory

Name	Description
<project_dir>/<component_name>/implement/results	
Implement script result files.	

[Back to Top](#)

<component_name>/simulation

The simulation directory contains the simulation test bench and environment for the example design.

Table 4-7: Simulation Directory

Name	Description
<project_dir>/<component_name>/simulation	
<component_name>_tb.v[hd]	Demonstration test bench

[Back to Top](#)

simulation/functional

The functional directory contains functional simulation scripts.

Table 4-8: Functional Directory

Name	Description
<project_dir>/<component_name>/simulation/functional	
Contains simulation scripts and waveform formats.	

[Back to Top](#)

simulation/timing

The timing directory contains timing simulation scripts.

Table 4-9: Functional Directory

Name	Description
<project_dir>/<component_name>/simulation/timing	
Contains simulation scripts and waveform formats.	

[Back to Top](#)

Implementation Scripts

The implementation script is either a shell script or batch file that processes the example design through the Xilinx tool flow. It is located at:

LINUX

```
<project_dir>/<component_name>/implement/implement.sh
<project_dir>/<component_name>/implement/planAhead_rdn.sh
```

Windows

```
<project_dir>/<component_name>/implement/implement.bat
<project_dir>/<component_name>/implement/planAhead_rdn.bat
```

The implement script performs these steps:

- Synthesizes the HDL example design files using XST or Synplicity Synplify Pro
- Runs NGDBuild to consolidate the core netlist and the example design netlist into the NGD file containing the entire design
- Maps the design to the target technology
- Place-and-routes the design on the target device
- Performs static timing analysis on the routed design using Timing Analyzer (TRCE)
- Generates a bitstream
- Enables Netgen to run on the routed design to generate a VHDL or Verilog netlist (as appropriate for the Design Entry project setting)

The Xilinx tool flow generates several output and report files. These are saved in the following directory which is created by the implement script:

```
<project_dir>/<component_name>/implement/results
```

Simulation Scripts

Functional Simulation

The test scripts are a ModelSim, IES, VCS or ISim macro that automate the simulation of the test bench. They are available from this location:

```
<project_dir>/<component_name>/simulation/functional/
```

The test scripts perform these tasks:

- Compiles the structural UNISIM simulation model
- Compiles HDL Example Design source code
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds signals of interest
- Runs the simulation to completion

Timing Simulation

The test scripts are a ModelSim, IES that automate the simulation of the test bench. They are available from this location:

```
<project_dir>/<component_name>/simulation/timing/
```

The test scripts perform these tasks:

- Compile the structural SIMPRIM simulation model
- Compile routed HDL Example Design
- Compile the demonstration test bench
- Start a simulation of the test bench
- Open a Wave window and adds signals of interest
- Run the simulation to completion

Example Design

The following files describe the example design for the Clocking Wizard core.

VHDL

```
<project_dir>/<component_name>/example_design/<component_name>_exdes.vhd
```

Verilog

```
<project_dir>/<component_name>/example_design/<component_name>_exdes.v
```

The top-level example designs adds clock buffers where appropriate to all of the input and output clocks. All generated clocks drive counters, and the high bits of each of the counters are routed to a pin. This allows the entire design to be synthesized and implemented in a target device to provide post place-and-route gate-level *simulation*.

Customizing the Example Design

The widths of the counter in the example design can be modified to make the counters wider or narrower. When the correct width is combined with the frequencies of the output clocks, this allows the high bits of the counters to be connected to a visible indicator, such as an LED.

Demonstration Test Bench

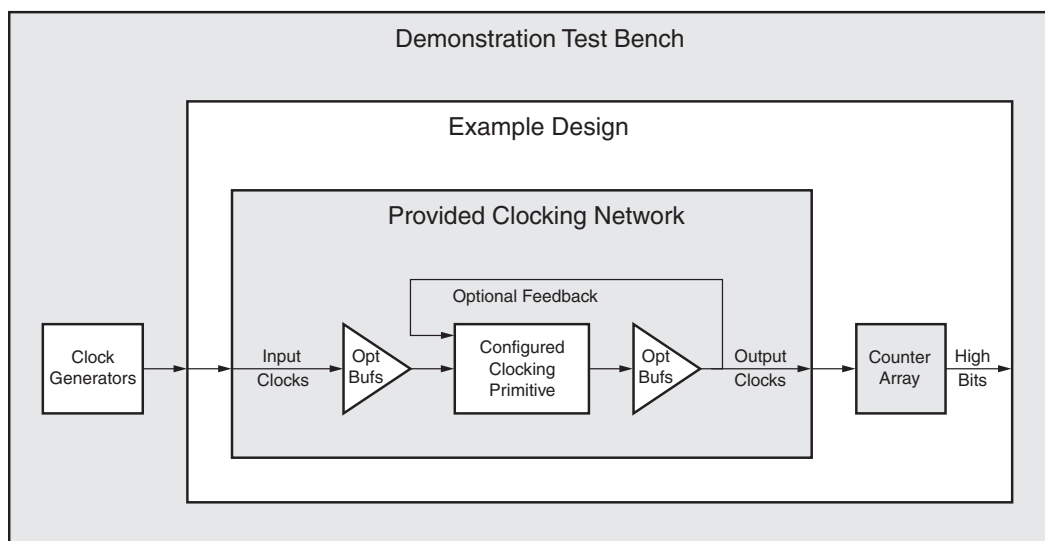


Figure 4-1: Clocking Wizard Demonstration Test Bench and Example Design

The following files describe the demonstration test bench.

VHDL

```
<project_dir>/<component_name>/simulation/<component_name>_tb.vhd
```

Verilog

```
<project_dir>/<component_name>/simulation/<component_name>_tb.v
```

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core.

The demonstration test bench performs these tasks:

- Generates input clock signals
- Runs the clocking primitive through the locking procedure
- In the event that a secondary input clock is enabled, changes the input clock from the primary source to the secondary source

Migration Guide

This information is provided to assist those designers who are experienced with the DCM and PLL Architecture Wizards. It highlights the differences between the old and new cores.

Differences between the Clocking Wizard and the Legacy DCM and PLL Wizards

There are several changes to the GUI and the wizard use model as described in the following subsections.

Primitive Selection

The old wizard required you to choose the correct GUI (DCM or PLL) before configuring the desired primitive.

The new wizard automatically selects the appropriate primitive and configures it based on desired parameters. You can choose to override this choice in the event that multiple primitives are available, as is the case for the Spartan®-6 device family.

Symbol Pin Activation

The old wizard had a symbol with clickable pins to enable a port.

For the new wizard, the symbol shows the ports that are currently active. To enable a port, enable the appropriate feature in the GUI. For example, enabling the secondary input clock enables the CLK_IN2 and CLK_IN_SEL ports and activates those ports in the symbol.

Parameter Override

The new wizard allows you to override any calculated parameter within the wizard by switching to override mode.

Port Display Conventions

The new wizard displays the superset of ports covering all device families. Ports that are not available for the selected target device are dimmed out. For example, if a Virtex[®]-6 device is selected, the STATUS port is dimmed out because it is not available for devices in that family. Information on the legal ports for a specific primitive can be found in the device family-specific FPGA or clocking resources User Guide at www.xilinx.com/support/documentation/index.htm.

Visibility of Clock Ports

The new wizard provides a clocking network that matches your requirements rather than making clock ports visible. As a result, your clock names will not match the exact names for the primitive. For example, while the “first” clock available for the Virtex-6 FPGA MMCM is CLKOUT0, the highest priority clock available to you is actually named CLK_OUT1. This change in numbering is especially important to consider if parameter overriding is desired.

GUI Information Gathering Order

Some of the information-gathering ordering has changed. For the new wizard the general flow is:

1. Select the clocking features.
2. Configure the input clock parameters.
3. Configure the output clock parameters.
4. Choose feedback and optional ports
5. View (and optionally override) calculated parameters.
6. Final summary pages.

For cascading clocking components, non-buffered input and output clocks are available for easy connection.