Clocking Wizard v6.0

LogiCORE IP Product Guide

Vivado Design Suite

PG065 April 20, 2022

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

AMD
XILINX

AMD
XILINX

Table of Contents

Introduction

The Clocking Wizard LogiCORE™ IP 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

- The selection of mixed-mode clock manager (MMCM) and phase-locked loop (PLL) primitives. Integrated design environment (IDE) options are enabled for the supported features for the primitives.
- The Safe Clock Startup feature enables a stable and valid clock at the output. Enabling the Sequencing feature provides sequenced output clocks.
- Accepts up to two input clocks and up to seven output clocks per clock network.
- Provides an AXI4-Lite interface for dynamically reconfiguring the clocking primitives for Multiply, Divide, Phase Shift/ Offset, or Duty Cycle.
- Automatically configures a clocking primitive based on the selected clocking features.
- Automatically calculates the 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+™ families, UltraScale™ families, Zynq® -7000 SoC, 7 Series
Supported User Interfaces	AXI4-Lite
Resources	Performance and Resource Utilization web page.
Special Features	PLL(E2/E3/E4), MMCM(E2/E3/E4), Spread Spectrum Clocking
Provided with Core	
Design Files	Verilog(2)
Example Design	Verilog
Test Bench	Verilog(2)
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 Advanced Simulator, Vivado Simulator
Synthesis Tools	Synplify PRO E-2012.03, Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 54102
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

1. For a complete listing of supported devices, see the Vivado IP Catalog.
2. The top RTL design file is delivered in Verilog and the sub-modules can still be in VHDL or Verilog.
3. A standalone C example can be found in the Vitis directory (<install_directory>/data/embeddedsw/XilinxProcessorIPLib/drivers/clk_wiz_vx_x). Linux OS and driver support information is available from the Xilinx Wiki page. Common clock Linux driver information for Zynq UltraScale+ MPSoCs is available from the Common Clock Framework Wiki page.
4. For the supported versions of third-party tools, see the Xilinx Design Tools: Release Notes Guide.

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 you 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 aids you in making design trade-off decisions.

Feature Summary

The clocking options are listed below:

- **Frequency Synthesis** allows output clocks to have different frequencies from the active input clock.
- **Spread Spectrum** provides modulated output clocks, which reduces the spectral density of the electromagnetic interference (EMI) generated by electronic devices. This feature is available for the MMCM(E2/E3/E4)_ADV primitive only. UNISIM simulation support for this feature is not currently available.
- **Phase Alignment** allows the output clock to be phase locked to a reference, such as the input clock pin for a device.
- **Minimize Power** allows you to minimize the amount of power needed for the primitive. This is at the possible expense of frequency, phase offset, or duty cycle accuracy.
- **Dynamic Phase Shift** allows you to change the phase relationship on the output clocks.
- **Dynamic Reconfiguration** allows you to change the programming of the primitive after device configuration. When this option is chosen, the AXI4-Lite interface is selected by default for reconfiguring the 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 the Maximize input jitter filtering feature.
- **Maximize Input Jitter filtering** 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 the Minimize output jitter feature.
- **Safe Clock Startup and Sequencing** can be used to get a stable and valid clock at the output. It also enables clocks in a particular sequence order as specified in the configuration.
- **Clock Monitor** helps you to monitor the clock inputs to the Clocking Wizard. It can monitor up to four clocks. You can monitor if the input frequency is out of range of the expected frequency, and detect clock stop and glitches in the clock.
- **Auto Primitive** instantiates the appropriate clocking primitive for your requirements. You do not need know the specification of the MMCM or PLL to judge which primitive fits into your requirements; the Wizard does this for you. This feature is available for UltraScale™ and UltraScale+™ devices only.

Applications

- The creation of clock networks with the required frequency, phase, and duty cycle, with reduced jitter.
- Electromagnetic interference reduction in electronic devices using the Spread Spectrum feature.

Licensing and Ordering

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 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 a mixed-mode clock manager (MMCM)(E2/E3/E4) or phase-locked loop (PLL)(E2/E3/E4) 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 the Spread Spectrum feature, which is helpful in reducing electromagnetic interference. Figure 2-1 shows a block diagram of the Clocking Wizard.

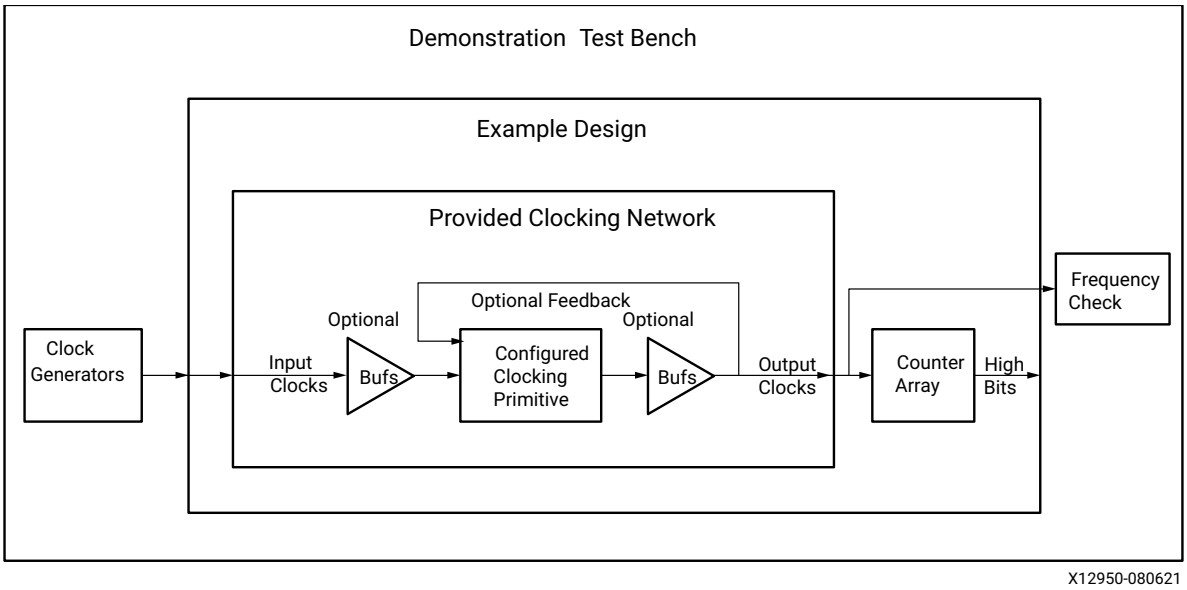

Figure 2-1: **Clocking Wizard Block Diagram**

Performance

Maximum Frequencies

For the maximum frequencies of the MMCM and PLL, refer to the following device data sheets:

- *Virtex-7 T and XT FPGAs Data Sheet* (DS183) [Ref 10]
- *Kintex-7 FPGAs Data Sheet* (DS182) [Ref 11]

- *Artix-7 FPGAs Data Sheet* (DS181) [Ref 16]
- *Artix UltraScale+ FPGA Data Sheet* (DS931) [Ref 17]
- *Kintex UltraScale™ FPGAs Data Sheet* (DS892) [Ref 12]
- *Virtex UltraScale FPGAs Data Sheet* (DS893) [Ref 13]
- *Zynq UltraScale+ MPSoC Data Sheet* (DS925) [Ref 14]
- *Kintex UltraScale+ FPGAs Data Sheet* (DS922) [Ref 15]

Power

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

Resource Utilization

Resource utilization is available in the Clocking Wizard IDE by clicking on the **Resource** tab. This does *not* include AXI4-Lite resources when Dynamic Reconfiguration is enabled.

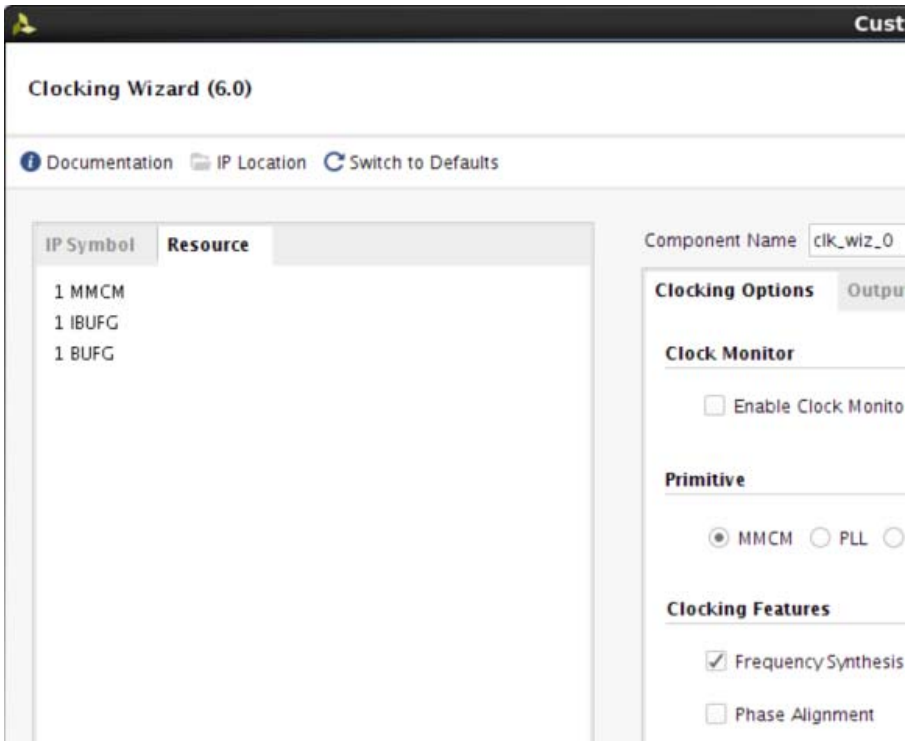

Figure 2-2: **Resource Tab**

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 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 2-1: **Clocking Wizard I/O**

Port [5]	I/O	Description
Input Clock Ports[1]		
clk_in1	I	Clock in 1: Single-ended primary input clock port. Available when single-ended primary clock source is selected.
clk_in1_p clk_in1_n	I	Clock in 1 Positive and Negative: Differential primary input clock port pair. Available when a differential primary clock source is selected.
clk_in2[2]	I	Clock in 2: Single-ended secondary input clock port. Available when a single-ended secondary clock source is selected.
clk_in2_p[2] clk_in2_n[2]	I	Clock in 2 Positive and Negative: Differential secondary input clock port pair. Available when a differential secondary clock source is selected.
clk_in_sel[2]	I	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	I	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 clkfb_in_n	I	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.
Output Clock Ports		
clk_out1	O	Clock Out 1: Output clock of the clocking network. clk_out1 is not optional.
clk_out1_ce	I	Clock Enable: Clock enable pin of the output buffer. Available when BUFGCE or BUFHCE or BUFR or BUFGCE_DIV buffers are used as output clock drivers.
clk_out1_clr	I	Counter reset for divided clock output: Available when BUFR or BUFGCE_DIV buffer is used as output clock driver.

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

Port [5]	I/O	Description
clk_out[n][3]	O	Clock Out[n]: Optional output clocks of the clocking network that are user specified. *n* can range in value from 2 to 7. For an MMCM, up to seven are available. For UltraScale™ PLLE3, up to two clocks are available and for 7 series/Zynq®-7000 PLLE2, up to six clocks are available.
clk_out[n]_ce[3]	I	Clock Enable: Clock enable pin of the output buffer. Available when BUFGCE or BUFHCE or BUFR or BUFGCE_DIV buffers are used as output clock drivers. *n* can range in value from 2 to 7.
clk_out[n]_clr[3]	I	Counter reset for divided clock output: Available when the BUFR buffer is used as output clock driver. *n* can range in value from 2 to 7.
clkfb_out	O	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	O	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	O	
Dynamic Reconfiguration Ports		
daddr[6:0]	I	Dynamic Reconfiguration Address: Address port for use in dynamic reconfiguration; active when den is asserted.
dclk	I	Dynamic Reconfiguration Clock: Clock port for use in dynamic reconfiguration.
den	I	Dynamic Reconfiguration Enable: Starts a dynamic reconfiguration transaction. Refer to DRP protocol details for more information.
di[15:0]	I	Dynamic Reconfiguration Data in: Input data for a dynamic reconfiguration write transaction; active when den is asserted.
do[15:0]	O	Dynamic Reconfiguration Data Out: Output data for a dynamic reconfiguration read transaction; active when drdy is asserted.
drdy	O	Dynamic Reconfiguration Ready: Completes a dynamic reconfiguration transaction.
dwe	I	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	I	Dynamic Phase Shift Clock: Clock for use in dynamic phase shifting.
psen	I	Dynamic Phase Shift Enable: Starts a dynamic phase shift transaction.

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

Port [5]	I/O	Description
psincdec	I	Dynamic Phase Shift increment/decrement: When 1, increments the phase shift of the output clock, when 0, decrements the phase shift.
psdone	O	Dynamic Phase Shift Done: Completes a dynamic phase shift transaction.
Status and Control Ports[4]		
reset/resetn	I	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	I	Power Down: When asserted, places the clocking primitive into a low power state, which stops the output clocks.
input_clk_ stopped	O	Input Clock Stopped: When asserted, indicates that the selected input clock is no longer toggling.
locked	O	Locked: When asserted, indicates that the output clocks are stable and usable by downstream circuitry.
cddcreq[6]	I	Clock Divide Dynamic Change (CDDC) request. This is asserted after last DRP request is performed, and then deasserted after the last DRDY.
cddcdone[6]	O	Clock Divide Dynamic Change (CDDC) done. When output counters are updated, this signal is asserted.
s_axi_aclk	I	AXI Clock.
s_axi_aresetn	I	AXI Reset, active-Low.
s_axi_awaddr[10:0]	I	AXI Write address. The write address bus gives the address of the write transaction.
s_axi_awvalid	I	Write address valid. This signal indicates that a valid write address and control information are available.
s_axi_awready	O	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
s_axi_wdata[31:0]	I	Write data.
s_axi_wstb[3:0]	I	Write strobes. This signal indicates which byte lanes to update in memory.
s_axi_wvalid	I	Write valid. This signal indicates that valid write data and strobes are available.
s_axi_wready	O	Write ready. This signal indicates that the slave can accept the write data.

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

Port [5]	I/O	Description
s_axi_bresp[1:0]	O	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)
s_axi_bvalid	O	Write response valid. This signal indicates that a valid write response is available.
s_axi_bready	I	Response ready. This signal indicates that the master can accept the response information.
s_axi_araddr[10:0]	I	Read address. The read address bus gives the address of a read transaction.
s_axi_arvalid	I	Read address valid. This signal indicates, when High, that the read address and control information is valid and remains stable until the address acknowledgment signal, s_axi_arready, is High.
s_axi_arready	O	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
s_axi_rdata[31:0]	O	Read data.
s_axi_rresp[1:0]	O	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	O	Read valid. This signal indicates that the required read data is available and the read transfer can complete.
s_axi_rready	I	Read ready. This signal indicates that the master can accept the read data and response information.
s_axis_aclk	I	The global clock signal. All streaming signals from Read interface of the FIFO are sampled on the rising edge of s_axis_aclk.
Clock Monitor Ports [7]		
ref_clk	I	This is the input reference clock used to monitor the user clocks. It is considered to be stable and error free.
user_clk0	I	User clock 0. This port is disabled when the ENABLE_PLL/ MMCM0 checkbox is enabled in the Vivado IDE.
user_clk1	I	User clock 1. This port is disabled when the ENABLE_PLL1/ MMCM1 checkbox is enabled in the Vivado IDE.
user_clk2	I	User input clock 2 to monitor.
user_clk3	I	User input clock 3 to monitor.

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

Port [5]	I/O	Description
clk_stop[3:0]	O	The bits for this port are High when clock is stopped on the respective user clock. Bit 0 - User clock 0 Bit 1 - User clock 1 Bit 2 - User clock 2 Bit 3 - User clock 3
clk_oor[3:0]	O	The bits for this port are High when input clock frequency is out of range than expected. Bit 0 - User clock 0 Bit 1 - User clock 1 Bit 2 - User clock 2 Bit 3 - User clock 3
clk_glitch[3:0]	O	The bits for this port are High where there is a glitch in the input clock. Bit 0 - User clock 0 Bit 1 - User clock 1 Bit 2 - User clock 2 Bit 3 - User clock 3
interrupt	O	This port gives the interrupts of the clock monitor feature.

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 2-1 is the default port list.
6. Ports used for dynamic change of output counter without reset. Available only in MMCME3 primitive.
7. These ports are available when the Clock Monitor feature is enabled.
8. The user must write to the interrupt enable register with all 1's (i.e., interrupt enable for clock stop, clock overrun, and clock underrun bits in the interrupt status register) for the Interrupt to be enabled.

Register Space

Table 2-2 shows the set of registers applicable when the Dynamic Reconfiguration mode is selected. All registers are accessed as 32-bit.

Table 2-2: **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 reconfiguration. The status of this bit is 0 during reconfiguration.
C_BASEADDR +0x08	Clock Monitor Error Status Register	0x00000000	R	This register gives the error status bits of the clock monitor feature.
C_BASEADDR +0x0C	Interrupt Status	0x00000000	R/W	Interrupt Status for Clock Stop, Clock Overrun, and Clock Underrun. These bits are gated by Interrupt enable bits. Interrupts corresponding to the enabled bits in Interrupt enable register would be updated in this register.
C_BASEADDR +0x10	Interrupt Enable	0x00000000	R/W	Interrupt Enable for Clock Stop, Clock Overrun, and Clock Underrun bits in the Interrupt status register.

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

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
Dynamic Reconfiguration Registers				
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. ***Note:*** You need not set any bit for specifying that the multiplier value is fractional. Just mention the fractional value in the register space. The value of CLKFBOUT fractional divide can be from 0 to 875 representing the fractional multiplied by 1000.
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 ***Note:*** You need not set any bit for specifying that the multiplier value is fractional. Just mention the fractional value in the register space.
C_BASEADDR + 0x20C	Clock Configuration Register 3	Default[(2)]: 0x00000000	R/W	Bit[31:0] = CLKOUT0_PHASE[(5)]

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

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
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)]
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)]

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

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
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]
C_BASEADDR + 0x25C	Clock Configuration Register 23	0x00000000	R/W	Bit[0] = LOAD / SEN Loads Clock Configuration Register values to the internal register used for dynamic reconfiguration and initiates reconfiguration state machine. This bit should be asserted when the required settings are already written into Clock Configuration Registers. This bit retains to 0, when the dynamic reconfiguration is done and the clock is locked. 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.
C_BASEADDR + 0x260 to C_BASEADDR + 0x7FC	Undefined	Undefined	N/A [1]	Do not read/write these registers.
C_BASEADDR + 0x260 to C_BASEADDR + 0x2FC	Undefined	Undefined	N/A	

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

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
Dynamic Reconfiguration Registers (when the Write DRP feature is enabled)				
C_BASEADDR +0x300	Power Register	FFFF	R/W	For more information, see *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6].
C_BASEADDR +0x304	CLKOUT0 Register 1	1145	R/W	
C_BASEADDR +0x308	CLKOUT0 Register 2	0000	R/W	
C_BASEADDR +0x30C	CLKOUT1 Register 1	1145	R/W	
C_BASEADDR +0x310	CLKOUT1 Register 2	00C0	R/W	
C_BASEADDR +0x314	CLKOUT2 Register 1 (Not available for PLLE3)	1145	R/W	

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

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR +0x318	CLKOUT2 Register 2 (Not available for PLLE3)	00C0	R/W	For more information, see *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6].
C_BASEADDR +0x31C	CLKOUT3 Register 1 (Not available for PLLE3)	1145	R/W	
C_BASEADDR +0x320	CLKOUT3 Register 2 (Not available for PLLE3)	00C0	R/W	
C_BASEADDR +0x324	CLKOUT4 Register 1 (Not available for PLLE3)	1145	R/W	
C_BASEADDR +0x328	CLKOUT4 Register 2 (Not available for PLLE3)	00C0	R/W	
C_BASEADDR +0x32C	CLKOUT5 Register 1	1145	R/W	
C_BASEADDR +0x330	CLKOUT5 Register 2	00C0	R/W	
C_BASEADDR +0x334	CLKOUT6 Register 1 (Not available for PLLE2 or PLLE3)	1145	R/W	
C_BASEADDR +0x338	CLKOUT6 Register 2 (Not available for PLLE2 or PLLE3)	00C0	R/W	
C_BASEADDR +0x33C	DIVCLK Register	1041	R/W	
C_BASEADDR +0x340	CLKFBOUT Register 1	1145	R/W	
C_BASEADDR +0x344	CLKFBOUT Register 2	0000	R/W	
C_BASEADDR +0x348	Lock Register 1	03e8	R/W	
C_BASEADDR +0x34C	Lock Register 2	7001	R/W	
C_BASEADDR +0x350	Lock Register 3	73E9	R/W	
C_BASEADDR +0x354	Filter Register 1	800	R/W	For more information, see *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6].
C_BASEADDR +0x358	Filter Register 2	9190	R/W	

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

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR +0x35C	Clock Configuration Register 24s	0000	R/W	Bit[0] = LOAD / SEN Loads Clock Configuration Register values to the internal register used for dynamic reconfiguration and initiates reconfiguration state machine. This bit should be asserted when the required settings are already written into Clock Configuration Registers. This bit retains to 0, when the dynamic reconfiguration is done and the clock is locked. 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.
C_BASEADDR + 0x360 to C_BASEADDR + 0x7FC	Undefined	Undefined	N/A	

Notes:

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/E4) primitive.
4. Eight-bit divide value.
5. Phase value = (Phase Requested) * 1000. For example, for a 45.5 degree phase, the required value is 45500 = 0xB1BC.
6. Phase values entered are signed numbers in the range +360000 to -360000.
7. Duty cycle value = (duty cycle in %) * 1000. For example, for a 50% duty cycle, the value is 50000 = 0xC350.
8. To enable the interrupt, the user must write to the interrupt enable register with all 1's. Clock overrun and clock underrun are bits gated by interrupt enable bits. Interrupts corresponding to the enabled bits in the interrupt enable register are updated in this register.

> ***Note:*** You need to write all the register sets with the required values, even if you want the change only in one particular register.

Clock Monitor Registers

The Clock Monitor error status register, the Interrupt status register, and the Interrupt enable register have the following bit map:

Table 2-3: **Clock Monitor Registers Bit Map**

Bit Number	Description
0	User clock 0 frequency is greater than the specifications.
1	User clock 1 frequency is greater than the specifications.
2	User clock 2 frequency is greater than the specifications.
3	User clock 3 frequency is greater than the specifications.
4	User clock 0 frequency is lesser than the specifications.
5	User clock 1 frequency is lesser than the specifications.
6	User clock 2 frequency is lesser than the specifications.
7	User clock 3 frequency is lesser than the specifications.
8	Glitch occurred in user clock 0.
9	Glitch occurred in user clock 1.
10	Glitch occurred in user clock 2.
11	Glitch occurred in user clock 3.
12	Clock stop on user clock 0.
13	Clock stop on user clock 1.
14	Clock stop on user clock 2.
15	Clock stop on user clock 3.
16-31	Undefined.

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. Jitter can be provided in the UI or PS from the GUI.
- If the same input clock is used by other logic in the design, provide no buffer (if the input clock is output of global buffer), or a global buffer option for the source type. If the input clock is used only by the core, provide a clock-capable pin as the source type.

Clocking

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

Resets

- The Clocking Wizard has an active-High asynchronous reset signal for the 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 deasserted 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, see Appendix B, Upgrading 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. See Feature Summary for more details.

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

You can configure the number of output clocks. 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. If the primitive selected is Auto, then all the maximum seven 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 configuration. You can expose any of the ports on the clocking primitive, and these are provided in the source code.

Primitive Override

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

Clock Monitor

The Clock Monitor feature allows you to monitor the clocks in a system; typically, the inputs to the MMCM/PLL. It detects changes in the frequency of the clock, glitches in the clock, or a clock stop. It also gives you the option to specify the tolerance required. For example, if you want to see an error only if the frequency is 1 MHz higher than requested, a tolerance of 1 MHz must be specified.

Clock Stop

The Clock Stop goes High when the clock is flat-lined for more than 10 clock cycles. Once the clock is stale for 10 or more clock cycles, it initiates the calculation to detect the stop and signals the `clock_stop` High. The clock stop signal will not become high immediately and can take up to 256 clock cycles from the time the clock is flat. This calculation requires a set of reference clock cycle periods, time at which the `clock_stop` will go High depends on the ratio of `user_clk` and `ref_clk`.

Clock Glitch

The Clock Monitor can detect a glitch in the user clock. The minimum glitch it can detect in the user clock is one clock period of the reference clock. The Clock Glitch condition might overlap with the Clock Overrun.

Clock Out of Range

The Clock Monitor detects if the user clock frequency exceeds or goes below the required frequency.

Note: The Clock Underrun signal might also go High during the stop condition, if the frequency of the signals goes much lower than the desired frequency.

Note: There might be cases where OOR/glitch is not detected if there is a large difference between the maximum user clock frequency (e.g., 280 MHz) and minimum user clock frequency (e.g., 10 MHz).

Auto Primitive

This feature helps to instantiate the clocking primitive which best fits your requirements, with high performance and better clock routing, while keeping the use of clocking resources to a minimum. All clocking features and optional ports are deselected when you select the primitive as **Auto**. You need to exclusively enable the options which are required. The following tables explain the selection criteria depending on the clocking features selected.

Note: This feature is available for UltraScale™ and UltraScale+™ devices only.

Table 3-1: **Auto Primitives**

Feature	BUFGCE_DIV	PLL	MMCM	Selection Criteria	Inferred Primitive
Phase Alignment	X	X	✓		MMCM
Spread Spectrum	X	X	✓		MMCM
Dynamic Phase Shift	X	X	✓		MMCM
Secondary Input Clock	X	X	✓		MMCM
Input_clk_stopped	X	X	✓		MMCM
Clock fb stopped	X	X	✓		MMCM
More than four output clocks	X	X	✓		MMCM
Use CDDC	X	X	✓		MMCM
Dynamic Reconfig	X	✓	✓	If number of output clocks > 2	MMCM
				If number of output clocks ≤ 2	PLL
Safe Clock Startup	X	✓	✓	If number of output clocks > 2	MMCM
				If number of output clocks ≤ 2	PLL

Table 3-1: **Auto Primitives *(Cont'd)***

Feature	BUFGCE_DIV	PLL	MMCM	Selection Criteria	Inferred Primitive
Out_freq >In_freq	X	✓	✓	If number of output clocks > 2	MMCM
				If number of output clocks ≤ 2	PLL
Reset	X	✓	✓	If number of output clocks > 4	MMCM
				If number of output clocks ≤ 4 The clocks must satisfy condition 2 and 3.	PLL
Non zero Phase Shift	X	✓	✓	If number of output clocks > 2	MMCM
				If number of output clocks ≤ 2	PLL
Locked	X	✓	✓	If number of output clocks > 4	MMCM
				If number of output clocks ≤ 4 The clocks must satisfy condition 2 and 3.	PLL
Power_down	X	✓	✓	If number of output clocks > 4	MMCM
				If number of output clocks ≤ 4 The clocks must satisfy condition 2 and 3.	PLL
Output Buffer selection	✓	✓	✓	If all output clocks satisfy the conditions 1 and 2	BUFGCE_DIV (Supports a maximum of four clocks)
				Else if clocks satisfy condition 2 and 3	PLL
				Else	MMCM

Condition 1

The following table gives the duty cycle values which are possible with the divide values from 1 to 8. Each clock must fall under any one of the divide and duty cycle combinations to satisfy condition 1.

Table 3-2: **Condition 1**

In_freq / Out<1-4>_freq	Out<1-4>_Duty_Cycle
1,2,4,6,8	50%
3	33% - 34%
5	40%
7	42% -43%

Condition 2

Condition 2 is satisfied when you select any one of the following buffers for all the output clocks:

1. Buffer
2. Buffer_with_CE
3. BUFGCE_DIV

Condition 3

Condition 3 is satisfied when the output clocks 2-7 satisfy the duty cycle and divide combinations specified in the following table.

Table 3-3: **Condition 3**

Out1_freq / Out<2-7>_freq	Out<1-7>_Duty_Cycle
1,2,4,6,8	50%
3	33% - 34%
5	40%
7	42% -43%

Auto Buffer Selection

In Auto mode, few output clocks are derived from the `clkout1` of the PLL through `BUFGCE_DIV`. If the number of clock outputs is greater than 4, the MMCM is inferred. In this case, `clkout1` matched routing must be enabled to derive the clocks from the MMCM.

Buffer options (`Buffer` or `Buffer with CE`) are provided to help you by instantiating the appropriate buffer, which helps in creating better clock routing. When you select these options, the Wizard determines the buffer to be instantiated by taking into consideration the conditions in the following table.

Note: This feature is available for UltraScale and UltraScale+ devices only.

Table 3-4: **Auto Buffer Selection**

Buffer Selected	Matched Routing	Buffer Inferred
Buffer	true	BUFGCE_DIV
	false	BUFG
Buffer_with_CE	true	BUFGCE_DIV
	false	BUFGCE

If you select specific buffers like `BUFG`, `BUFGCE`, `BUFGCE_DIV`, or `No_Buffer`, the Wizard instantiates them specifically.

Matched Routing

- Enabling matched routing on the clocks conveys the information to the implementation tool to use the same `CLOCK_ROOT` for the selected clocks. This is performed by setting the `CLOCK_DELAY_GROUP` property on the corresponding clock nets in the Xilinx® design constraints (XDC) file for the IP.
- For clocks without matched routing, the clock skew on timing paths with other clocks is not optimized. This can lead to difficult timing closure. Use matched routing preferably for high-frequency clocks with many clock domain crossing paths.
- The Wizard infers the `BUFGCE_DIV` as buffer when matched routing is selected. This buffer helps in better matched routing.

Optimal Clocking Structure

This feature helps to provide optimal clocking structure when an explicit primitive (MMCM or PLL) is selected along with auto buffer. This is not applicable to 7 series devices because the auto buffer feature is not supported in 7 series devices.

Whenever you select the check box, the IP creates the optimal clocking structure. This option will be helpful because it is backward compatible with the earlier clocking structure.

New Parameters Added

- User parameter: OPTIMIZE_CLOCKING_STRUCTURE_EN
  - Takes true and false as values with the default value being false.
- Model parameter: C_OPTIMIZE_CLOCKING_STRUCTURE_EN
  - The value of this parameter is 1 if the above user parameter is true, and 0 if it is false.

Example Configuration

Example 1: All Seven Clocks Active with MMCM as Primitive

Table 3-5: **Example 1 Configuration**

Clock	Frequency (MHz)	Duty Cycle
clk_out2	300 (= clk_out1/2)	50
clk_out3	200 (= clk_out1/3)	33.33
clk_out4	150 (= clk_out1/4)	50
clk_out5	120 (= clk_out1/5)	40
clk_out6	100 (= clk_out1/6)	50
clk_out7	100 (= clk_out1/6)	50

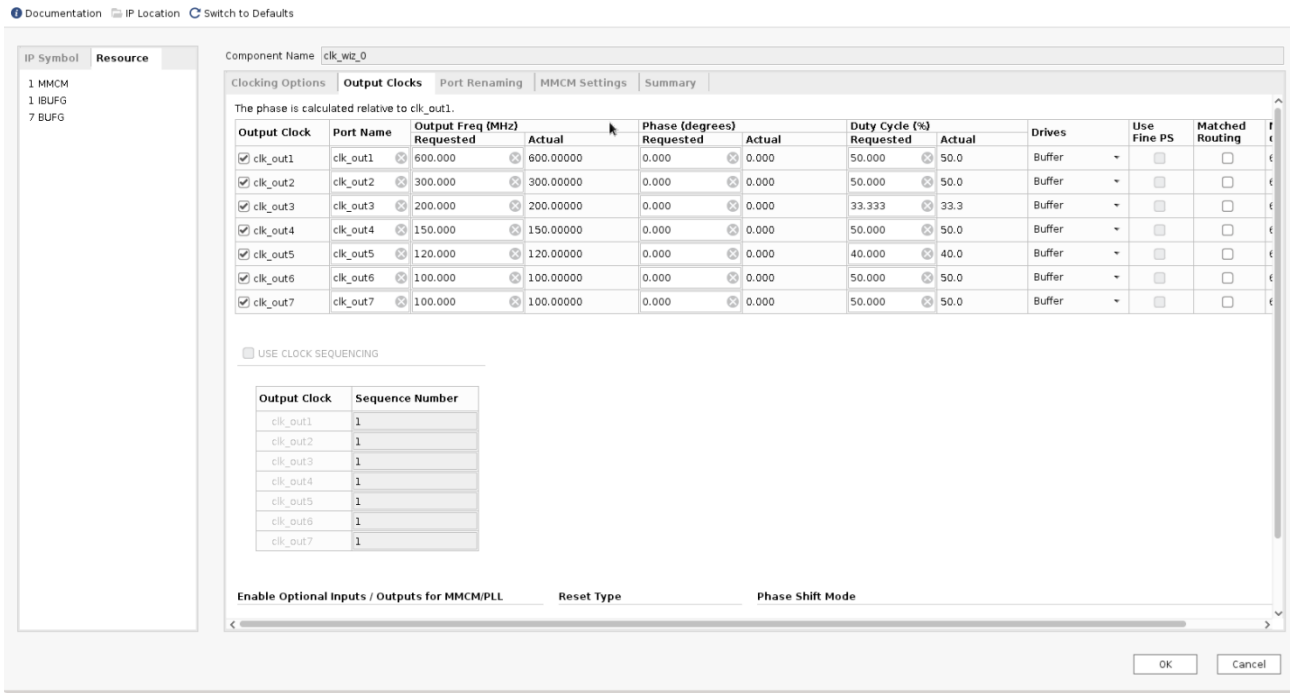

Figure 3-1: **Example 1 Configuration**

1. Optimize clock structure not enabled:
   - The resource tab shows one MMCM, one IBUFG, and seven BUFG.
   - The clocking structure is as follows (which is non-optimal):

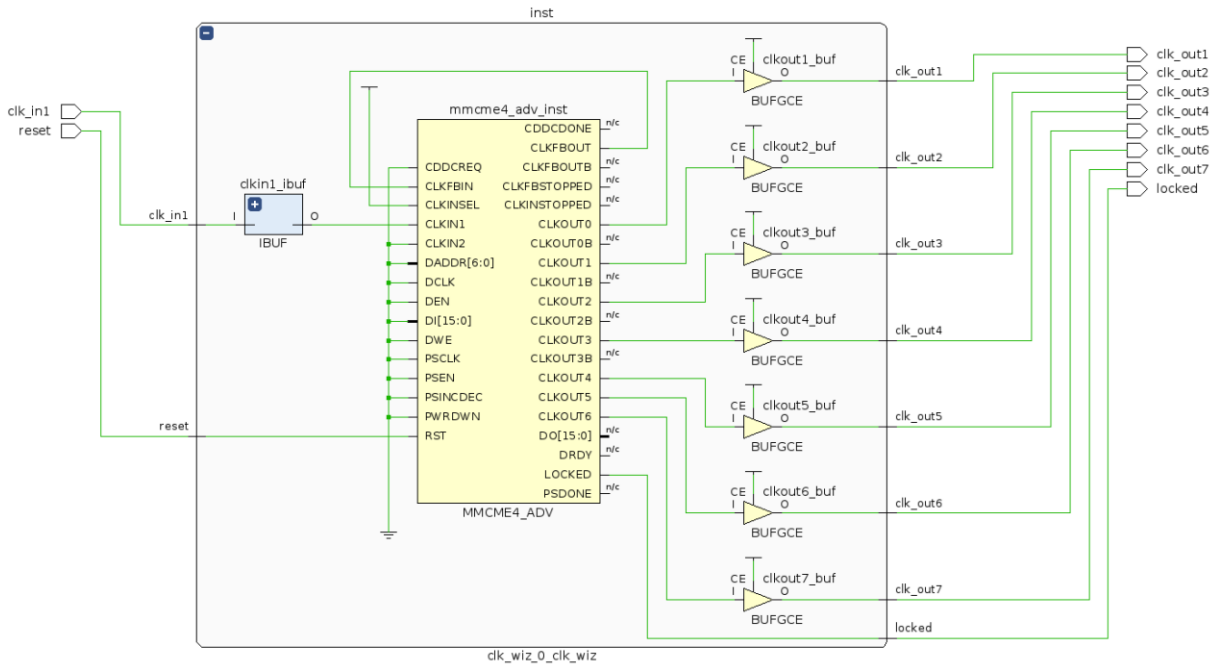

Figure 3-2: **Optimize Clock Structure Not Enabled**

- The summary table is as follows:

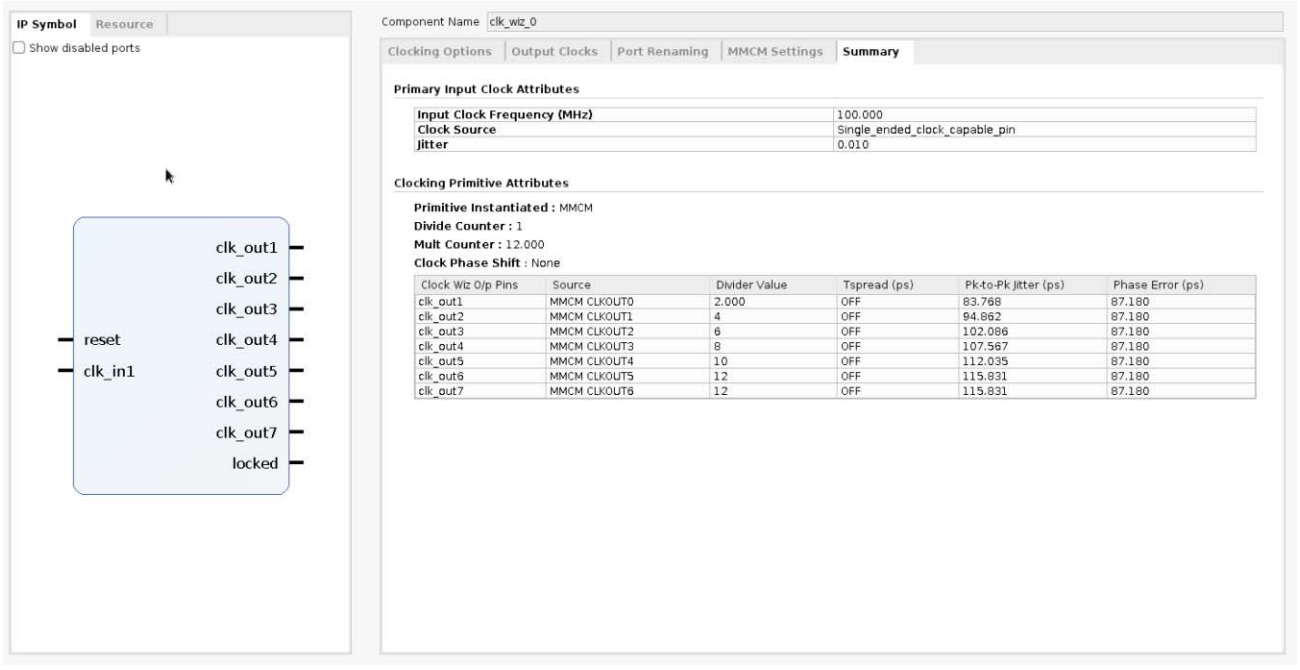

Figure 3-3: **Summary Table for Optimize Clock Structure Not Enabled**

2. Optimize clock structure enabled:
    - The resource tab is updated immediately after enabling optimized clocking structure:

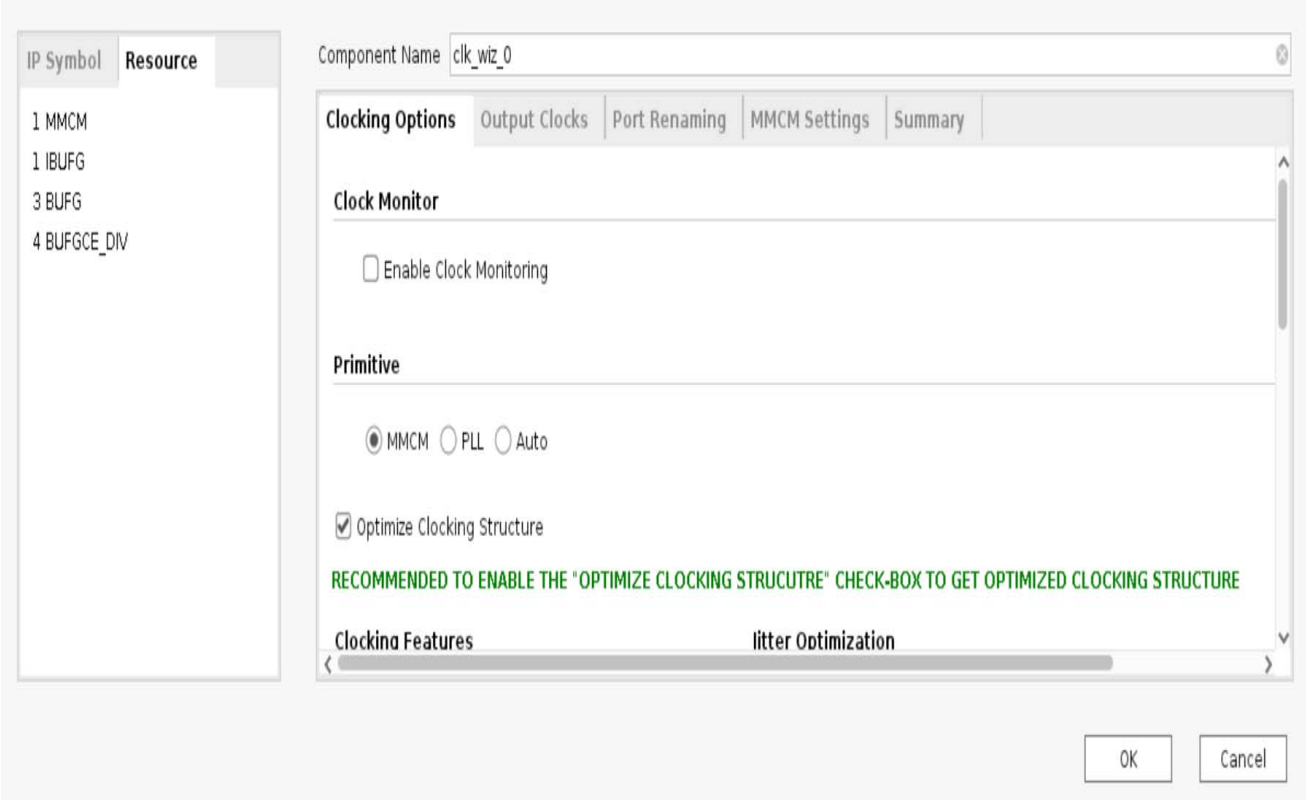

Figure 3-4: **Optimize Clock Structure Enabled**

- The clocking structure is as follows:

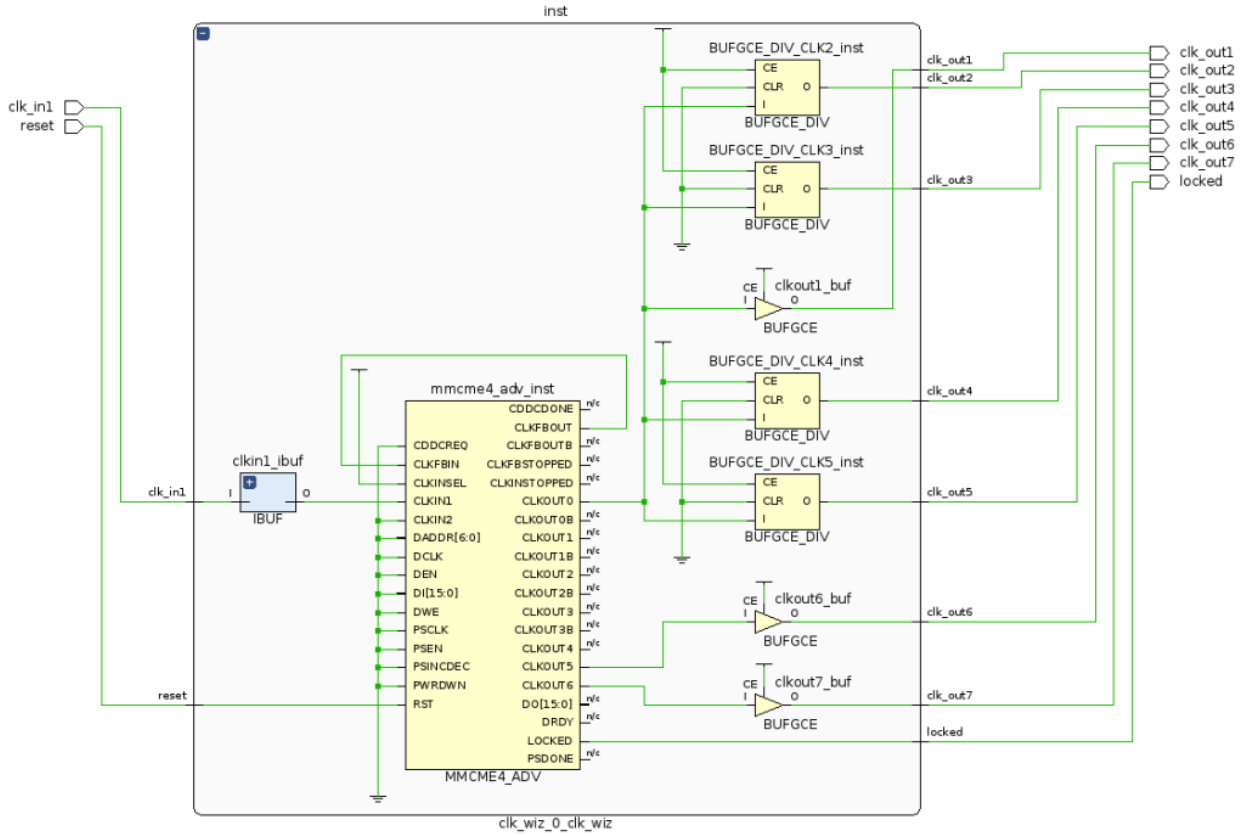

Figure 3-5: **Clocking Structure for Optimize Clock Structure Enabled**

- The summary table is as follows:

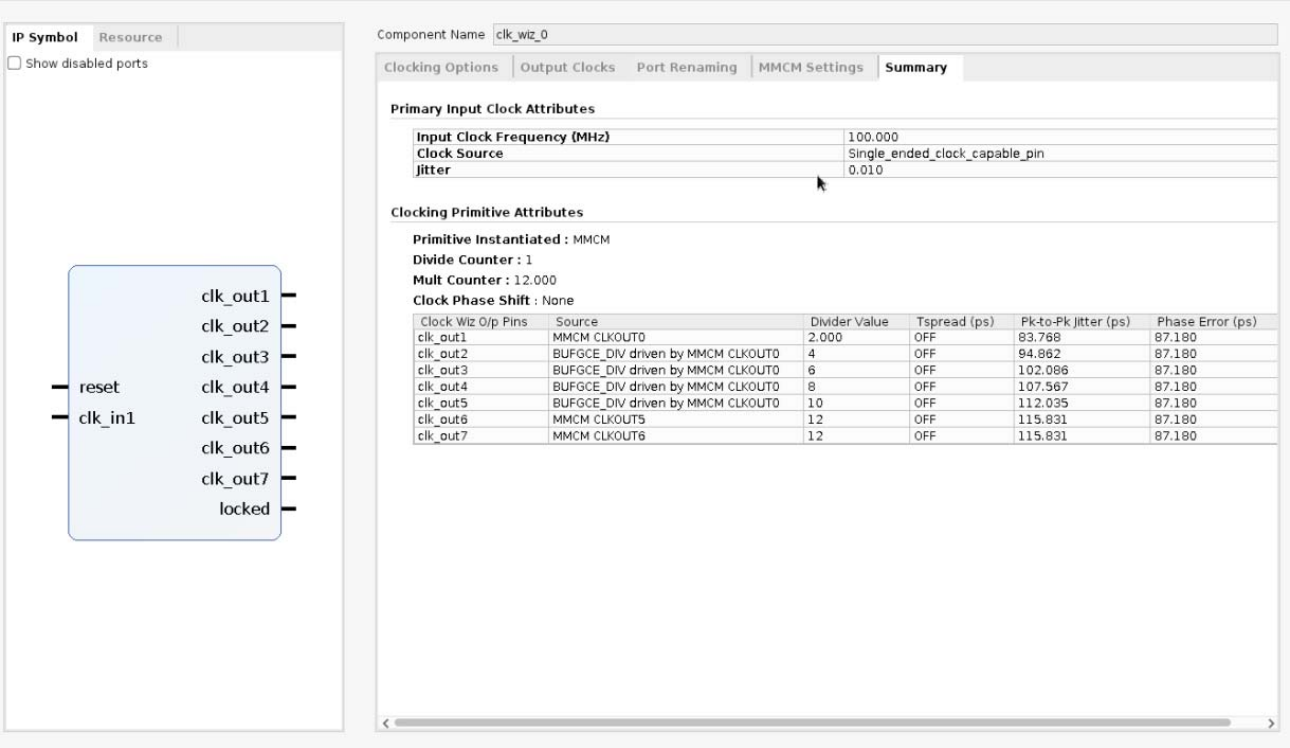

Figure 3-6: **Summary Table for Optimize Clock Structure Enabled**

Example 2: Both Clocks Active with PLL as Primitive

Table 3-6: **Example 2 Configuration**

Clock	Frequency (MHz)	Duty Cycle
clk_out2	300 (= clk_out1/2)	50

1. Optimize clock structure not enabled:
   - The resource tab shows one PLL, one IBUFG, and two BUFGs.
   - The summary table is updated as follows:

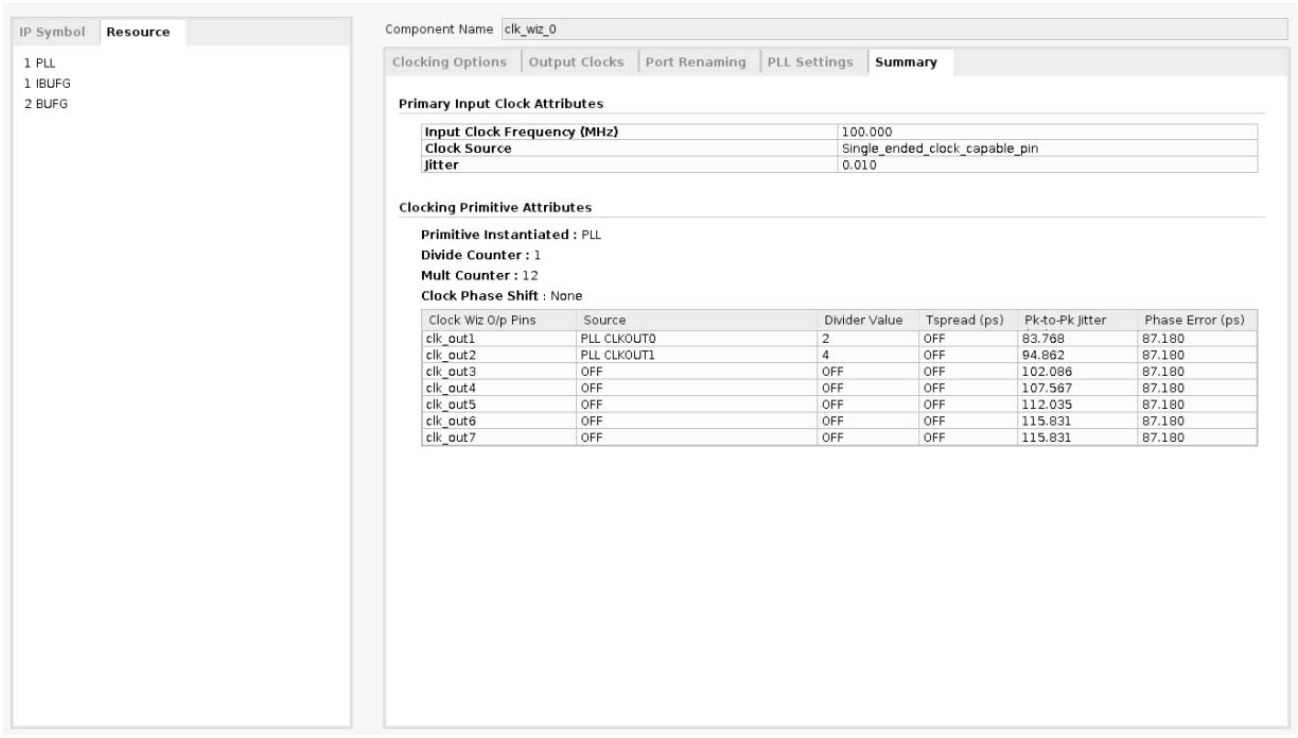

Figure 3-7: **Summary Table for Optimize Clock Structure Not Enabled**

- The clocking structure is as follows:

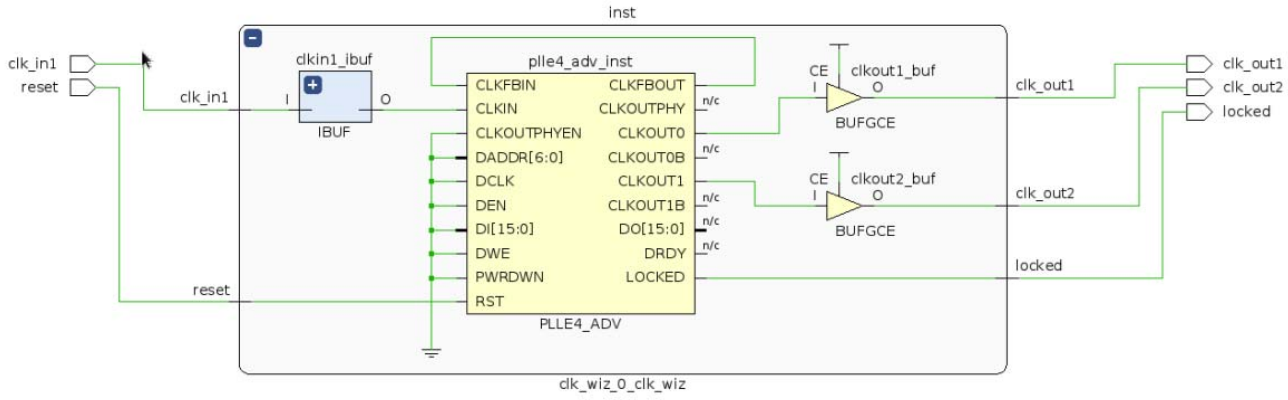

Figure 3-8: **Clocking Structure for Optimize Clock Structure Not Enabled**

2. Optimize clock structure enabled:
   - The resource tab shows one PLL, one IBUFG, one BUFG, and one BUFGCE_DIV.
   - The summary table is updated as follows:

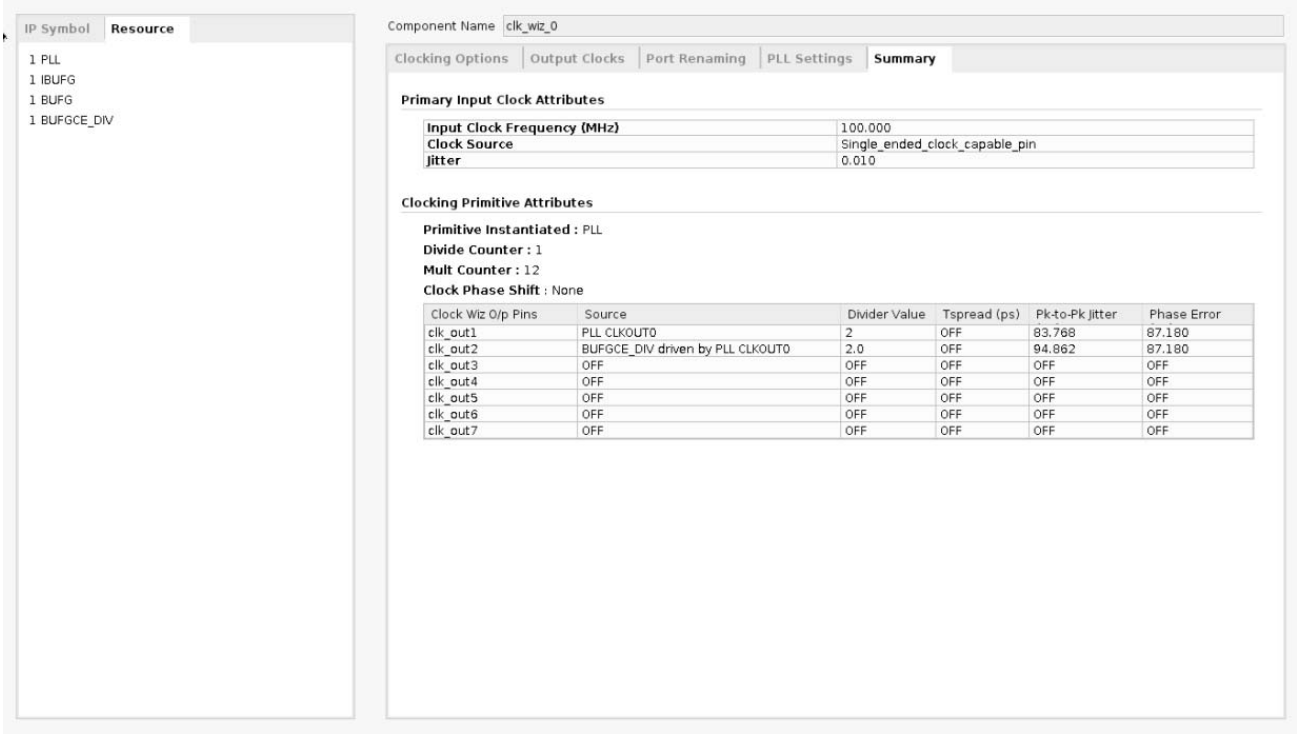

Figure 3-9: **Summary Table for Optimize Clock Structure Enabled**

- The clocking structure is as follows:

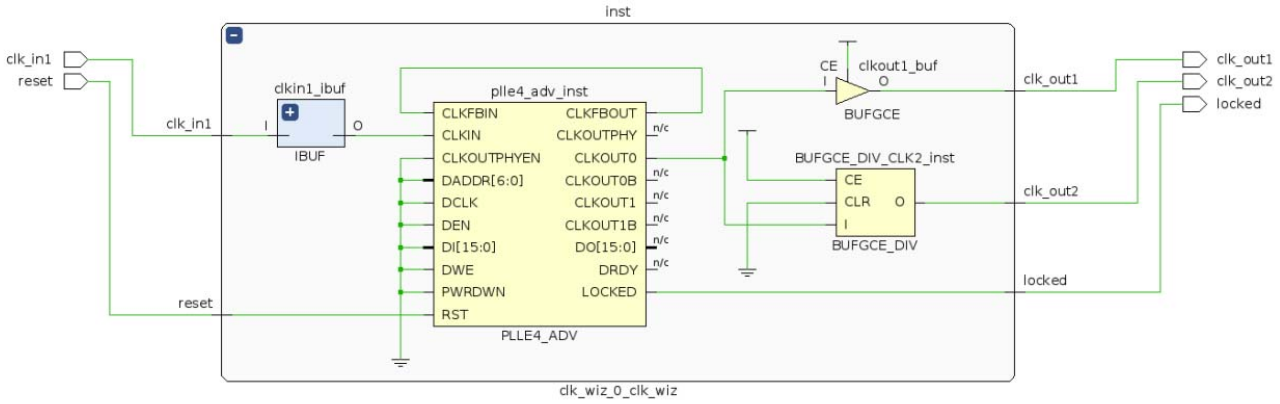

Figure 3-10: **Clocking Structure for Optimize Clock Structure Enabled**

Summary

This feature is available only for UltraScale and UltraScale+ devices. When matched routing is enabled on the clocks and Auto Primitive is selected, the Wizard generates the clocks in such a way that they have the same source and are derived through `BUFGCE_DIV`. These derived clocks have minimum skew. 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.

Design Environment

Figure 3-11 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 simple clocking network in a 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.

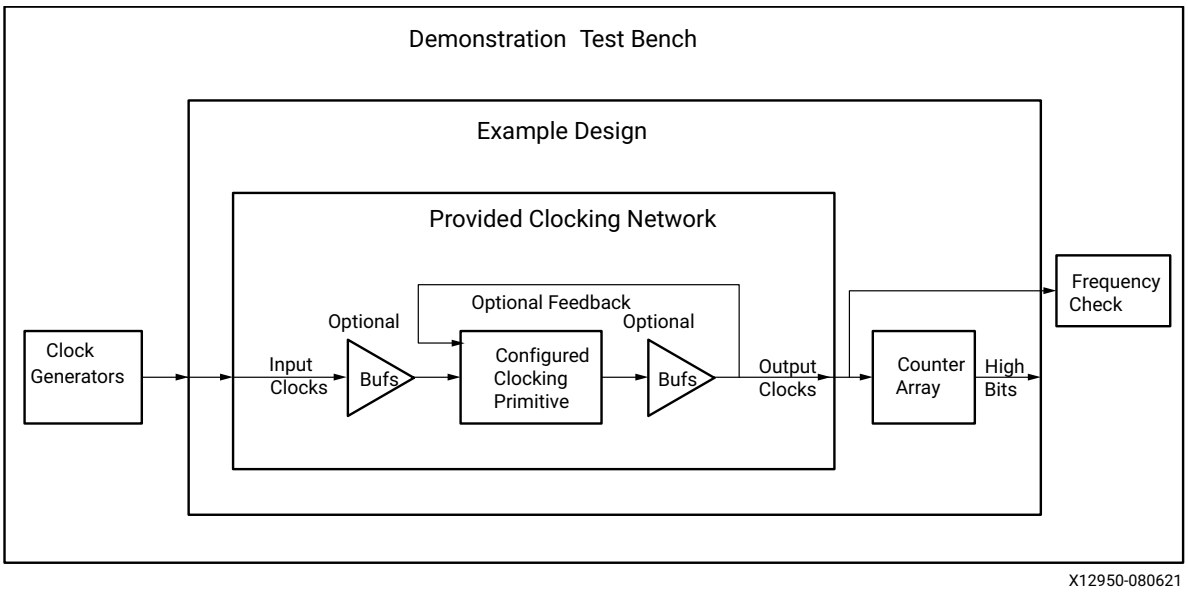

Figure 3-11: **Clocking Network and Support Modules**

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 3-12. Details of these segments are described in the following sections.

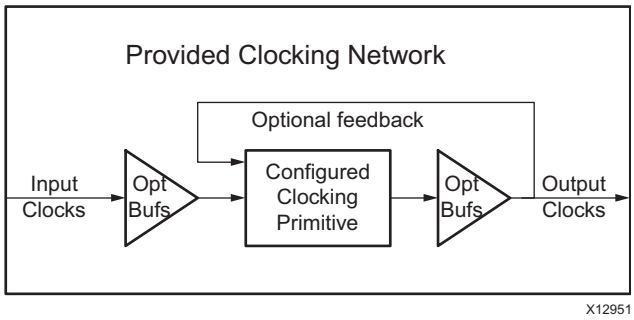

Figure 3-12: **Provided Clocking Network**

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 selected buffer type.

Primitive Instantiation

The primitive, selected either by you or the Wizard, is instantiated into the network. Parameters on primitives are set by the Wizard, and you can override them. 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 and 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

This section includes information about using Xilinx® tools to customize and generate the core in the Vivado Design Suite.

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.

You can customize the IP for use in the 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 right-click menu.

For details, see *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and *Vivado Design Suite User Guide: Getting Started* (UG810) [Ref 4].

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 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 tab of the IDE allows you to identify the required features of the clocking network and configure the input clocks.

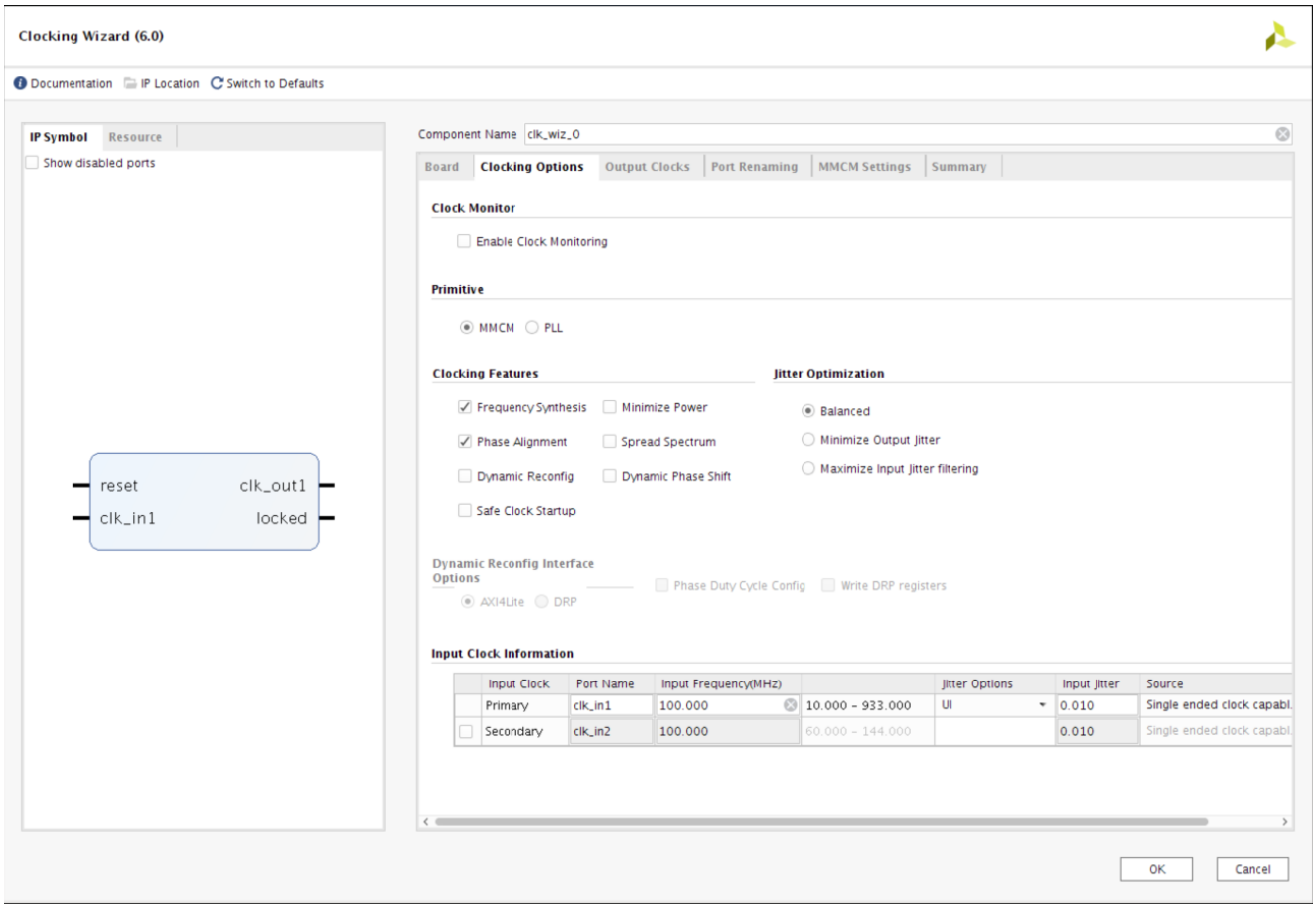

Figure 4-1: **Clocking Options for 7 Series MMCM (Spread Spectrum Unselected)**

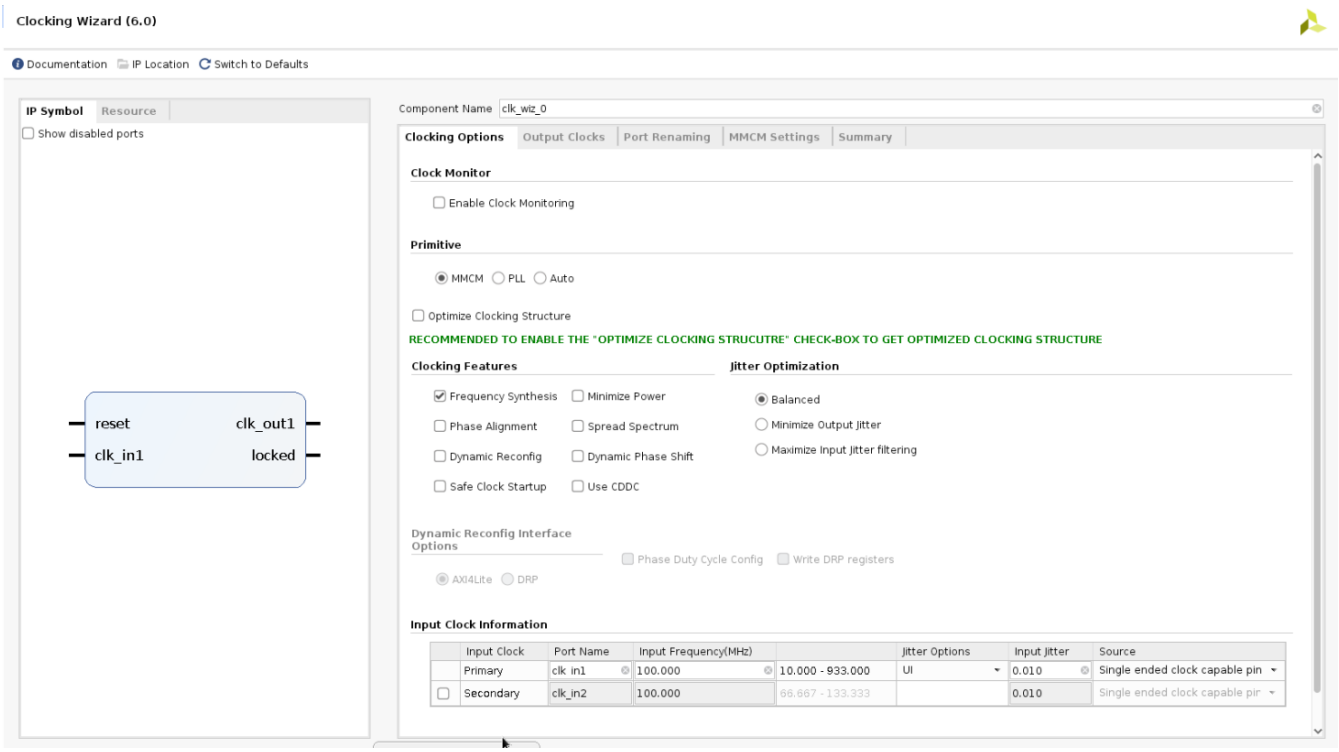

Figure 4-2: **Clocking Options with Spread Spectrum Unselected for UltraScale and UltraScale+**

Note: The **Auto** option under **Primitive** is available only for UltraScale and UltraScale+™ devices.

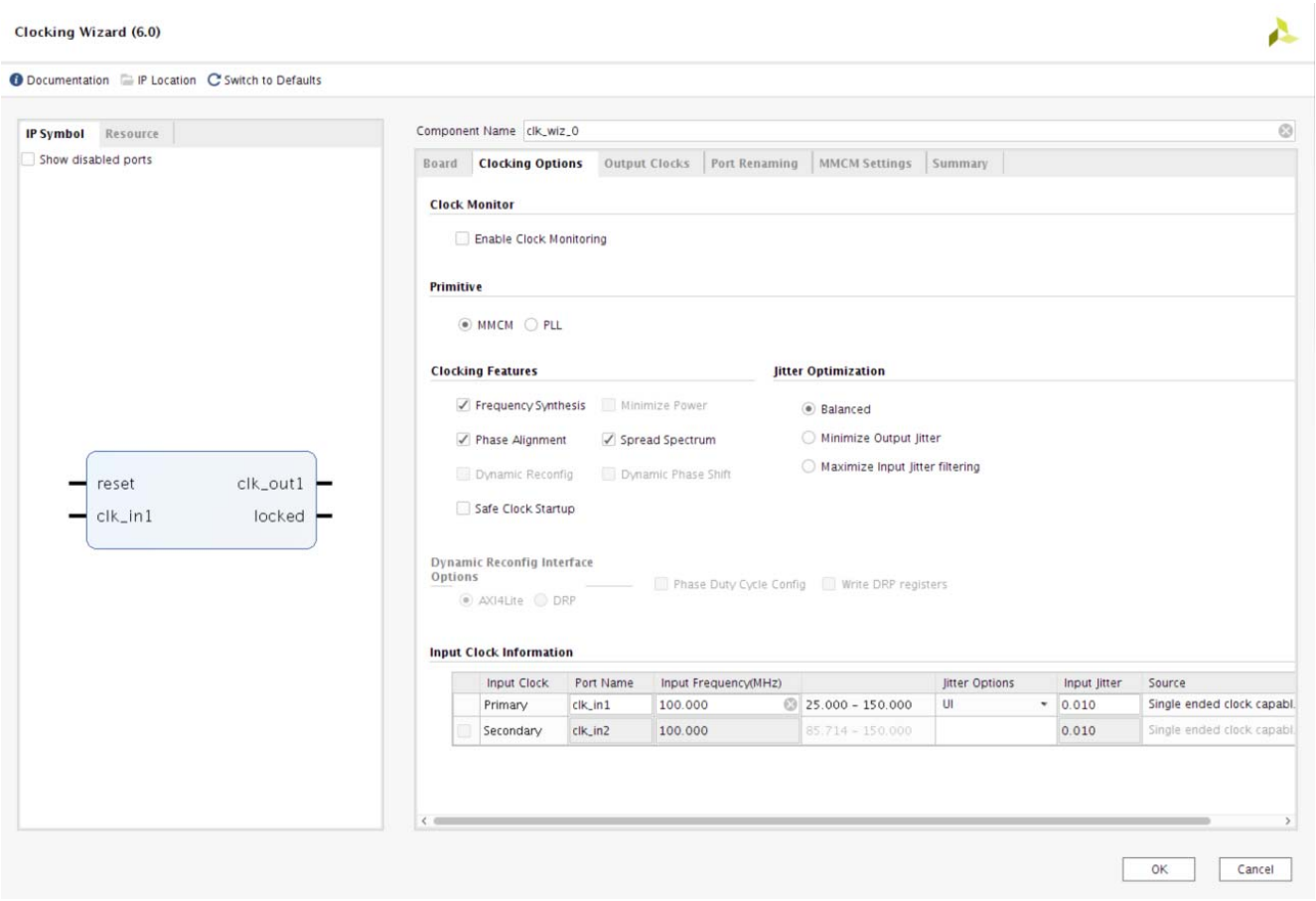

Figure 4-3: **Clocking Options for 7 Series MMCM (Spread Spectrum Selected)**

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, the frequency, phase and clock domain properties of the output clocks are automatically propagated and any change in input clock properties reflects on all the outputs.

Note: The Port Renaming tab has been removed from the Clocking Wizard interface because the IP does not support renaming of the ports in IP integrator.

Details of the clocking options are listed below:

- **Frequency Synthesis** allows output clocks to have different frequencies from the active input clock.

- **Spread Spectrum** provides modulated output clocks, which reduces the spectral density of the electromagnetic interference (EMI) generated by electronic devices. This feature is available for the MMCM(E2/E3/E4)_ADV primitive only. The **Minimize Power** and **Dynamic Reconfig** features are not available when **Spread Spectrum** is selected.
- **Phase Alignment** allows the output clock to be phase locked to a reference, such as the input clock pin for a device. The default value of phase alignment is set to false for UltraScale and UltraScale+ primitives. This feature uses extra clock routes in UltraScale and UltraScale+ designs when MMCMs are used.

RECOMMENDED: Use this feature only if you specifically require it. Extra clock routes can be used by implementation tools for high fanout signals instead of the Phase Alignment feature.

Note: The Phase Alignment option is not available for the UltraScale PLL primitive.

- **Minimize Power** allows you to minimize the amount of power needed for the primitive. This is at the possible expense of frequency, phase offset, or duty cycle accuracy.
- **Dynamic Phase Shift** allows you to change the phase relationship on the output clocks.
- **Dynamic Reconfiguration** allows you to change the programming of the primitive after device configuration. When this option is chosen, the AXI4-Lite interface is selected by default for reconfiguring the clocking primitive. The DRP interface can be selected if direct access to MMCM/PLL DRP register is required. See Dynamic Reconfiguration through AXI4-Lite for more information.
- 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 the Maximize input jitter filtering feature.
- **Maximize Input Jitter filtering** 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 the Minimize output jitter feature.
- **Safe Clock Startup** enables a stable and valid clock at the output using `BUFGCE` after Locked is sampled High for eight input clocks. The sequencing feature enables clocks in a sequence according to the number entered in the IDE. The delay between two enabled output clocks in sequence is eight cycles of the second clock in the sequence clock. This feature is useful for a system where modules need to start operating one after the other.
- **Optimize Clocking Structure**. This option is only available to the user for the MMCM and PLL primitives. When this option is enabled, the IP generates the optimal clocking structure for the explicit primitive in combination with the auto buffer selection.

Configuring Input Clocks

If **Auto** is selected under **Primitive**, all the above options would be unselected except **Frequency Synthesis**. There are two input clocks available, and depending on the selection, the reference clock can be switched from one to another. The IDE provides the 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

Note: These input frequency ranges vary with the device selected.

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 Xilinx design constraints (XDC) 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 the buffer type that drives the input clock, and this is then instantiated in the provided source code. If the 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.

In IP integrator, when the Clocking Wizard IP is selected to target a board part, the frequency values that are generated to the primary and secondary clocks are displayed in a floating number format. For example, if the primary clock frequency is 100 MHz, it is displayed as 100.000 instead of 100.

Output Clock Settings

The requirements for each selected output clock can be configured on the **Output Clocks** tab in the IDE (Figure 4-4).

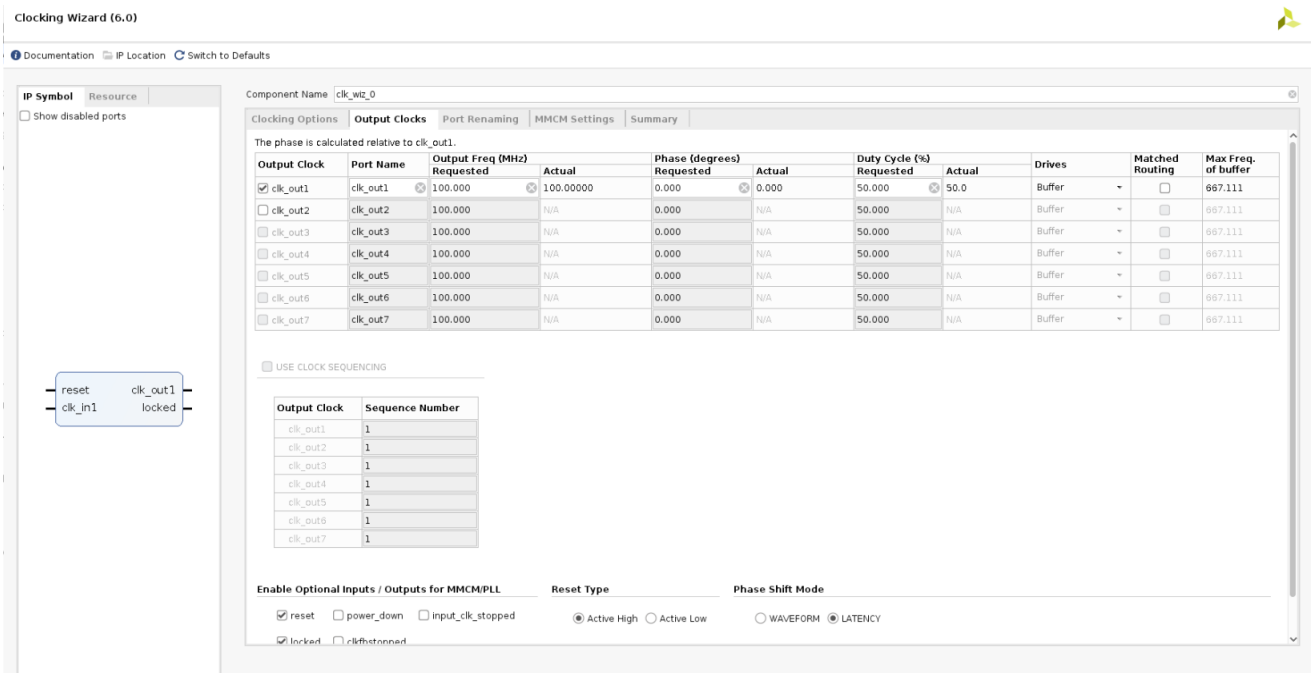

Figure 4-4: **Output Clocks for MMCM (Spread Spectrum Unselected)**

Configuring Output Clocks

To enable an output clock, click on the box located next to it. Output clocks must be enabled sequentially. You can rename the output clocks in the output clock table itself.

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. Actual frequencies are calculated to limit the values to three decimal places. 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 a 180° phase shift is requested on `clk_out2`, `clk_out3`, `clk_out4`, or `clk_out5`, 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, and so on, are identical to the previous clock. If `clk_out1` is configured with 100 MHz and a 0° phase shift, and `clk_out2` is configured with 100 MHz and 180° phase shift, `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, clk_out3 uses its own phase settings and is connected to clkout2 of the MMCM. If you have another clock, clk_out4, with a 180° phase shift compared to clk_out3, 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.

Note: The **Max Freq. of buffer** column in the **Output Clocks** tab of the Clocking Wizard (as shown in Figure 4-4) shows the maximum frequency of the clock that the selected output buffer can drive. When you select a buffer, ensure that the required frequency is within the range of frequencies that the buffer can support. Otherwise, you might get timing violations.

If you select **Dynamic Phase Shift** clocking, the **Use Fine PS** check boxes become available. 'These checkboxes allow you to enable the variable fine phase shift on the 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 the **Safe Clock Startup** feature is enabled on the first tab of the GUI, the **Use Clock Sequencing** table is active and the sequence number for each enabled clock is available for configuration. In this mode, only BUFGCE is allowed as a driver of the clock outputs.

Both 7 series and UltraScale devices support the 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 synthesizable frequencies by a factor of eight. For example, if the CLKIN frequency is 100 MHz and the M divide value is set to 8, 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).

Note: The fractional divide values entered in override mode must be in multiples of 0.125. Otherwise, the IP returns an error saying that the value must be a multiple of 0.125.

When using the fractional divider, the duty cycle is not programmable for outputs used in the fractional mode. Fractional divide is not allowed in fixed or dynamic phase shift mode. The CDDC feature is not available in the fractional divide mode for UltraScale devices. See *7 Series FPGAs Clocking Resources User Guide* (UG472) [Ref 9] and *UltraScale Architecture Clocking Resources User Guide* (UG572) [Ref 8] for more information.

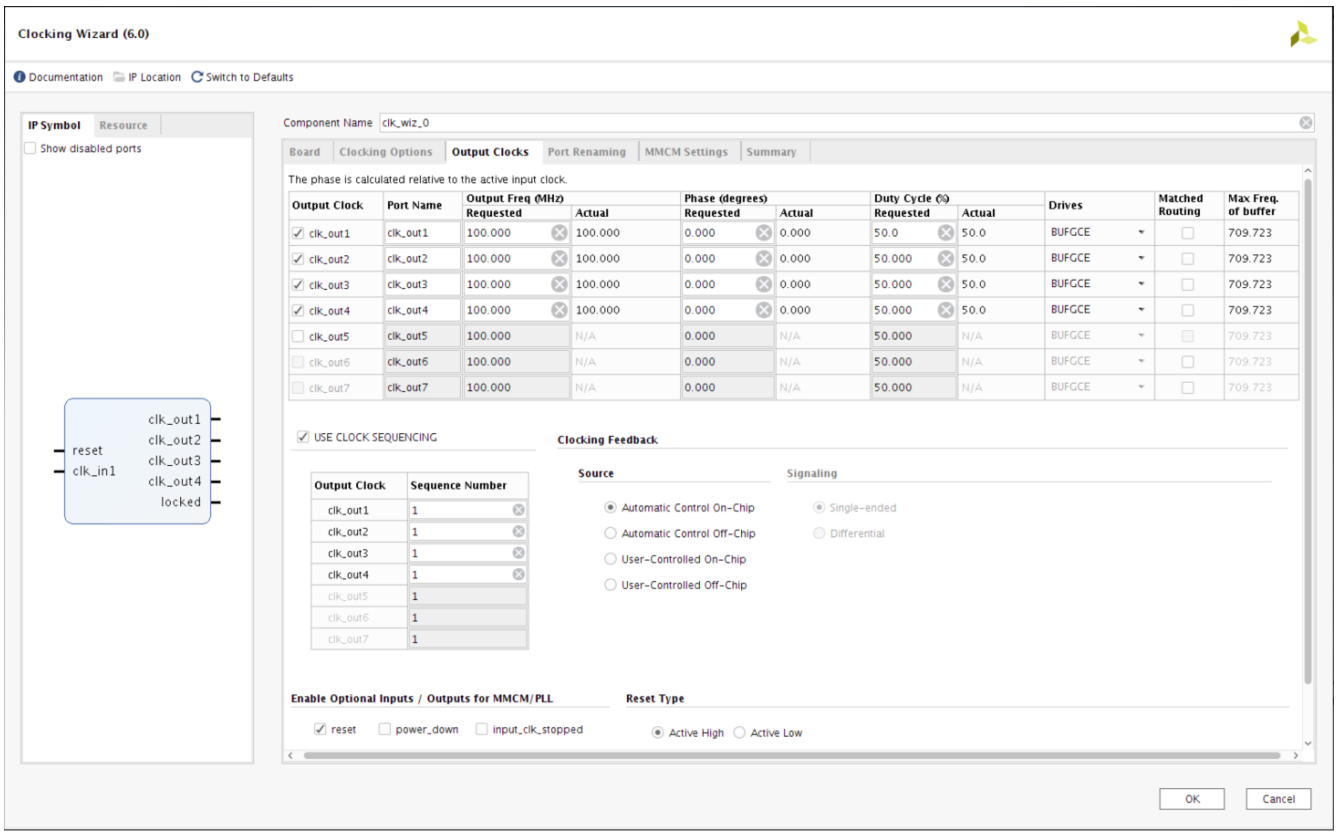

Figure 4-5: **Output Clocks with Safe Clock Start Up and Clock Sequencing for 7 Series MMCM**

You can configure the sequence number from 1 to the maximum number of clocks selected. The Clocking Wizard does not allow any break in the sequence from one to the maximum in the table. The frequency of the output clock in the sequence must not be more than eight times that of the output clock next in sequence. For details of the clocking behavior in this mode, see Figure 4-6 and Figure 4-7.

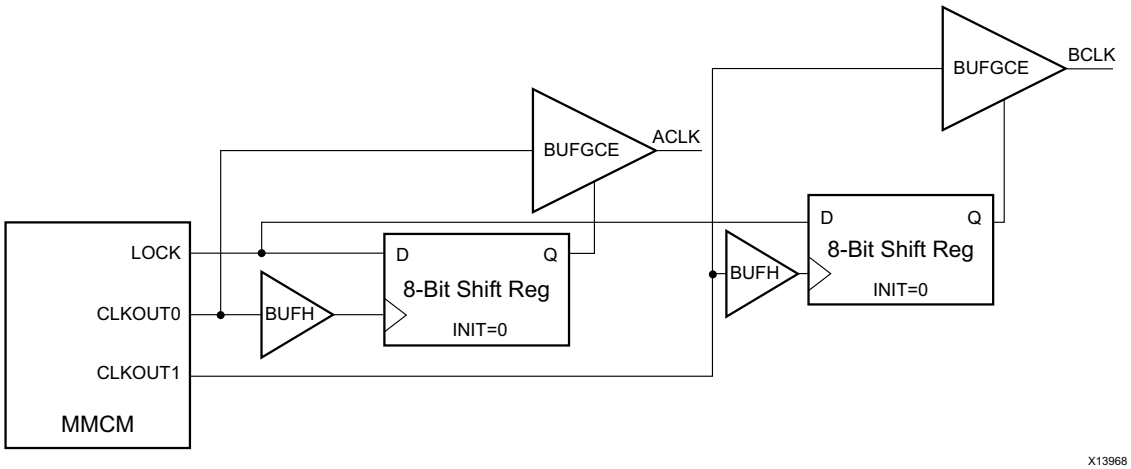

Figure 4-6: **Safe Clock Start Up**

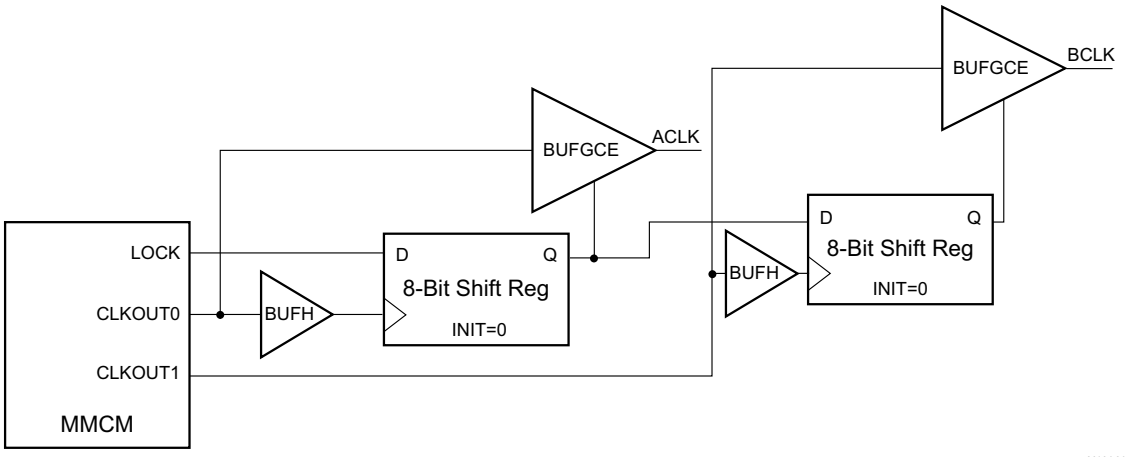

Figure 4-7: **Safe Clock Start Up with Sequencing**

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.

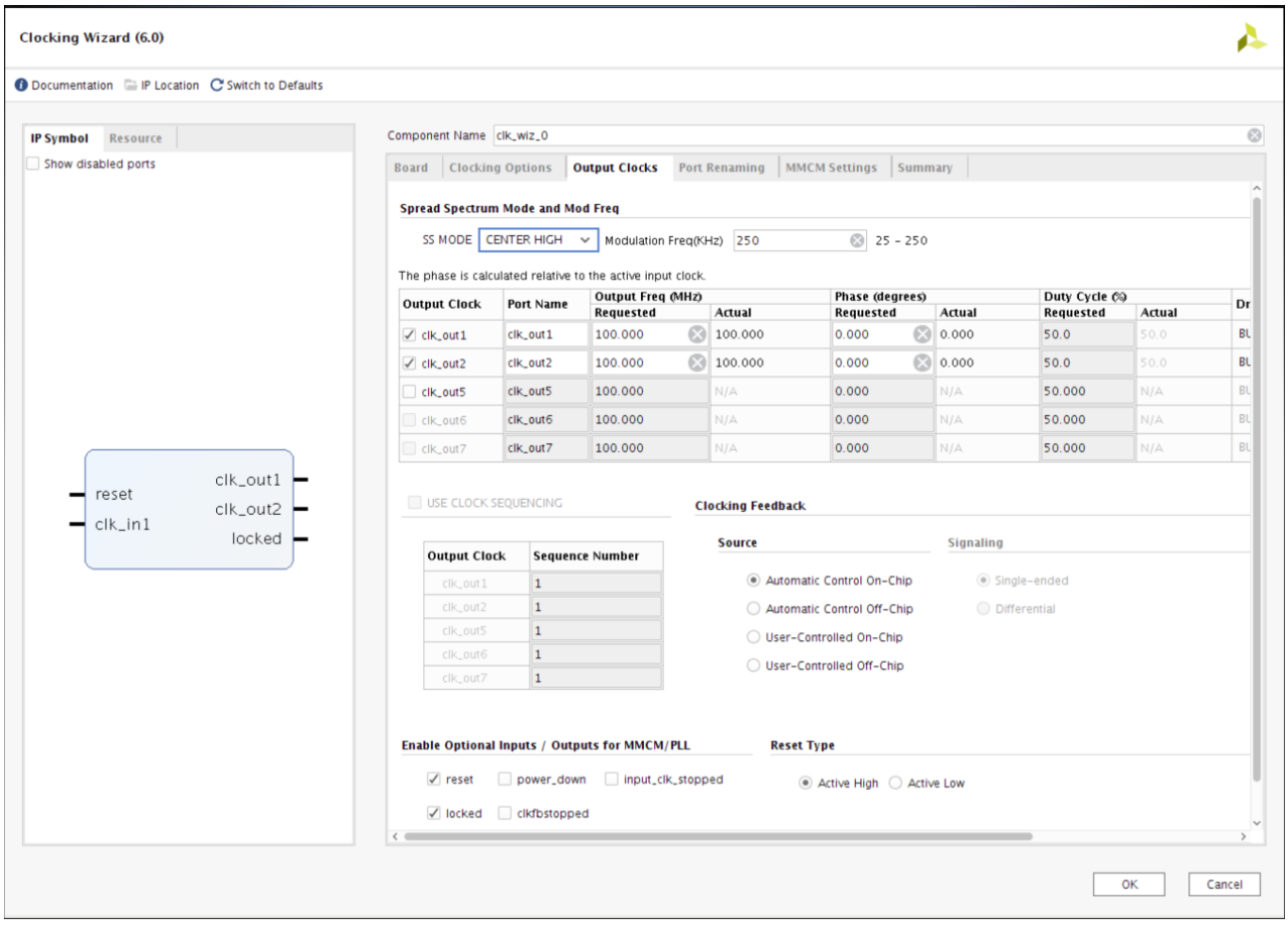

Figure 4-8: **Output Clocks for 7 Series MMCM (Spread Spectrum Selected)**

There are four modes available for SS mode:

- DOWN_LOW
- DOWN_HIGH
- CENTER_LOW
- CENTER_HIGH

The available modulation frequency range is 25 – 250 KHz. Spread spectrum calculation details are described in Figure 4-9 and Figure 4-10.

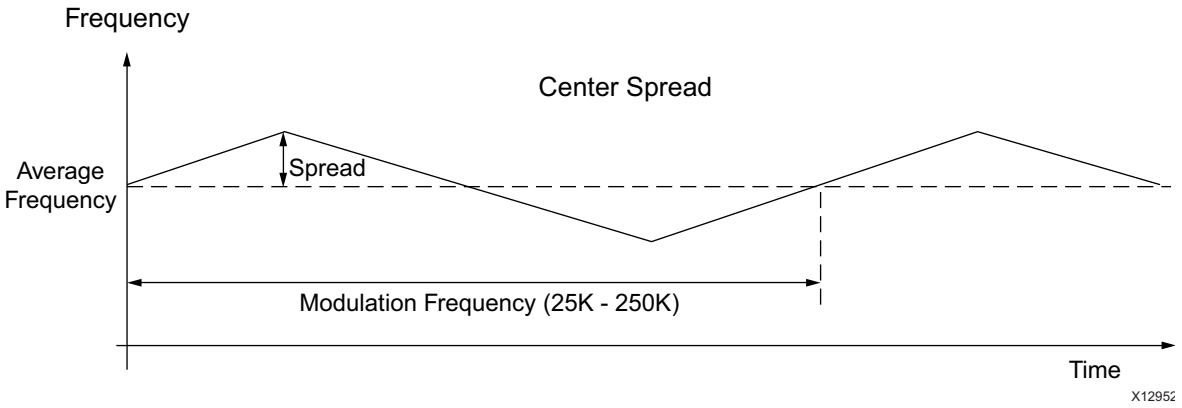

Figure 4-9: **Spread Spectrum Mode (Center Spread)**

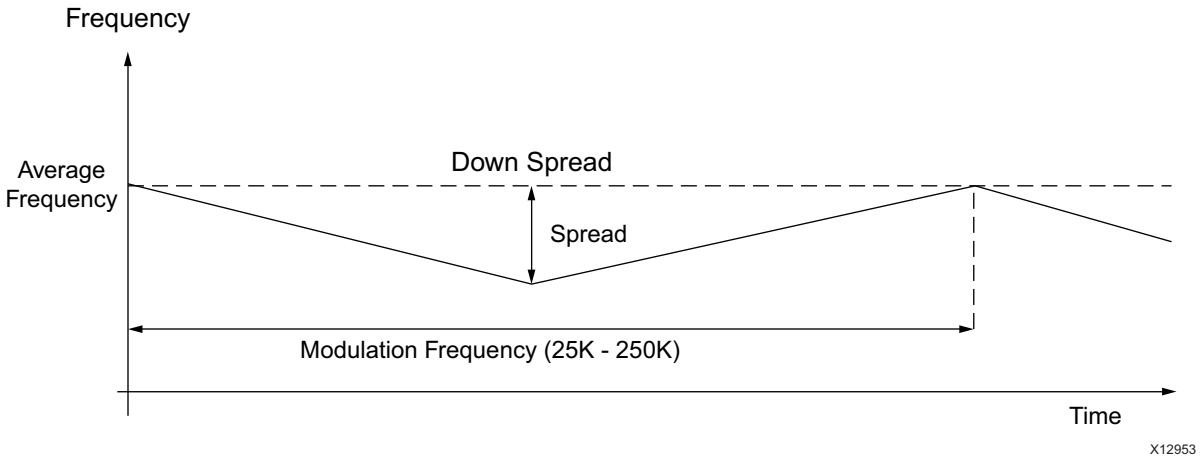

Figure 4-10: **Spread Spectrum Mode (Down Spread)**

Note: Input_clock_frequency is in Hz.

For spread:

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

- If (SS_Mode = CENTER_LOW) :=>
  - spread (ps) = +/- [1/(Input_clock_frequency*(M-0.125*4)/D/O) - 1/(Input_clock_frequency*M/D/O)]
- If (SS_Mode = DOWN_HIGH) :=>
  - spread (ps) = + [1/(Input_clock_frequency*(M-0.125*4)/D/O) - 1/(Input_clock_frequency*M/D/O)]
- If (SS_Mode = DOWN_LOW) :=>
  - spread (ps) = + [1/(Input_clock_frequency*(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. The same calculation is done in the Wizard to get an actual modulation frequency value.
  - Based on how O2 and O3 are 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: *The actual modulation frequency might 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 the Reset Type as active-High or active-Low when **RESET** is enabled. The 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 the logic required to route the feedback signals to the I/O. The **Output Clocks** IDE tab (Figure 4-8) provides information to configure the rest of the clocking network.

Note: When you select the UltraScale PLL, choosing clocking feedback option is not available.

Output Clock Jitter and Phase Error

You can query the jitter and phase error on any of the output clocks of the Clocking Wizard IP. For example, if the component name is **clk_wiz_0**, the jitter and phase error for `clk_out1` can be made available by entering the following commands in Tcl Console:

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

Phase Shift Mode

In UltraScale and UltraScale+, you need to communicate to the tool the type of phase shift required. Whether the phase shifted clock must be modeled into the clock as waveform or latency is determined by this option. No multi-cycle constraint is needed when the phase is modeled as latency.

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 as per your requirement. By selecting **Allow override mode**, the overridden values are used rather than the calculated values as primitive parameters. The Wizard uses the settings shown in Figure 4-11 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.*

The 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 the CLKOUT0* parameters, and not the CLKOUT1* parameters for a MMCM or PLL.

Note: The OVERRIDE_PRIMITIVE parameter is disabled when the selected primitive is **Auto**. You can only update the attributes of the resulting primitive.

MMCM Counter Cascading

The CLKOUT6 divider (counter) can be cascaded with the CLKOUT4 divider. This provides the capability of an output divider that is larger than 128. The CLKOUT6 counter feeds the input of the CLKOUT4 divider. There is a static phase offset between the output of the cascaded divider and all other output dividers. You need to select CLKOUT4_CASCADE option to use this feature. If you are using this option, you need to use the override mode to generate all the required frequencies.

CLKFBOUT_USE_FINE_PS is a variant of the CLKFBOUT counter that enables fine phase shifting for CLKFBOUT. Select the **Dynamic Phase Shift** option to use fine phase shifting for CLKFBOUT.

You can only select the **STARTUP_WAIT** option in the Override mode, that works with the **Configuration CLK_Cycle** option, to wait for the Clock Manager to lock before completing the startup sequence.

See *7 Series FPGAs Clocking Resources User Guide* (UG472) [Ref 9] and *UltraScale Architecture Clocking Resources User Guide* (UG572) [Ref 8] for more information.

The generated source code contains the input and output clock summaries shown in the next summary page, as shown in Figure 4-16.

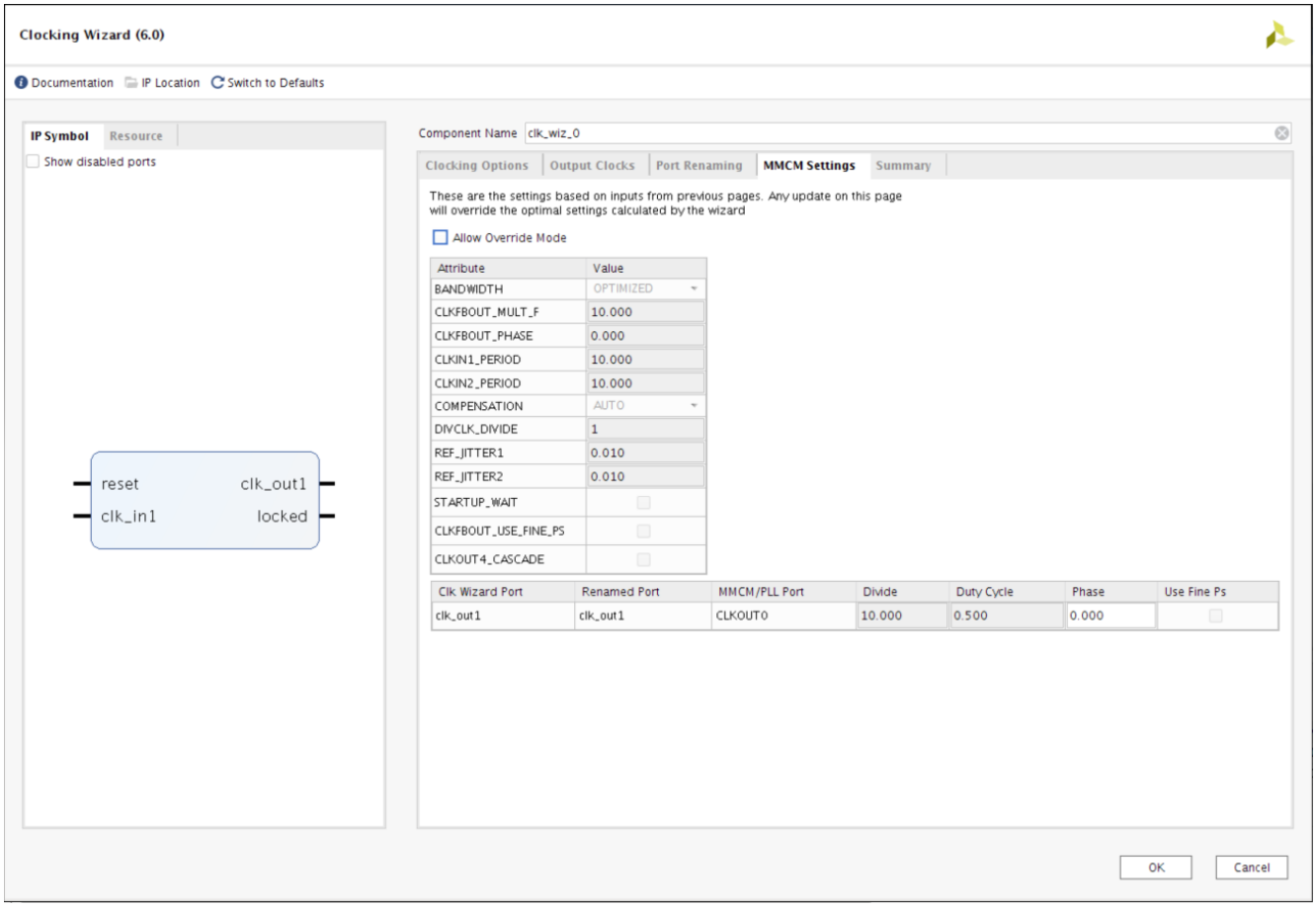

Figure 4-11: **Primitive Override Screen (Spread Spectrum Unselected)**

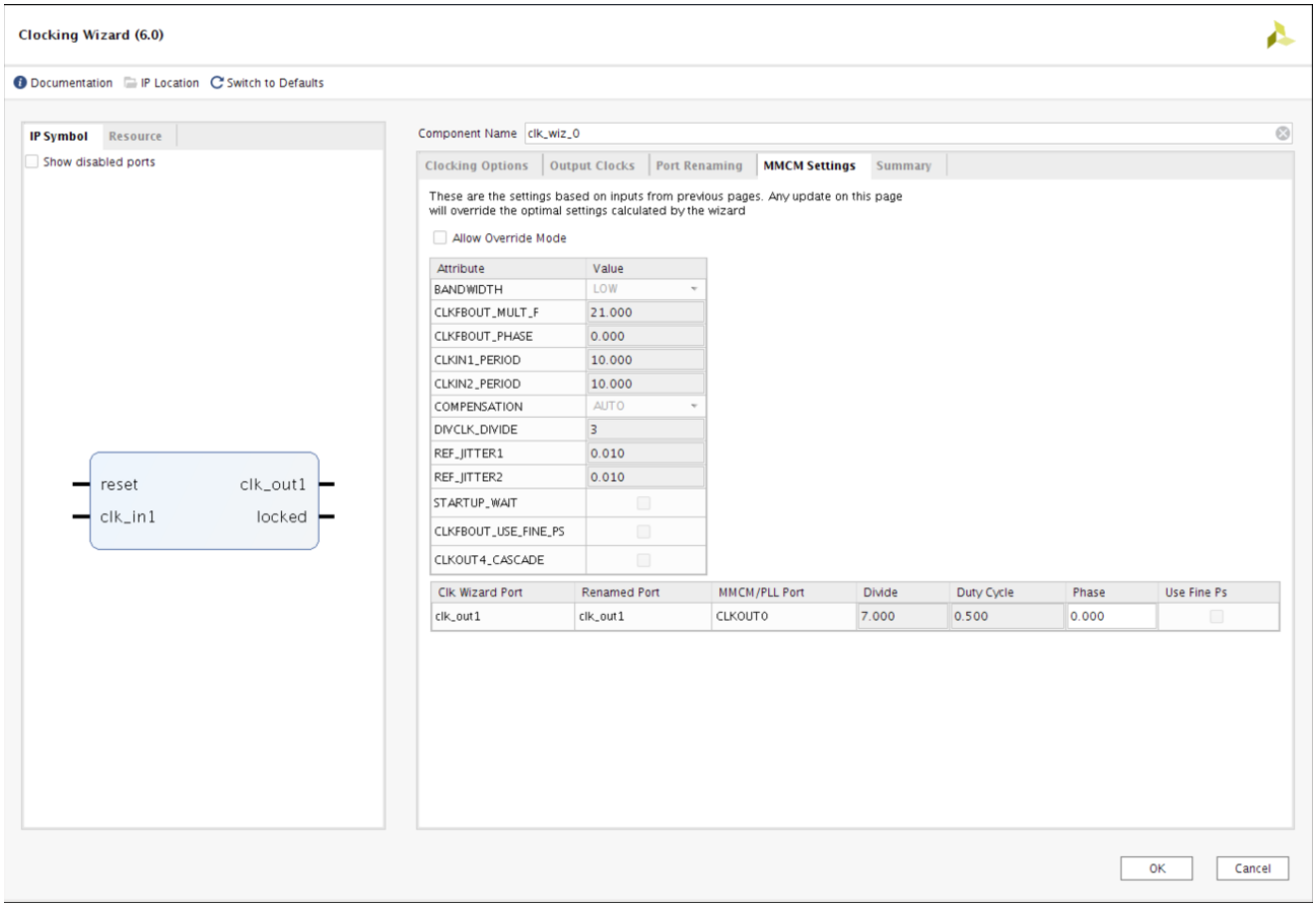

Figure 4-12: **Primitive Override Screen (Spread Spectrum Selected)**

Port Renaming

The first summary page (Figure 4-14) 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.

Note: The renaming of the `reset` port is not currently supported.

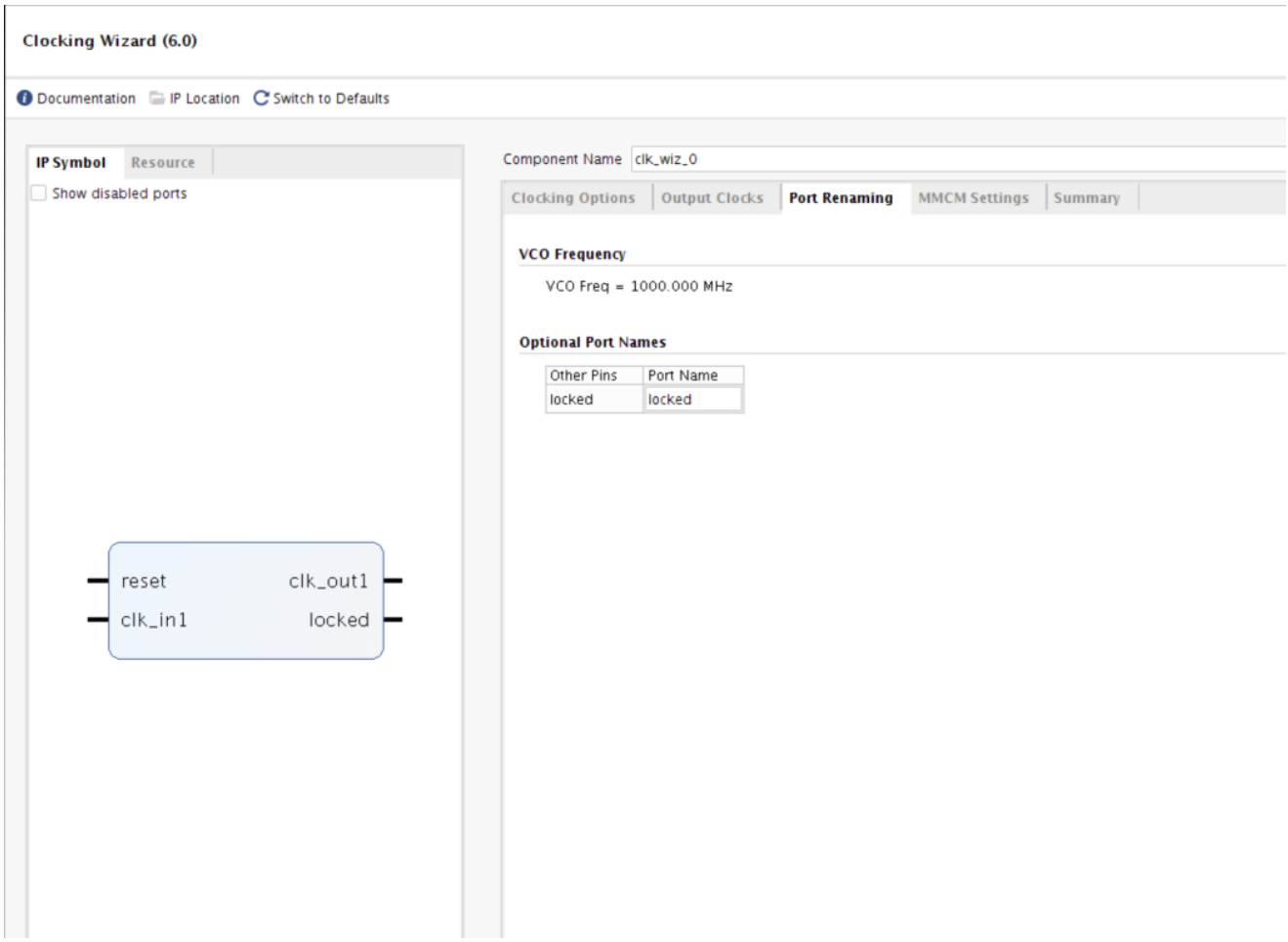

Figure 4-13: **Port Renaming (Spread Spectrum Unselected)**

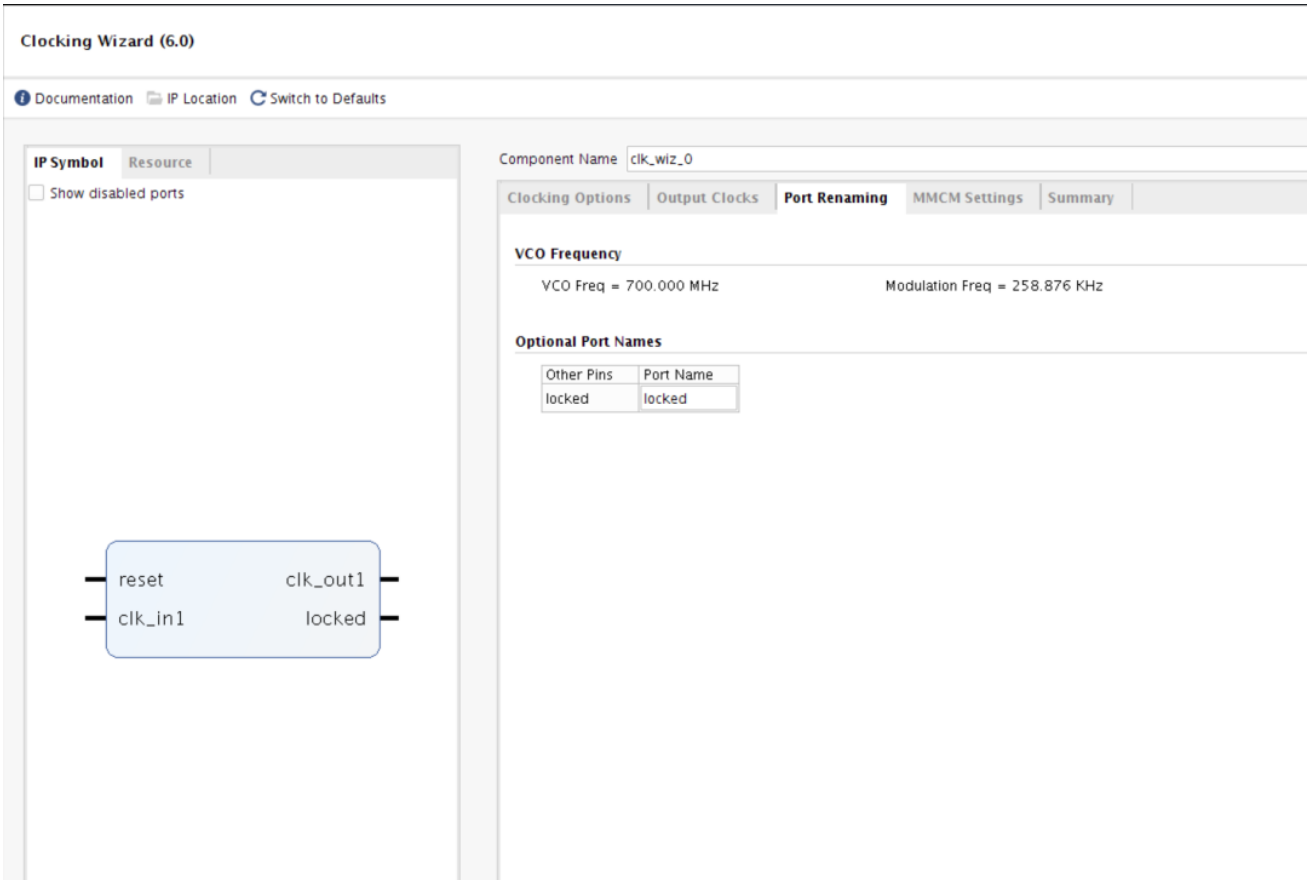

Figure 4-14: **Port Renaming (Spread Spectrum Selected)**

Note: The Port Renaming feature is not supported in Vivado IP integrator.

Input Clocking Summary

The information entered on the first tab of the IDE 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, the actual modulation frequency is provided as a 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, type the port name in the adjacent text box. The text boxes contain the default names. In the case of the 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.*

Clock Monitor

The Clock Monitor feature is a part of the Clocking Wizard IP. It allows you to monitor the clock in a given system for clock loss or out-of-range errors. In Zynq or Zynq UltraScale devices, the clock monitored can be either a processing system (PS) clock or a programmable logic (PL) clock. In FPGAs, the clock monitored can be an arbitrary clock.

- **Clock Stop**: The clock is flat lined.
- **Clock Glitch**: Variation in the duty cycle of the clock.
- **Overrun**: The number of transitions in the clock is higher than expected.
- **Underrun**: The number of transitions in the clock is lower than expected.

Note: **Overrun** and **Underrun** are termed as out-of-range errors.

The IP provides scalable logic to monitor four clocks.

- **Reference Clock Frequency**: The reference clock frequency determines the frequency of the clock to be monitored.
- **Channel Clock Frequency**: You can choose the frequency of the clock to be monitored based on the value of reference clock.
- **Tolerance**: You can program the precision required to monitor the clock.

Note: Only integer values of tolerance are accepted.

- **Enable_PLL/MMCM (0-1)**: Enabling this option monitors the input clock to the MMCM/PLL.

Note: If the Enable_PLL/MMCM options are enabled in the IDE, ensure that the primary/secondary clock frequency does not exceed 300 MHz. Users cannot request user clock frequencies (USER_CLK_FREQ_0/1/2/3) beyond 300 MHz.

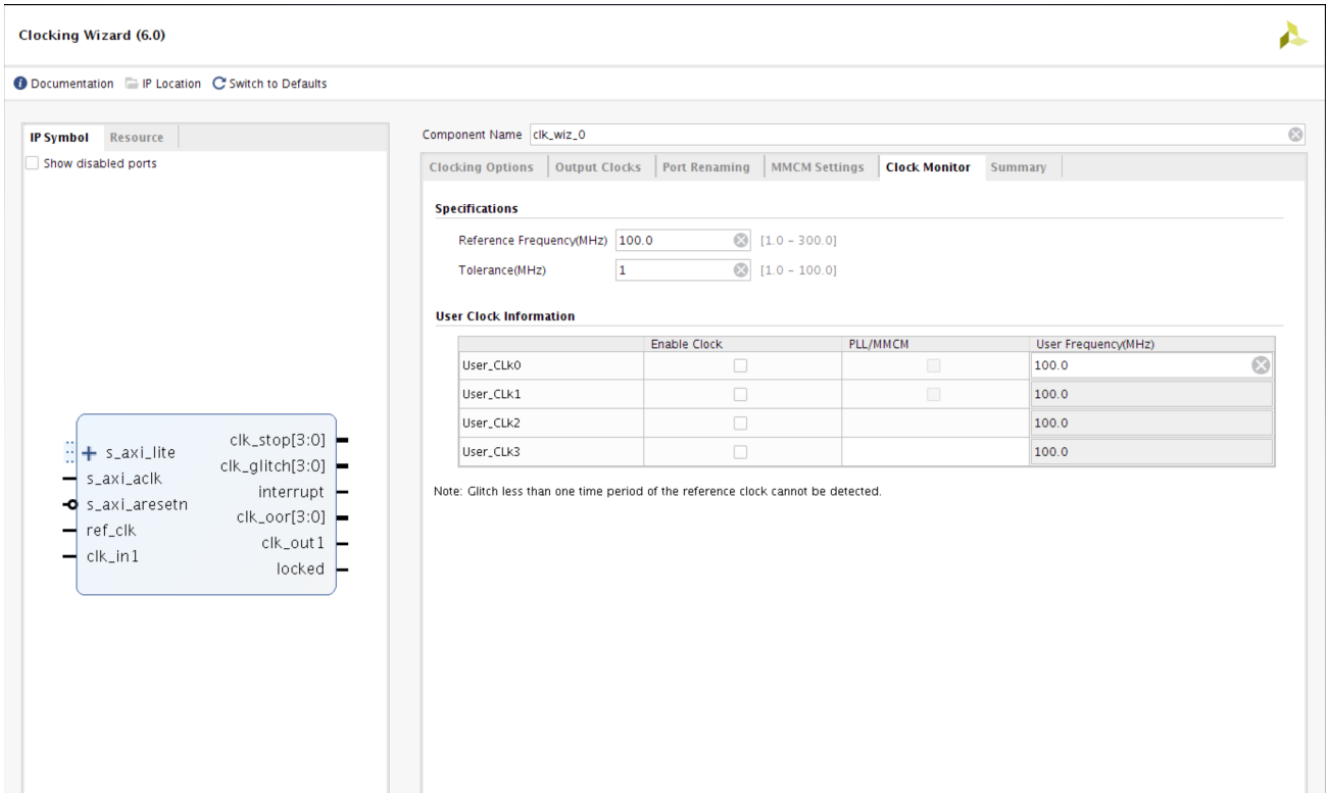

Figure 4-15: **Clock Monitor Settings**

Summary

The summary page (Figure 4-16) contains general summary information.

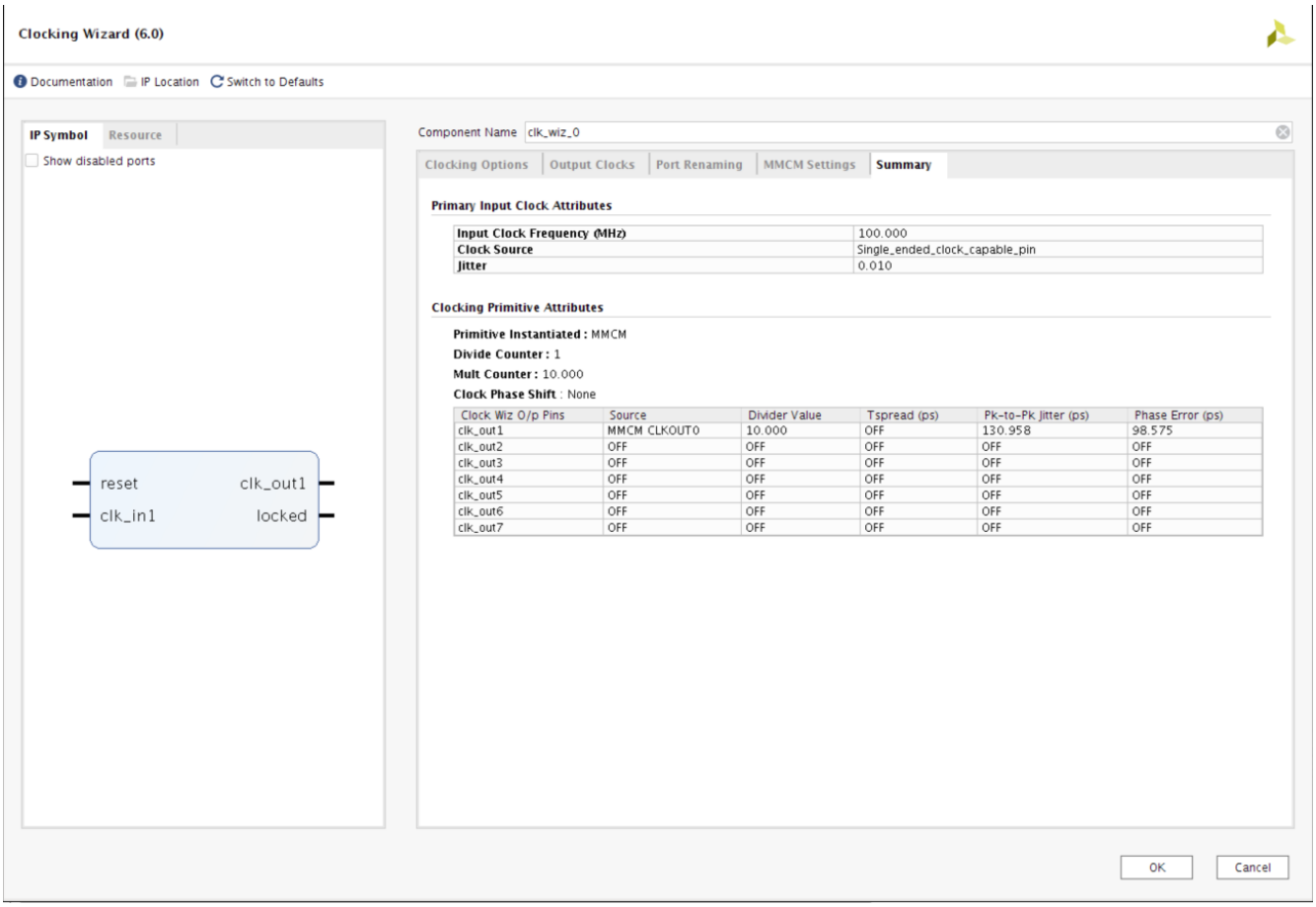

Figure 4-16: **Summary Screen**

The **Summary** page gives you the information about the connections between the Wizard output clocks and the primitive output clocks. The **Source** column in the last table specifies the primitive output pin. It states whether the clock is generated directly from the primitive or derived using `BUFGCE_DIV`.

Resource Estimate Summary

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

Xilinx Power Estimator Summary

Input parameters to the Xilinx power estimator 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 the selected interface is **AXI4-Lite**. This feature is not supported when **Spread Spectrum** is enabled. Mixed-language RTL is delivered by the core when the AXI4-Lite interface is used. To reconfigure the phase and duty cycle, select **Phase Duty Cycle Config**. Enabling this option utilizes DSP resources. By default, this option is deselected to optimize the design for area. Resource utilization for the AXI4-Lite interface configuration of the 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	8	38

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

The Clocking Wizard core uses a configuration state machine listed in *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6] and extends from two fixed-state configurations to program any valid range of Multiply, Divide, Phase and Duty Cycle. In this state machine, state 1 corresponds to default state configured through the Clocking Wizard interface. State 2 corresponds to the user configuration loaded into the Clock Configuration Register detailed in Table 2-2. State 2 values are also initialized with the state 1 values so that a valid configuration is stored by default. All the dynamic reconfiguration registers are to be updated whenever you want to reprogram the clock.

Dynamic Reconfiguration uses resources for internal calculations in the wizard. The user provides CLKFBOUTMULT, DIVCLK_DIVIDE, CLKOUTxPHASE, and CLKOUTx_DUTY other user understandable parameters. They cannot be directly mapped to MMCM/PLL DRP registers. User understandable attributes are converted to MMCM DRP registers and written into primitive. This calculation is done in the wizard using few function call and makes the wizard utilize the resources mentioned in *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6].

If the phase duty cycle configuration parameter is not enabled, the wizard only calls functions related to frequency. This option must be enabled to change phase and duty cycle dynamically. Next, wizard calls the functions related to phase and duty cycle at the cost of resources. If resources cannot be used, use wizard to call these functions during IP generation and write a different address set containing direct DRP data. Here wizard does not do any calculations, it guides the user with the value needed to be written into the registers. Write DRP must be selected to enable this feature.

To do a dynamic reconfiguration, follow the below steps:

1. Write all the Clock Configuration Registers, and then check for the status register.
2. Before writing into the C_BASEADDR + 0x200 register detailed in Table 4-1, make sure that these values result in a valid VCO frequency range of MMCM/PLL which is calculated using the following 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.
3. If the status register value is 0x1, start the reconfiguration by writing Clock Configuration Register 23 with 0x3.

CAUTION! *The fractional enable bit in Clock Configuration Register 0 must only be enabled if the value of the clock `FBOUT MULT` is a non-integer. Similarly, the fractional enable bit in Clock Configuration Register 2 must only be enabled if the value of `CLKOUT0_DIVISE` is a non-integer.*

Write DRP Feature

The main advantage of this feature is to avoid the usage of DSPs in the core. This option can be selected in Dynamic Reconfiguration mode and is valid only for the AXI4-Lite interface. This feature allows you to write directly to the primitive registers through AXI. When this feature is selected, a tab is enabled in the IDE which lists the register addresses and the values which need to be written into them. To execute this feature, follow the below steps:

1. Generate the Clocking Wizard IP. Enable the **Dynamic Reconfig** option and select the **Write DRP registers** feature.
2. Open another Clocking Wizard with the same input clock and the features as intended.
3. Change the output clock features in the **Output Clocks** tab of the Vivado IDE as required for dynamic reconfiguration.
4. The table in the **Write DRP registers** tab is updated for the required clocks. Use these register set values for dynamically reconfiguring the initial Clocking Wizard.

An example of the write DRP feature is described in Example for Dynamic Reconfiguration Using Write to DRP.

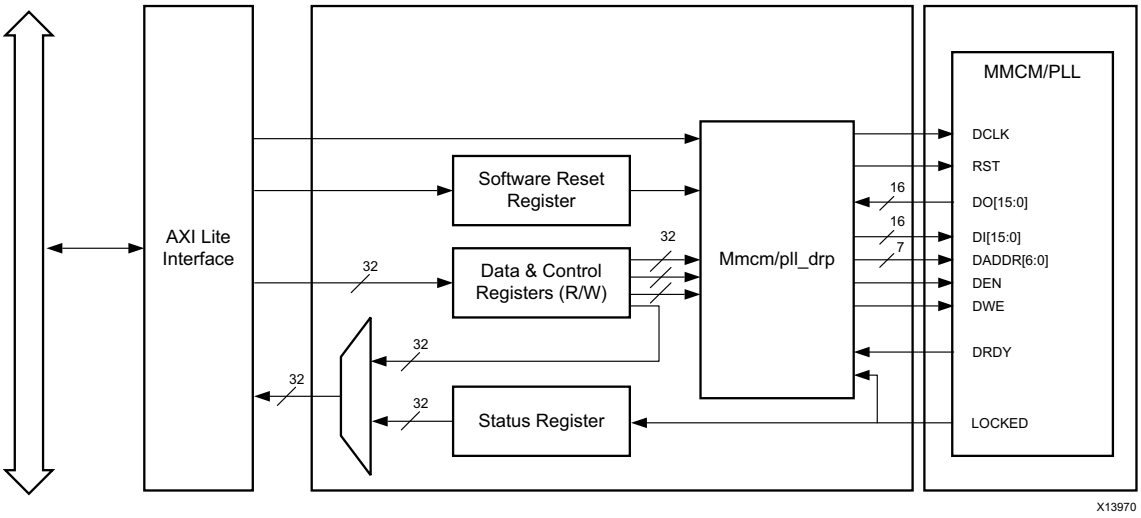

Figure 4-17: **Dynamic Reconfiguration Using AXI4-Lite Interface**

Example for Dynamic Reconfiguration through AXI4-Lite

Note: For Clocking Wizard v5.2 and earlier, you need to write 0x00000007 followed by 0x00000002 into Clock Configuration Register 23, to consolidate the redundant bits that the IP has upgraded. Now you can initiate the transaction by writing 0x00000003 to the clock configuration register 23, but the backward compatibility still exists.

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

1. Configure 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 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` gives 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 Clock Configuration Register 23 (Address: C_BASEADDR + 0x25C) with 0x00000003 to set the `LOAD` and `SEN` bits.
4. Wait for the locked signal. The new frequency can be checked at the `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 0x00000001.

Refer to Chapter 5, Example Design for more details.

Example for Dynamic Reconfiguration Using Write to DRP

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

1. The output clock frequency needs to be reconfigured to 50 MHz with a phase shift of 90°, as shown below:

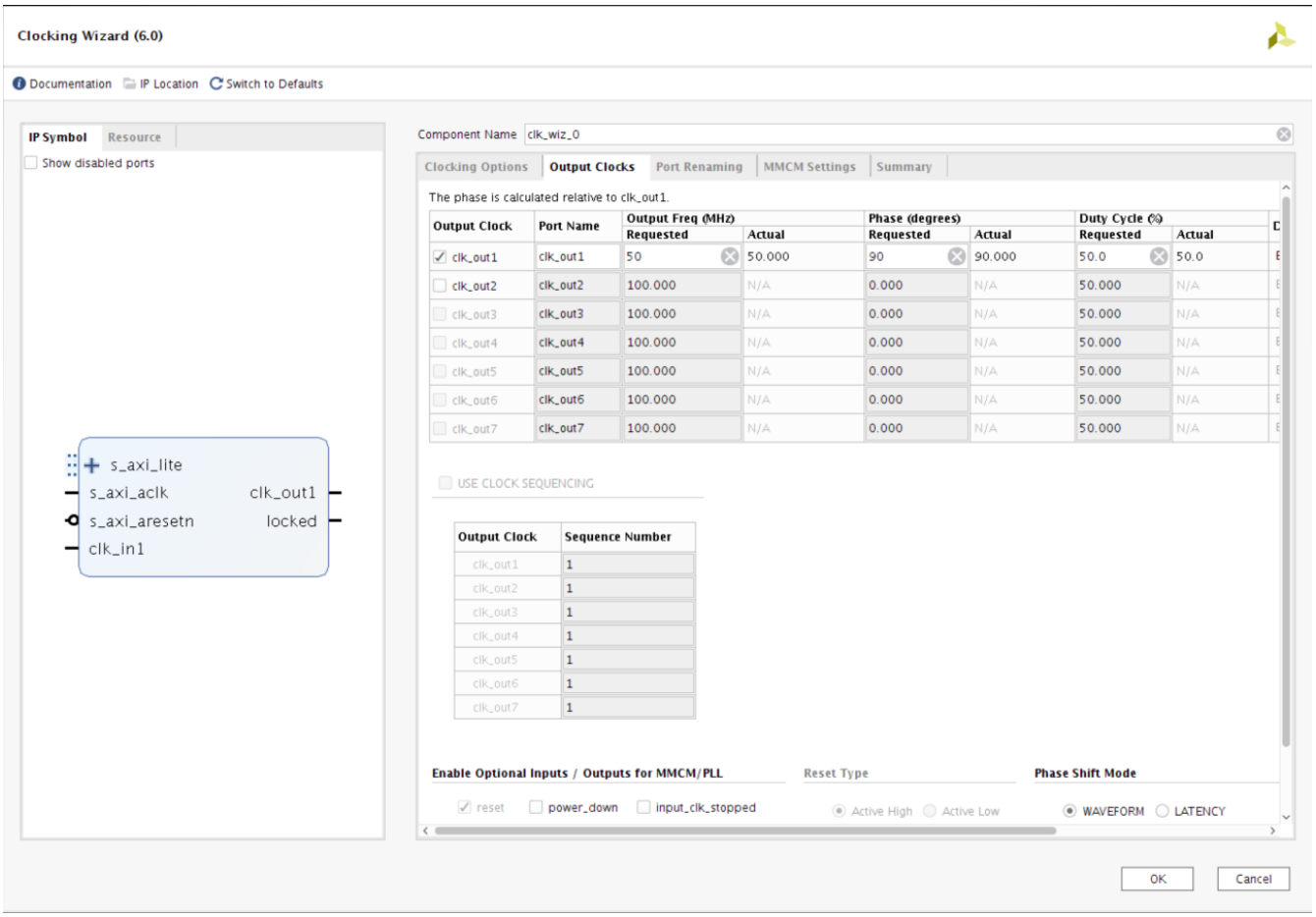

Figure 4-18: **Output Clocks Tab**

2. The table in the **DRP Registers** tab gets updated for the user requirements on the clock, as shown below:

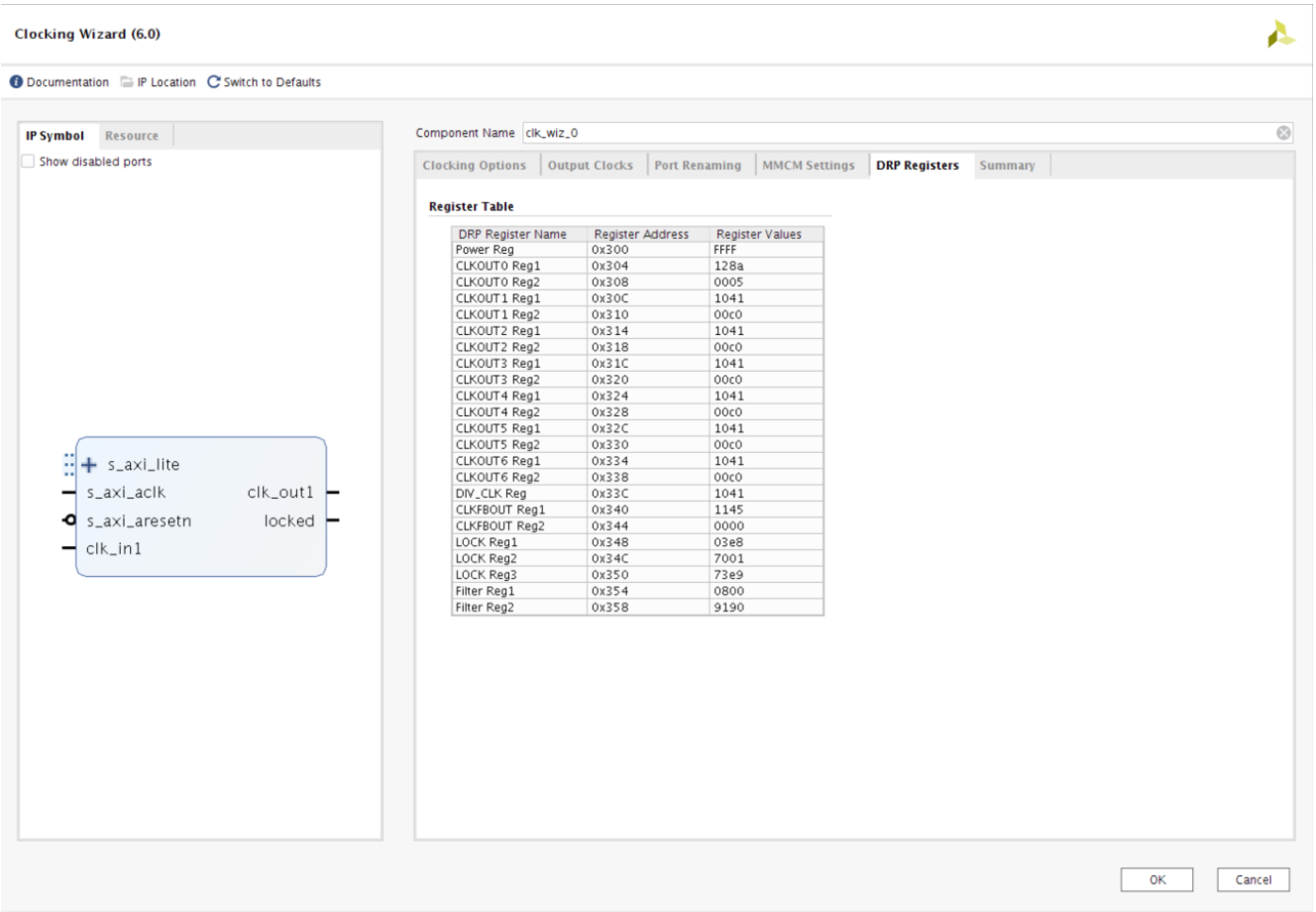

Figure 4-19: **DRP Registers Tab**

3. The table specifies all the AXI registers with the address and the data which needs to be written into it. Configure all the AXI Registers with respect to the table.
4. Configure Clock Configuration Register 24 (Address: C_BASEADDR + 0x35C) with 0x00000003 to set the `LOAD` and `SEN` bits.
5. Wait for the locked signal. The new frequency can be checked at the `clkout1` output port.

IMPORTANT: *When Dynamic Reconfiguration is selected using the DRP interface, refer to MMCM and PLL Dynamic Reconfiguration (XAPP888) [Ref 6] for the steps to reconfigure the clocking primitive. You can use the MMCM register details to configure different M and D values.*

Output Generation

For details, see 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. The following command signifies the setting of +/- 100 ps (= 0.1 nsec) peak to peak jitter on the primary clock port propagating through input port clk_in1.

```
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, meaning that core-level XDC constraints are applied first and are then 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 the 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 the Vivado design tools. If you open an example project, the simulation scripts are generated in the following working directory:

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

You can run fast simulation using the `unifast_ver` or `unifast` libraries of `MMCME2_ADV` and `PLLE2_ADV`. This results in a hundred-fold improvement in simulation run time.

Simulation Waveforms for the Safe Clock Startup Feature

Simulation when Safe Clock Startup is true is illustrated in Figure 4-20.

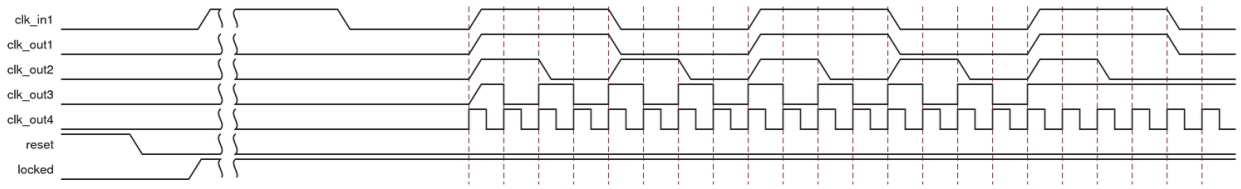

Figure 4-20: **Simulation When Safe Clock Startup is True**

Figure 4-21 illustrates simulation when Safe Clock Startup is true and Use Clock Sequencing is true, with the required sequence number in the table.

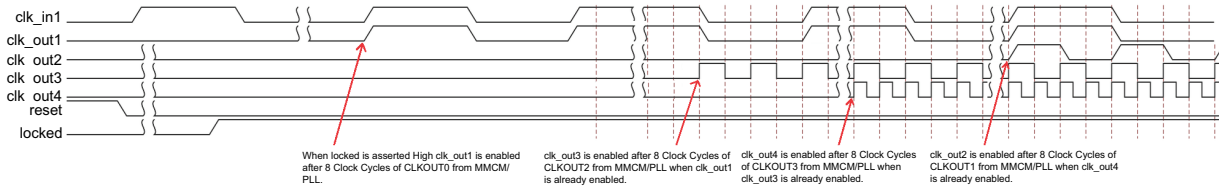

Figure 4-21: **Simulation when Safe Clock Startup is True and Use Clock Sequencing is True**

IMPORTANT: *For cores targeting 7 series or Zynq®-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

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].

Example Design

In the Vivado® design tools, the `open_example_project [get_ips <component_name>]` parameter in the 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 the example project.

Directory and File Contents

The `open_example_project [get_ips <component_name>]` parameter creates an `example_project` directory in the working area. The example design contains the counters on all the output clocks and the MSBs of these counters are used as output to observe on LEDs on board.

Example Design

The following file describes the example design for the Clocking Wizard core.

- 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.

Note: Example design files are delivered only in Verilog.

Test Bench

This chapter contains information about the provided test bench in the Vivado® Design Suite environment. The following file describes the demonstration test bench:

- Verilog

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

The demonstration test bench is a simple Verilog program designed to exercise the example design and the core. It performs frequency calculations as well as checks of all the output clocks. It reports all the output clock frequencies, and if any of the output clocks is not generating the required frequency, it reports an error.

Note: Test bench files are delivered only in Verilog.

AMD XILINX

Appendix A

Verification, Compliance, and Interoperability

Simulation

Verified with all the supported simulators.

Hardware Testing

Hardware testing is performed for all the features on the Kintex®-7 KC705 Evaluation Kit using the provided example design. Hardware testing for the Clock Monitor feature is performed on the Kintex UltraScale™ KCU105 Evaluation Kit.

Upgrading

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. In 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 in the relevant clocking resources User Guide at the Xilinx Support web page.

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 do 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. In the new Wizard, the general flow is as follows:

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 your 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 2-1.

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 contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

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 or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. 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 you 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.

Master Answer Record for the Clocking Wizard core

AR: 54102

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if the you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

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 Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows the 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 Vivado debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature 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 you use 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.

Debug for Dynamic Reconfiguration

If the configuration of your Clocking Wizard is not as per the required dynamic clock, ensure to verify the following:

- The VCO frequency falls into the valid range for the given multiply and the divide values.
- The fractional enable bit must be set only for the non-integer values.

Note: The VCO range varies based on the device selected.

Safe Clock Startup Timing Failures for UltraScale and UltraScale+ Devices

For UltraScale™ or UltraScale+™ devices, when the Safe Clock Startup feature is enabled and the clocks are operating at more than 300 MHz frequency, the design might not meet timing. Use the following steps to meet the timing during implementation:

1. Add an additional BUFGCE cascaded in the failing clock path along with the existing BUFGCE in safe clock startup path. Figure C-1 shows the failing path for the clk_out1. Insert New BUFGCE between MMCM and the BUFGCE2 as shown below.

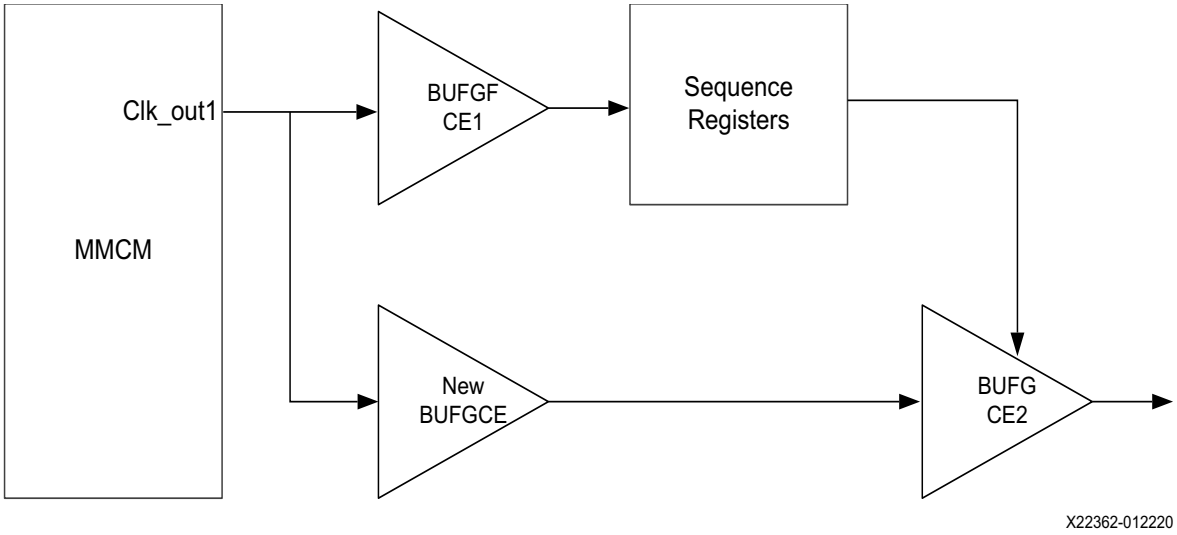

Figure C-1: **Safe Clock Startup Failing Path**

2. Apply the DONT_TOUCH constraint to the newly added BUFGCE. The DONT_TOUCH is necessary to prevent opt_design from removing the newly inserted cascaded BUFGCE.

```
(* dont_touch = "true" *) wire <<signal_name>>;
```

3. Set CLOCK_DELAY_GROUP and USER_CLOCK_ROOT between the newly added cascaded BUFGCE and the BUFGCE for the safe startup circuit to ensure that they are balanced.

```
set_property CLOCK_DELAY_GROUP group_bufgce [get_nets <<BUFG_CE1_net>>]
set_property CLOCK_DELAY_GROUP group_bufgce [get_nets <<new_BUFGCE_net>>]
set_property CLOCK_REGION <<CLOCK_REGION_XX_YY>> [get_cells <<BUFG_CE1>>]
set_property CLOCK_REGION <<CLOCK_REGION_XX_YY>> [get_cells <<BUFG_CE2>>]
set_property CLOCK_REGION <<CLOCK_REGION_XX_YY>> [get_cells <<new_BUFGCE>>]
```

CLOCK_REGION is needed to place the cascaded BUFGCE buffers in the same clock region to reduce the delay between them.

Note: <CLOCK_REGION_XX_YY> is the CLOCK_REGION where the MMCM resides and should be in the same region as the MMCM clock input pin.

AMD XILINX

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product 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)
10. *Virtex-7 T and XT FPGAs Data Sheet* (DS183)
11. *Kintex-7 FPGAs Data Sheet* (DS182)
12. *Kintex UltraScale FPGAs Data Sheet* (DS892)
13. *Virtex UltraScale FPGAs Data Sheet* (DS893)
14. *Zynq UltraScale+ MPSoC Data Sheet* (DS925)
15. *Kintex UltraScale+ FPGAs Data Sheet* (DS922)
16. *Artix-7 FPGAs Data Sheet* (DS181)
17. *Artix UltraScale+ FPGA Data Sheet* (DS931)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/20/2022	6.0	• Added the Artix UltraScale and UltraScale+ Data sheets support in Maximum Frequencies. • Updated the content in the MMCM Counter Cascading section. • Updated the note in the Example for Dynamic Reconfiguration through AXI4-Lite section.
08/06/2021	6.0	• Updated output clock ports in Table 2-1. • Added note to Clock Out of Range. • Added Optimal Clocking Structure. • Added bullet to Selecting Clocking Features. • Updated note about fractional divide values to Configuring Output Clocks. • Added note to Debug for Dynamic Reconfiguration.
03/18/2021	6.0	• Added note to Table 2-1 and Table 2-2. • Updated the number of clock cycles in the Clock Stop section. • Added content to the Dynamic Reconfiguration through AXI4-Lite section. • Added a description to the command line in the Required Constraints section. • Updated settings in step 3 of Safe Clock Startup Timing Failures for UltraScale and UltraScale+ Devices.
02/05/2020	6.0	• Updated IP Facts. • Updated binary in note in Example for Dynamic Reconfiguration through AXI4-Lite.
02/25/2019	6.0	• Added note explaining that the Port Renaming tab has been removed in Selecting Clocking Features. • Added note about OVERRIDE_PRIMITIVE parameter in Overriding Calculated Parameters. • Added note about Dynamic Reconfiguration in the DRP interface in Example for Dynamic Reconfiguration Using Write to DRP.
04/04/2018	6.0	• Updated for core version • Updated phase alignment feature in Selecting Clocking Features • Updated Configuring Output Clocks section in Chapter 4, Design Flow Steps
10/04/2017	5.4	• Added Wiki link of clock framework for Linux Operating System • Updated for minor corrections
04/05/2017	5.4	• Updated for core version • Updated IP GUI screens in Chapter 4, Design Flow Steps
10/05/2016	5.3	• Added a special feature AUTO in primitive selection. • Added Auto Inference section in Chapter 4, Design Flow Steps.
06/08/2016	5.3	• Added a sub section heading for the register space. • Segregated the Clocking Wizard I/O table contents.

Date	Version	Revision
04/06/2016	5.3	• Updated for core version • Added support for clock monitoring. • Added the write DRP feature for non-utilization of DSP resources in dynamic reconfiguration. • Consolidated the redundant bits of clock configuration register 23.
11/18/2015	5.2	• Added support for UltraScale+ architecture-based devices.
09/30/2015	5.2	• Updated for core version and IP GUI screens • Removed the VHDL support for example design and simulation files • Removed the unsupported Port-Renaming Tab from GUI in IP Integrator
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

SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.