

LogiCORE™ IP Clocking Wizard v1.5

Getting Started Guide

UG521 April 19, 2010



Xilinx is providing this product documentation, hereinafter “Information,” to you “AS IS” with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice.

XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.

© 2004 - 2010 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE 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.

Table of Contents

Revision History	2
Preface: About This Guide	
Guide Contents	5
Conventions	6
Typographical	6
Online Document	7
List of Acronyms	8
Chapter 1: Introduction	
System Requirements	9
Operating Systems	9
About the Core	9
Recommended Design Experience	10
Additional Core Resources	10
Technical Support	10
Feedback	10
Clocking Wizard	10
Document	10
Chapter 2: Core Architecture	
Input Clocks	11
Primitive Instantiation	11
Feedback	11
Output Clocks	12
I/O Signals	12
Chapter 3: Generating the Core	
Clocking Features	17
Selecting Clocking Features	18
Clock Manager Type (Primitive Selection)	18
Configuring Input Clocks	19
Output Clock Settings	20
Configuring Output Clocks	20
Feedback and I/O	21
Selecting Optional Ports	22
Choosing Feedback	22
Primitive Overrides	22
Overriding Calculated Parameters	24
User Provided Comment	24

Clock Summary	25
Input Clocking Summary	25
Output Clocking Summary	25
Core Summary	26
Resource Estimate Summary	26
XPower Estimator Summary	26
File Report Summary	26

Chapter 4: Detailed Example Design

Directory and File Contents	28
<project directory>	28
<project directory>/<component name>	28
<component name>/example design	29
<component name>/doc	29
<component name>/implement	29
implement/results	29
<component name>/simulation	30
simulation/functional	30
Implementation Scripts	30
Simulation Scripts	31
Functional Simulation	31
Example Design	31
Customizing the Example Design	31
Demonstration Test Bench	32

Appendix A: Migration Guide

Differences in Wizards	33
------------------------------	----

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 provided example design and test bench.

Guide Contents

This guide contains the following chapters:

- [Preface, “About this Guide”](#) introduces the organization and purpose of this guide and the conventions used in this document.
- [Chapter 1, “Introduction,”](#) describes the core and related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.
- [Chapter 2, “Core Architecture,”](#) describes the generated clocking network.
- [Chapter 3, “Generating the Core,”](#) provides complete information about the GUI, including detailing information about entering parameters to generate the desired clocking network.
- [Chapter 4, “Detailed Example Design,”](#) describes the provided example design and test bench.
- [Appendix A, “Migration Guide,”](#) describes the usage differences between the new Clocking Wizard and the old PLL and DCM Architecture Wizards.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
Courier font	Messages, prompts, and program files that the system displays. Signal names also.	speed grade: - 100
Courier bold	Literal commands that you enter in a syntactical statement	ngdbuild <i>design_name</i>
Helvetica bold	Commands that you select from a menu	File →Open
	Keyboard shortcuts	Ctrl+C
<i>Italic font</i>	Variables in a syntax statement for which you must supply values	ngdbuild <i>design_name</i>
	References to other manuals	See the <i>User Guide</i> for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> connected.
Dark Shading	Items that are not supported or reserved	This feature is not supported
Square brackets []	An optional entry or parameter. However, in bus specifications, such as bus [7:0] , they are required.	ngdbuild [<i>option_name</i>] <i>design_name</i>
Braces { }	A list of items from which you must choose one or more	lowpwr = { on off }
Vertical bar	Separates items in a list of choices	lowpwr = { on off }
Angle brackets < >	User-defined variable or in code samples	<directory name>
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	allow block <i>block_name loc1 loc2 ... locn</i> ;

Convention	Meaning or Use	Example
Notations	The prefix '0x' or the suffix 'h' indicate hexadecimal notation	A read of address 0x00112975 returned 45524943h.
	An '_n' means the signal is active low	usr_teof_n is active low.

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section “ Guide Contents ” for details. See “ Title Formats ” in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the <i>FPGA User Guide</i> .
Blue, underlined text	Hyperlink to a website (URL)	Go to www.xilinx.com for the latest speed files.

List of Acronyms

The following table describes acronyms used in this manual.

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)
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) to implement a clocking network matched to your requirements and is designed to support both Verilog and VHDL design environments. In addition, the example design and simulation test bench delivered with the core is provided in both Verilog and VHDL.

System Requirements

Operating Systems

Windows

- Windows XP Professional 32-bit/64-bit
- Windows Vista Business 32-bit/64-bit

Linux

- Red Hat Enterprise Linux WS v4.0 32-bit/64-bit
- Red Hat Enterprise Desktop v5.0 32-bit/64-bit (with Workstation Option)
- SUSE Linux Enterprise (SLE) desktop and server v10.1 32-bit/64-bit

About the Core

The Clocking Wizard core is a Xilinx® CORE Generator™ software IP core, included in the latest Intellectual Property (IP) Update on the Xilinx IP Center.

Recommended Design Experience

The Clocking Wizard is designed to be used by those with any level of experience. It is the Xilinx recommended method to create to your clocking network. It guides all 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 more information about the Clocking Wizard, do the following

1. Go to the IP center (<http://www.xilinx.com/ipcenter/>).
2. Under Browse for Intellectual Property, select the “+” sign next to By Function.
3. Select FPGA Features and Design. The Clocking Wizard documents will be 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, please 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, please 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 provides source HDL which implements a clocking network including a clocking primitive, such as a DCM_SP, DCM_CLKGEN, PLL_BASE, or MMCM_ADV and associated circuitry including buffers and clock pins. The network is designed in segments that are illustrated in [Figure 2-1](#), and described in the following sections.

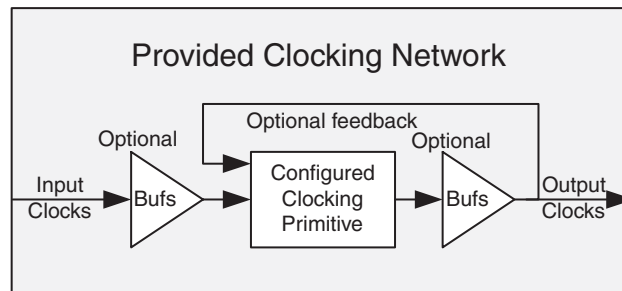


Figure 2-1: Provided Clocking Network

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 of 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, these ports are exposed to the user. Any port that is not exposed is appropriately 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) or Spartan®-6 FPGA DCM_CLKGEN primitives.

For more information 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	I/O	Description
Input Clock Ports^a		
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	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, up to seven are available. For a PLL 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	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^b		
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^c		
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^d		
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 ^e	Input	Power Down: When asserted, places the clocking primitive into low power state, which stops the output clocks.
STATUS[2:0] ^f	Output	Status: Contains information about the state of the primitive. Please see the user guide for specific bit descriptions.
FREEZE ^g	Input	Freeze: When asserted, prevents tap adjustment in the event that the input clock is lost.
INPUT_CLK_STOPPED ^h	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	I/O	Description
LOCKED	Output	Locked: When asserted, indicates that the output clocks are stable and usable by downstream circuitry.
CLK_VALID ⁱ	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.

- a. At least one input clock is required; any design will 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.
- b. Dynamic reconfiguration ports are available only for the Spartan-6 FPGA DCM_CLKGEN primitive.
- c. Dynamic phase shift ports are available for Virtex-6 FPGA MMCM or Spartan-6 FPGA DCM primitives.
- d. Exposure of every status and control port is individually selectable.
- e. The power-down port is available for the Virtex-6 FPGA MMCM primitive.
- f. The status port is available for the Spartan-6 FPGA DCM and DCM_CLKGEN primitives.
- g. The freeze port is available for the Spartan-6 FPGA DCM_CLKGEN primitive.
- h. The input clock stopped port is available for the Virtex-6 FPGA MMCM and Spartan-6 FPGA DCM and DCM_CLKGEN primitives.
- i. clk_valid port is available for DCM and DCM_CLKGEN.

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; simply 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.

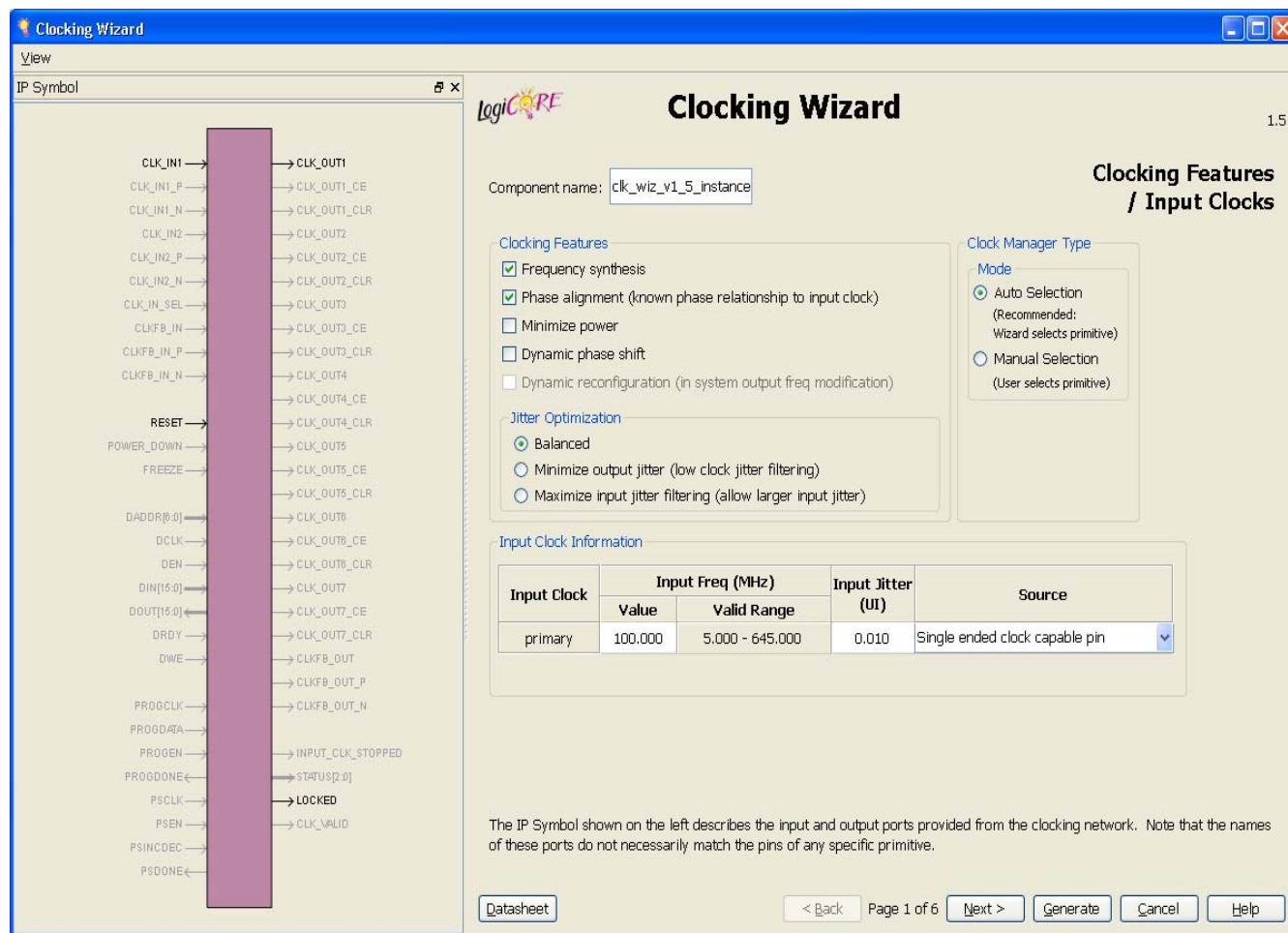


Figure 3-1: Main Screen - Clocking Features (Spartan-6 Version Depicted)

Selecting Clocking Features

The available clocking features are shown for the device selected. You can select as many features as needed; however, some features consume additional resources, and some may have ramifications such as increased power consumption. Additionally, not all features are available at the same time. For example, selecting one feature may automatically dim (make unavailable) other features. Likewise, there are also some features that when selected remove functionality within the GUI.

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.
- **Minimization of 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 Allowance of larger input jitter.
- **Allow larger input jitter.** This feature allows for larger input jitter on the input clocks, but may negatively impact the jitter on the output clocks. This feature is not available with Minimization of output jitter.
- **Dynamic phase shift.** This feature allows you to change the phase relationship on the output clocks.
- **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 reconfiguration.** This feature allows you to change the programming of the primitive after device configuration.

Clock Manager Type (Primitive Selection)

Virtex®-6 devices contain MMCM primitives, which encapsulate all clocking needs. Therefore, this option is not available for a Virtex-6 FPGA based project. For a Spartan-6 FPGA based projects, the wizard automatically picks the appropriate primitive (DCM_SP, DCM_CLKGEN, or PLL_BASE), based on clocking features. 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 made unavailable.

Configuring Input Clocks

Select the box next to the secondary input clock to enable an additional input clock. Depending on the frequency of the secondary input clock, this may 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 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, since the skew introduced by a non-clock pin is not matched by the primitive.

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.

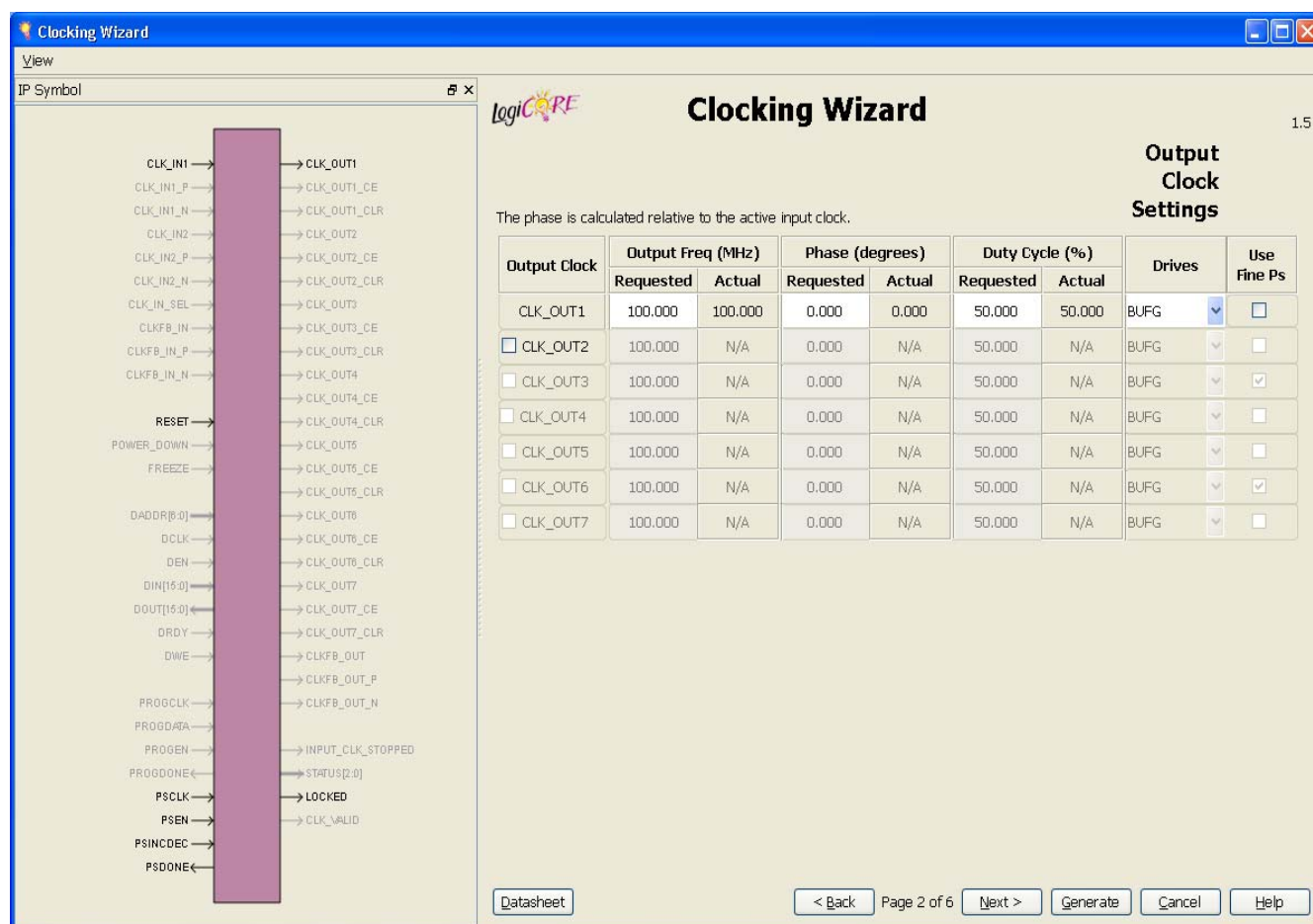


Figure 3-2: Output Clock Settings Screen (Virtex-6 FPGA Version Depicted)

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 solve the clocking network exactly. In the event that a solution cannot be found, best attempt values are provided and are shown in the actual value column. Output frequency is solved before phase, which is solved before duty cycle. CLK_OUT1 is solved before CLK_OUT2, which is solved before CLK_OUT3, and so on. Therefore, finding a solution for CLK_OUT1 frequency has a higher priority. Values are calculated 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 can not be exactly provided.

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 can not 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.

If you choose the **Dynamic phase shift** clocking feature in a Virtex-6 FPGA design, the Use Fine Ps check boxes are available. Select the appropriate check box for any clock that requires dynamic phase shift. The wizard will reset requested phase field to "0.000" when Use Fine Ps is selected. Refer to 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.

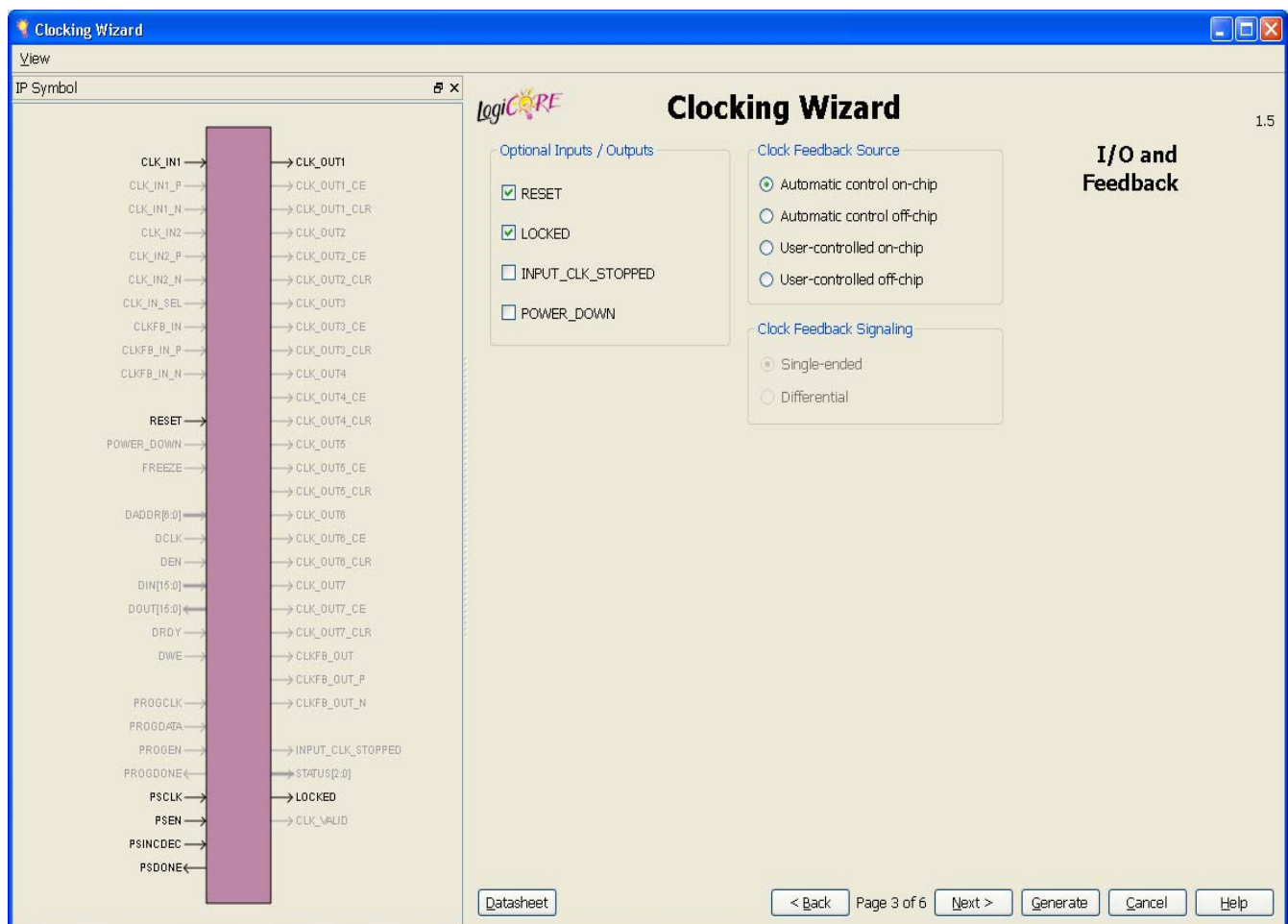


Figure 3-3: Feedback, and I/O Configuration Screen (Virtex-6 FPGA Version Depicted)

Selecting Optional Ports

All other optional ports that are not handled by selection of specific clocking features are listed under Optional Inputs/Outputs. 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 IO 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 IO.

Primitive Overrides

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

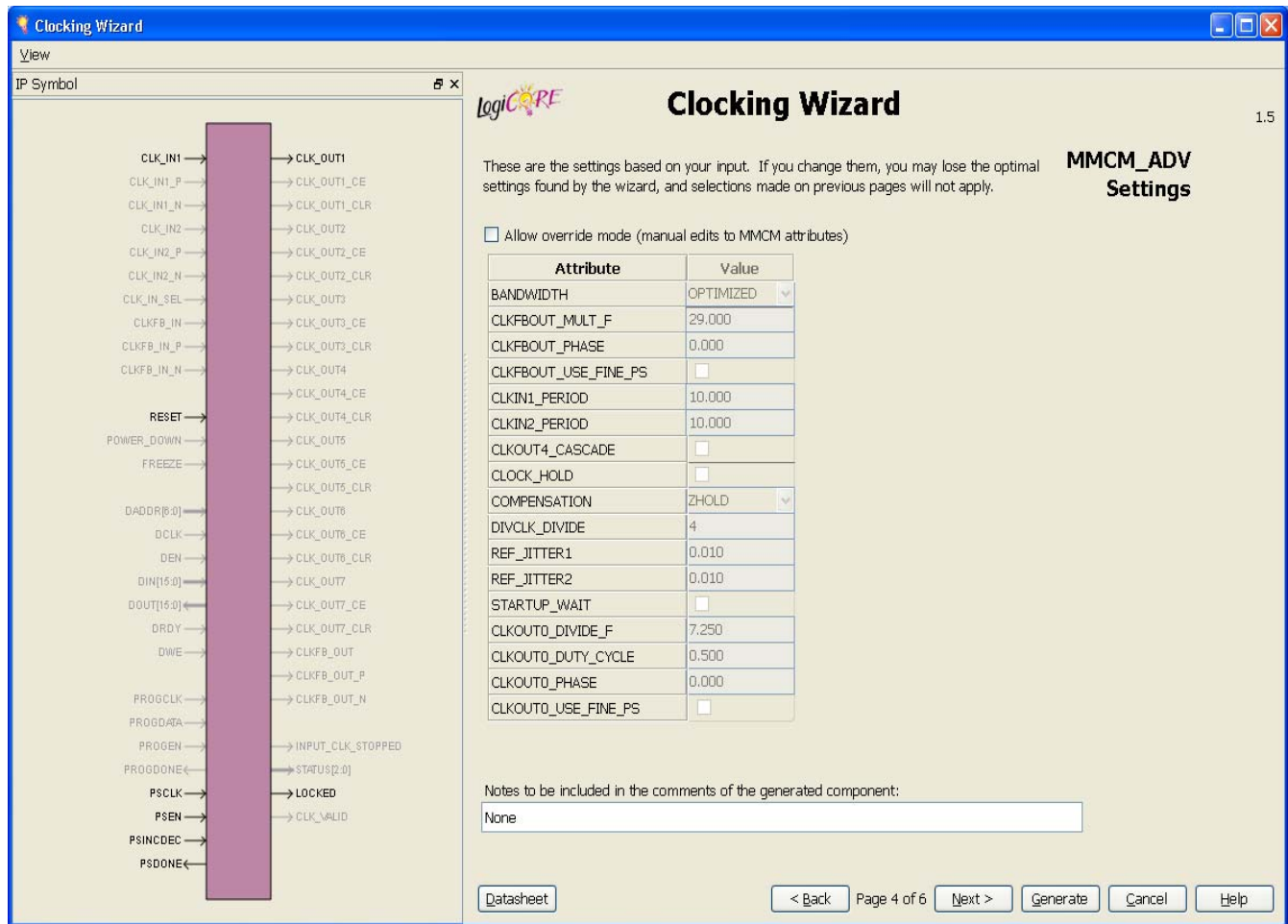


Figure 3-4: Primitive Override Screen (Virtex-6 FPGA Version Depicted)

Overriding Calculated Parameters

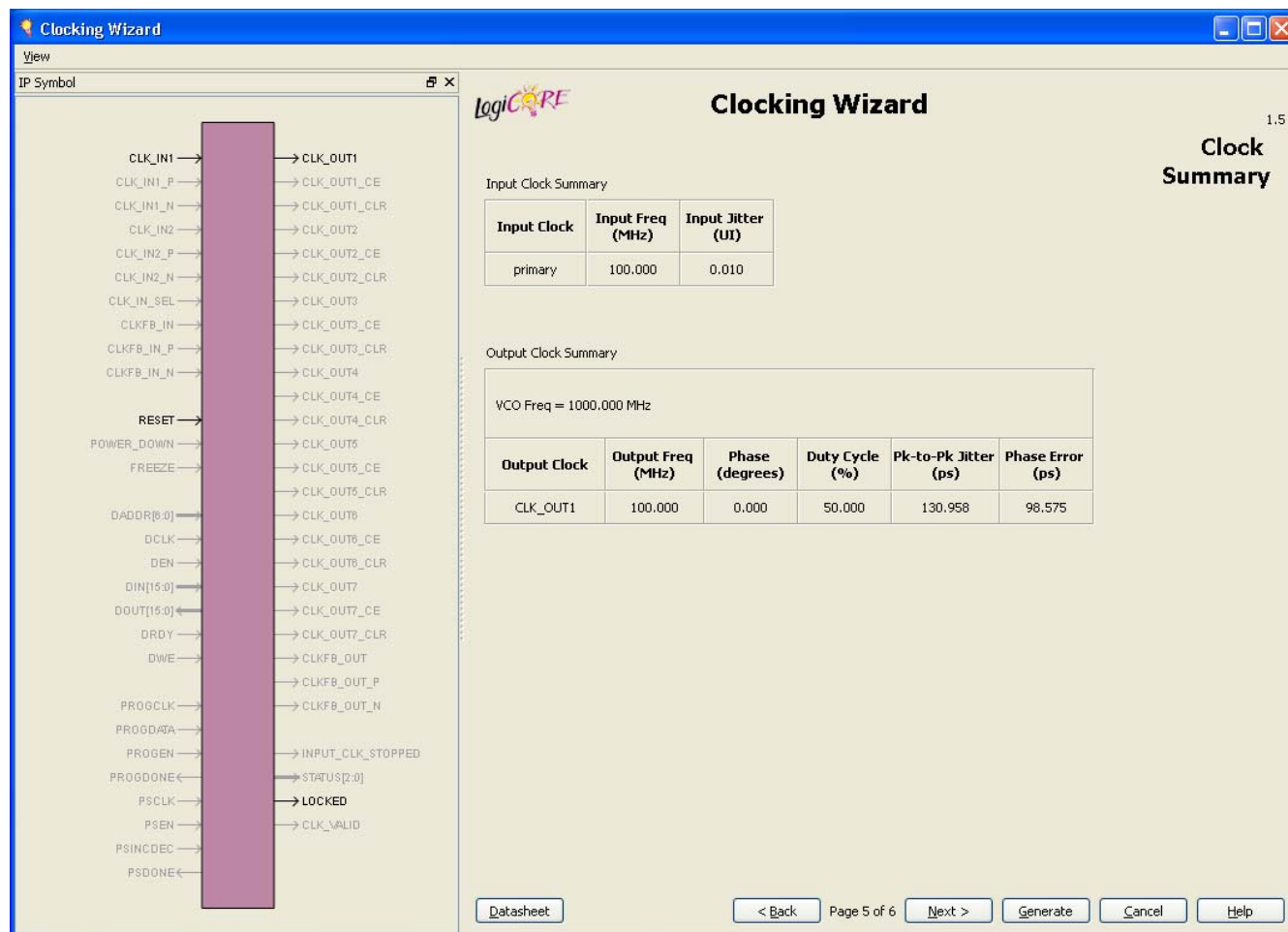
The clocking wizard selects optimal settings for the parameters of the clocking primitive. You can override any of these calculated parameters if you wish. By selecting “Allow override mode,” the overridden values are used rather than the calculated values as primitive parameters. The wizard uses the 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—as the wizard will implement 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 will actually need to modify `CLKOUT0` * parameters, and not the `CLKOUT1` * parameters for a MMCM or PLL.

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

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.



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.

Core Summary

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

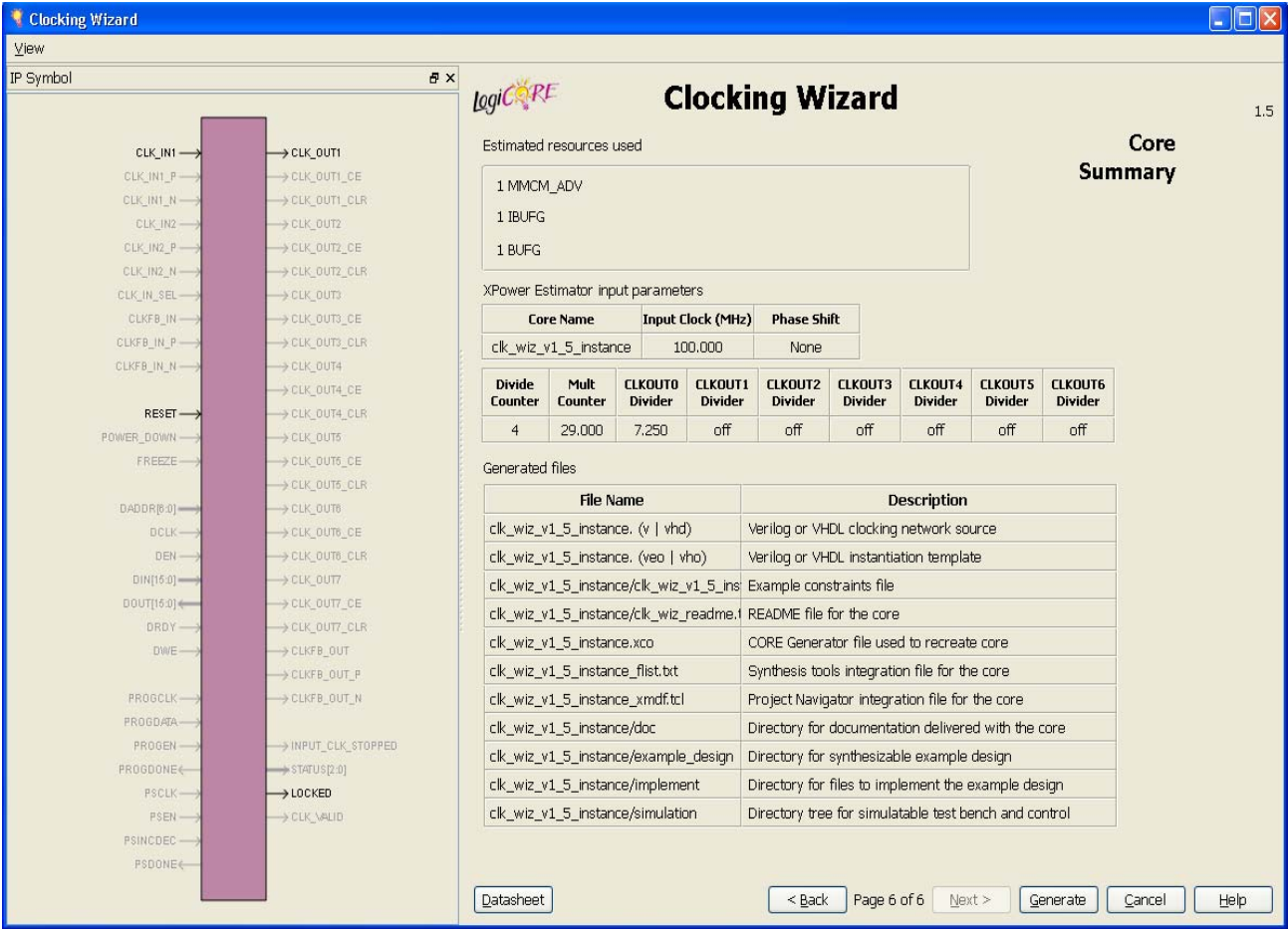


Figure 3-6: Core Summary Screen

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. For more information, please 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
 -  **Implementation Scripts**
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 release notes file provided with the core, which may include last-minute changes and updates.

Table 4-2: Component Name Directory

Name	Description
<project_dir>/<component_name>	
clocking_wizard_v1_5_release_notes.txt	Clocking Wizard v1.5 release notes 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 v1.5 Data Sheet
clk_wiz_gsg521.pdf	Clocking Wizard v1.5 Getting Started Guide

[Back to Top](#)

<component name>/implement

The implement directory contains the core implementation script files.

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 provided with the core.

Table 4-8: Functional Directory

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

Windows

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

The implement script performs the following 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 the following location:

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

The test scripts perform the following 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

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

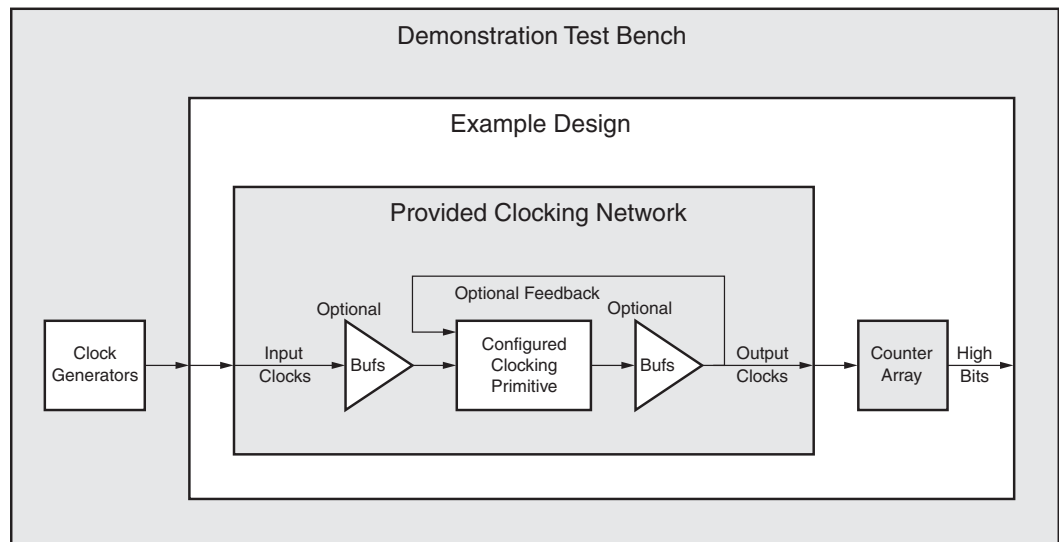


Figure 4-1: Clocking Wizard Demonstration Test Bench and Example Design

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 the following 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 in Wizards

There are several changes to the GUI and the wizard use model.

- 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.
- 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.
- The new wizard allows you to override any calculated parameter within the wizard by switching to override mode.
- The new wizard shows all possible ports for all possible devices. For example, the STATUS port is always displayed, but this port is not available for the Virtex®-6 device family. However, the correct optional ports and clocking features are available in the GUI for each specific device; if a port can not be selected by a clocking feature or an optional port, it is not available in that device. Please see the User Guides for the appropriate devices and primitives for more information. These User Guides can be downloaded from www.xilinx.com/support/documentation/index.htm.
- 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.

- Some of the information-gathering ordering has changed. For the new wizard the general flow is:
 1. Select the clocking features.
 - a. Configure the input clock parameters.
 - b. Configure the output clock parameters.
 - c. Choose feedback and optional ports
 - d. View (and optionally override) calculated parameters.
 - e. Final summary pages.
- For cascading clocking components, non-buffered input and output clocks are available for easy connection.