Clocking Wizard v5.1

LogiCORE IP Product Guide

Vivado Design Suite

PG065 April 1, 2015

XILINX
ALL PROGRAMMABLE™

Table of Contents

Introduction

The LogiCORE™ IP Clocking Wizard core simplifies the creation of 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 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

- Selection of mixed-mode clock manager (MMCM)/phase-locked loop (PLL) primitives. GUI options are enabled for the supported features for the primitives.
- Safe Clock Startup feature enables stable and valid clock at the output. Enabling Sequencing provides sequenced output clocks.
- Accepts up to two input clocks and up to seven output clocks per clock network.
- Provides AXI4-Lite interface for dynamically reconfiguring the clocking primitives for Multiply, Divide, Phase, or Duty Cycle.
- 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.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale™ Architecture, Zynq®-7000, 7 Series
Supported User Interfaces	AXI4-Lite
Resources	See Table 2-2.
Special Features	PLL(E2/E3), MMCM(E2/E3), Spread Spectrum Clocking
Provided with Core	
Design Files	Verilog and VHDL
Example Design	Verilog and VHDL
Test Bench	Verilog and VHDL
Constraints File	.xdc (Xilinx Design Constraints)
Simulation Model	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Instantiation Template	Verilog and VHDL Wrapper
Supported S/W Driver	Not Applicable
Tested Design Flows	
Design Entry Tools	Vivado® Design Suite
Simulation	Mentor Graphics Questa®SIM, Vivado Simulator
Synthesis Tools	Synplify PRO E-2012.03, Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete listing of supported devices, see the Vivado IP Catalog.

Features (continued)

- Automatically implements overall configuration that supports phase shift and duty cycle requirements.
- Supports Spread Spectrum clocking for MMCM(E2/E3) and allows users to select valid range of modulation frequency, mode and input/output clocks.
- Optionally buffers clock signals.
- 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 is a Xilinx IP core that can be generated using the Xilinx Vivado design tools, 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 you 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.
- **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 MMCM(E2/E3)_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 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, AXI4-Lite interface is selected by default for reconfiguring clocking primitive.
- **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'.
- **Safe Clock Startup and Sequencing**. This feature is useful to get stable and valid clock at the output. It also enables Clocks in a particular sequence order as specified in the configuration.

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

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado™ Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, 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 mixed-mode clock manager (MMCM)(E2/E3) or phase-locked loop (PLL)(E2/E3) 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. Figure 2-1 shows a block diagram of the Clocking Wizard.

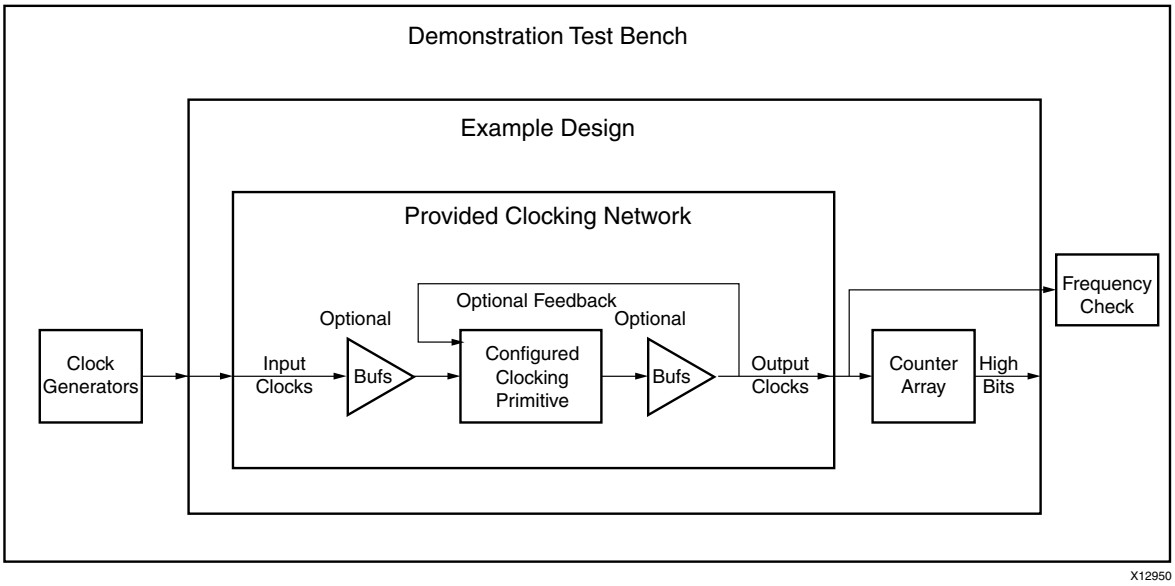

Figure 2-1: **Clocking Wizard Block Diagram**

Performance

Maximum Frequencies

Table 2-1 shows the maximum frequencies for Virtex®-7 devices. The maximum frequencies are same for MMCM and PLL.

Table 2-1: **Maximum Frequency Virtex-7 Devices**

Clock	Speed Grade		
	-1	-2	-3
Input	800 MHz	933 MHz	1066 MHz
Output	800 MHz	933 MHz	1066 MHz
S_AXI_ACLK	250 MHz	250 MHz	250 MHz

Table 2-2 shows the minimum frequencies of MMCM in Virtex®-7 devices.

Table 2-2: **Minimum Frequency of MMCM in Virtex-7 Devices**

Clock	Speed Grade		
	-1	-2	-3
Input	10	10	10
Output	10	10	10

Table 2-3 shows the minimum frequencies of PLL in Virtex®-7 devices.

Table 2-3: **Minimum Frequency of PLL in Virtex-7 Devices**

Clock	Speed Grade		
	-1	-2	-3
Input	19	19	19
Output	19	19	19

Power

- Minimize power feature minimizes the amount of power needed for the primitive at the possible expense of frequency, phase offset, or duty cycle accuracy.
- Power Down input pin when asserted, places the clocking primitive into low power state, which stops the output clocks.

Resource Utilization

Resource utilization is available in the Clocking Wizard GUI by clicking the **Resource** tab. This does not include AXI4-Lite resources when Dynamic Reconfiguration is enabled. Refer to Dynamic Reconfiguration through AXI4-Lite for more information.

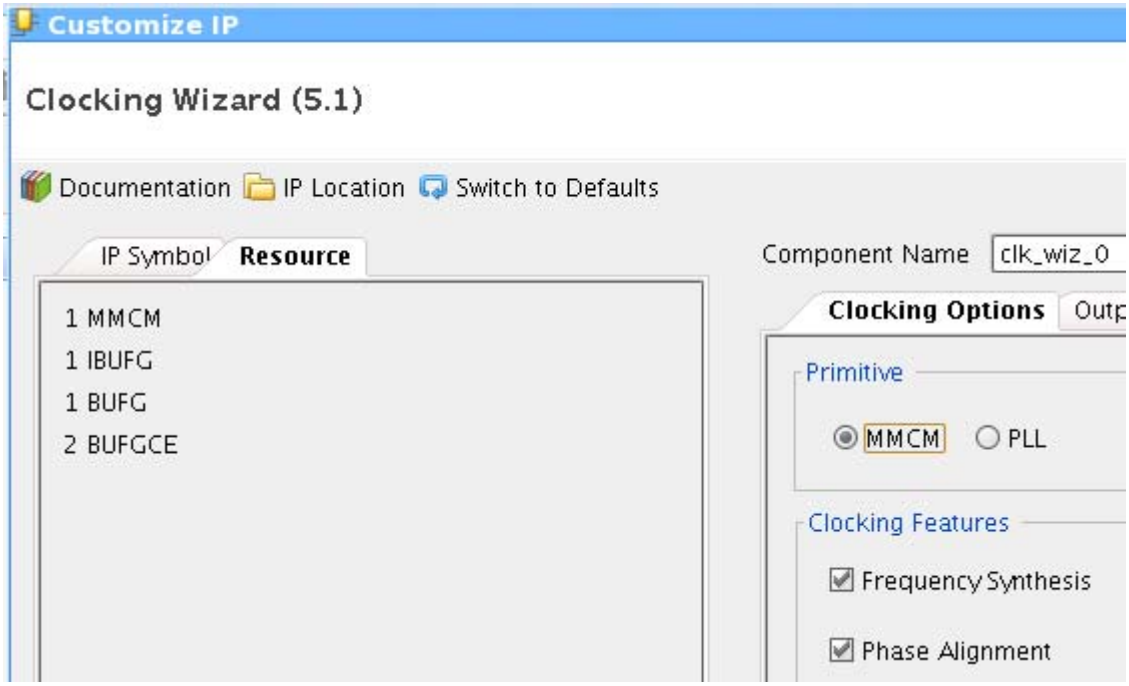

Figure 2-2: **Resource Tab**

Port Descriptions

Table 2-4 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 determine which ports are actually available to be configured. For example, when Dynamic Reconfiguration is selected, these ports are exposed. Any port that is not exposed is appropriately tied off or connected to a signal labeled *unused* in the delivered source code.

Table 2-4: **Clocking Wizard I/O**

Port [5]	I/O	Description
Input Clock Ports [1]		
clk_in1	Input	**Clock in 1**: Single-ended primary input clock port. Available when single-ended primary clock source is selected.
clk_in1_p	Input	**Clock in 1 Positive and Negative**: Differential primary input clock port pair. Available when a differential primary clock source is selected.
clk_in1_n		
clk_in2 [2]	Input	**Clock in 2:** Single-ended secondary input clock port. Available when a single-ended secondary clock source is selected.
clk_in2_p [2]	Input	**Clock in 2 Positive and Negative:** Differential secondary input clock port pair. Available when a differential secondary clock source is selected.
clk_in2_n [2]		
clk_in_sel [2]	Input	**Clock in Select:** When '1', selects the primary input clock; When '0', the secondary input clock is selected. Available when two input clocks are specified.
clkfb_in	Input	**Clock Feedback in**: Single-ended feedback in port of the clocking primitive. Available when user-controlled on-chip, user controller-off chip, or automatic control off-chip feedback option is selected.
clkfb_in_p	Input	**Clock Feedback in: Positive and Negative:** Differential feedback in port of the clocking primitive. Available when the automatic control off-chip feedback and differential feedback option is selected.
clkfb_in_n	Input	
Output Clock Ports		
clk_out1	Output	**Clock Out 1:** Output clock of the clocking network. clk_out1 is not optional.
clk_out1_ce	Input	**Clock Enable:** Chip enable pin of the output buffer. Available when BUFGCE or BUFHCE or BUFR buffers are used as output clock drivers.
clk_out1_clr	Input	**Counter reset for divided clock output:** Available when BUFR buffer is used as output clock driver.
clk_out2_n [3]	Output	**Clock Out 2 - n:** Optional output clocks of the clocking network that are user-specified. For an MMCM, up to seven are available. For UltraScale PLLE3, up to two clocks are available and for 7 series/Zynq-7000PLLE2, up to six clocks are available

Table 2-4: **Clocking Wizard I/O *(Cont'd)***

Port [5]	I/O	Description
clk_out[2-n]_ce [3]	Input	**Clock Enable:** Chip enable pin of the output buffer. Available when BUFGCE or BUFHCE or BUFR buffers are used as output clock drivers.
clk_out[2-n]_clr [3]	Input	**Counter reset for divided clock output:** Available when BUFR buffer is used as output clock driver.
clkfb_out	Output	**Clock Feedback Out:** Single ended feedback port of the clocking primitive. Available when the user-controlled feedback or automatic control off chip with single ended feedback option is selected.
clkfb_out_p	Output	**Clock Feedback Out: Positive and Negative:** Differential feedback output port of the clocking primitive. Available when the user-controlled off-chip feedback and differential feedback option is selected.
clkfb_out_n	Output	
Dynamic Reconfiguration Ports		
daddr[6:0]	Input	**Dynamic Reconfiguration Address:** Address port for use in dynamic reconfiguration; active when den is asserted
dclk	Input	**Dynamic Reconfiguration Clock:** Clock port for use in dynamic reconfiguration
den	Input	**Dynamic Reconfiguration Enable:** Starts a dynamic reconfiguration transaction. Refer to DRP protocol details for more information.
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

Table 2-4: **Clocking Wizard I/O *(Cont'd)***

Port [5]	I/O	Description
psdone	Output	**Dynamic Phase Shift Done:** Completes a dynamic phase shift transaction
Status and Control Ports [4]		
reset/resetn	Input	**Reset (Active High)/Resetn (Active Low):** 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
cddcreq[6]	Input	Clock Divide Dynamic Change (CDDC) request. This is asserted after last DRP request is performed and then de-asserted after last DRDY
cddcdone[6]	Output	Clock Divide Dynamic Change (CDDC) done. When output counters are updated this signal is asserted
s_axi_aclk	Input	AXI Clock
s_axi_aresetn	Input	AXI Reset, Active-Low
s_axi_awaddr[10:0]	Input	AXI Write address. The write address bus gives the address of the write transaction.
s_axi_awvalid	Input	Write address valid. This signal indicates that a valid write address and control information are available.
s_axi_awready	Output	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
s_axi_wdata[31:0]	Input	Write data
s_axi_wstb[3:0]	Input	Write strobes. This signal indicates which byte lanes to update in memory.
s_axi_wvalid	Input	Write valid. This signal indicates that valid write data and strobes are available.
s_axi_wready	Output	Write ready. This signal indicates that the slave can accept the write data.
s_axi_bresp[1:0]	Output	Write response. This signal indicates the status of the write transaction 00 = OKAY (normal response) 10 = SLVERR (error condition) 11 = DECERR (not issued by core)

Table 2-4: **Clocking Wizard I/O *(Cont'd)***

Port [5]	I/O	Description
s_axi_bvalid	Output	Write response valid. This signal indicates that a valid write response is available.
s_axi_bready	Input	Response ready. This signal indicates that the master can accept the response information.
s_axi_araddr[10:0]	Input	Read address. The read address bus gives the address of a read transaction.
s_axi_arvalid	Input	Read address valid. This signal indicates, when High, that the read address and control information is valid and remains stable until the address acknowledgement signal, s_axi_arready, is High.
s_axi_arready	Output	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
s_axi_rdata[31:0]	Output	Read data
s_axi_rresp[1:0]	Output	Read response. This signal indicates the status of the read transfer. 00 = OKAY (normal response) 10 =SLVERR (error condition) 11 = DECERR (not issued by core)
s_axi_rvalid	Output	Read valid. This signal indicates that the required read data is available and the read transfer can complete.
s_axi_rready	Input	Read ready. This signal indicates that the master can accept the read data and response information.
s_axis_aclk	Input	The global clock signal. All streaming signals from Read interface of the FIFO are sampled on the rising edge of s_axis_aclk.

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 primitive chosen is UltraScale PLL or Spread Spectrum is selected for MMCM.
3. The clk_out3 and clk_out4 ports 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 2-4 is the default port list.
6. Ports used for dynamic change of output counter without reset. Available only in MMCME3 primitive.

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 (if the input clock is output of global buffer), or global buffer option for source type. If the input clock is used only by core, 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.

If you are already familiar with the Digital Clock Manager (DCM) and Phase-Locked Loop (PLL) wizards, refer to Appendix B, Migrating for information on usage differences.

Clocking Features

Major clocking-related functional features desired and specified 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
- Safe Clock Startup and Sequencing

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 you to specify the input clock jitter either in UI or PS units using a drop-down menu.

Output Clocks

The number of output clocks is user-configurable. The maximum number allowed depends upon the selected device or primitive and the interaction of the major clocking features you specify. For MMCM(E2/E3) maximum seven, PLLE2 maximum six and PLLE3 maximum two output clocks can be configured. Input the 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.

TIP: *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. 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.

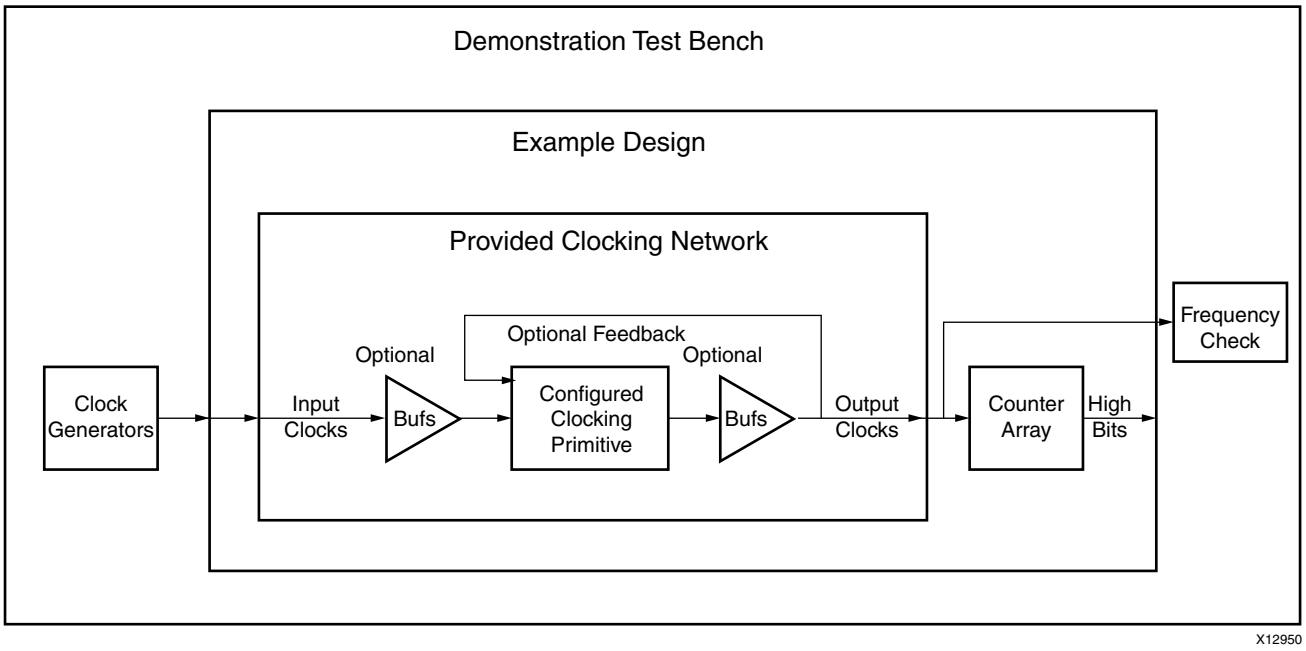

Figure 3-1: **Clocking Network and Support Modules**

Core Architecture

The Clocking Wizard generates source code HDL to implement a clocking network. The generated clocking network typically consists of a clocking primitive (MMCM(E2/E3)_ADV or PLL(E2/E3)_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.

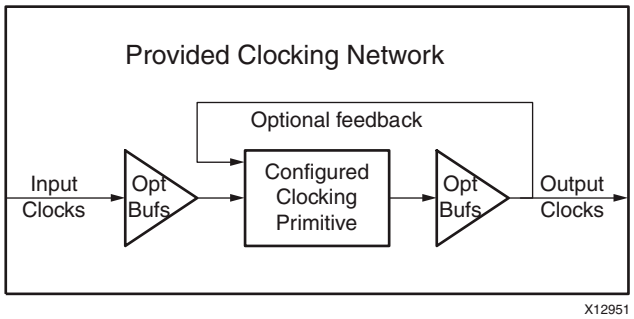

Figure 3-2: **Provided Clocking Network**

Input Clocks

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

Primitive Instantiation

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

Feedback

If phase alignment is not selected, the feedback output port on the primitive is automatically tied to the feedback input port. If phase alignment with automatic feedback is selected, the connection is made, but the path delay is matched to that of clk_out1. If user-controlled feedback is selected, the feedback ports are exposed.

Output Clocks

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

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. Any port that is not exposed is either tied off or connected to a signal labeled *unused* in the delivered source code.

IMPORTANT: *Not all ports are available for all devices or primitives; for example, Dynamic Phase Shift is not available when Spread Spectrum is selected.*

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows in the IP Integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 7]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5]

Customizing and Generating the Core

Vivado Integrated Design Environment (IDE)

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click on the selected IP or select the Customize IP command from the toolbar or popup menu.

For details, see the sections, "Working with IP" and "Customizing IP for the Design" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and the "Working with the Vivado IDE" section in the *Vivado Design Suite User Guide: Getting Started* (UG810) [Ref 4].

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 7] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl console.

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

Clock Manager Type (Primitive Selection)

In Zynq®-7000 and 7 series devices, MMCME2 and PLLE2 primitives are available for the clocking needs. In UltraScale architecture, MMCME3 and PLLE3 primitives are available for clocking needs. You have the option to configure either of these by selecting the primitive. Features are enabled or disabled depending on the primitive selected.

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.

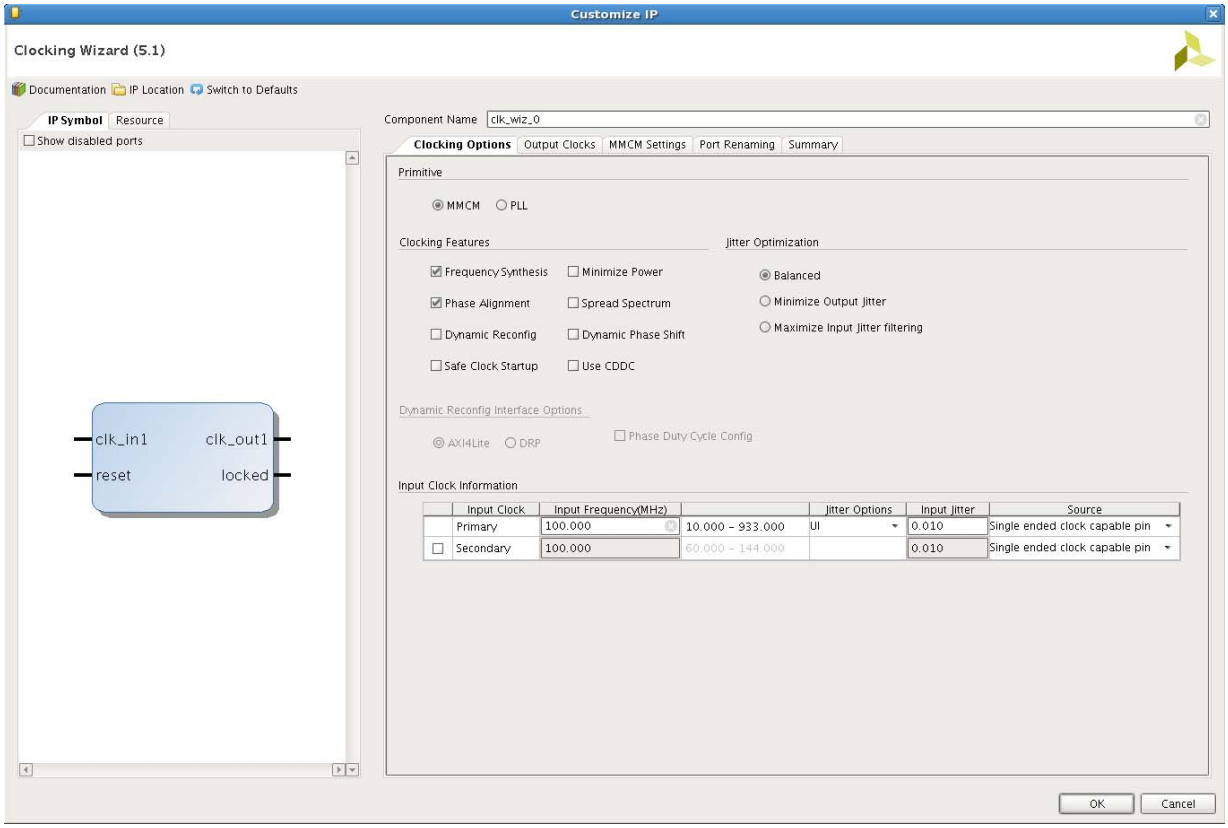

Figure 4-1: **Clocking Options for 7 Series MMCM (Spread Spectrum Unselected)**

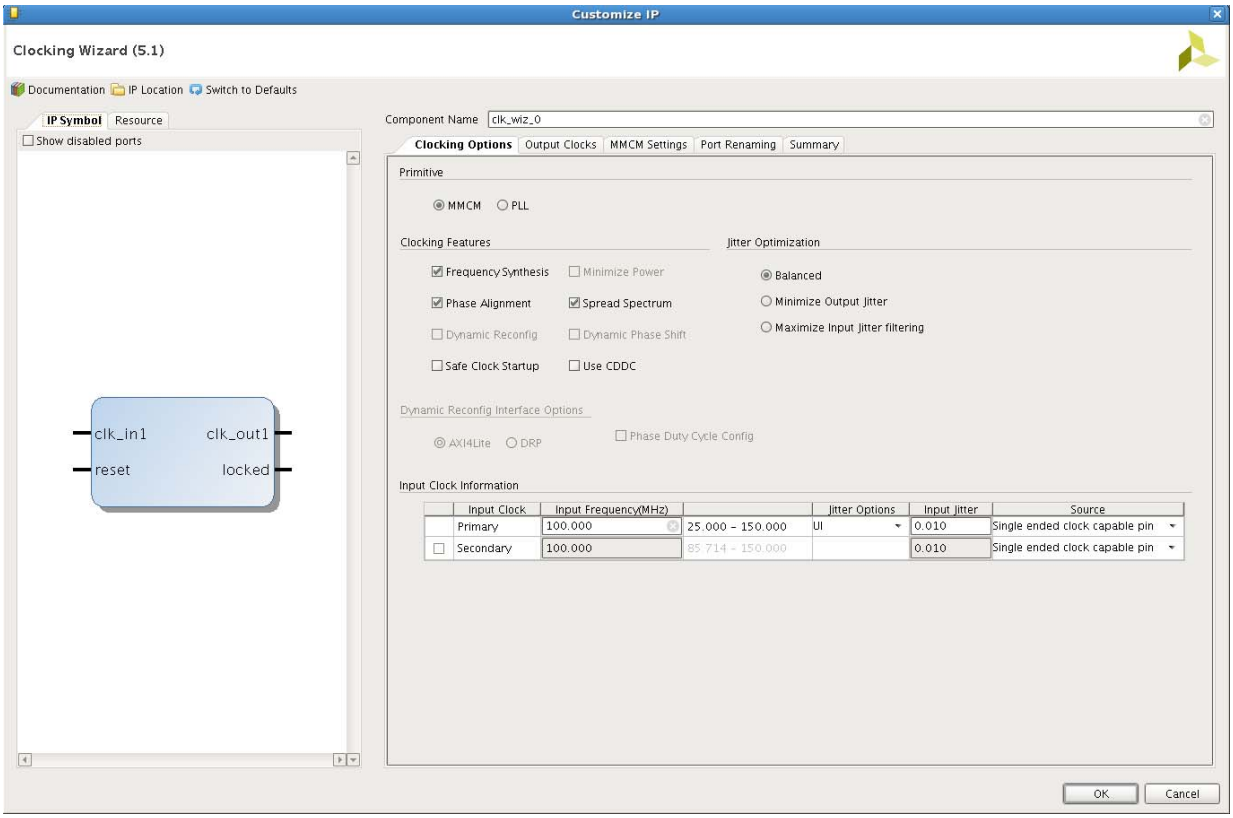

Figure 4-2: **Clocking Options for 7 Series MMCM (Spread Spectrum Selected)**

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.

When using IP Integrator, Frequency, Phase and Clock Domain properties of the output clocks are automatically propagated and any change on input clock properties reflect on all the outputs.

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 MMCM(E2/E3) primitive. Minimize power, Dynamic Reconfig features are not available when Spread Spectrum is TRUE.
- **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. This feature is not available when Spread Spectrum feature is selected.
- **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, AXI4-Lite interface is selected by default for reconfiguring clocking primitive. DRP interface can be selected if direct access to MMCM/PLL DRP register is required. Refer to Dynamic Reconfiguration through AXI4-Lite for more information.
- **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'.
- **Safe Clock Startup and Sequencing.** Safe Clock Startup feature enables stable and valid clock at the output using BUFGCE after Locked is sampled High for 8 input clocks. Sequencing feature enables Clocks in a sequence according to the number entered through GUI. Delay between two enabled output clocks in sequence is 8 cycle of second clock in the sequence clock. This feature is useful for a system where modules need to be start operating one after the other.

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 – 1066 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 drop-down menu. 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.

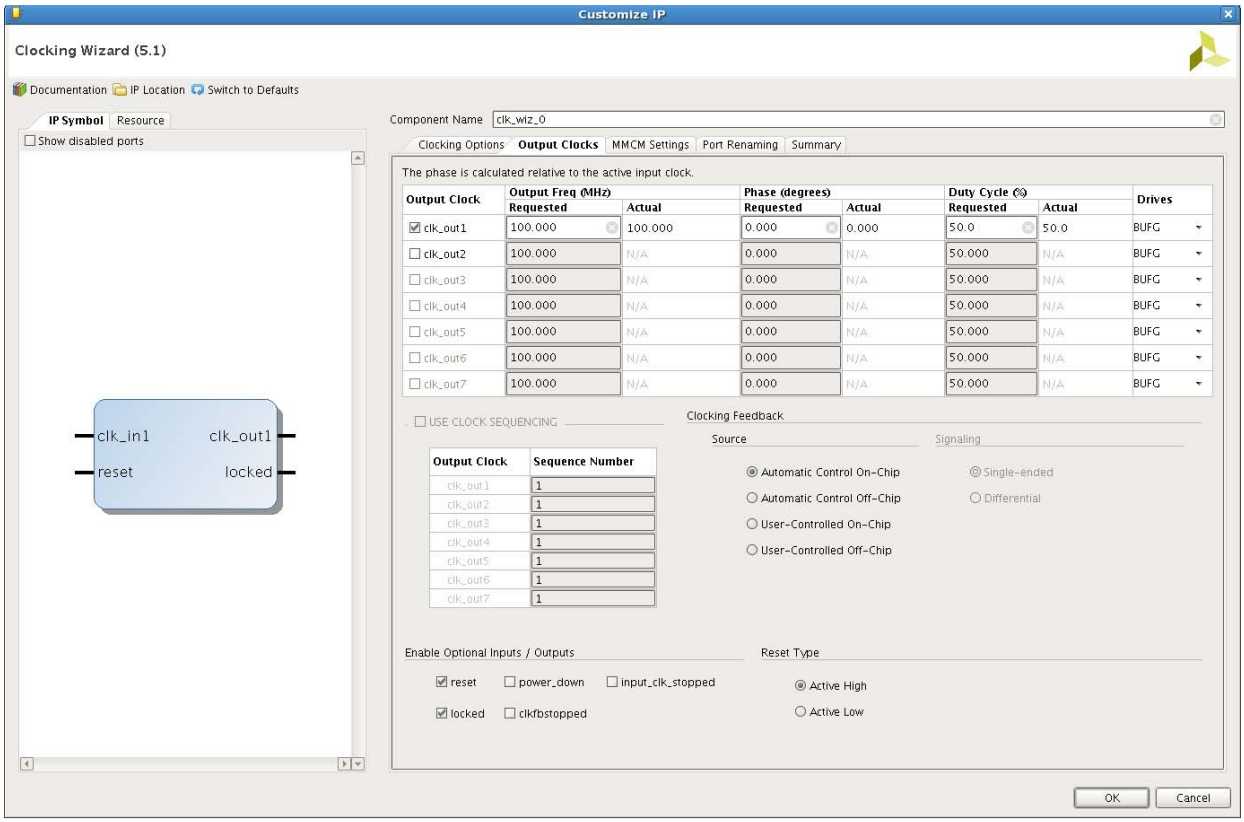

Figure 4-3: **Output Clocks for 7 Series MMCM (Spread Spectrum Unselected)**

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 `clk_out1` > `clk_out2` > `clk_out3`, 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 180° phase shift is requested on `Clk_out2`, `clk_out3`, `clk_out4`, or `clk_out5`, then the Wizard connects any of these clocks to previous clocks Inverted clock outputs (`clkout[0:3]B`) of MMCM/PLL, as compared to the previous clock and other properties like Frequency, duty cycle, etc., are identical to the previous clock. Consider that `clk_out1` is configured with 100 MHz and 0° phase shift and `clk_out2` is configured 100 MHz with 180° phase shift. Then `clk_out2` is connected to `clkout0b`. If `clk_out1` and `clk_out2` are 180° phase shifted and `CLK_OUT2` and `clk_out3` are 180° phase shifted, then `clk_out3` uses it's own phase settings and is connected to `clkout2` of MMCM. If you have another clock `CLK_OUT4` with 180° phase shift compared to `clk_out3`, then `clk_out4` is connected to `clkout2b`.

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. You 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 MMCM(E2/E3). 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 **Safe Clock Startup** feature is enabled on the first tab of the GUI, the **Use Clock Sequencing** table is active and Sequence number for each enabled clock is available for the configuration. In this mode only BUFGCE is allowed as Drives of the clock outputs.

Both 7 series and UltraScale devices support MMCM fractional divide functionality in increments of 1/8th (0.125) for CLKFBOUT and CLKOUT0, and can support greater clock frequency synthesis.

The resolution of the fractional divide is 1/8 or 0.125 degrees, effectively increasing the number of synthesizeable frequencies by a factor of eight. For example, if the CLKIN frequency is 100 MHz and the M divide value is set to 8, then the VCO frequency is 800 MHz. CLKOUT0 can be used to further fractionally divide the 800 MHz VCO frequency (for example, CLKOUT0_DIVIDE = 2.5 resulting in a 320 MHz output frequency).

When using the fractional divider, the duty cycle is not programmable for outputs used in the fractional mode.

Fractional divide is not allowed in either fixed or dynamic phase-shift mode. CDDC feature is not available in the fractional divide mode for UltraScale devices.

Refer to *7 Series FPGAs Clocking Resources User Guide* (UG47) [Ref 9] and *UltraScale Architecture Clocking Resources User Guide* (UG572) [Ref 8] for more information.

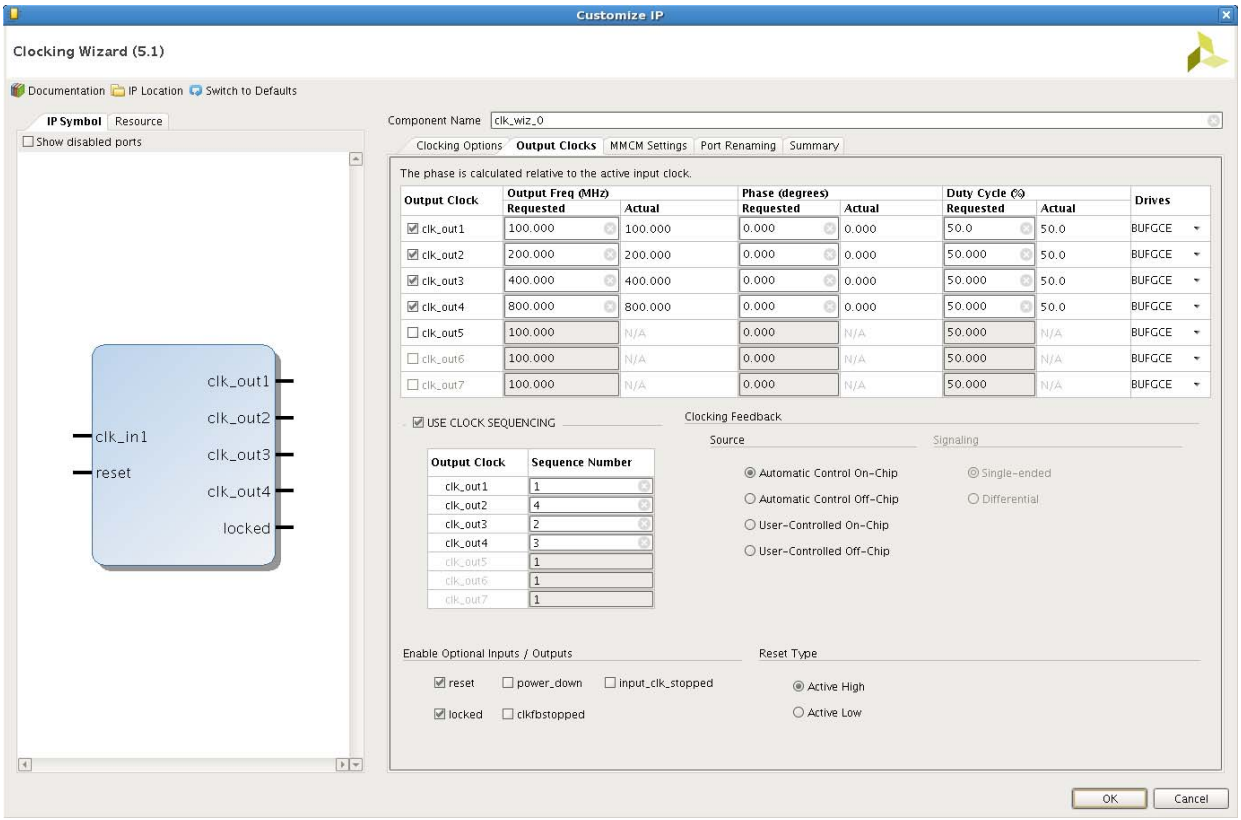

Figure 4-4: **Output Clocks with Safe Clock Start Up and Clock Sequencing for 7 Series MMCM**

You can configure the sequence number from 1 to the maximum number of clocks selected. Clocking Wizard does not allow any break in the sequence from one to maximum in the table. Clock Frequency of the output clock in Sequence should not be more than eight times of the output clock next in sequence.

For details of the clocking behavior in this mode, refer to Figure 4-5 and Figure 4-6.

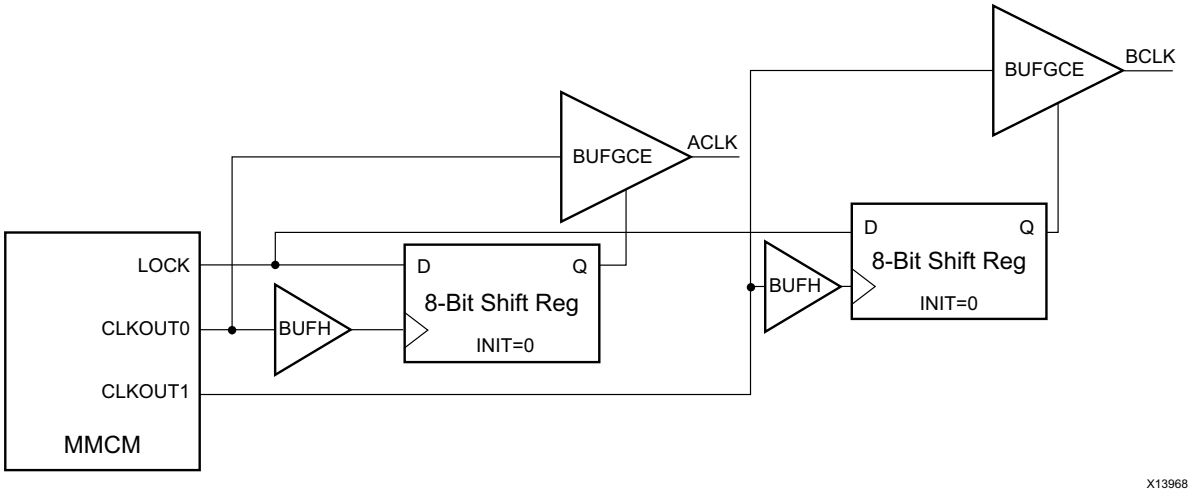

Figure 4-5: **Safe Clock Start Up**

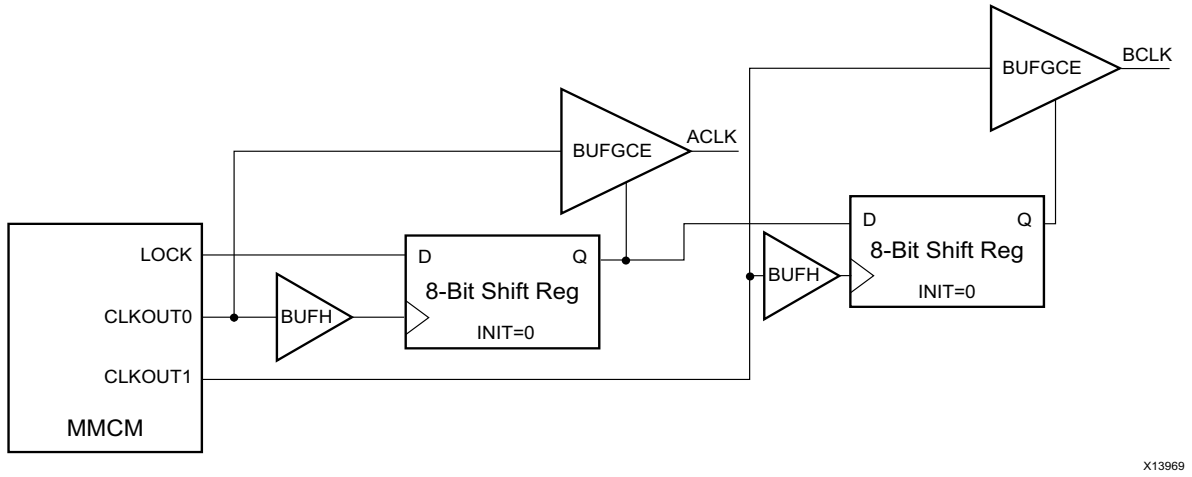

Figure 4-6: **Safe Clock Start Up with Sequencing**

When Spread Spectrum (SS) is selected, `CLK_OUT<3>` and `CLK_OUT<4>` are not available. Divide values of these outputs are used for SS modulation frequency generation.

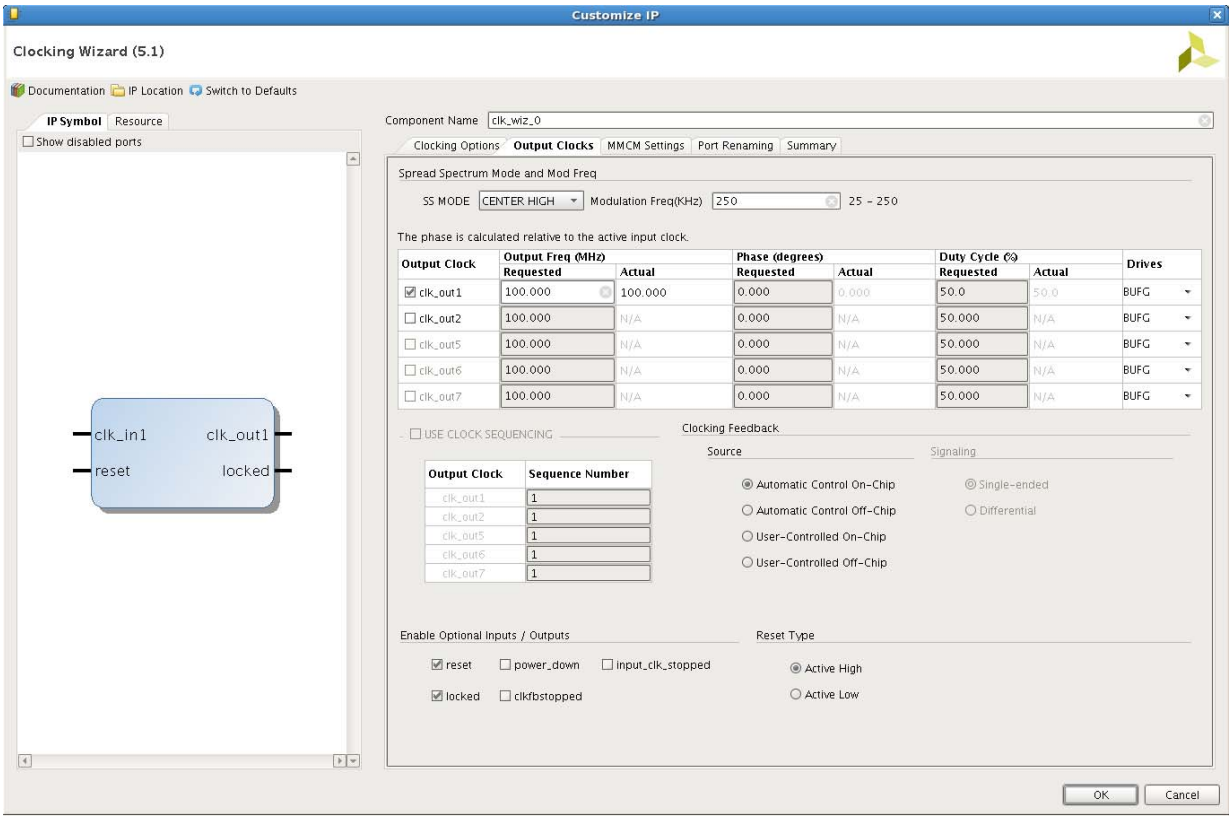

Figure 4-7: **Output Clocks for 7 Series MMCM (Spread Spectrum Selected)**

There are four modes available for SS Mode:

- DOWN_LOW
- DOWN_HIGH
- CENTER_LOW
- CENTER_HIGH

Available Modulation Frequency range is 25 – 250 KHz

Spread Spectrum calculation details are described in Figure 4-8 and Figure 4-9.

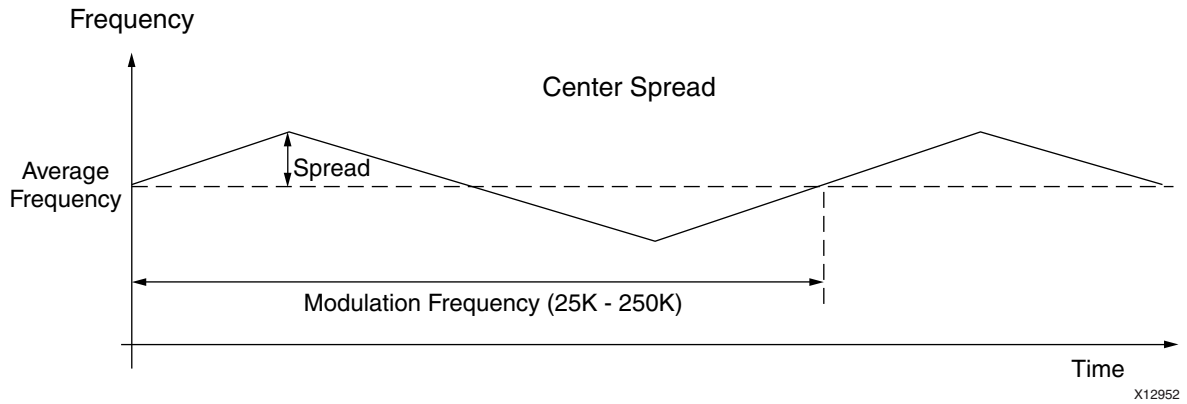

Figure 4-8: **Spread Spectrum Mode (Center Spread)**

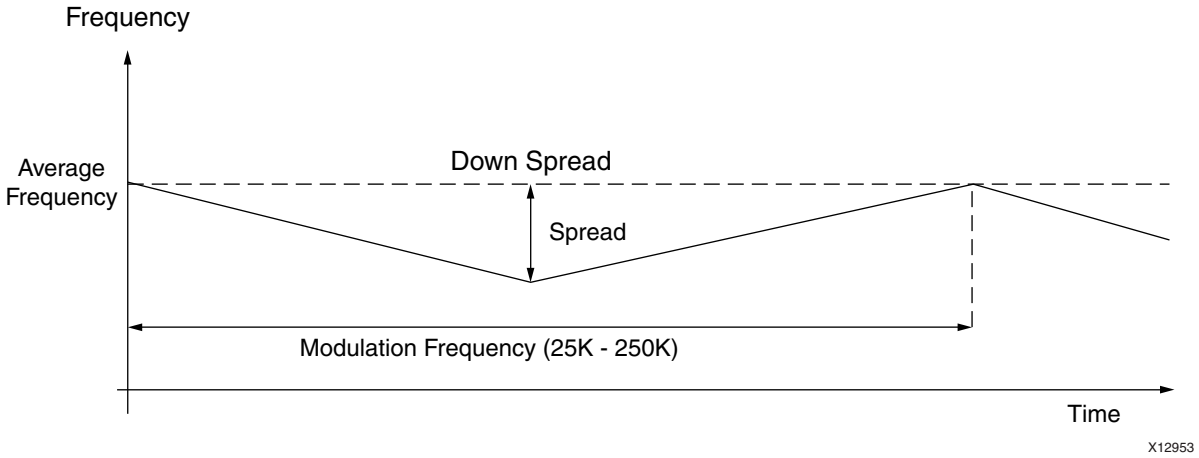

Figure 4-9: **Spread Spectrum Mode (Down Spread)**

Note: Input_clock_frequency is in Hz unit.

For spread:

- If (SS_Mode = CENTER_HIGH) :=>
  - *spread (ps) = +/- [1/(Input_clock_frequncy*(M-0.125*4)/D/O) - 1/ (Input_clock_frequency*M/D/O)]*
- If (SS_Mode = CENTER_LOW) :=>
  - *spread (ps) = +/- [1/(Input_clock_frequncy*(M-0.125*4)/D/O) - 1/ (Input_clock_frequency*M/D/O)]*
- If (SS_Mode = DOWN_HIGH) :=>
  - *spread (ps) = + [1/(Input_clock_frequncy*(M-0.125*4)/D/O) - 1/ (Input_clock_frequency*M/D/O)]*
- If (SS_Mode = DOWN_LOW) :=>

- spread (ps) = + [1/(Input_clock_frequncy*(M-0.125*4)/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)]

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

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.

Reset Type

You can select Reset Type as Active High or Active Low when **RESET** is enabled. Default value is Active High.

RECOMMENDED: *Xilinx recommends using the Active High reset in the design.*

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.

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

Output Clock Jitter and Phase Error

You can query the jitter and phase error on any of the output clocks of Clocking Wizard IP core. For example, if core name is **clk_wiz_0**, then by using following commands, jitter and phase error for `clk_out1` are available in Tcl console.

```
get_property CONFIG.CLKOUT1_JITTER [get_ips clk_wiz_0]
get_property CONFIG.CLKOUT1_PHASE_ERROR [get_ips clk_wiz_0]
```

Primitive Overrides

One or more pages of device and primitive specific parameter overrides are displayed.

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.

IMPORTANT: *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 PLLE2.

The generated source code contains the input and output clock summaries shown in the next summary page, as shown in Figure 4-10.

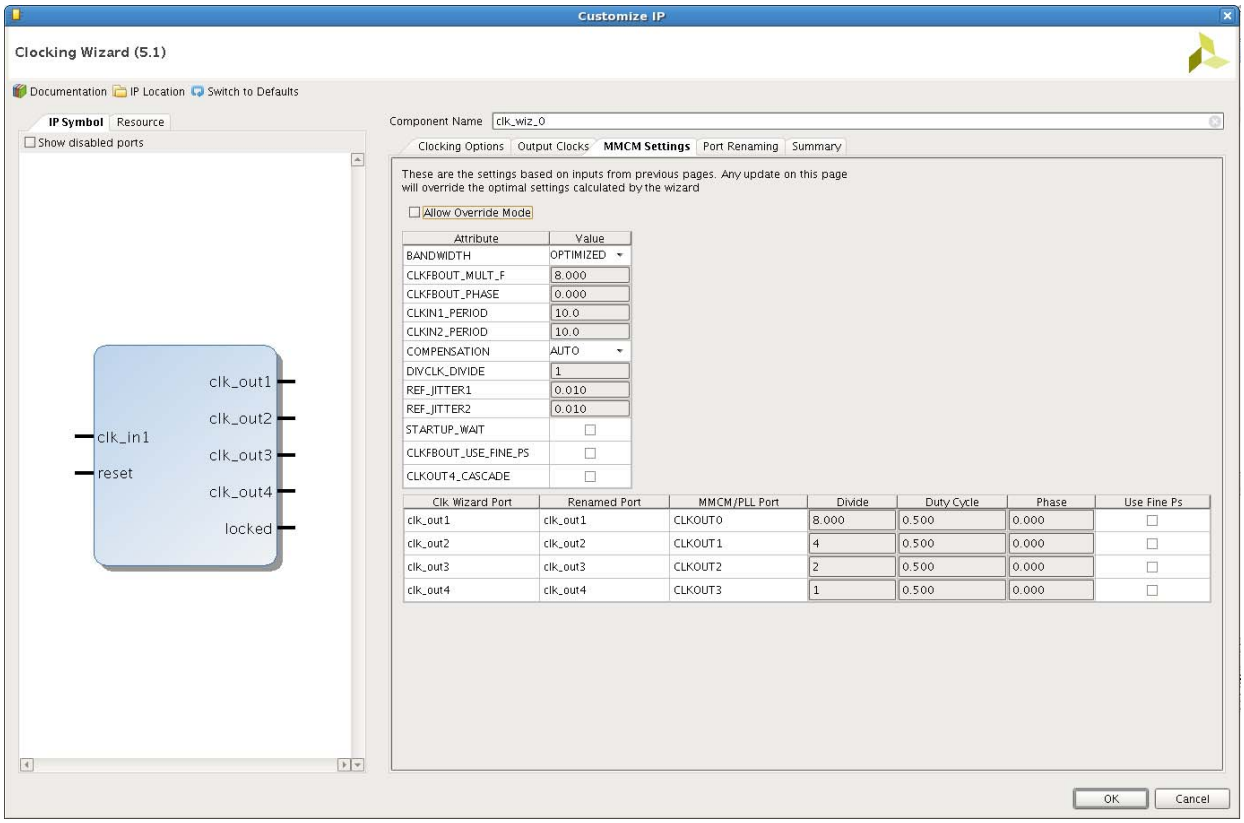

Figure 4-10: **Primitive Override Screen (Spread Spectrum Unselected)**

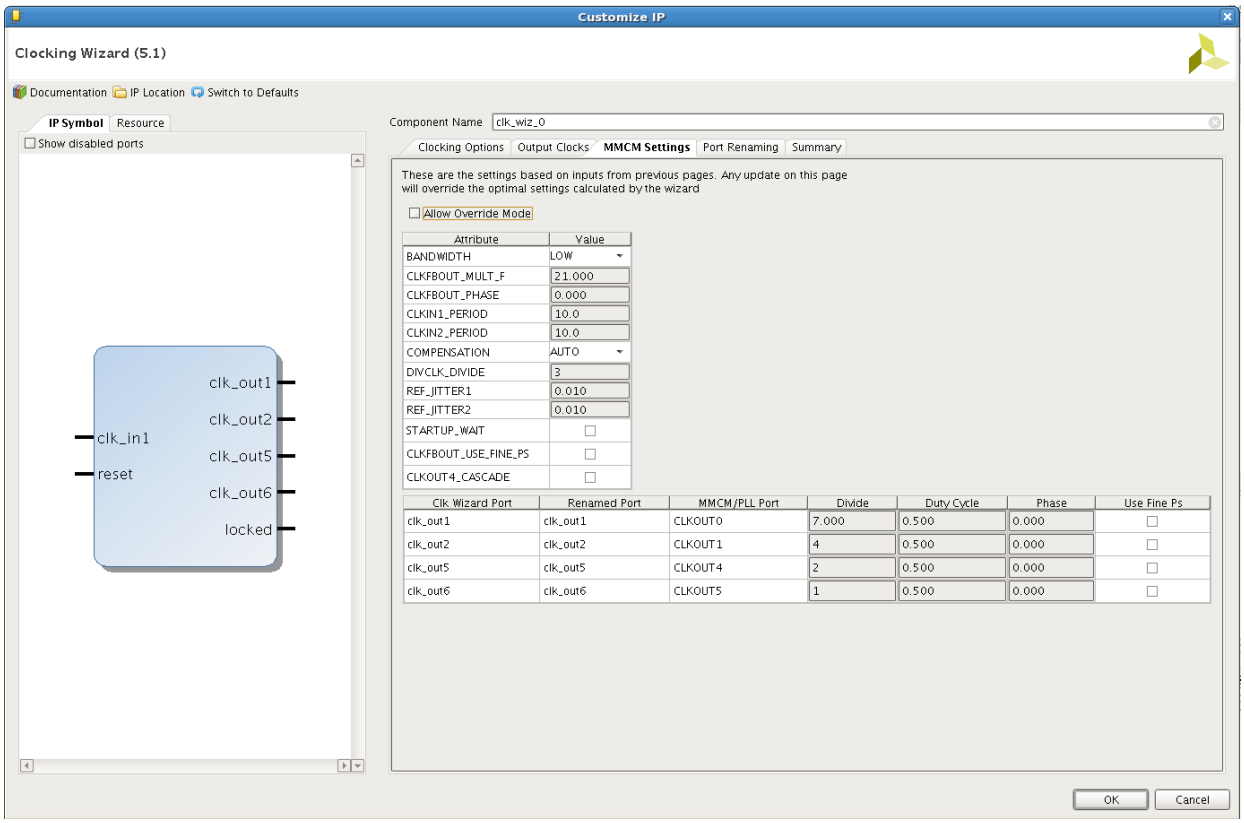

Figure 4-11: **Primitive Override Screen (Spread Spectrum Selected)**

Port Renaming

The first summary page (Figure 4-13) 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.

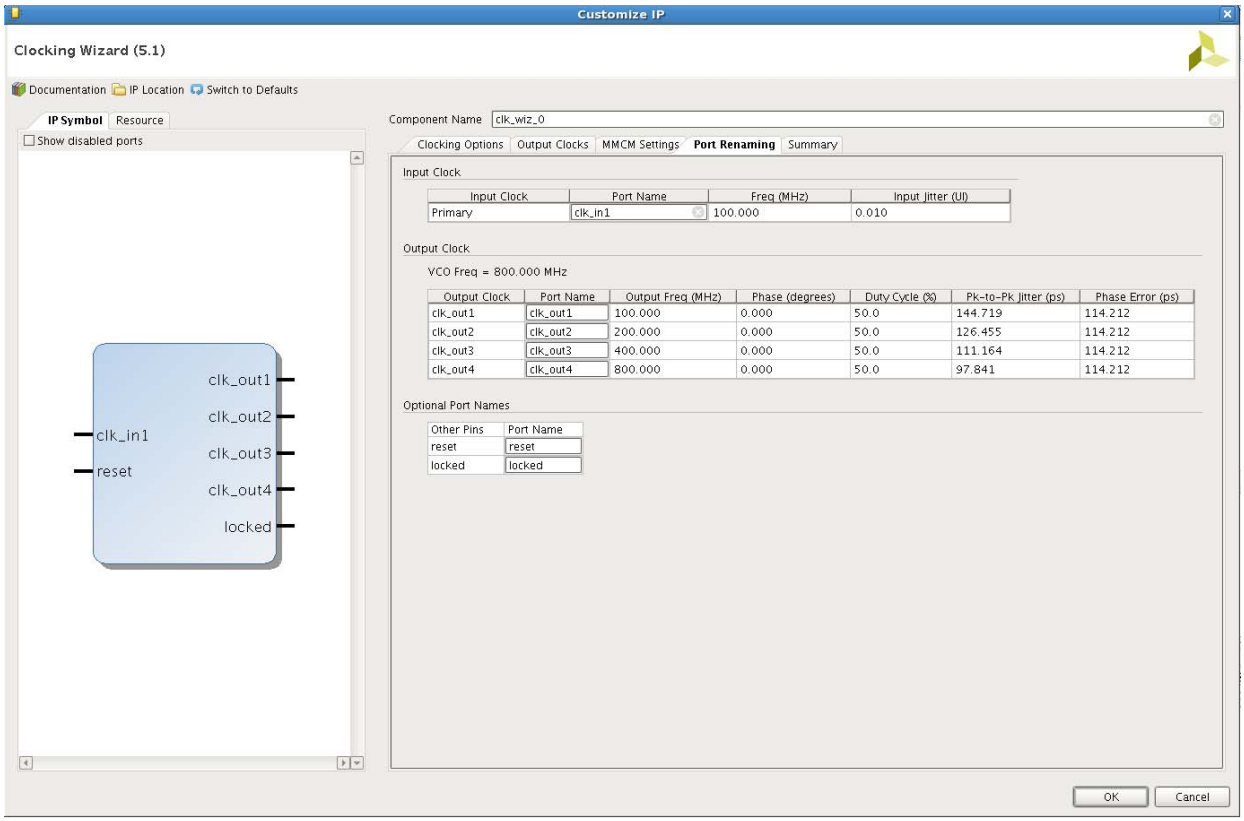

Figure 4-12: **Port Renaming (Spread Spectrum Unselected)**

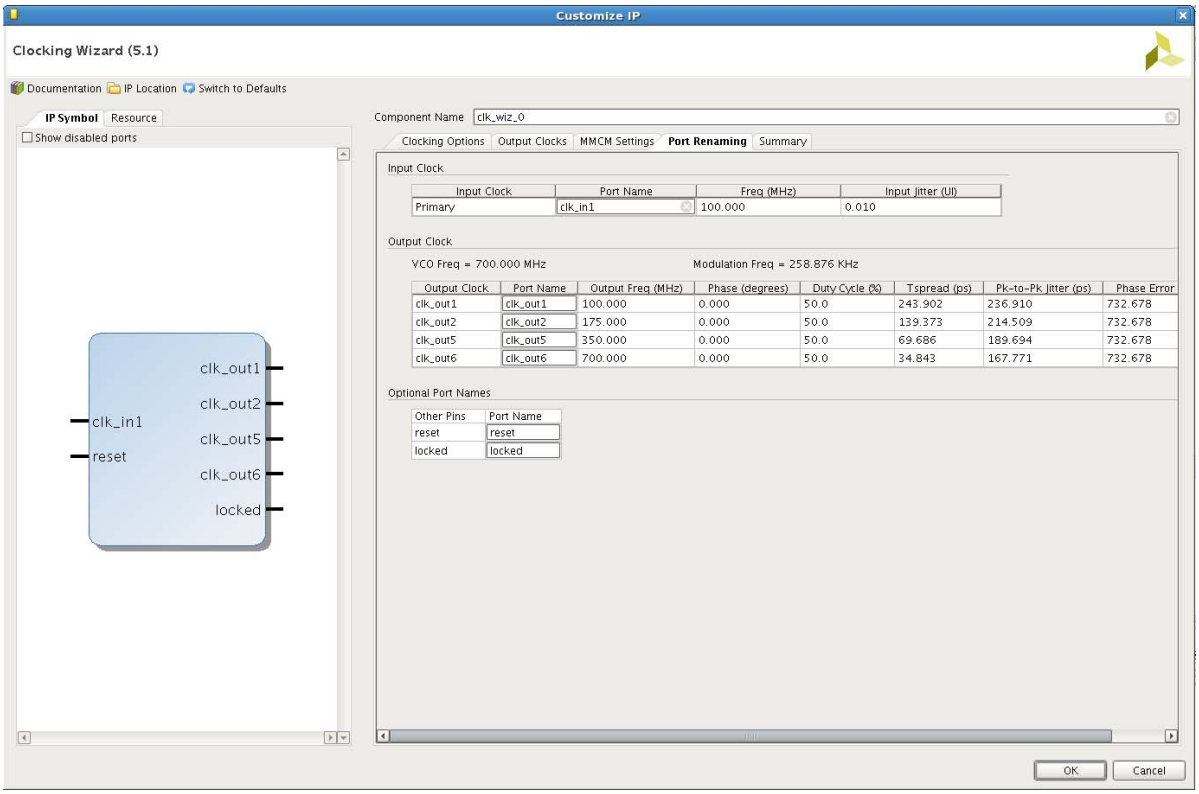

Figure 4-13: **Port Renaming (Spread Spectrum Selected)**

Input Clocking Summary

Information entered on the first page of the GUI 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`.

IMPORTANT: *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.*

Summary

The summary page (Figure 4-14) contains general summary information.

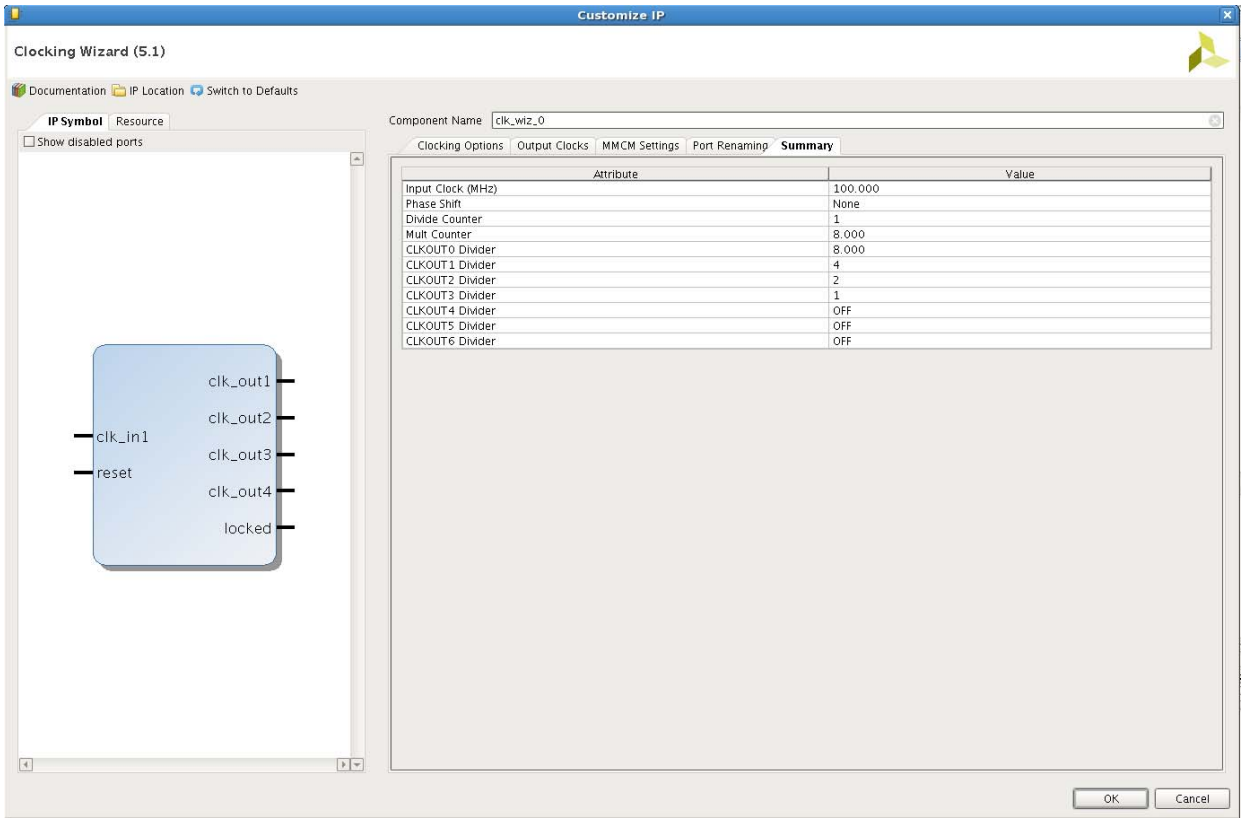

Figure 4-14: **Summary Screen**

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.

Dynamic Reconfiguration through AXI4-Lite

The Clocking Wizard core provides an AXI4-Lite interface for the dynamic reconfiguration of the clocking primitive MMCM/PLL. This interface is enabled when **Dynamic Reconfig** is enabled and Interface selection is **AXI4-Lite**. This feature is not supported when **Spread**

Spectrum is enabled. Mixed language RTL is delivered by the core when AXI4-Lite interface is used. For reconfiguring Phase and Duty Cycle, set Phase Duty Cycle config to **True**. This configuration utilizes DSP resources. By default, this option is set to **False** to optimize the design for area. Resource utilization for the AXI4-Lite interface configuration of Clocking Wizard IP core using Kintex-7 part xc7k325t is described in Table 4-1.

Table 4-1: **Kintex-7 FPGA Resource Utilization with AXI4-Lite Interface**

Site Type	Used when Phase Duty Cycle config = false	Used when Phase Duty Cycle config = true
Slice LUTs	1071	15323
Slice Registers	1426	1504
DSPs	0	38

Table 2-2 provides details of the signals of AXI4-Lite and Table 4-2 provides details of the clock configuration registers.

The Clocking Wizard core uses a configuration state machine listed in *MMCM and PLL Dynamic Reconfiguration* [Ref 6] and extends from two fixed state configuration to program any valid range of Multiply, Divide, Phase and Duty Cycle. In this state machine, State 1 corresponds to default state configured through Clocking Wizard interface. State 2 corresponds to user-configuration loaded into the clock configuration register detailed in Table 4-2. State 2 values are also initialized with the State 1 values so that a valid configuration is stored by default. You should update only those registers which are required to change the output clock behavior.

You should first write all the required clock configuration registers and then check for the status register. If status register value is 0x1, start the reconfiguration by writing Clock Configuration Register 23 with 0x7. The next write should be 0x2 before the Locked goes High. If the original configuration is needed at any time, then writing this register with value 0x4 and then 0x0 restores the original settings.

Before writing into C_BASEADDR + 0x200 register detailed in table 4-2, please make sure that these values result in a valid VCO frequency range of MMCM/PLL which is calculated using the equation:

VCO Frequency = (Input Clock Frequency) * (CLKFBOUT_MULT)/DIVCLK_DIVIDE

For details on the VCO range, refer to the DC and Switching Characteristics section of the applicable device data sheet.

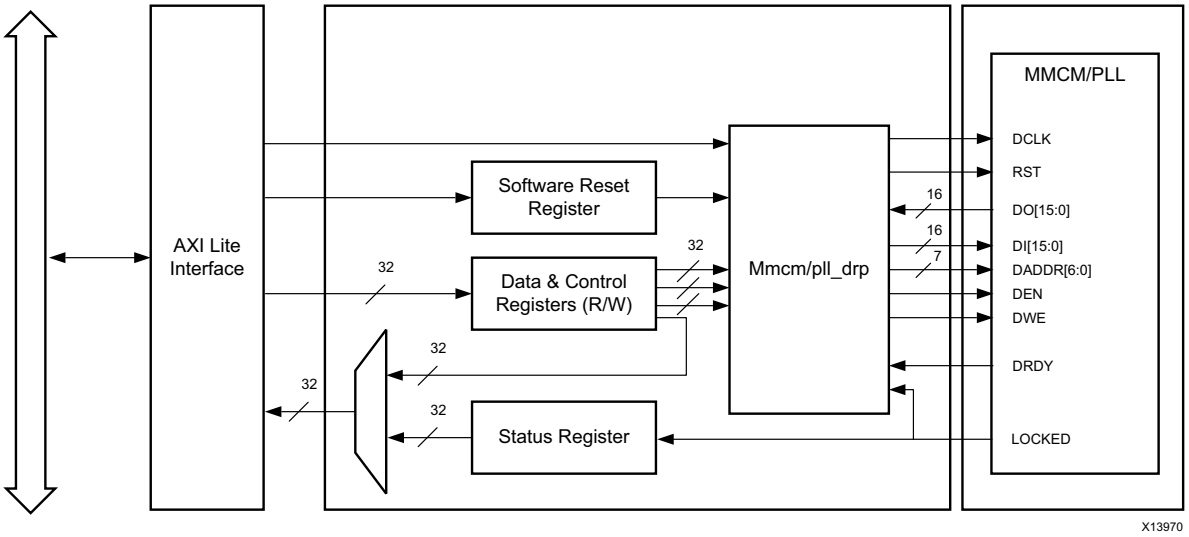

Figure 4-15: **Dynamic Reconfiguration using AXI4-Lite Interface**

Table 4-2: **Clock Configuration Registers**

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x00	Software Reset Register (SRR)	N/A	W(1)	Software Reset Register To activate software reset, the value 0x0000_000A must be written to the register. Any other access, read or write, has undefined results.
C_BASEADDR + 0x04	Status Register (SR)	0x00000000	R	Status Register Bit[0] = Locked When '1' MMCM/PLL is Locked and ready for the reconfiguration. Status of this bit is '0' during the reconfiguration.
C_BASEADDR + 0x200	Clock Configuration Register 0	Default(2): 0x01010A00	R/W	Bit[7:0] = DIVCLK_DIVIDE Eight bit divide value applied to all output clocks. Bit[15:8] = CLKFBOUT_MULT Integer part of multiplier value i.e. For 8.125, this value is 8 = 0x8. Bit[25:16] = CLKFBOUT_FRAC Multiply(3) Fractional part of multiplier value i.e. For 8.125, this value is 125 = 0x7D. Bit[26] = CLKFBOUT_FRAC_EN(3) This bit should be set to 1 for Fractional multiplication. Setting this bit to 0 uses the default configuration. The value of CLKFBOUT fractional divide can be from 0 to 875 representing the factional multiplied by 1000.

Table 4-2: **Clock Configuration Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x204	Clock Configuration Register 1	Default[2]: 0x00000000	R/W	Bit[31:0] = CLKFBOUT_PHASE Phase values entered are Signed Number for +/- phase.
C_BASEADDR + 0x208	Clock Configuration Register 2	Default[2]: 0x0004000a	R/W	Bit[7:0] = CLKOUT0_DIVIDE Integer part of clkout0 divide value For example, for 2.250, this value is 2 = 0x2 Bit[17:8] = CLKOUT0_FRAC Divide[3] Fractional part of clkout0 divide value For example, for 2.250, this value is 250 = 0xFA Bit[18] = CLKOUT0_FRAC_EN[3] This bit should be set to 1 for Fractional division
C_BASEADDR + 0x20C	Clock Configuration Register 3	Default[2]: 0x00000000	R/W	Bit[31:0] = CLKOUT0_PHASE[5]
C_BASEADDR + 0x210	Clock Configuration Register 4	Default[2]: 0x0000C350	R/W	Bit[31:0] = CLKOUT0_DUTY Duty cycle value = (Duty Cycle in %) * 1000 For example, for 50% duty cycle, value is 50000 = 0xC350
C_BASEADDR + 0x214	Clock Configuration Register 5	Default[2]: 0x0000000A	R/W	Bit[7:0] = CLKOUT1_DIVIDE[4] Eight bit clkout1 divide value
C_BASEADDR + 0x218	Clock Configuration Register 6	Default[2]: 0x00000000	R/W	Bit[31:0] = CLKOUT1_PHASE[5] Phase values entered are Signed Number for +/- phase
C_BASEADDR + 0x21C	Clock Configuration Register 7	Default[2]: 0x0000C350	R/W	Bit[31:0] = CLKOUT1_DUTY[6]
C_BASEADDR + 0x220	Clock Configuration Register 8	Default[2]: 0x0000000A	R/W	Bit[7:0] = CLKOUT2_DIVIDE[4]
C_BASEADDR + 0x224	Clock Configuration Register 9	Default[2]: 0x00000000	R/W	Bit[31:0] = CLKOUT2_PHASE[5]
C_BASEADDR + 0x228	Clock Configuration Register 10	Default[2]: 0x0000C350	R/W	Bit[31:0] = CLKOUT2_DUTY[6]
C_BASEADDR + 0x22C	Clock Configuration Register 11	Default[2]: 0x0000000A	R/W	Bit[7:0] = CLKOUT3_DIVIDE[4]

Table 4-2: **Clock Configuration Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x230	Clock Configuration Register 12	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT3_PHASE(5)
C_BASEADDR + 0x234	Clock Configuration Register 13	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT3_DUTY(6)
C_BASEADDR + 0x238	Clock Configuration Register 14	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT4_DIVIDE(4)
C_BASEADDR + 0x23C	Clock Configuration Register 15	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT4_PHASE(5)
C_BASEADDR + 0x240	Clock Configuration Register 16	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT4_DUTY(6)
C_BASEADDR + 0x244	Clock Configuration Register 17	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT5_DIVIDE(4)
C_BASEADDR + 0x248	Clock Configuration Register 18	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT5_PHASE(5)
C_BASEADDR + 0x24C	Clock Configuration Register 19	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT5_DUTY(6)
C_BASEADDR + 0x250(3)	Clock Configuration Register 20	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT6_DIVIDE(4)
C_BASEADDR + 0x254(3)	Clock Configuration Register 21	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT6_PHASE(5)
C_BASEADDR + 0x258(3)	Clock Configuration Register 22	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT6_DUTY(6)

Table 4-2: **Clock Configuration Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x25C	Clock Configuration Register 23	0x00000000	R/W	Bit[0] = LOAD Loads Clock Configuration Register values to the internal register used for dynamic reconfiguration. This bit should be asserted when the required settings are already written into Clock Configuration Registers and then de-asserted with next write. Bit[1] = SADDR When written 0, default configuration done in the Clocking Wizard GUI is loaded for dynamic reconfiguration. When written 1, setting provided in the Clock Configuration Registers are used for dynamic reconfiguration. Bit[2] = SEN Writing 1 to this bit starts the dynamic reconfiguration state machine. This bit should be asserted and then deasserted with next write. Configuring this bit to 0 has no impact on the configuration register values.
C_BASEADDR + 0x260 to C_BASEADDR + 0x7FC	Undefined	Undefined	N/A (1)	Don't read/write these registers.

1. Reading of this register returns an undefined value.
2. Initialized with configuration settings done by the clocking algorithm.
3. Valid only for MMCM(E2/E3) primitive
4. Eight bit divide value
5. Phase value = (Phase Requested) * 1000 i.e. for 45.5 degree phase, required value is 45500 = 0xB1BC.
6. Note: Phase values entered are Signed Number in the range +360000 to -360000
7. duty cycle value = (Duty Cycle in %) * 1000 i.e. for 50% duty cycle, value is 50000 = 0xC350

Examples Dynamic Reconfiguration through AXI4-Lite

The input and output clock frequencies are 100 MHz in the Clocking Wizard by default.

To achieve 200 MHz frequency on clkout1 in the Clocking Wizard core with primary input clock frequency at 100 MHz (VCO frequency at 1000 MHz), perform the following steps:

1. Configure the Clock Configuration register 0 (Address: C_BASEADDR + 0x200) with 0x00000A01.

   Writing this value sets `DIVCLK_DIVIDE` value to 1 and `CLKFBOUT_MULT` to 10.

2. Configure the Clock Configuration Register 2 (Address: C_BASEADDR + 0x208) with 0x00000005.

   Writing this value sets `CLKOUT0_DIVIDE` to 5. The VCO frequency being 1000 MHz, dividing it by `CLKOUT0_DIVIDE` will give the 200 MHz frequency on the `clkout1` in the IP. Check for the status register, if the status register value is 0x1, then go to step 3.

3. Configure the Clock Configuration Register 23 (Address: C_BASEADDR + 0x25C) with 0x00000007 to set the LOAD and SEN bits.
4. Configure the Clock Configuration Register 23 (Address: C_BASEADDR + 0x25C) with 0x2 before the Locked goes High. This resets the LOAD and SEN bits.
5. Wait for the locked signal. The new frequency can be checked at `clkout1` output port.

   Note: You can reset to the default settings by configuring the Clock Configuration Register 23 (Address: C_BASEADDR + 0x25C) with the value 0x00000004, followed by writing 0x00000000 (SADDR bit is set to 0).

Refer to the Chapter 5, Detailed Example Design for more details.

Output Generation

For details, see "Generating IP Output Products" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Constraining the Core

Required Constraints

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

```
create_clock -period 10.0 [get_ports clk_in1]
set_input_jitter [get_clocks -of_objects [get_ports clk_in1]] 0.1
```

The core level XDC has early processing order so core level XDC constraints are applied first and then are overridden by the user-provided constraints.

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.

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5].

You can simulate the example design using 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:

```
example_project/<component_name>_example/<component_name>_example.sim/sim_1/
```

You can run fast simulation using `unifast_ver` or `unifast` libraries of `MMCME2_ADV` and `PLLE2_ADV`. This improves simulation runtime by 100X.

Simulation Waveforms for the Safe Clock Startup Feature

Simulation when Safe Clock Startup is true is illustrated in Figure 4-16.

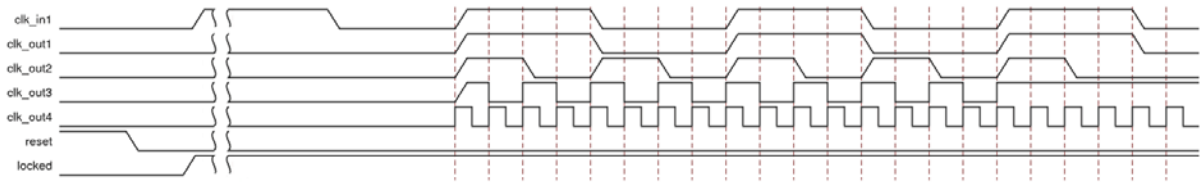

Figure 4-16: **Simulation When Safe Clock Startup is True**

Figure 4-17 illustrates simulation when Safe Clock Startup is true and Use Clock Sequencing is true with required sequence number in the table as indicted in Figure 4-4.

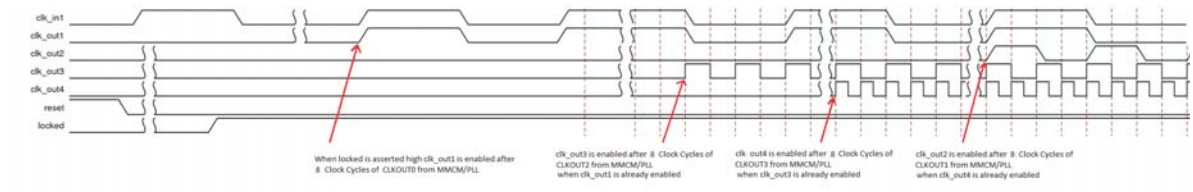

Figure 4-17: **Simulation when Safe Clock Startup is true and Use Clock Sequencing is true**

Synthesis and Implementation

For details about synthesis and implementation, see "Synthesizing IP" and "Implementing IP" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

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.

Test Bench

This chapter contains information about the provided test bench in the Vivado® Design Suite environment.

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.

Appendix A

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.

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.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, *see the ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 1].

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`.

IMPORTANT: *This change in numbering is especially important to consider if parameter overriding is desired.*

GUI Information Gathering Order

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

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

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

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Parameter Changes

Added the `INTERFACE_SELECTION` parameter in the IDE for selecting the **AXI4-Lite**, **DRP** or **None** for DRP register access.

```
CLKOUT<1-7>_JITTER parameter added to query the Peak to Peak Jitter on the output
clocks
CLKOUT<1-7>_PHASE_ERROR parameter added to query the phase error on the output clock.
```

Port Changes

Added optional AXI4-Lite ports (`s_axi_*`). See Table 2-4.

Other Changes

Improved safe clock logic to remove glitches on clock outputs for odd multiples of input clock frequencies.

Xilinx does not recommend to upgrading the Clocking Wizard IP that has been targeted on a board to a device part. For assistance, contact Xilinx Support.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the Clocking Wizard core, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

Documentation

This product guide is the main document associated with the Clocking Wizard core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

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.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core are listed below, and can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Answer Records for the Clocking Wizard core

AR 54102
http://www.xilinx.com/support/answers/54102.htm

Contacting 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.

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Note: Access to WebCase is not available in all cases. Please login to the WebCase tool to see your specific support options.

Debug Tools

There are many tools available to address Clocking Wizard core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Lab Edition

Vivado® Lab Edition inserts logic analyzer and virtual I/O cores directly into your design. Vivado Lab Edition also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 3].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The ChipScope debugging tool is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the ChipScope debugging tool for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `LOCKED` port.
- If your outputs go to 0, check your licensing.

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this user guide:

1. *ISE to Vivado Design Suite Migration Guide* (UG911)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Programming and Debugging* (UG908).
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *MMCM and PLL Dynamic Reconfiguration* (XAPP888)
7. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
8. *UltraScale Architecture Clocking Resources User Guide* (UG572)
9. *7 Series FPGAs Clocking Resources User Guide* (UG472)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/01/2015	5.1	• Added the Minimum Input frequencies for MMCM and PLL for Virtex-7 devices. • Added an example describing the reference steps when using the AXI Interface for using the dynamic reconfiguration interface.
10/01/2014	5.1	Added UltraScale architecture support and User Parameters mapping table.
10/01/2014	5.1	Added UltraScale architecture support and User Parameters mapping table.
04/02/2014	5.1	Updated Configuring Output Clock section. Added Resource utilization for AXI4-Lite interface using Kintex-7 device.
12/18/2013	5.1	Added UltraScale Architecture support.
10/02/2013	5.1	Updated for to synch doc version with core version. Added Migration information.
03/20/2013	1.3	Updated for core version, added XCI parameters and Safe Clock Startup diagrams and waveforms.
12/18/2012	1.2	Updated for core version, Active Low RESET support, and Vivado GUI screens.
10/16/2012	1.1	Updated for core version and Vivado GUI screens.
07/25/2012	1.0	Initial release of Product Guide, replacing DS709 and UG521.

Please Read: Important Legal Notices