LogiCORE IP Clocking Wizard 3.6 (ISE) / 4.2 (Vivado)

Product Guide

PG065 July 25, 2012
Table of Contents

IP Facts

Chapter 1: Overview
  About the Core ................................................................. 7
  Recommended Design Experience ....................................... 7
  Feature Summary ............................................................. 7
  Applications ...................................................................... 8
  Licensing and Ordering Information ................................. 9

Chapter 2: Product Specification
  Performance ....................................................................... 10
  Resource Utilization ......................................................... 11
  Port Descriptions ............................................................. 11

Chapter 3: Designing with the Core
  General Design Guidelines ............................................... 14
  Clocking ........................................................................... 14
  Resets ............................................................................... 14
  Functional Overview ......................................................... 15
  Core Architecture ............................................................ 18

Chapter 4: Customizing and Generating the Core
  GUI .................................................................................... 21
  Clocking Features ............................................................. 21
  Output Clock Settings ......................................................... 25
  Feedback and I/O ............................................................... 29
  Primitive Overrides .......................................................... 31
  Clock Summary (First Page) ............................................... 32
  Core Summary (Second Page) ............................................. 35
  Parameter Values in the XCI File ........................................ 35
  Output Generation ............................................................ 41
Chapter 5: Constraining the Core

Required Constraints ................................................................. 43
Device, Package, and Speed Grade Selections ................................. 43
Clock Frequencies ........................................................................ 43
Clock Management ........................................................................ 43
Clock Placement .......................................................................... 43
Banking ......................................................................................... 44
I/O Standard and Placement .......................................................... 44

Chapter 6: Detailed Example Design

Directory and File Contents .......................................................... 45
Example Design ............................................................................ 45
Demonstration Test Bench ................................................................ 46
Simulation ..................................................................................... 46

Chapter 7: Customizing and Generating the Core

Clocking Features ......................................................................... 48
Output Clock Settings .................................................................... 52
Feedback and I/O .......................................................................... 57
Primitive Overrides ........................................................................ 58
Clock Summary (First Page) ........................................................... 60
Core Summary (Second Page) ......................................................... 62

Chapter 8: Constraining the Core

Required Constraints ..................................................................... 63
Device, Package, and Speed Grade Selections ................................. 63
Clock Frequencies ....................................................................... 63
Clock Management ....................................................................... 63
Clock Placement .......................................................................... 63
Banking ......................................................................................... 64
I/O Standard and Placement .......................................................... 64

Chapter 9: Detailed Example Design

Directory and File Contents .......................................................... 66
Implementation Scripts .................................................................... 69
Simulation Scripts ......................................................................... 70
Example Design ............................................................................ 71
Demonstration Test Bench ............................................................. 72
Appendix A: Verification, Compliance, and Interoperability

Simulation ................................................................. 74
Hardware Testing ....................................................... 74

Appendix B: Application Software Development

Device Drivers ........................................................... 75

Appendix C: Migrating

Differences between the Clocking Wizard and the Legacy DCM and PLL Wizards .......... 76

Appendix D: Debugging

Appendix E: Additional Resources

Xilinx Resources .......................................................... 79
Solution Centers ........................................................... 79
Technical Support ......................................................... 79
Revision History .......................................................... 80
Notice of Disclaimer ..................................................... 80
SECTION I: SUMMARY

IP Facts
Overview
Product Specification
Designing with the Core
Introduction

The LogiCORE™ IP Clocking Wizard core (v3.6 for ISE and v4.2 for Vivado tools) 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.

Features

- 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 calculates Voltage Controlled Oscillator (VCO) frequency for primitives with an oscillator, and provides multiply and divide values based on input and output frequency requirements
- Automatically implements overall configuration that supports phase shift and duty cycle requirements
- Supports Spread Spectrum clocking for MMCME2 and allows users to select valid range of modulation frequency, mode and input/output clocks.
- Optionally buffers clock signals

<table>
<thead>
<tr>
<th>Core Specifics</th>
<th>Supported Device Family(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Virtex-7, Kintex™-7, Artix™-7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported User Interfaces</th>
<th>Not Applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>See Table 2-2.</td>
</tr>
<tr>
<td>Special Features</td>
<td>PLL2E, MMCME2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Provided with Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Files</td>
</tr>
<tr>
<td>Example Design</td>
</tr>
<tr>
<td>Test Bench</td>
</tr>
<tr>
<td>Constraints File</td>
</tr>
<tr>
<td>Simulation Model</td>
</tr>
<tr>
<td>Instantiation Template</td>
</tr>
<tr>
<td>Supported S/W Driver</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tested Design Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Entry Tools</td>
</tr>
<tr>
<td>Simulation(2)</td>
</tr>
<tr>
<td>Synthesis Tools(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided by Xilinx @ <a href="http://www.xilinx.com/support">www.xilinx.com/support</a></td>
</tr>
</tbody>
</table>

Notes:
1. For a complete listing of supported devices, see the release notes for this core.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
Features (continued)

• Provides the ability to override the selected clock primitive and any calculated attribute
• Provides timing estimates for the clock circuit and Xilinx® Power Estimator (XPE) parameters
• Provides a synthesizable example design including the clocking network and a simulation test bench
• Provides optional ports for the selected primitive
Overview

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.

About the Core

The Clocking Wizard v3.6 is a Xilinx IP core, can be generated using the Xilinx ISE CORE Generator™ software, which is a standard component of the Xilinx ISE Design Suite.

The Clocking Wizard v4.2 is a Vivado™ IP core, included with the latest Vivado 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.

Feature Summary

Clocking features include:

- **Frequency synthesis.** This feature allows output clocks to have different frequencies than the active input clock.
Applications

- **Spread Spectrum.** This feature provides modulated output clocks which reduces the spectral density of the electromagnetic interference (EMI) generated by electronic devices. This feature is available for only MMCME2_ADV primitive. Unisim simulation support for this feature is not available in current release.

- **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'.

- **Fast Simulation.** This feature generates fast simulation configuration and scripts for running fast simulation using unifast_ver and unifast model of MMCME2_ADV and PLLE2_ADV. This improves simulation runtime by 100X.

Applications

- Creation of clock network having required frequency, phase and duty cycle with reduced jitter

- Electromagnetic Interference reduction in electronic devices using Spread Spectrum feature
Licensing and Ordering Information

The LogiCORE IP Clocking Wizard core is provided free of charge under the terms of the Xilinx End User License Agreement. The core can be generated using the Xilinx Vivado software. This version of the core can be generated using the Vivado system v2012.2.

For details, visit the Clocking Wizard product web page. Information about additional Xilinx LogiCORE IP modules is available at the Xilinx IP Center. For pricing and availability of other Xilinx LogiCORE IP modules and software, contact your local Xilinx sales representative.
Product Specification

Clocking Wizard helps create the clocking circuit for the required output clock frequency, phase and duty cycle using MMCME2 or PLLE2 primitive. It also helps verify the output generated clock frequency in simulation, providing a synthesizable example design which can be tested on the hardware. It also supports Spread Spectrum feature which is helpful in reducing Electromagnetic interference.

Performance

Maximum Frequencies

- Input Clock - 800 MHz
- Output Clock - 800 MHz
Resource Utilization

Resource utilization is available in the Clocking Wizard GUI in the Resource tab.

Port Descriptions

Table 2-1 describes the input and output ports provided from the clocking network. All ports are optional, with the exception being that at least one input and one output clock are required. The options selected by the user determine which ports are actually available to be configured. 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.

Table 2-1: Clocking Wizard I/O

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK_IN1</td>
<td>Input</td>
<td><strong>Clock in 1:</strong> Single-ended primary input clock port. Available when single-ended primary clock source is selected.</td>
</tr>
</tbody>
</table>
### Port Descriptions

**Table 2-1: Clocking Wizard I/O (Cont’d)**

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK_IN1_P</td>
<td>Input</td>
<td><strong>Clock in 1 Positive and Negative</strong>: Differential primary input clock port pair. Available when a differential primary clock source is selected.</td>
</tr>
<tr>
<td>CLK_IN1_N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLK_IN2 [2]</td>
<td>Input</td>
<td><strong>Clock in 2</strong>: Single-ended secondary input clock port. Available when a single-ended secondary clock source is selected.</td>
</tr>
<tr>
<td>CLK_IN2_N [2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLK_IN_SEL [2]</td>
<td>Input</td>
<td><strong>Clock in Select</strong>: When ‘1’, selects the primary input clock; When ‘0’, the secondary input clock is selected. Available when two input clocks are specified.</td>
</tr>
<tr>
<td>CLKFB_IN</td>
<td>Input</td>
<td><strong>Clock Feedback in</strong>: 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.</td>
</tr>
<tr>
<td>CLKFB_IN_P</td>
<td></td>
<td><strong>Clock Feedback in: Positive and Negative</strong>: Differential feedback in port of the clocking primitive. Available when the automatic control off-chip feedback and differential feedback option is selected.</td>
</tr>
<tr>
<td>CLKFB_IN_N</td>
<td>Input</td>
<td></td>
</tr>
</tbody>
</table>

**Output Clock Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK_OUT1</td>
<td>Output</td>
<td><strong>Clock Out 1</strong>: Output clock of the clocking network. CLK_OUT1 is not optional.</td>
</tr>
<tr>
<td>CLK_OUT1_CE</td>
<td>Input</td>
<td><strong>Clock Enable</strong>: Chip enable pin of the output buffer. Available when BUFGCE or BUFGCE or BUFR buffers are used as output clock drivers.</td>
</tr>
<tr>
<td>CLK_OUT1_CLR</td>
<td>Input</td>
<td><strong>Counter reset for divided clock output</strong>: Available when BUFR buffer is used as output clock driver.</td>
</tr>
<tr>
<td>CLK_OUT2-n [3]</td>
<td>Output</td>
<td><strong>Clock Out 2 - n</strong>: 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.</td>
</tr>
<tr>
<td>CLK_OUT2-n_CE [3]</td>
<td>Input</td>
<td><strong>Clock Enable</strong>: Chip enable pin of the output buffer. Available when BUFGCE or BUFGCE or BUFR buffers are used as output clock drivers.</td>
</tr>
<tr>
<td>CLK_OUT2-n_CLR [3]</td>
<td>Input</td>
<td><strong>Counter reset for divided clock output</strong>: Available when BUFR buffer is used as output clock driver.</td>
</tr>
<tr>
<td>CLKFB_OUT</td>
<td>Output</td>
<td><strong>Clock Feedback Out</strong>: 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.</td>
</tr>
<tr>
<td>CLKFB_OUT_P</td>
<td>Output</td>
<td><strong>Clock Feedback Out: Positive and Negative</strong>: Differential feedback output port of the clocking primitive. Available when the user-controlled off-chip feedback and differential feedback option is selected.</td>
</tr>
<tr>
<td>CLKFB_OUT_N</td>
<td>Output</td>
<td></td>
</tr>
</tbody>
</table>

**Dynamic Reconfiguration Ports for MMCME2**

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DADDR[6:0]</td>
<td>Input</td>
<td><strong>Dynamic Reconfiguration Address</strong>: Address port for use in dynamic reconfiguration; active when DEN is asserted</td>
</tr>
<tr>
<td>DCLK</td>
<td>Input</td>
<td><strong>Dynamic Reconfiguration Clock</strong>: Clock port for use in dynamic reconfiguration</td>
</tr>
<tr>
<td>DEN</td>
<td>Input</td>
<td><strong>Dynamic Reconfiguration Enable</strong>: Starts a dynamic reconfiguration transaction</td>
</tr>
</tbody>
</table>
### Port Descriptions

#### 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.

#### 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 Phase Shift Ports (2)
- **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 Input:**
  **Power Down:** When asserted, places the clocking primitive into low power state, which stops the output clocks.
- **INPUT_CLK_STOPPED Output:**
  **Input Clock Stopped:** When asserted, indicates that the selected input clock is no longer toggling.
- **LOCKED Output:**
  **Locked:** When asserted, indicates that the output clocks are stable and usable by downstream circuitry.

### Table 2-1: Clocking Wizard I/O (Cont’d)

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI[15:0]</td>
<td>Input</td>
<td><strong>Dynamic Reconfiguration Data in:</strong> Input data for a dynamic reconfiguration write transaction; active when DEN is asserted</td>
</tr>
<tr>
<td>DO[15:0]</td>
<td>Output</td>
<td><strong>Dynamic Reconfiguration Data Out:</strong> Output data for a dynamic reconfiguration read transaction; active when DRDY is asserted</td>
</tr>
<tr>
<td>DRDY</td>
<td>Output</td>
<td><strong>Dynamic Reconfiguration Ready:</strong> Completes a dynamic reconfiguration transaction</td>
</tr>
<tr>
<td>DWE</td>
<td>Input</td>
<td><strong>Dynamic Reconfiguration Write Enable:</strong> When asserted, indicates that the dynamic reconfiguration transaction is a write; active when DEN is asserted</td>
</tr>
<tr>
<td>PSCLK</td>
<td>Input</td>
<td><strong>Dynamic Phase Shift Clock:</strong> Clock for use in dynamic phase shifting</td>
</tr>
<tr>
<td>PSEN</td>
<td>Input</td>
<td><strong>Dynamic Phase Shift Enable:</strong> Starts a dynamic phase shift transaction</td>
</tr>
<tr>
<td>PSINCDEC</td>
<td>Input</td>
<td><strong>Dynamic Phase Shift increment/decrement:</strong> When '1'; increments the phase shift of the output clock, when '0', decrements the phase shift</td>
</tr>
<tr>
<td>PSDONE</td>
<td>Output</td>
<td><strong>Dynamic Phase Shift Done:</strong> Completes a dynamic phase shift transaction</td>
</tr>
<tr>
<td>RESET</td>
<td>Input</td>
<td><strong>Reset:</strong> When asserted, asynchronously clears the internal state of the primitive, and causes the primitive to re-initiate the locking sequence when released</td>
</tr>
<tr>
<td>POWER_DOWN</td>
<td>Input</td>
<td><strong>Power Down:</strong> When asserted, places the clocking primitive into low power state, which stops the output clocks</td>
</tr>
<tr>
<td>INPUT_CLK_STOPPED</td>
<td>Output</td>
<td><strong>Input Clock Stopped:</strong> When asserted, indicates that the selected input clock is no longer toggling</td>
</tr>
<tr>
<td>LOCKED</td>
<td>Output</td>
<td><strong>Locked:</strong> When asserted, indicates that the output clocks are stable and usable by downstream circuitry</td>
</tr>
</tbody>
</table>

### Notes:
1. At least one input clock is required; any design has at least a CLK_IN1 or a CLK_IN1_P/CLK_IN1_N port.
2. Not available when Spread Spectrum is selected.
3. CLK_OUT3 and CLK_OUT4 are not available when Spread Spectrum is selected.
4. Exposure of every status and control port is individually selectable.
5. This version of clocking wizard supports naming of ports as per requirements. The list mentioned in Table 2-1 is the default port list.
Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

General Design Guidelines

- Provide the available input clock information for Frequency and Jitter.
- If the same input clock is used by other logic in the design then provide No buffer (incase the input clock is output of global buffer) or global buffer option for source type. If the input clock is used only by core then provide clock capable pin as source type.

Clocking

Up to seven output clocks with different frequencies can be generated for required circuitry.

 Resets

- Clocking Wizard has active high Asynchronous reset signal for clocking primitive.
- The core must be held in RESET during clock switch over
- When the input clock or feedback clock is lost, the CLKINSTOPPED or CLKFBSTOPPED status signal is asserted. After the clock returns, the CLKINSTOPPED signal is unasserted and a RESET must be applied.
Functional Overview

The Clocking Wizard is an interactive Graphical User Interface (GUI) that creates a clocking network based on design-specific needs. The required clock network parameters are organized in a linear sequence so that you can select only the desired parameters. Using the wizard, experienced users can explicitly configure their chosen clocking primitive, while less experienced users can let the wizard automatically determine the optimal primitive and configuration -- based on the features required for their individual clocking networks.

Users already familiar with the Digital Clock Manager (DCM) and Phase-Locked Loop (PLL) wizards can refer to the Migration Guide Appendix in the Clocking Wizard Getting Started Guide for information on usage differences.

Clocking Features

Major clocking-related functional features desired and specified by the user can be used by the wizard to select an appropriate primitive. Incompatible features are automatically dimmed out to help the designer evaluate feature trade-offs.

Clocking features include:

- Frequency synthesis
- Phase alignment
- Spread Spectrum
- Minimization of output jitter
- Allowance of larger input jitter
- Minimization of power
- Dynamic phase shift
- Dynamic reconfiguration
- Fast Simulation

Input Clocks

One input clock is the default behavior, but two input clocks can be chosen by selecting a secondary clock source. Only the timing parameters of the input clocks in their specified units is required; the wizard uses these parameters as needed to configure the output clocks.
Input Clock Jitter Option

The wizard allows the user to specify the input clock jitter either in UI or PS units using a radio button.

Output Clocks

The number of output clocks is user-configurable. The maximum number allowed depends upon the selected device and the interaction of the major clocking features you specify. Users can simply input their desired timing parameters (frequency, phase, and duty cycle) and let the clocking wizard select and configure the clocking primitive and network automatically to comply with the requested characteristics. If it is not possible to comply exactly with the requested parameter settings due to the number of available input clocks, best-attempt settings are provided. When this is the case, the clocks are ordered so that CLK_OUT1 is the highest-priority clock and is most likely to comply with the requested timing parameters. The wizard prompts you for frequency parameter settings before the phase and duty cycle settings.

Note: The port names in the generated circuit can differ from the port names used on the original primitive.

Clock Buffering and Feedback

In addition to configuring the clocking primitive within the device, the wizard also assists with constructing the clocking network. Buffering options are provided for both input and output clocks. If a clock output requires special buffers like BUFPLL which the wizard does not generate in the design, alert messages are flagged to the user. Feedback for the primitive can be user-controlled or left to the wizard to automatically connect. If automatic feedback is selected, the feedback path is matched to timing for CLK_OUT1.

Optional Ports

All primitive ports are available for user-configuration. You can expose any of the ports on the clocking primitive, and these are provided as well in the source code.

Primitive Override

All configuration parameters are also user-configurable. In addition, should a provided value be undesirable, any of the calculated parameters can be overridden with the desired settings.
Summary

The Clocking Wizard provides a summary for the created network. Input and output clock settings are provided both visually and as constraint files. In addition, jitter numbers for the created network are provided along with a resource estimate. Lastly, the wizard provides the input setting for PLL and MMCM based designs for Xilinx Power Estimator (XPE) in an easy-to-use table.

Design Environment

Figure 3-1 shows the design environment provided by the wizard to assist in integrating the generated clocking network into a design. The wizard provides a synthesizable and downloadable example design to demonstrate how to use the network and allows you to place a very simple clocking network in your device. A sample simulation test bench, which simulates the example design and illustrates output clock waveforms with respect to input clock waveforms, is also provided.
Core Architecture

The Clocking Wizard generates source code HDL to implement a clocking network. The generated clocking network typically consists of a clocking primitive (MMCME2_ADV or PLLE2_ADV) plus some additional circuitry which typically includes buffers and clock pins. The network is divided into segments as illustrated in Figure 3-2. Details of these segments are described in the following sections.

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

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 Phase Shift is not available when Spread Spectrum is selected.
SECTION II: VIVADO DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Detailed Example Design
Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core in the Vivado Design Suite.

GUI

This chapter describes the Vivado tools Graphical User Interface (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 (Figure 4-1, Figure 4-2) allows you to identify the required features of the clocking network and configure the input clocks.
Figure 4-1: Clocking Features (Spread Spectrum Unselected)
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.

Clocking features include:

- **Frequency synthesis.** This feature allows output clocks to have different frequencies than the active input clock.

- **Spread Spectrum (SS).** This feature provides modulated output clocks which reduces the spectral density of the electromagnetic interference (EMI) generated by electronic devices. This feature is available only for MMCME2 primitive.

- **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. This feature is not available when Spread Spectrum feature is selected.
Clocking Features

- **Dynamic phase shift.** This feature allows you to change the phase relationship on the output clocks. This feature is not available when Spread Spectrum feature is selected.

- **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'.

- **Fast Simulation.** This feature generates fast simulation configuration and scripts for running fast simulation using unifast_ver and unifast model of MMCME2_ADV and PLLE2_ADV. This improves simulation runtime by 100X.

Clock Manager Type (Primitive Selection)

In Virtex-7, Kintex-7, and Artix-7 devices MMCME2 and PLLE2 primitives for the clocking needs. The user has the option to configure either of these by selecting the primitive. Features are enabled or disabled depending on the primitive selected.

Configuring Input Clocks

There are two input clocks available and depending on selection reference clock can be switched from one to another. GUI provides option to select the secondary input clock to enable the additional input clock. If Spread Spectrum feature is selected, secondary input clock is disabled in the Clocking wizard. 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.)

Valid input frequency ranges are:

- Frequency when SS is unselected: 10 - 800 MHz
- Frequency when SS is selected: 25 - 150 MHz

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 XDC (Xilinx Design Constraints 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.

Output Clock Settings

The second page of the GUI (Figure 4-3) configures requirements for the output clocks. Each selected output clock can be configured on this screen.

Figure 4-3: Output Clock Settings (Spread Spectrum Unselected)

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 \texttt{CLK\_OUT1} > \texttt{CLKOUT2} > \texttt{CLKOUT3}, and so on. Therefore, finding a solution for \texttt{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 \texttt{CLK\_OUT1}.

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, the 'Use Fine Ps' check boxes are available. 'Use Fine Ps' allows you to enable the Variable Fine Phase Shift on 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.

When Spread Spectrum (SS) is selected, \texttt{CLK\_OUT<3>} and \texttt{CLK\_OUT<4>} are not available. Divide values of these outputs are used for SS modulation frequency generation.
There are four modes available for SS Mode:

- DOWN_LOW
- DOWN_HIGH
- CENTER_LOW
- CENTER_HIGH

Available Modulation Frequency range:

- 25 to 250 KHz

Spread Spectrum calculation details are described in Figure 4-5 and Figure 4-6.
Output Clock Settings

Note: Input_clock_frequency is in Hz unit.

For spread:

- If (SS_Mode = CENTER_HIGH) :=>
  \[ \text{spread (ps)} = \pm \left\{ \frac{1}{\text{Input_clock_frequency} \times (M-0.125 \times 4) / (D/O)} - \frac{1}{\text{Input_clock_frequency} \times M / (D/O)} \right\} \]

- If (SS_Mode = CENTER_LOW) :=>
  \[ \text{spread (ps)} = \pm \left\{ \frac{1}{\text{Input_clock_frequency} \times (M-0.125 \times 2) / (D/O)} - \frac{1}{\text{Input_clock_frequency} \times M / (D/O)} \right\} \]

- If (SS_Mode = DOWN_HIGH) :=>
  \[ \text{spread (ps)} = + \left\{ \frac{1}{\text{Input_clock_frequency} \times (M-0.125 \times 4) / (D/O)} - \frac{1}{\text{Input_clock_frequency} \times M / (D/O)} \right\} \]

- If (SS_Mode = DOWN_LOW) :=>

Figure 4-5: Spread Spectrum Mode (Center Spread)

Figure 4-6: Spread Spectrum Mode (Down Spread)
spreads (ps) = $\frac{1}{(\text{Input\_clock\_frequency}\times(M-0.125\times2)/D/O)} - 1/$
$\frac{1}{(\text{Input\_clock\_frequency}\times M/D/O)}$

Where $M$ is CLKFBOUT_MULT_F, $D$ is DIVCLK_DIVIDE, and $O$ is respective CLKOUTx_DIVIDE.

- For Modulation Frequency:
  - $O2$ and $O3$ are calculated by the bitgen in implementation. Same calculation is done in the wizard to get actual modulation frequency value.
  - Then based on what $O2$ and $O3$ is calculated, the actual modulation frequency is calculated:
    - If (SS\_Mode = CENTER\_HIGH or SS\_Mode = CENTER\_LOW)
      $\text{Actual\_modulation\_frequency\ (average)} = \frac{(\text{Input\_clock\_frequency}\times M/D)}{(O2 \times O3) / 16}$
    - If (SS\_Mode = DOWN\_HIGH)  $\text{Actual\_modulation\_frequency\ (average)} = 0.5 \times [rac{(\text{Input\_clock\_frequency}\times M/D)}{(O2 \times O3) / 8} + \frac{(\text{Input\_clock\_frequency}\times (M-0.5)/D)}{(O2 \times O3) / 8}]$
    - If (SS\_Mode = DOWN\_LOW)  $\text{Actual\_modulation\_frequency\ (average)} = 0.5 \times [rac{(\text{Input\_clock\_frequency}\times M/D)}{(O2 \times O3) / 8} + \frac{(\text{Input\_clock\_frequency}\times (M-0.25)/D)}{(O2 \times O3) / 8}]$

*Note:* Actual modulation frequency may deviate within +/- 10% of the requested modulation frequency for some settings.

---

**Feedback and I/O**

The third GUI screen (Figure 4-7) provides information to configure the rest of the clocking network.
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 4-8.

**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 MMCME2 or PLL2.
The generated source code contains the input and output clock summaries shown in the next summary page (Figure 4-9). You can manually add an additional comment in the box at the bottom of the page. Use underscores in place of spaces.

**User Provided Comment**

Bandwidth is always set to low for Spread Spectrum clocking.

---

**Clock Summary (First Page)**

The first summary page (Figure 4-10) 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 XDC.
Input Clocking Summary

Information entered on the first page of the GUI (Figure 4-10) 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. When Spread Spectrum is enabled, actual modulation frequency is provided as reference.
Tspread is the actual spread as calculated in Configuring Output Clocks.

**Port Names**

The Wizard allows you to name the ports according to their needs. If you want to name the HDL port for primary clock input, simply type in the port name in the adjacent text box. The text boxes contain the default names. In the case of Primary clock input, the default name is CLK_IN1.

**Note:** Be careful when changing the port names, as it 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 4-10) contains general summary information.

Figure 4-12: 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.

Parameter Values in the XCI File

Table 4-1 defines valid entries for the XCI parameters.
Table 4-1: XCI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPONENT_NAME</td>
<td>clk_wiz_v4_2_0</td>
</tr>
<tr>
<td>primitive</td>
<td>MMCME2</td>
</tr>
<tr>
<td>primtype_sel</td>
<td>mmcm_adv</td>
</tr>
<tr>
<td>clock_mgr_type</td>
<td>auto</td>
</tr>
<tr>
<td>USE_FREQ_SYNTH</td>
<td>TRUE</td>
</tr>
<tr>
<td>USE_SPREAD_SPECTRUM</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_PHASE_ALIGNMENT</td>
<td>TRUE</td>
</tr>
<tr>
<td>USE_MIN_POWER</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_DYN_PHASE_SHIFT</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_DYN_RECONFIG</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_FAST_SIMULATION</td>
<td>FALSE</td>
</tr>
<tr>
<td>JITTER_SEL</td>
<td>No_Jitter</td>
</tr>
<tr>
<td>PRIM_IN_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>IN_FREQ_UNITS</td>
<td>Units_MHz</td>
</tr>
<tr>
<td>IN_JITTER_UNITS</td>
<td>Units_UI</td>
</tr>
<tr>
<td>RELATIVE_INCLK</td>
<td>REL_PRIMARY</td>
</tr>
<tr>
<td>USE_INCLK_SWITCHOVER</td>
<td>FALSE</td>
</tr>
<tr>
<td>SECONDARY_IN_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>SECONDARY_PORT</td>
<td>CLK_IN2</td>
</tr>
<tr>
<td>SECONDARY_SOURCE</td>
<td>SingleEnded_clock_capable_pin</td>
</tr>
<tr>
<td>JITTER_OPTIONS</td>
<td>UI</td>
</tr>
<tr>
<td>CLKNIN1_UI_JITTER</td>
<td>0.01</td>
</tr>
<tr>
<td>CLKN2_UI_JITTER</td>
<td>0.01</td>
</tr>
<tr>
<td>PRIM_IN_JITTER</td>
<td>0.01</td>
</tr>
<tr>
<td>SECONDARY_IN_JITTER</td>
<td>0.01</td>
</tr>
<tr>
<td>CLKN1_JITTER_PS</td>
<td>100</td>
</tr>
<tr>
<td>CLKN2_JITTER_PS</td>
<td>100</td>
</tr>
<tr>
<td>CLKOUT2_USED</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLKOUT3_USED</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLKOUT4_USED</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLKOUT5_USED</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLKOUT6_USED</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLKOUT7_USED</td>
<td>FALSE</td>
</tr>
<tr>
<td>NUM_OUT_CLKS</td>
<td>1</td>
</tr>
<tr>
<td>CLK_OUT1_USE_FINE_PS_GUI</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
### Table 4-1: XCI Parameters (Cont’d)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK_OUT2_USE_FINE_PS_GUI</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLK_OUT3_USE_FINE_PS_GUI</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLK_OUT4_USE_FINE_PS_GUI</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLK_OUT5_USE_FINE_PS_GUI</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLK_OUT6_USE_FINE_PS_GUI</td>
<td>FALSE</td>
</tr>
<tr>
<td>CLK_OUT7_USE_FINE_PS_GUI</td>
<td>FALSE</td>
</tr>
<tr>
<td>PRIMARY_PORT</td>
<td>CLK_IN1</td>
</tr>
<tr>
<td>CLK_OUT1_PORT</td>
<td>CLK_OUT1</td>
</tr>
<tr>
<td>CLK_OUT2_PORT</td>
<td>CLK_OUT2</td>
</tr>
<tr>
<td>CLK_OUT3_PORT</td>
<td>CLK_OUT3</td>
</tr>
<tr>
<td>CLK_OUT4_PORT</td>
<td>CLK_OUT4</td>
</tr>
<tr>
<td>CLK_OUT5_PORT</td>
<td>CLK_OUT5</td>
</tr>
<tr>
<td>CLK_OUT6_PORT</td>
<td>CLK_OUT6</td>
</tr>
<tr>
<td>CLK_OUT7_PORT</td>
<td>CLK_OUT7</td>
</tr>
<tr>
<td>DADDR_PORT</td>
<td>DADDR</td>
</tr>
<tr>
<td>DCLK_PORT</td>
<td>DCLK</td>
</tr>
<tr>
<td>DRDY_PORT</td>
<td>DRDY</td>
</tr>
<tr>
<td>DWE_PORT</td>
<td>DWE</td>
</tr>
<tr>
<td>DIN_PORT</td>
<td>DIN</td>
</tr>
<tr>
<td>DOUT_PORT</td>
<td>DOUT</td>
</tr>
<tr>
<td>DEN_PORT</td>
<td>DEN</td>
</tr>
<tr>
<td>PSCLK_PORT</td>
<td>PSCLK</td>
</tr>
<tr>
<td>PSEN_PORT</td>
<td>PSEN</td>
</tr>
<tr>
<td>PSINCDEC_PORT</td>
<td>PSINCDEC</td>
</tr>
<tr>
<td>PSDONE_PORT</td>
<td>PSDONE</td>
</tr>
<tr>
<td>CLKOUT1_REQUESTED_OUT_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>CLKOUT1_REQUESTED_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>CLKOUT1_REQUESTED_DUTY_CYCLE</td>
<td>50</td>
</tr>
<tr>
<td>CLKOUT2_REQUESTED_OUT_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>CLKOUT2_REQUESTED_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>CLKOUT2_REQUESTED_DUTY_CYCLE</td>
<td>50</td>
</tr>
<tr>
<td>CLKOUT3_REQUESTED_OUT_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>CLKOUT3_REQUESTED_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>CLKOUT3_REQUESTED_DUTY_CYCLE</td>
<td>50</td>
</tr>
<tr>
<td>CLKOUT4_REQUESTED_OUT_FREQ</td>
<td>100</td>
</tr>
</tbody>
</table>
### Table 4-1: XCI Parameters (Cont’d)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKOUT4_REQUESTED_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>CLKOUT4_REQUESTED_DUTY_CYCLE</td>
<td>50</td>
</tr>
<tr>
<td>CLKOUT5_REQUESTED_OUT_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>CLKOUT5_REQUESTED_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>CLKOUT5_REQUESTED_DUTY_CYCLE</td>
<td>50</td>
</tr>
<tr>
<td>CLKOUT6_REQUESTED_OUT_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>CLKOUT6_REQUESTED_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>CLKOUT6_REQUESTED_DUTY_CYCLE</td>
<td>50</td>
</tr>
<tr>
<td>CLKOUT7_REQUESTED_OUT_FREQ</td>
<td>100</td>
</tr>
<tr>
<td>CLKOUT7_REQUESTED_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>CLKOUT7_REQUESTED_DUTY_CYCLE</td>
<td>50</td>
</tr>
<tr>
<td>USE_MAX_I_JITTER</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_MIN_O_JITTER</td>
<td>FALSE</td>
</tr>
<tr>
<td>PRIM_SOURCE</td>
<td>Single-ended_clock_capable_pin</td>
</tr>
<tr>
<td>CLKOUT1_DRIVES</td>
<td>BUFG</td>
</tr>
<tr>
<td>CLKOUT2_DRIVES</td>
<td>BUFG</td>
</tr>
<tr>
<td>CLKOUT3_DRIVES</td>
<td>BUFG</td>
</tr>
<tr>
<td>CLKOUT4_DRIVES</td>
<td>BUFG</td>
</tr>
<tr>
<td>CLKOUT5_DRIVES</td>
<td>BUFG</td>
</tr>
<tr>
<td>CLKOUT6_DRIVES</td>
<td>BUFG</td>
</tr>
<tr>
<td>CLKOUT7_DRIVES</td>
<td>BUFG</td>
</tr>
<tr>
<td>FEEDBACK_SOURCE</td>
<td>FDBK_AUTO</td>
</tr>
<tr>
<td>CLKFB_IN_SIGNALING</td>
<td>SINGLE</td>
</tr>
<tr>
<td>CLKFB_IN_PORT</td>
<td>CLKFB_IN</td>
</tr>
<tr>
<td>CLKFB_IN_P_PORT</td>
<td>CLKFB_IN_P</td>
</tr>
<tr>
<td>CLKFB_IN_N_PORT</td>
<td>CLKFB_IN_N</td>
</tr>
<tr>
<td>CLKFB_OUT_PORT</td>
<td>CLKFB_OUT</td>
</tr>
<tr>
<td>CLKFB_OUT_P_PORT</td>
<td>CLKFB_OUT_P</td>
</tr>
<tr>
<td>CLKFB_OUT_N_PORT</td>
<td>CLKFB_OUT_N</td>
</tr>
<tr>
<td>PLATFORM</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>SUMMARY_STRINGS</td>
<td>empty</td>
</tr>
<tr>
<td>USE_LOCKED</td>
<td>TRUE</td>
</tr>
<tr>
<td>CALC_DONE</td>
<td>empty</td>
</tr>
<tr>
<td>USE_RESET</td>
<td>TRUE</td>
</tr>
<tr>
<td>USE_POWER_DOWN</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
### Table 4-1: XCI Parameters (Cont’d)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE_STATUS</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_FREEZE</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_CLK_VALID</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_INCLK_STOPPED</td>
<td>FALSE</td>
</tr>
<tr>
<td>USE_CLKFB_STOPPED</td>
<td>FALSE</td>
</tr>
<tr>
<td>RESET_PORT</td>
<td>RESET</td>
</tr>
<tr>
<td>LOCKED_PORT</td>
<td>LOCKED</td>
</tr>
<tr>
<td>POWER_DOWN_PORT</td>
<td>POWER_DOWN</td>
</tr>
<tr>
<td>CLK_VALID_PORT</td>
<td>CLK_VALID</td>
</tr>
<tr>
<td>STATUS_PORT</td>
<td>STATUS</td>
</tr>
<tr>
<td>CLK_IN_SEL_PORT</td>
<td>CLK_IN_SEL</td>
</tr>
<tr>
<td>INPUT_CLK_STOPPED_PORT</td>
<td>INPUT_CLK_STOPPED</td>
</tr>
<tr>
<td>CLKFB_STOPPED_PORT</td>
<td>CLKFB_STOPPED</td>
</tr>
<tr>
<td>SS_MODE</td>
<td>CENTER_HIGH</td>
</tr>
<tr>
<td>SS_MOD_FREQ</td>
<td>250</td>
</tr>
<tr>
<td>OVERRIDE_MMCM</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_NOTES</td>
<td>None</td>
</tr>
<tr>
<td>MMCM_DIVCLK_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>MMCM_BANDWIDTH</td>
<td>OPTIMIZED</td>
</tr>
<tr>
<td>MMCM_CLKFBOUT_MULT_F</td>
<td>10</td>
</tr>
<tr>
<td>MMCM_CLKFBOUT_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKFBOUT_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKIN1_PERIOD</td>
<td>10</td>
</tr>
<tr>
<td>MMCM_CLKIN2_PERIOD</td>
<td>10</td>
</tr>
<tr>
<td>MMCM_CLKOUT4_CASCADE</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLOCK_HOLD</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_COMPENSATION</td>
<td>ZHOLD</td>
</tr>
<tr>
<td>MMCM_REF_JITTER1</td>
<td>0.01</td>
</tr>
<tr>
<td>MMCM_REF_JITTER2</td>
<td>0.01</td>
</tr>
<tr>
<td>MMCM_STARTUP_WAIT</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKOUT0_DIVIDE_F</td>
<td>10</td>
</tr>
<tr>
<td>MMCM_CLKOUT0_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>MMCM_CLKOUT0_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKOUT0_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKOUT1_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>MMCM_CLKOUT1_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>MMCM_CLKOUT1_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKOUT1_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKOUT2_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>MMCM_CLKOUT2_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>MMCM_CLKOUT2_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKOUT2_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKOUT3_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>MMCM_CLKOUT3_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>MMCM_CLKOUT3_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKOUT3_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKOUT4_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>MMCM_CLKOUT4_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>MMCM_CLKOUT4_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKOUT4_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKOUT5_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>MMCM_CLKOUT5_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>MMCM_CLKOUT5_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKOUT5_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>MMCM_CLKOUT6_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>MMCM_CLKOUT6_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>MMCM_CLKOUT6_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>MMCM_CLKOUT6_USE_FINE_PS</td>
<td>FALSE</td>
</tr>
<tr>
<td>OVERRIDE_PLL</td>
<td>FALSE</td>
</tr>
<tr>
<td>PLL_NOTES</td>
<td>None</td>
</tr>
<tr>
<td>PLL_BANDWIDTH</td>
<td>OPTIMIZED</td>
</tr>
<tr>
<td>PLL_CLKFBOUT_MULTI</td>
<td>4</td>
</tr>
<tr>
<td>PLL_CLKFBOUT_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>PLL_CLK_FEEDBACK</td>
<td>CLKFBOUT</td>
</tr>
<tr>
<td>PLL_DIVCLK_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>PLL_CLKIN_PERIOD</td>
<td>10</td>
</tr>
<tr>
<td>PLL_COMPENSATION</td>
<td>SYSTEM_SYNCHRONOUS</td>
</tr>
<tr>
<td>PLL_REF_JITTER</td>
<td>0.01</td>
</tr>
<tr>
<td>PLL_CLKOUT0_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>PLL_CLKOUT0_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Output Generation

Vivado IP Catalog outputs the core as a netlist that can be inserted into a processor interface wrapper or instantiated directly in an HDL design. The output is placed in the `<project directory>`.

File Details

The IP is generated by the Vivado tool in `<project_name>/project_name.ssrc/sources_1/ip/<component_name>`. The file and directory structure is as follows:

```
<component_name>.v[hd]
<component_name>_clk_wiz.v[hd]
<component_name>.veo
<component_name>.xdc
<component_name>_ex.tcl
<component_name>.xci
<component_name>.xml
<component_name>/

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

Table 4-1: XCI Parameters (Cont’d)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL_CLKOUT0_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>PLL_CLKOUT1_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>PLL_CLKOUT1_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>PLL_CLKOUT1_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>PLL_CLKOUT2_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>PLL_CLKOUT2_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>PLL_CLKOUT2_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>PLL_CLKOUT3_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>PLL_CLKOUT3_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>PLL_CLKOUT3_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>PLL_CLKOUT4_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>PLL_CLKOUT4_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>PLL_CLKOUT4_PHASE</td>
<td>0</td>
</tr>
<tr>
<td>PLL_CLKOUT5_DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>PLL_CLKOUT5_DUTY_CYCLE</td>
<td>0.5</td>
</tr>
<tr>
<td>PLL_CLKOUT5_PHASE</td>
<td>0</td>
</tr>
</tbody>
</table>
simulation/  
<component_name>_tb.v[hd]

functional/  
<component_name>_config.v[hd]
simulate_mti.do
simulate_ncsim.sh
ucli_commands.key
wave.do
simcmds.tcl
simulate_mti.sh
simulate_vcs.sh
simulate_xsim.sh
vcs_session.tcl
wave.sv

timing/  
sdf_cmd_file
simulate_mti.do
simulate_mti.sh
simulate_ncsim.sh
simulate_vcs.sh
simulate_xsim.sh
ucli_commands.key
vcs_session.tcl
wave.do
Constraining the Core

**Required Constraints**

At least one clock constraint is required for period and jitter.

```
NET "CLK_IN1" TNM_NET = "CLK_IN1";
TIMESPEC "TS_CLK_IN1" = PERIOD "CLK_IN1" 10 ns HIGH 50% INPUT_JITTER 200 ps;
```

**Device, Package, and Speed Grade Selections**

Supports all packages, speed grades and devices.

**Clock Frequencies**

See [Maximum Frequencies in Chapter 2](#)

**Clock Management**

The core can generate a maximum of seven output clocks with different frequencies.

**Clock Placement**

No clock placement constraint is provided.
Banking

Bank selection is not provided in xdc file.

I/O Standard and Placement

No I/O or placement constraints are provided.
Detailed Example Design

In Vivado design tools, the `open_example_project [get_ips <component_name>]` parameter in tcl console invokes a separate example design project where it creates `<component_name>_exdes` as top module for synthesis and `<component_name>_tb` as top module for simulation. You can run implementation or simulation of the example design from example project.

Directory and File Contents

The `open_example_project [get_ips <component_name>]` parameter creates example_project directory in the working area.

Example design contains the counters on all the output clocks and MSBs of these counters are used as output to observe on LEDs on board.

Example Design

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

- **VHDL**
  
  `<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/example_design/<component_name>_exdes.vhd`

- **Verilog**
  
  `<project_name>/<project_name>.srcs/sources_1/ip/<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.
Demonstration Test Bench

The following files describe the demonstration test bench.

- VHDL

  `<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/simulation/<component_name>_tb.vhd`

- Verilog

  `<project_name>/<project_name>.srcs/sources_1/ip/<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. It does Frequency calculation and check of all the output clocks. It reports all the output clock frequency and if any of the output clocks is not generating the required frequency then it reports ERROR.

Simulation

User can simulate the example design from the `<project_name>/`<project_name>.srcs/sources_1/ip/component_name>/simulation/` area in functional or timing directories using available scripts or the open_example_project flow in Vivado design tools.

If you open an example project, then the simulation scripts are generated in the working directory in:

- For Linux

  `example_project/<component_name>.sim/sim_1/compile.sh`

- For Windows

  `example_project/<component_name>.sim/sim_1/compile.bat`
SECTION III: ISE DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Detailed Example Design
Chapter 7

Customizing and 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.

![Clocking Features GUI](image)

*Figure 7-1: Clocking Features (Spread Spectrum Unselected)*
Figure 7-2: Clocking Features (Spread Spectrum Selected)
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.

- **Spread Spectrum (SS).** This feature provides modulated output clocks which reduces the spectral density of the electromagnetic interference (EMI) generated by electronic devices. This feature is available only for MMCME2 primitive.

- **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 features 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.

Valid input frequency ranges are:

- Frequency when SS is unselected: 10 - 800 MHz
- Frequency when SS is selected: 25 - 150 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 7-3) configures requirements for the output clocks. Each selected output clock can be configured on this screen.

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 \texttt{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.

For Spartan6 DCM\_SP, Clocking Wizard supports output clock frequency selection in descending order. In this \texttt{CLK\_OUT1} is selected as CLKFX and \texttt{CLK\_OUT2} is selected as CLKDV. For example if you select 42 MHz as input and want to generate 21 MHz and 10.5 MHz output clock then you need to use \texttt{CLK\_OUT1} for 21 MHz and \texttt{CLK\_OUT2} for 10.5 MHz.

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 \textbf{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 \textit{Virtex-6 FPGA Clocking Resources User Guide} for more details.

When Spread Spectrum (SS) is selected, \texttt{CLK\_OUT\<3\>} and \texttt{CLK\_OUT\<4\>} are not available. Divide values of these outputs are used for SS modulation frequency generation.
There are four modes available for SS Mode:

- DOWN_LOW
- DOWN_HIGH
- CENTER_LOW
- CENTER_HIGH

Available Modulation Frequency range:

- 25 to 250 KHz

Spread Spectrum calculation details are described in Figure 7-5 and Figure 7-6.
**Note:** Input\_clock\_frequency is in Hz unit.

For spread:

- If (SS\_Mode = CENTER\_HIGH) :=>
  - $spread (ps) = +/- \left[ \frac{1}{Input\_clock\_frequency \times (M - 0.125 \times 4) / D / O} - \frac{1}{Input\_clock\_frequency \times M / D / O} \right]$

- If (SS\_Mode = CENTER\_LOW) :=>
  - $spread (ps) = +/- \left[ \frac{1}{Input\_clock\_frequency \times (M - 0.125 \times 2) / D / O} - \frac{1}{Input\_clock\_frequency \times M / D / O} \right]$

- If (SS\_Mode = DOWN\_HIGH) :=>
  - $spread (ps) = + \left[ \frac{1}{Input\_clock\_frequency \times (M - 0.125 \times 4) / D / O} - \frac{1}{Input\_clock\_frequency \times M / D / O} \right]$

- If (SS\_Mode = DOWN\_LOW) :=>
Output Clock Settings

- spread (ps) = + [1/(Input_clock_frequency*(M-0.125*2)/D/O) - 1/(Input_clock_frequency*M/D/O)]

Where M is CLKFBOUT_MULT_F, D is DIVCLK_DIVIDE, and O is respective CLKOUTx_DIVIDE.

- For Modulation Frequency:
  - O2 and O3 are calculated by the bitgen in implementation. Same calculation is done in the wizard to get actual modulation frequency value.
  - Then based on what O2 and O3 is calculated, the actual modulation frequency is calculated:
    - If (SS_Mode = CENTER_HIGH or SS_Mode = CENTER_LOW) Actual_modulation_frequency (average) = (Input_clock_frequency*M/D) / (O2 * O3) / 16
    - If (SS_Mode = DOWN_HIGH) Actual_modulation_frequency (average) = 0.5 *
      [[[Input_clock_frequency*M/D) / (O2 * O3) / 8] + ((Input_clock_frequency*(M-0.5)/D) / (O2 * O3) / 8)]
    - If (SS_Mode = DOWN_LOW) Actual_modulation_frequency (average) = 0.5 *
      [[[Input_clock_frequency*M/D) / (O2 * O3) / 8] + ((Input_clock_frequency*(M-0.25)/D) / (O2 * O3) / 8)]

Note: Actual modulation frequency may deviate within +/- 10% of the requested modulation frequency for some settings.
Feedback and I/O

The third GUI screen (Figure 7-7) provides information to configure the rest of the clocking network.

![Figure 7-7: Feedback and I/O Configuration Screen](image)

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 7-8.

![Primitive Override Screen (Spread Spectrum Unselected)](image)

*Figure 7-8: Primitive Override Screen (Spread Spectrum Unselected)*
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 7-10). You can manually add an additional comment in the box at the bottom of the page. Use underscores in place of spaces. Bandwidth is always set to low for Spread Spectrum clocking.

Clock Summary (First Page)

The first summary page (Figure 7-10) 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 7-10: Clock Summary Screen (Spread Spectrum Unselected)
Clock Summary (First Page)

Tspread is the actual spread as calculated in Configuring Output Clocks.

Input Clocking Summary

Information entered on the first page of the GUI (Figure 7-1) 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 you to name the ports according to your needs. If you want to name the HDL port for primary clock input, simply type the 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 7-12) contains general summary information.

---

**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 9, Detailed Example Design.
Chapter 8

Constraining the Core

Required Constraints
At least one clock constraint is required for period and jitter.

```
NET "CLK_IN1" TNM_NET = "CLK_IN1";
TIMESPEC "TS_CLK_IN1" = PERIOD "CLK_IN1" 10 ns HIGH 50% INPUT_JITTER 200 ps;
```

Device, Package, and Speed Grade Selections
Supports all packages, speed grades and devices.

Clock Frequencies
See Maximum Frequencies in Chapter 2

Clock Management
The core can generate a maximum of seven output clocks with different frequencies.

Clock Placement
No clock placement constraint is provided.
Banking

Bank selection is not provided in ucf file.

I/O Standard and Placement

No I/O or placement constraints are provided.
Chapter 9

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

- `<component name>/implement/results`
  Results directory, created after implementation scripts are run, and contains implement script results

- `<component name>/simulation`
  Simulation scripts

- `<component name>/simulation/functional`
  Functional simulation files

- `<component name>/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 9-1:  Project Directory

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

<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 9-2:  Component Name Directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clocking_wizard_v3_5_readme.txt</td>
<td>Clocking Wizard v3.5 readme file.</td>
</tr>
<tr>
<td>&lt;component_name&gt;.ucf</td>
<td>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.</td>
</tr>
</tbody>
</table>
<component name>/example design

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

Table 9-3: Example Design Directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;project dir&gt;/&lt;component name&gt;/example_design</td>
<td>Implementable Verilog or VHDL example design, with all created clocks driving a counter.</td>
</tr>
<tr>
<td>&lt;component name&gt;_exdes.v[hd]</td>
<td>Constraint files containing the timing parameters for the example design clock network.</td>
</tr>
</tbody>
</table>

<component name>/doc

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

Table 9-4: Doc Directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;project dir&gt;/&lt;component name&gt;/doc</td>
<td>Clocking Wizard v4.2 Product Guide</td>
</tr>
</tbody>
</table>

<component name>/implement

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

Table 9-5: Implement Directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;project dir&gt;/&lt;component name&gt;/implement</td>
<td>Scripts and projects to implement the example design</td>
</tr>
</tbody>
</table>

implement/results

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

Table 9-6: Results Directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;project dir&gt;/&lt;component name&gt;/implement/results</td>
<td>Implement script result files.</td>
</tr>
</tbody>
</table>
<component name>/simulation

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;project_dir&gt;/&lt;component_name&gt;/simulation</td>
<td></td>
</tr>
<tr>
<td>&lt;component_name&gt;_tb.v[hd]</td>
<td>Demonstration test bench</td>
</tr>
</tbody>
</table>

**Table 9-7: Simulation Directory**

**simulation/functional**

The functional directory contains functional simulation scripts.

**Table 9-8: Functional Directory**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;project_dir&gt;/&lt;component_name&gt;/simulation/functional</td>
<td>Contains simulation scripts and waveform formats.</td>
</tr>
</tbody>
</table>

**simulation/timing**

The timing directory contains timing simulation scripts.

**Table 9-9: Functional Directory**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;project_dir&gt;/&lt;component_name&gt;/simulation/timing</td>
<td>Contains simulation scripts and waveform formats.</td>
</tr>
</tbody>
</table>
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:

\[\text{<project_dir>/<component_name>/simulation/functional/}\]

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:

\[\text{<project_dir>/<component_name>/simulation/timing/}\]

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 9-1: 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
SECTION IV: APPENDICES

Verification, Compliance, and Interoperability

Migrating

Debugging

Additional Resources
Verification, Compliance, and Interoperability

Simulation

Verified with all the supported simulators.

Hardware Testing

Hardware testing is performed for all the features on Kintex-7 KC705 Evaluation Kit using the provided example design.
Appendix B

Application Software Development

Device Drivers
Migrating

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.

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

Debugging

See Solution Centers in Appendix E for information helpful to the debugging progress.
Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:


Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

See the IDS Embedded Edition Derivative Device Support web page (www.xilinx.com/ise/embedded/ddsupport.htm) for a complete list of supported derivative devices for this core.

See the IP Release Notes Guide (XTP025) for more information on this core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:
Revision History

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
</table>

Notice of Disclaimer

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 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, 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.