

UG0586
User Guide
RTG4 FPGA Clocking Resources



a  **MICROCHIP** company



a  MICROCHIP company

Microsemi Headquarters

One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113

Outside the USA: +1 (949) 380-6100

Sales: +1 (949) 380-6136

Fax: +1 (949) 215-4996

Email: sales.support@microsemi.com

www.microsemi.com

©2021 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

1	Revision History	1
1.1	Revision 11.0	1
1.2	Revision 10.0	1
1.3	Revision 9.0	1
1.4	Revision 8.0	1
1.5	Revision 7.0	1
1.6	Revision 6.0	1
1.7	Revision 5.0	2
1.8	Revision 4.0	2
1.9	Revision 3.0	2
1.10	Revision 2.0	2
1.11	Revision 1.0	2
2	Clocking Resources Overview	3
2.1	Introduction	3
2.2	Clocking Scheme Overview	4
3	FPGA Fabric Global Network Architecture	6
3.1	Introduction	6
3.2	Global Network Architecture	6
3.2.1	SerDes Output Clocks	8
3.2.2	Dedicated Global I/Os	8
3.2.3	Fabric CCC	10
3.2.4	Global Buffer (GB)	11
3.2.5	Row Global Buffers (RGB)	13
3.3	Design Recommendations	13
3.3.1	Global Macros	13
3.3.2	Managing Global Signals	14
3.3.3	Place and Route	17
3.3.4	Global Net Clock Jitter	19
3.4	SmartPower Usage Tips	24
4	On-Chip Oscillator	26
4.1	Introduction	26
4.2	Functional Description	26
4.3	How to Use On-Chip Oscillator	27
5	Fabric Clock Conditioning Circuitry	28
5.1	Introduction	28
5.1.1	Features	28
5.2	System-Level Block Diagram	29
5.3	Fabric CCC Locations	30
5.4	Functional Description	30
5.4.1	Fabric CCC Output Clocks	34
5.4.2	Fabric CCC Clock Sources	34
5.4.3	Fabric PLL Circuitry	37
5.4.4	GPMUX	40
5.4.5	GPD	41
5.4.6	SRG	42

5.4.7	Clock Gating Logic (CGL)	44
5.4.8	RX Clock Recovery (SpaceWire)	46
5.5	How to Use SpaceWire Mode and Glitch Filter Mode using RX Clock Recovery Block	47
5.6	Guidelines for using CCC with Simultaneous Switching Registers	48
5.7	Fabric CCC Configuration	49
5.7.1	Fabric CCC Static Configuration	49
5.7.2	Fabric CCC Dynamic Configuration	49
5.8	Fabric CCC Configuration Registers	49
5.8.1	Fabric CCC Configuration Registers Bit Definitions	51
5.8.2	Fabric CCCs Multiplexers Selection Control	58
5.9	How to Use Fabric CCC(s)	60
5.9.1	Basic	60
5.9.2	Advanced Options	60
5.9.3	PLL Options	62
5.9.4	PLL/CCC Configuration Register Report	64
5.9.5	Using RTG4 FCCC with Enhanced PLL Calibration	66
5.9.6	Simulation Support	77

Figures

Figure 1	Clocking Scheme Overview	4
Figure 2	Global Signal Routing	7
Figure 3	Global Network Architecture for RT4G150 Devices	8
Figure 4	Dedicated Global I/Os (SouthWest corner) Assignment in and RT4G150 Devices	9
Figure 5	Various Sources Feeding Global Buffers	11
Figure 6	Row Global Signals Driving Clusters	13
Figure 7	Synthesize Options Dialog Box	15
Figure 8	Example of Global Net Report 1	17
Figure 9	Example of Global Net Report 2	18
Figure 10	SmartPower Window	24
Figure 11	Switching Frequencies	24
Figure 12	RTG4 On-Chip Oscillator Clock Sourcing Capabilities	26
Figure 13	50 MHz RC Oscillator and Fabric CCC Connectivity	27
Figure 14	Fabric CCC System-Level Block Diagram	29
Figure 15	Fabric CCC Locations in the RTG4 Devices	30
Figure 16	Fabric CCC Block Diagram	31
Figure 17	Fabric PLL Circuitry	37
Figure 18	GPMUX Input Clock Sources	40
Figure 19	GPD Block Diagram	41
Figure 20	Assertion of GPD Synchronous Reset – GPD_SYNCRST_N	42
Figure 21	Releasing of GPD Synchronous Reset – GPD_SYNCRST_N	42
Figure 22	GPD_SYNCRST_N Released after PLL Locks	43
Figure 23	GPD Output Resynchronization after PLL Locks	44
Figure 24	CGLMUX Block Diagram	45
Figure 25	CGL Circuit	45
Figure 26	CGL Operation Waveform	45
Figure 27	SpaceWire Rx Clock Recovery Block Diagram	46
Figure 28	SpaceWire Rx Clock Recovery Waveform	46
Figure 29	SpaceWire	47
Figure 30	Glitch Filter	48
Figure 31	Advanced Options	61
Figure 32	PLL Options	62
Figure 33	Design Window	64
Figure 34	PLL/CCC Configuration Report File	65
Figure 35	CCC Registers Configuration Report	65
Figure 36	Selecting CCC Being Instantiated in Enhanced PLL Calibration Core	67
Figure 37	Package Pin Number Shown on Advanced Configuration Tab per CCC	68
Figure 38	Graphical Example of Generated RTG4FCCC with Enhanced PLL Calibration	68
Figure 39	HDL Module Hierarchy for RTG4 FCCC with Enhanced PLL Calibration	70
Figure 40	CorePLL_ELOCK Block Diagram	70
Figure 41	RTG4 FCCC with Enhanced PLL Calibration Configuration Report	73
Figure 42	Example SmartDesign Canvas with the HDL+ Component	77
Figure 43	Output Clocks Settings	77
Figure 44	Fabric CCC Reference Clock Selection	78
Figure 45	Fabric CCC and On-chip Oscillator Connectivity	78
Figure 46	Fabric CCC Configuration	79
Figure 47	Dedicated Global I/O Selection	79
Figure 48	PLL Auto-Reset Circuit	80
Figure 49	PLL Auto-Reset Circuit HDL Files	80
Figure 50	RTG4 FCCC with Enhanced PLL Calibration Configurator	81
Figure 51	FCCC Calibration SmartDesign	81

Tables

Table 1	Maximum Clocking Resources for RTG4 Family Devices	3
Table 2	Maximum Global Resources for RTG4 Devices	6
Table 3	Global Buffers Assignment to Fabric CCC Global Outputs	12
Table 4	Global Buffers Assignment to SerDes Blocks	13
Table 5	Global Macros	14
Table 6	Effective Device Flip-Flop Toggle Rate	20
Table 7	Estimation of Worst-Case Clock Jitter for a - 1 Speed Grade RTG4 Design	21
Table 8	Fabric CCC Port Description	31
Table 9	Dedicated Global I/Os Connections to the Fabric CCC/PLLs	35
Table 10	Control Signals for PLL Output and Power State	40
Table 11	Fabric CCC Register Map	49
Table 12	FCCC_RFMUX_CR	51
Table 13	FCCC_RFDIV_CR	51
Table 14	FCCC_FBMUX_CR	51
Table 15	FCCC_FBDIV_CR	51
Table 16	FCCC_CGLMUX0_CR	51
Table 17	FCCC_CGLMUX1_CR	52
Table 18	FCCC_CGLMUX2_CR	52
Table 19	FCCC_CGLMUX3_CR	52
Table 20	FCCC_GPMUX0_CR	52
Table 21	FCCC_GPMUX1_CR	53
Table 22	FCCC_GPMUX2_CR	53
Table 23	FCCC_GPMUX3_CR	53
Table 24	FCCC_GPD0_CR	53
Table 25	FCCC_GPD1_CR	54
Table 26	FCCC_GPD2_CR	54
Table 27	FCCC_GPD3_CR	54
Table 28	FCCC_PLL_CR0	54
Table 29	FCCC_PLL_CR1	54
Table 30	FCCC_PLL_CR2	55
Table 31	FCCC_PLL_CR3	55
Table 32	FCCC_GPDS_SYNC_CR	55
Table 33	FCCC_PLL_CR4	55
Table 34	FCCC_PLL_CR5	55
Table 35	FCCC_GPD0_SYNC_CR	56
Table 36	FCCC_PLL_CR6	56
Table 37	FCCC_PLL_CR7	56
Table 38	FCCC_GPD1_SYNC_CR	57
Table 39	FCCC_GPD2_SYNC_CR	57
Table 40	FCCC_GPD3_SYNC_CR	58
Table 41	FCCC_PDLY_CR	58
Table 42	FCCC_RX_RECOVERY_CR	58
Table 43	Fabric CCCs Multiplexers Selection Control	59

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

1.1 Revision 11.0

The following is a summary of the changes in revision 11.0 of this document.

- Information about [Using RTG4 FCCC with Enhanced PLL Calibration](#), page 66 was updated.
- Information about [Enhanced PLL Calibration Fabric IP](#), page 69 was updated.
- Information about [Performing Dynamic CCC Configuration with RTG4FCCCCECALIB](#), page 74 was added.
- Information about PLL_ARST_N port was updated. See [Table 8](#), page 31, [Table 10](#), page 40, and [PLL Options](#), page 62.
- Updated RCLKINT symbol. See [Table 5](#), page 14.

1.2 Revision 10.0

The following is a summary of the changes in revision 10.0 of this document.

- Information about [Global Net Clock Jitter](#), page 19 was updated.
- Information about [SmartPower Usage Tips](#), page 24 was added.
- Information about driving CLK_50MHz pin was updated. See [Use Model 4: PLL in Triple Module Redundant \(TMR\) Mode](#), page 80.
- Information about programmable delay value was corrected. See [RDLY](#) and [FBDLY](#), page 38.

1.3 Revision 9.0

Information about [Global Net Clock Jitter](#), page 19 was added.

1.4 Revision 8.0

The following is a summary of the changes in revision 8.0 of this document.

- Added section [Using RTG4 FCCC with Enhanced PLL Calibration](#), page 66.
- Added section [Selecting CCC Location and CCC Instance\(s\) being Configured](#), page 66.
- Added section [Enhanced PLL Calibration Fabric IP](#), page 69.
- Added section [Selecting CCC GLx Outputs to Ensure Placement Success](#), page 73
- Updated section [How to Use SpaceWire Mode and Glitch Filter Mode using RX Clock Recovery Block](#), page 47
- Updated [Table 28](#), page 54
- Updated section [PLL Options](#), page 62

1.5 Revision 7.0

The following is a summary of the changes in revision 7.0 of this document.

- Information about PLL external feedback was updated. See [PLL Core](#), page 37.
- Information about [Lock Generation Circuit](#), page 39 was updated.
- Information about lock controls was updated. See [PLL Options](#), page 62.
- Information about LOCKWIN signal was updated. See [Table 28](#), page 54.

1.6 Revision 6.0

The following is a summary of the changes in revision 6.0 of this document.

- Information about [GPD](#), page 41 was updated.
- Information about [READY_VDDPLL](#) signal was added. See [PLL Options](#), page 62.
- Information about Triple Module Redundant (TMR) mode was added. See [Use Model 4: PLL in Triple Module Redundant \(TMR\) Mode](#), page 80.

- Information about programmable delay step was updated. See RDLY and FBDLY, page 38.
- Information about GPD Operating Modes, page 43 was updated.
- Information about Clock Gating Logic (CGL), page 44 was updated.
- Information about de-glitching filter was added. See How to Use SpaceWire Mode and Glitch Filter Mode using RX Clock Recovery Block, page 47.

1.7 Revision 5.0

The following is a summary of the changes in revision 5.0 of this document.

- Information about internal and external feedback of PLL was updated. See PLL Core, page 37.
- Information about clock gating logic was updated. See Clock Gating Logic (CGL), page 44.
- Information about global network architecture for RTG4 150 devices was updated. See Figure 3, page 8.

1.8 Revision 4.0

Information about global buffers assignment to SerDes block was added, see Table 4, page 13.

1.9 Revision 3.0

The following is a summary of the changes in revision 3.0 of this document.

- Removed reference to RT4G075 device.
- Updated RX Clock Recovery (SpaceWire), page 46.
- Updated Note in Global Macros, page 13.
- Updated Figure 18, page 40 and Figure 24, page 45.
- Updated Table 43, page 59.
- Information about Design automation was added, see Managing Global Signals, page 14.
- Information about CQ352 package device was added, see CCC/PLL Restrictions for the CQ352 Package Devices, page 36.
- Information about PLL external feedback was updated, see PLL Core, page 37.
- Updated Lock Generation Circuit, page 39.
- Information about clock gating logic was updated, see Clock Gating Logic (CGL), page 44.
- Information about switching registers was added, see Guidelines for using CCC with Simultaneous Switching Registers, page 48.
- Information about how to use SpaceWire mode and glitch filter mode was added, see How to Use SpaceWire Mode and Glitch Filter Mode using RX Clock Recovery Block, page 47.
- Updated Fabric CCC Dynamic Configuration, page 49.
- Updated MUX selection input 8, 9, 10, and 11 in Table 43, page 59.
- Information about PLL/CCC configuration registers was added, see PLL/CCC Configuration Register Report, page 64.

1.10 Revision 2.0

The following is a summary of the changes in revision 2.0 of this document.

- Added How to Use On-Chip Oscillator, page 27.
- Added How to Use SpaceWire Mode and Glitch Filter Mode using RX Clock Recovery Block, page 47 and How to Use Fabric CCC(s), page 60.
- Updated Figure 14, page 29 and Figure 15, page 30.
- Updated the document with FTC inputs (SAR 63141).

1.11 Revision 1.0

Revision 1.0 was the first publication of this document.

2 Clocking Resources Overview

2.1 Introduction

This chapter provides an overview of the RTG4 FPGA clocking resources and device clocking scheme. The following table lists the maximum number of clocking resources available on each RTG4 family device.

Table 1 • Maximum Clocking Resources for RTG4 Family Devices

Resource	RTG4 Device Part Number
	RT4G150
50 MHz RC oscillator ¹	1
Fabric CCCs ²	8
Global buffers	24
Dedicated global I/Os	36
SerDes global clock outputs ³	12

1. Radiation-hardened RC Oscillator.
2. Each CCC has a dedicated radiation-hardened triple redundant PLL for clock synchronization and clock synthesis.
3. SerDes global output clocks are not SET mitigated.

The on-chip RC oscillator is a high-precision hardened 50 MHz embedded RC Oscillator that can be used in conjunction with the on-chip CCCs to generate clocks of varying frequencies and phases. Refer to the [On-Chip Oscillator](#), page 26 for more information.

The RTG4 fabric CCC is radiation hardened and can generate four different clock outputs with a maximum frequency of 400 MHz. Each fabric CCC has a radiation hardened dedicated triple redundant PLL for flexible clocking in the FPGA fabric. In addition, each CCC includes clock gating logic and general purpose clock dividers that are all radiation hardened. Fabric CCCs can also provide a base reference clock to the on-chip hard IP blocks: FDDR and high-speed serial interfaces. Refer to the [Fabric Clock Conditioning Circuitry](#), page 28 for more information.

The RTG4 FPGA fabric offers a low-skew radiation hardened global network that provides effective clock distribution throughout the FPGA fabric and has extensive support for multiple clock domains. The global network is composed of global buffers (GB) to distribute low-skew clock signals or high-fanout nets to all fabric resources. RTG4 devices have a total of 24 GBs where each GB produces 24 global clock signals to the left and 24 global clock signals to the right for a total of 48 half-chip global networks, refer to the preceding table.

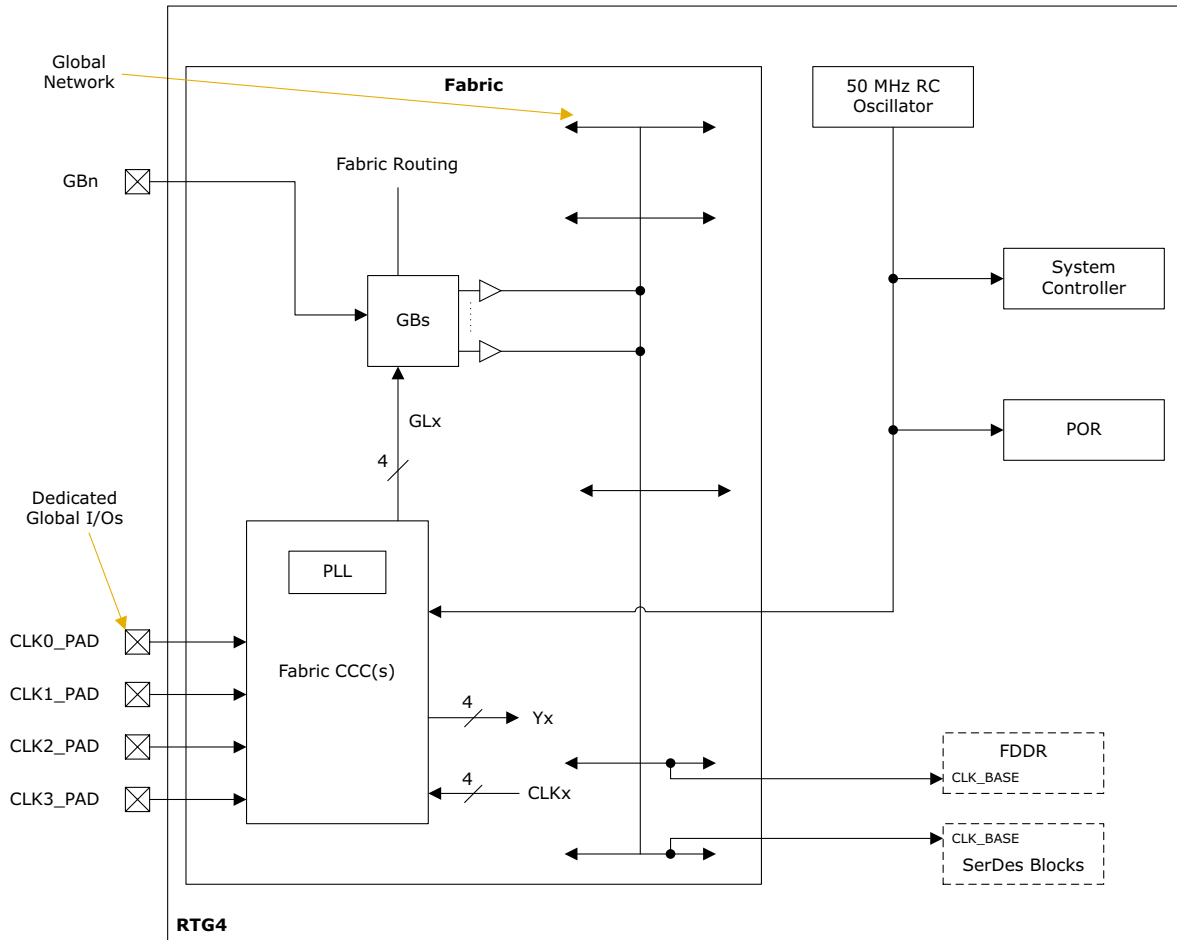
Dedicated global I/Os drive the GBs directly and are the primary source for connecting external clock inputs to the internal global clock network. There are 36 dedicated global I/Os. Refer to the [FPGA Fabric Global Network Architecture](#), page 6 for more information about the global network and dedicated global I/Os.

The Libero SoC design software provides clock management macros for static configuration of the CCCs.

2.2 Clocking Scheme Overview

The following figure shows the top-level RTG4 device clocking scheme. It shows the inputs and outputs for one fabric CCC; each fabric CCC has a similar set of inputs and outputs.

Figure 1 • Clocking Scheme Overview



The GBs in the FPGA fabric distribute global signals to the entire FPGA fabric with low-skew. As shown in the preceding figure, GBs can be driven from multiple sources:

- Dedicated global I/Os
- Fabric CCCs
- FPGA fabric routing

All 36 dedicated global I/Os have direct access to the GBs. Out of the 36 dedicated global I/Os, 32 can reach the fabric CCC. It is also possible to access the GBs from a regular I/O or an FPGA fabric internal signal through the FPGA fabric routing. The four global clock outputs (GLx, x = 0 to 3) of each fabric CCC have radiation-hardened hardwired connection to the GBs. In addition to global clock outputs (GLx), the equivalent four core clocks (Yx, x = 0 to 3) can drive the fabric routing resources in the FPGA fabric.

The on-chip oscillator—50 MHz RC oscillator—has hardwired connections to the System Controller, power-on-reset (POR), and all the fabric CCCs.

Each fabric CCC can have an independent reference clock from one of the following clock sources:

- Four dedicated global I/Os (CLKw_PAD, w = 0 to 3)
- FPGA fabric inputs (CLKx, x = 0 to 3)
- On-chip oscillator

Each fabric CCC has its own dedicated global I/Os. For instance, a dedicated global I/O belongs to the northeast side CCCs is referred to as CCC_NEz_CLKIw, where z represents CCC number and w represents one of the four associated dedicated global I/Os (CLKw_PAD). The dedicated global I/Os have a hardwired connection to the fabric CCCs whereas the FPGA fabric inputs are routed through FPGA fabric routing nets. It must be noted that there is a delay due to the fabric routing that a net goes through before reaching the reference input of a CCC. Refer to the [Dedicated Global I/Os](#), page 8 for more information.

The base clocks (CLK_BASE) to the on-chip hard IP blocks (FDDR and SerDes blocks) must come through the FPGA fabric. For the FDDR, the CLK_BASE is the base clock to the FDDR clock controller. The clock controller generates aligned clocks to all the FDDR sub-blocks for proper operation and synchronous communication with user logic in the FPGA fabric.

For the SerDes block, the CLK_BASE is used for the AXI3 Master and Slave interfaces. It is also used as the reference clock to the SPLL, which is used to achieve interface timing across the fabric to SERDESIF. Each base clock can be generated from any one of the fabric CCCs or a clock source (internal or external) through the global network. The FDDR and SerDes blocks subsystems also have their own clock controllers with a dedicated radiation-hardened PLL for generating the required clocks. Refer to the [UG0573: RTG4 FPGA High Speed DDR Interfaces User Guide](#) and [UG0567: RTG4 FPGA High Speed Serial Interfaces User Guide](#) for more information about FDDR and SERDESIF clocking.

3 FPGA Fabric Global Network Architecture

3.1 Introduction

The RTG4 FPGA fabric offers a low-skew fully SET hardened global network for effective distribution of high-fanout nets including clock signals. The global network has an extensive support for multiple clock domains. This chapter describes the global network architecture and global resources. For information about how to implement asynchronous resets in RTG4 devices, see [UG0741: RTG4 FPGA I/O User Guide](#).

3.2 Global Network Architecture

The RTG4 global network is a tightly coupled, hardwired, radiation hardened, and dedicated routing network between the following global resources:

- Dedicated Global I/Os
- Fabric CCC
- Global Buffer (GB)
- Row Global Buffers (RGB)

The following table lists the maximum global resources available in the RTG4 devices.

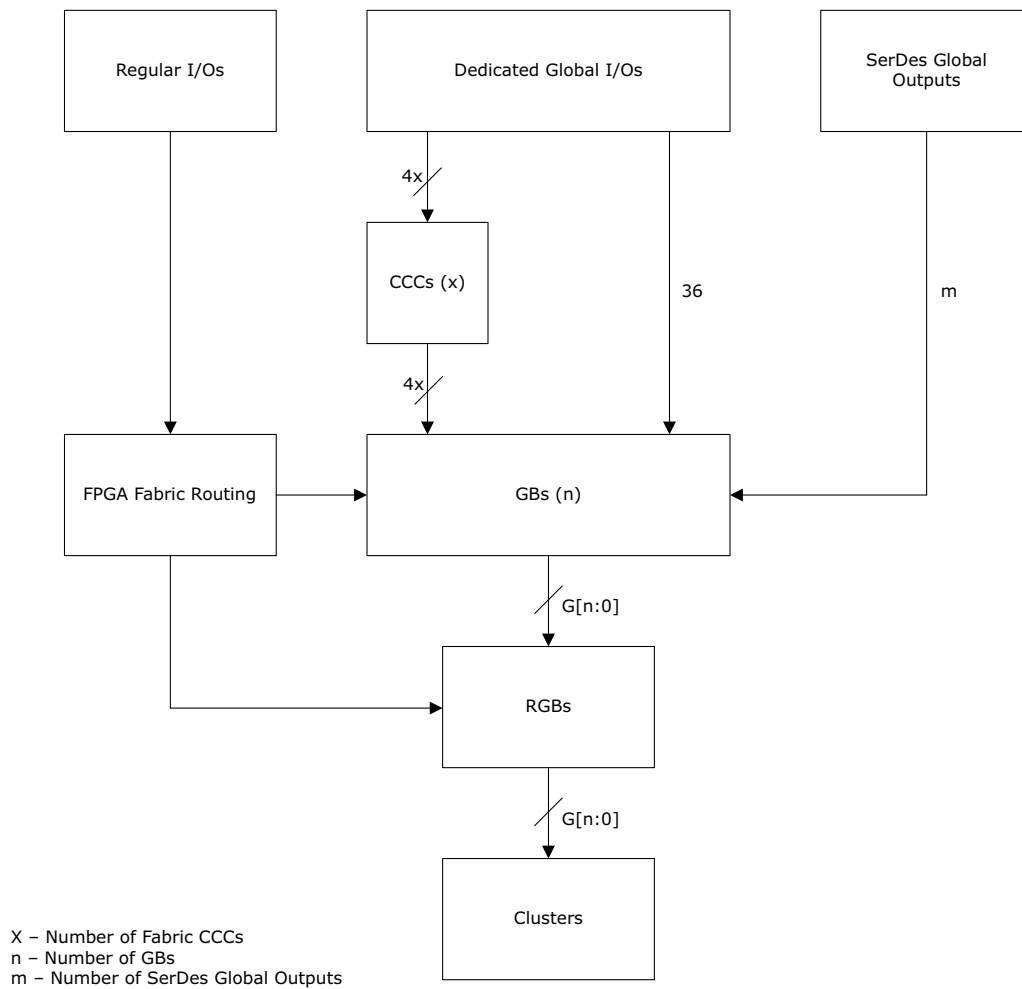
Table 2 • Maximum Global Resources for RTG4 Devices

Resource	RT4G150 Device
Fabric CCCs ¹	8
Global buffers (GB)	24
Dedicated global I/Os	36
SerDes global clock outputs ²	12

1. Each CCC has a dedicated radiation-hardened triple redundant PLL for clock synchronization and clock synthesis.
2. SerDes global output clocks are not SET mitigated.

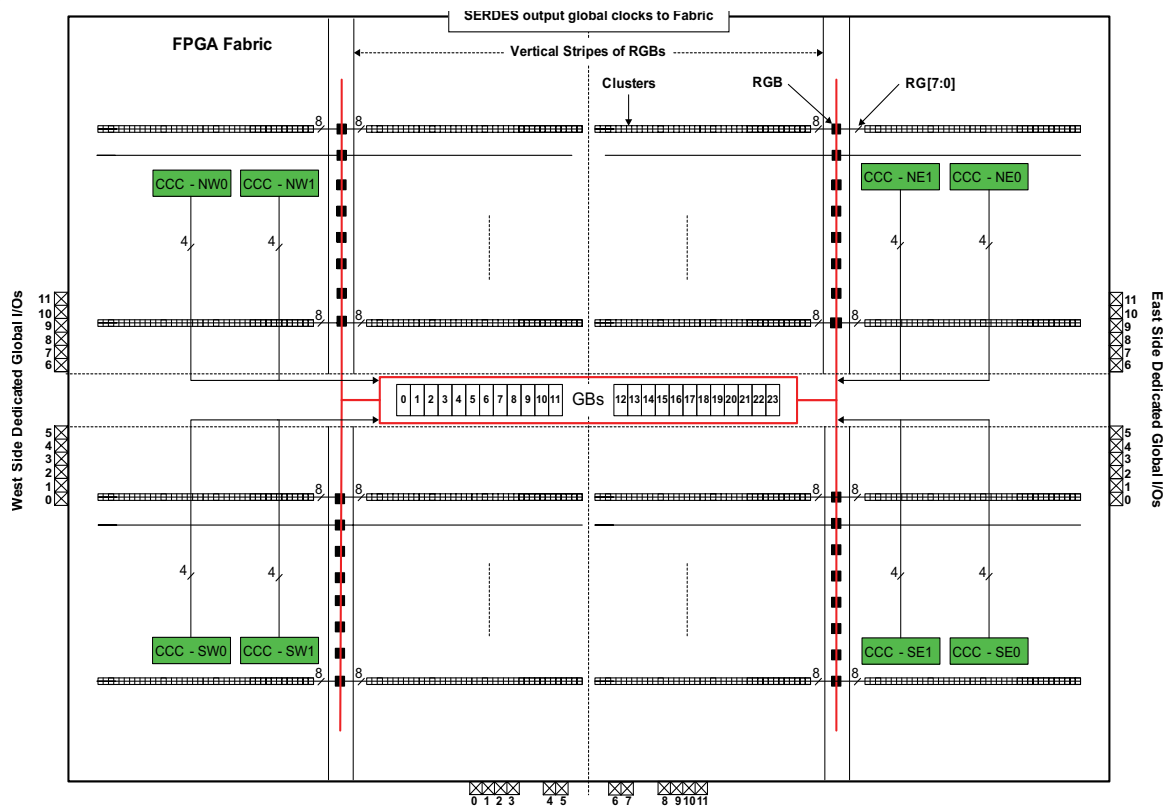
The RTG4 devices use two vertical stripes and 24 GBs (2VS24), eight fabric CCCs (two in each corner of I/O ring), and 36 dedicated external global I/Os global network architecture. The 24 GBs can drive 24 global clocks to the left and 24 global clocks to the right producing a total of 48 half-chip globals or a left and right pair can be used to send a global low-skew clock signal to the entire device.

Figure 2, page 7 shows the global signal routing hierarchy to logic clusters. Refer to the [UG0574: RTG4 FPGA Fabric User Guide](#) for more information on the fabric logic cluster. Global signals reach the logic clusters through row global signals (RGs) generated by an associated row global buffer (RGB). RGBs are located on the vertical stripes and can drive any input of the 4-input LUT and/or any input of the STMR-D flip-flop except the ADn and SD inputs which must be connected to a power net (GND or VCC). RGBs can be accessed from GBs and FPGA fabric routing. GBs can be accessed from dedicated global I/Os, CCC global outputs, SerDes global output, and the FPGA fabric routing. GBs can also be accessed through FPGA fabric routing and from clocks coming from regular I/Os.

Figure 2 • Global Signal Routing

The following figure shows the global network architecture for RTG4 family devices.

Figure 3 • Global Network Architecture for RT4G150 Devices



3.2.1 SerDes Output Clocks

In the RTG4150 device, there are a total of 11 different clocks generated out of each SerDes block when the block is configured in EPCS mode. These clock outputs are available to the fabric for data exchange between the fabric and the SerDes block. The output clocks use the regular routing through the fabric. In this case, these clocks are not SET mitigated. You have an option to select, using the SerDes block configurator in the software, two out of the 11 different clocks to be routed onto a hardwired global clock network that drives into the GB. Those two globally routed clocks will be SET mitigated and will use the dedicated global clock network. For more information on SerDes output clocks and SerDes radiation hardening, refer to the SerDes Block Network in EPCS section in the [UG0567: RTG4 FPGA High Speed Serial Interfaces User Guide](#).

Note: However, the source of these two clocks from the SerDes will not be radiation hardened.

3.2.2 Dedicated Global I/Os

The RTG4 user I/Os are grouped into multi-standard I/Os (MSIO and MSIOD) and DDRIOs. Some of these user I/Os, referred to as dedicated global I/Os, are dual-use I/Os that are capable of driving the global routing network or local routing network. Dedicated global I/Os can be used to bring in external clock signals as inputs to the FPGA fabric. Dedicated global I/Os can be used as regular I/Os, as either input or output for any design signal, if they are not utilized for clocking. Dedicated global I/Os are located on each of the three sides (south, east, and west) of the FPGA fabric.

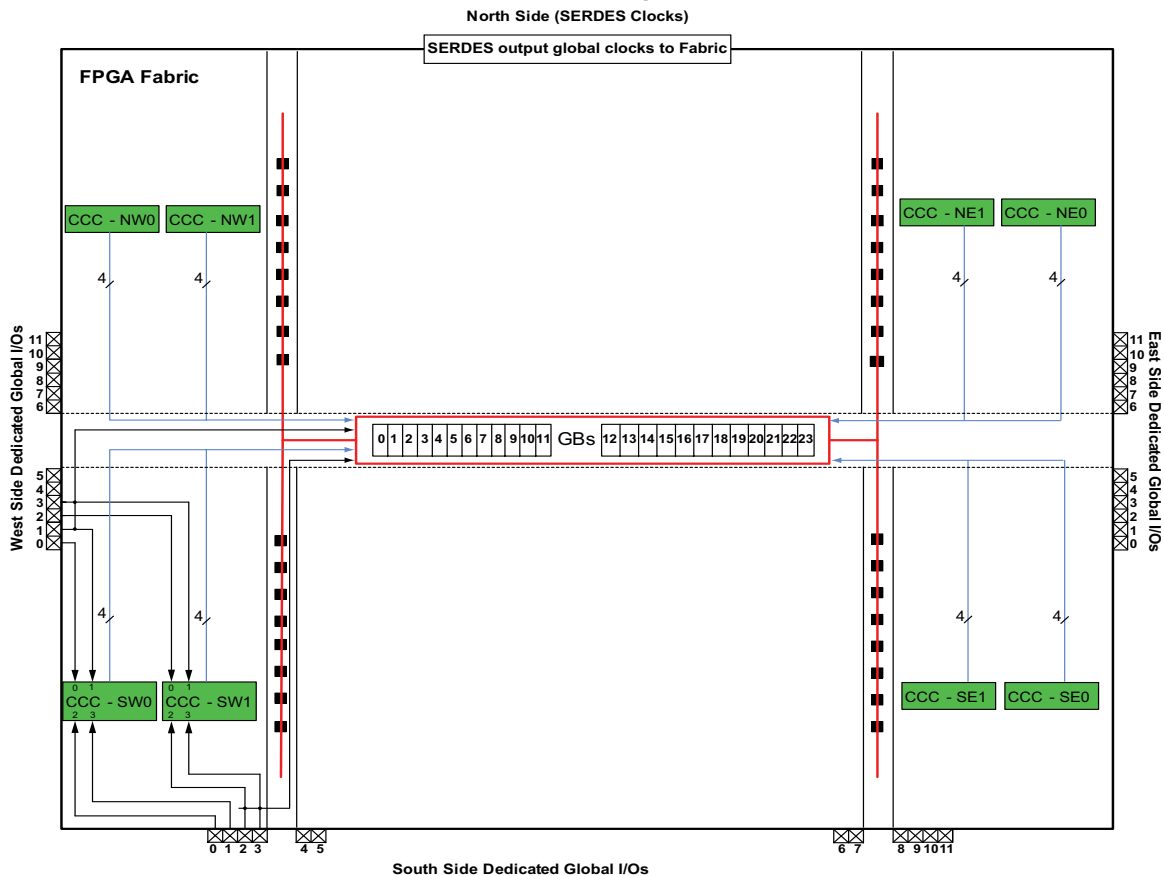
All 36 dedicated global I/Os have direct access to GBs. Out of 36 dedicated global I/Os, there are 32 that can reach the fabric CCC. Each fabric CCC has four dedicated global I/Os as inputs and can drive up to four GBs. Each of the four CCC outputs can drive up to 3 GBs. As such, each CCC can drive up to 12 different GBs. The dedicated global I/Os connect to fabric CCCs and GBs through a hardwired connection. When MSIO or MSIOD input buffers are used to drive the GBs or CCC blocks, the input buffers and associated clock wiring is radiation hardened if the dedicated clock routes are used.

Dedicated global I/Os are routed to GBs by the Libero SoC routing software as follows:

- If a dedicated global I/O, which has direct access to GBs is selected, the routing software directly routes the dedicated global I/O input to an associated GB. There are no routing delays associated with this dedicated connection. If a regular I/O is used instead, then there will be routing delays associated with that connection.
- If a dedicated global I/O, which has direct access to a fabric CCC (CCC macro instantiated in the design with input source selected as dedicated global I/O) is selected, the routing software selects an associated fabric CCC. In this case, the outputs of the fabric CCC reach the associated GBs.

The 2VS24 global network architecture has 36 dedicated global I/Os, which can access GBs directly or through CCCs. The following figure shows the dedicated global I/Os assignment in the lower left corner of the FPGA fabric in RT4G150 devices. The remaining dedicated global I/Os are assigned to CCCs and/or GBs as shown in [Table 3](#), page 12.

Figure 4 • Dedicated Global I/Os (SouthWest corner) Assignment in and RT4G150 Devices



3.2.2.1 Naming Conventions for Dedicated Global I/Os

Due to the comprehensive and flexible nature of dedicated global I/Os, a naming scheme is used to provide the detailed information on each I/O.

The dedicated global I/O uses the generic name **IOxyBz/GBn/CCC_xyz_CLKlw** or **IOxyBz/GBn_n/CCC_xyz_CLKlw**, where:

- IO: Type of I/O—MSIO, MSIOD, or DDRIO
- x: I/O pair number
- y: Differential I/O polarity—P (positive) or N (negative)
- Bz: Bank number
- GBn: Dedicated global I/Os that drive the GBs directly, where n is 0 to 23
- GBn_n: Dedicated global I/O that drives different Gs instead of one direct GB. For example, GB0_11 means that this particular global I/O can drive any one of the GBs 0 to 11.
- CCC_xyz_CLKlw represents the dedicated global I/Os that drive GBs through CCCs, where:
 - xy: Location—NE, SW, SE, or NW
 - z: CCC number—0 or 1
 - l: Clock input
 - w: One of the four dedicated global I/Os associated with each CCC—CLKI0, CLKI1, CLKI2, or CLKI3

Some of the dedicated global I/Os are multiplexed with hard IP blocks such as FDDR. These multiplexed or multi-function dedicated global I/Os act as regular I/Os and cannot be used for accessing the global network or CCCs when the associated hard IP block is enabled in the design. Refer to the [DS0130: RTG4 Pin Descriptions](#) for more information on the functions supported by the dedicated global I/Os. The name of a pin shows the functionalities for which that pin can be configured and used.

Example pin name: **DDRIO94PB0/FDDR_E_ADDR15/GB12_23/CCC_NE0_CLKI3**

The above example pin is a multi-purpose I/O, which is configured as an FDDR I/O (FDDR_E_ADDR15) when the FDDR is enabled. If FDDR is not used in the design, this I/O can be configured as dedicated global I/O which can drive the CCC_NE0_CLKI3 input port of CCC_NE0 or directly drive one of the GB 12 to 23 (GB12_23).

Table 3, page 12 shows the corresponding GB number of each CCC global (GL#) drivers.

3.2.2.2 Dedicated Global I/O Voltage Standards

Dedicated global I/Os are located in different I/O banks with each bank having its own supply and ground pins. The voltage standards supported by dedicated global I/Os are based on the I/O bank it is located in. The voltage standard for a dedicated global I/O can be set using I/O Editor available in the Libero SoC software. Refer to the "Supported Voltage Standards" table in the [UG0574: RTG4 FPGA Fabric User Guide](#) for I/O standards supported by each I/O bank. Dedicated global I/Os can be configured in Single-ended mode or Differential mode. Differential mode is implemented with a fixed I/O pair and cannot be split with adjacent I/Os. According to the naming convention, differential I/O pairs are denoted with an I/O pair number and their polarity (P and N). In Single-ended mode, the I/O pair operates as two independent I/Os. All the configuration and data inputs/outputs are separate and use names ending with P and N to differentiate between the I/Os.

3.2.2.3 Unused Dedicated Global I/O Configuration

Unused dedicated global I/Os behave similarly to unused regular User I/Os. When regular User I/Os (MSIO, MSIOD, DDRIO) are not used, the Libero SoC software configures the I/O as input buffer disabled, output buffer tristated with weak pull-up. For the I/Os unused conduction and recommendations, refer to the [RTG4 CG1657 Package Pin Assignment Table](#).

3.2.3 Fabric CCC

Fabric CCCs enable flexible clocking schemes to the logic implemented in the FPGA fabric, and can also provide the base clock for on-chip hard IP blocks—FDDR and SerDes blocks. Each fabric CCC operates with a dedicated radiation-hardened triple redundant PLL and generates clock signals of varying frequency and phase. Each fabric CCC generates up to four different global clocks (GL0, GL1, GL2, and GL3) and four core clocks (Y0, Y1, Y2, and Y3).

The generated global clocks drive GBs and core clocks drive the local routing resources in the FPGA fabric. The fabric CCCs core clock outputs (Y_x) can be used to drive internal logic without using global network resources. Core clocks (Y_x) introduce additional delay because of FPGA fabric routing. Core clocks are useful when global network resources must be conserved and utilized for other timing-critical paths. The GBs associated with the global clock outputs (GL_x) are available to user logic, if the global clock outputs are disabled. Each fabric CCC has four dedicated global I/Os as inputs. Each fabric CCC output (GL_x/Y_x), as well as the reference clock, can be driven from any one of these four dedicated global I/Os.

Fabric CCCs are labeled according to their location in the FPGA fabric floor plan. For instance, the fabric CCCs located in the northeast corner are labeled CCC-NE0 and CCC-NE1. Refer to the [Fabric Clock Conditioning Circuitry](#), page 28 for more information on fabric CCCs. The entire CCC block is hardened with the exception of the clock inputs from the fabric, delay line, and feedback dividers. RFDIV and FBDIV, which are not hardened.

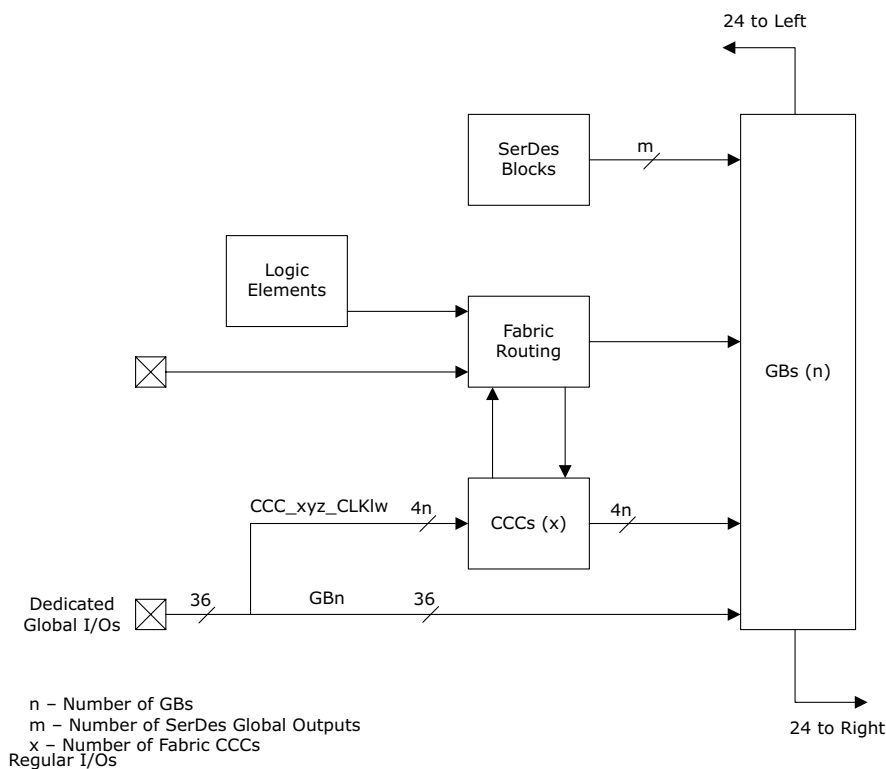
Note: Any SET pulses on the delay lines are filtered by the PLL.

3.2.4 Global Buffer (GB)

The global buffer (GB) is a multiplexer that generates an independent global signal. The global signal can drive all fabric resources with low-skew. The GBs can be driven from multiple sources such as dedicated global I/Os, fabric CCCs, SerDes blocks, and fabric routing. For example, an input signal connected to a dedicated global I/O can route directly to a GB or through a CCC. Input signals connected to regular I/Os or any internal logic module can also be connected to the GB through fabric routing. The following figure shows the sources feeding into GBs which finally feed into RGBs through $G[0-23]$ to span the complete fabric area.

GBs can also be fed through regular I/Os, in which case the signals are first routed to fabric routing and then reach the GBs. Any signal generated from logic modules can reach GBs through fabric routing.

Figure 5 • Various Sources Feeding Global Buffers



The following table shows the assignment of fabric CCC global outputs (GLx, x = 0 to 3) to the global buffers. Each global output of the fabric CCC is associated with three global buffers. For instance, GB0 can be accessed from GL0 of any one of the four fabric CCCs present on the west side.

Table 3 • Global Buffers Assignment to Fabric CCC Global Outputs

CCC-SW0	CCC-SW1	CCC-NW0	CCC-NW1	CCC-SE0	CCC-SE1	CCC-NE0	CCC-NE1	GB Number
GL0								0, 4, 8
	GL0							
		GL0						
			GL0					
GL1								1, 5, 9
	GL1							
		GL1						
			GL1					
GL2								2, 6, 10
	GL2							
		GL2						
			GL2					
GL3								3, 7, 11
	GL3							
		GL3						
			GL3					
				GL0				12, 16, 20
					GL0			
						GL0		
							GL0	
				GL1				13, 17, 21
					GL1			
						GL1		
							GL1	
				GL2				14, 18, 22
					GL2			
						GL2		
							GL2	
				GL3				15, 19, 23
					GL3			
						GL3		
							GL3	

The following table lists the assignment of global buffers to the SerDes blocks.

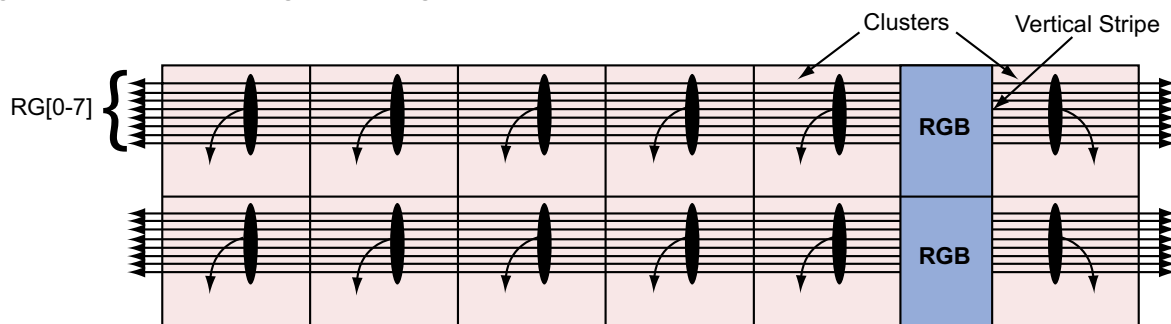
Table 4 • Global Buffers Assignment to SerDes Blocks

SerDes block	GB Number
SERDES_0	0, 2
SERDES_1	4, 6
SERDES_2	8, 10
SERDES_3	12, 14
SERDES_4	16, 18
SERDES_5	20, 22

3.2.5 Row Global Buffers (RGB)

RGBs are situated on the vertical stripes of the global network architecture inside the FPGA fabric. Each RGB drives logic clusters, consisting of 12 logic modules, located on left and right branches using RGs, refer the following figure. The global signals from the GBs are routed to RGBs, which are then fed into the clusters through RGs. Each GB has access to all RGBs available on the vertical stripes as the global network is segmented. Each RGB is independent and can be driven by fabric routing in addition to being driven by GBs. This facilitates the use of RGBs to drive regional clocks spanning a small fabric area. RGBs drive 1/2 the width of the device - either to the centered left or centered right.

Figure 6 • Row Global Signals Driving Clusters



3.3 Design Recommendations

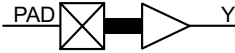


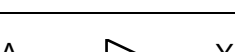
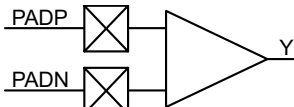
In general, Microchip recommends that all clocks in a design must be routed using the global routing resources in the device. This ensures that clock skew is minimized throughout the design. Furthermore, global resource usage ensures that predictable levels of clock jitter can be accounted for during static timing analysis. This guideline should be applied when selecting clock input pin assignments, when selecting CCC reference clock assignments, and when generating clocks inside the FPGA. The following sections provide recommendations for using the global network in a design.

3.3.1 Global Macros

Global macros can be used for assigning signals to the global network. The following table shows the global macros available for the RTG4 devices. The CLKBUF macro allows you to route the clock coming from the dedicated global I/Os to GBs. Dedicated global I/Os, which have direct access to GBs are available as input pads to these macros. The CLKINT macro route the internal clock signals to GBs through FPGA fabric routing. The RCLKINT macro route the internal clock signals to RGBs through FPGA fabric routing.

Note: Clock nets that are generated and routed in the fabric up to the RCLKINT (RGB) entry point are not radiation hardened. Once the generated nets get to the RGB entry point, from that point forward they are fully hardened. It is fully hardened, if the entry point to the RGB is coming from the GB.

Table 5 • Global Macros

Macro Name	Description	Functional Symbol
CLKBUF	To drive a clock signal coming from input pad to a global buffer.	
CLKINT	To drive a global buffer from FPGA fabric routing.	
CLKINT_PRESERVE	Similar to CLKINT, but the signals routed through the CLKINT_PRESERVE macro will never be demoted or optimized by the Libero Compile tool.	
RCLKINT	To drive a row global net from FPGA fabric.	
CLKBUF_DIFF	To drive a clock signal coming from differential input pad to a global buffer. Use Libero SoC I/O Constraint Editor to select the desired differential standard.	

In addition to these global macros, the CCC macros global outputs (GLx) also drive the GBs. Dedicated global I/Os and their I/O standard can be assigned using **I/O Constraint Editor** in the Libero SoC software. The Libero place-and-route software runs the pre-layout checker and checks the validity of dedicated global I/O assignment.

A regular I/O can be routed to the global network through the FPGA fabric. To allow regular I/O to access the global network, the CLKINT macro can be instantiated or PDC constraints can be used to promote signals from regular I/O or internal signal to the global network.

3.3.2 Managing Global Signals

Assigning high fan-out nets to the global clock network is an effective way of reducing routing congestion and minimizing skew. Due to its high propagation delays, the global clock network is not recommended for use in timing-critical data paths.

The clock macros can be used for assigning signals to the global clock network:

- The CLKBUF and CLKBUF_DIFF macro connects a Dedicated global I/O to GB. Dedicated global I/Os have direct hardwired routing to GBs.
- The CLKINT macro connects fabric routed signal to GB. The CLKINT macro must be used to connect a regular I/O to GB through the FPGA fabric.
- The RCLKINT macro connects a fabric routed signal to RGB.

The CCCs and transceivers drive GBs through hardwired routing.

The Libero SoC software supports automated global buffer allocation to minimize the user intervention. The allocation strategy for global buffers employs the following priority:

- User-inserted global clock macros
- Clock nets
- Asynchronous reset/set nets
- Very high fan-out nets

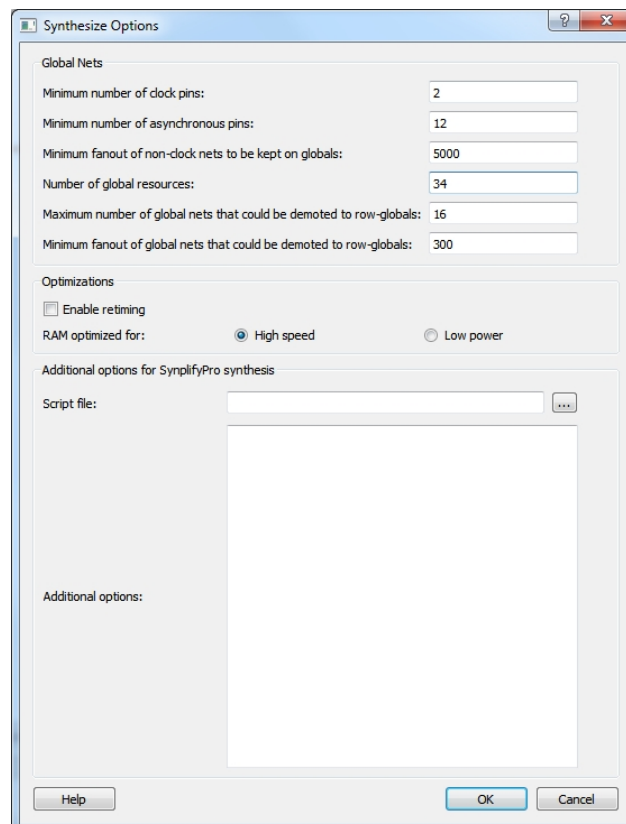
In Libero tool, the default fan-out threshold for global net promotion is larger for data pins (pins involved in register-to-register paths) than asynchronous logic pins (pins involved in register-to-asynchronous paths).

Due to this, the automated design flow is more likely to employ global nets on register-to-asynchronous paths than register-to-register paths. The reasoning for this is that asynchronous pins are not normally timing-critical, and routing them on global nets reduce routing congestion. However, register-to-asynchronous paths are functionally equivalent to register-to-register paths from the perspective of achieving timing closure. As a result, when designing register-to-asynchronous paths, ensure that timing critical connections do not unnecessarily employ global nets.

If a design contains a failing register-to-asynchronous timing path, check if the path drives a global net in SmartTime. This is done by examining the path and looking for a GB between its launching and latching registers. If a GB is present, you may be able to improve the likelihood of timing closure by demoting the net to a fabric-routed net. An asynchronous net can be demoted by increasing the fan-out threshold of asynchronous pins above the fan-out of the asynchronous net. Alternatively, you can manually adjust the RTL by moving timing-critical pins from high-fanout asynchronous nets to lower fan-out nets.

Users have the option of setting the minimum fan-out for automatic global assignment. The fan-out threshold values are set in the Libero tool to automate clock pin promotion to global nets. In the Libero Design Flow window, expand Implement Design, right-click Synthesize, and choose Configure Options. This opens the Synthesize Options dialog box, as shown in the following figure.

Figure 7 • Synthesize Options Dialog Box



The following options specify the threshold value for nets promotion or demotion:

- **Minimum number of clock pins** – Specifies the fan-out threshold value for clock pin promotion. The default value is 2.
- **Minimum number of asynchronous pins** – Specifies the fan-out threshold value for Asynchronous pin promotion. The default value is 12. This feature is not available in the RT4G150_ES device.
- **Minimum fan-out of non-clock nets to be kept on globals** – Specifies the fan-out threshold value for data pin promotion to global resources. It is the minimum fan-out of non-clock (data) nets to be kept on global nets (no demotion). The default value is 5,000 (must be between 1,000 and 200,000). If you run out of global resources for your design, increase this number. If a CLKINT net with fan-out less than this threshold value has data pins along with some clock or asynchronous reset/set pin, move all the data pins to the CLKINT driver net.
- **Number of global resources** – Specifies the number of global resources to be used in the design. The default value is 34 and you could increase its value up to 48.
- **Maximum number of global nets that could be demoted to row-globals** – Specifies the maximum number of global nets that could be demoted to RGB resources. The default value is 16.

Minimum fan-out of global nets that could be demoted to row-globals – Specifies the minimum fan-out of global nets that could be demoted to RGB resources. The default value is 300.

It is undesirable to have high fan-out clock nets demoted using RGB resources because it may result in high skew. If you run out of global resources for your design, reduce this number to allow more globals to be demoted to RGB resources.

Note: Hardwired connections to global resources, such as connections from CCCs and Dedicated Global I/Os cannot be controlled by synthesizer options.

After synthesis, the compiler tool performs the following steps to assign nets to global buffers:

1. Sorting all CLKINT nets in the following priority order.
 - Fan-out, only if fan-out \geq threshold value specified by minimum fanout of non-clock nets to be kept on globals
 - Number of clock pins
 - Number of asynchronous reset/set pins
 - Number of data pins
2. Determining the number of GB resources available for CLKINT nets after allocating them to any of the CLKBUF, CLKBUF_DIFF, and CLKINT_PRESERVE.
3. Demoting CLKINT nets from the sorted list that are beyond the limit specified by the number of global resources.
 - If such a net has at least the number of pins specified by minimum fanout of global nets that could be demoted to row-globals, replace the CLKINT with an RCLKINT macro. Limit the number of nets demoted to RCLKINT to the count specified by maximum number of global nets that could be demoted to row-globals.
 - Otherwise, merge the net with the driver of the CLKINT.

The HDL source file or SmartDesign schematic is the preferred place for defining which signals must be assigned to a global network using global macro instantiation. A signal with high-fanout may have logic replication, if it is not promoted to a global during synthesis.

3.3.3 Place and Route

Place and Route step aligns placement of I/Os, CCC, and GB resources while preserving any user locked placement, finding feasible I/O bank technology solution and optimizing the number of hardwired connections to reduce clock insertion delays. An I/O may be hardwired to a GB as well as route directly to data pins in the fabric.

The CLKINT nets are mapped to GB resources and the placement algorithm resolves the GB bandwidth per vertical half. The RCLKINT nets are mapped to RGB resources with routed inputs and the placement algorithm resolves the RGB bandwidth along with RGRESET bandwidth per row.

The power-driven option optimizes the number of rows occupied by a global net or a local clock net, resulting drop in performance (on average).

After performing the place and route, a report of global nets is generated. User can open the report tab to view the global net report. The user should review the generated global net report to understand and minimize clocks that are routed on the FPGA fabric routing instead of global routing resources. The following figures show an example of global net report.

Figure 8 • Example of Global Net Report 1

Global Nets Information

	<u>From</u>	<u>GB Location</u>	<u>Net Name</u>	<u>Fanout</u>
1	GBR[1]	(709, 153)	CLK_18_ibuf_RNI7605/U0_YWn	2266
2	GBR[12]	(720, 153)	CLK_12_ibuf_RNI1837/U0_Ywn	623
3	GBL[19]	(727, 154)	CLK_12_ibuf_RNIVQ04/U0_Y	558
4	GBL[7]	(715, 154)	MTD_0/CCC_0/CCC_INST/INST_CCC_IP:GL0	512
5	GBL[0]	(708, 154)	FCCC_2/CCC_INST/INST_CCC_IP:GL0	256
6	GBL[3]	(711, 154)	FCCC_2/CCC_INST/INST_CCC_IP:GL1	128
7	GBL[11]	(719, 154)	I7/I0/I4/vcxclk_1_5m	64
8	GBL[2]	(710, 154)	I7/I0/I4/clk_3_088m_0	32

I/O to GB Connections

	<u>Port Name</u>	<u>Pin #</u>	<u>I/O Function</u>	<u>From</u>	<u>From Location</u>	<u>To</u>
1	CLK_18	K5	MSIOD119PB7/GB1	CLK_18_ibuf/U0/U_IOIN:Y	West IO #2(0, 58)	GBR[1]
2	CLK_12	A17	DDRIO76PB0/GB12	CLK_12_ibuf/U0/U_IOPAD:Y	North IO #6(564, 205)	GBR[12]
3	CLK	F27	MSIO335PB4/GB19	CLK_ibuf/U0/U_IOPAD:Y	South IO #9(1134, 1)	GBL[19]

Fabric to GB Connections

	<u>From</u>	<u>From Location</u>	<u>To</u>	<u>Net name</u>	<u>Net type</u>	<u>Fanout</u>
1	I7/I0/I4/vcxclk_1_5m:Q	(361, 61)	GBR[11]	I7/I0/I4/vcxclk_1_5m	ROUTED	2
2	I7/I0/I4/clk_3_088m:Q	(368, 61)	GBR[7]	I7/I0/I4/clk_3_088m_0	ROUTED	3

Figure 9 • Example of Global Net Report 2CCC to GB Connections

	From	From Location	To	Net Name	Net Type	Fanout
1	MTD_0/CCC_0/CCC_INST/INST_CCC_IP:GL0	CCC-NW0 (0, 254)	GBL[7]	MTD_0/CCC_0/GL0_net	HARDWIRED	1
2	FCCC_2/CCC_INST/INST_CCC_IP:GL0	CCC-NW1 (18, 254)	GBL[0]	FCCC_2/GL0_net	HARDWIRED	1
3	FCCC_2/CCC_INST/INST_CCC_IP:GL1	CCC-NW1 (18, 254)	GBL[3]	FCCC_2/GL1_net	HARDWIRED	1

CCC Input Connections

	From	From Location	To	CCC Location	Net Name	Net Type	Fanout
1	SERDES_IF2_0/SERDESIF_INST/INST_S ERDESIF_IP:EPCS_RXCLK[1]	(12, 2)	FCCC_2/CCC_INST/INST_CCC_IP:CLK0	CCC-NW1 (18, 254)	SERDES_IF2_0_EPCS_3_RX_CLK	ROUTED	1
2	SERDES_IF2_0/SERDESIF_INST/INST_S ERDESIF_IP:EPCS_RXCLK_1	(12, 2)	FCCC_0/CCC_INST/INST_CCC_IP:CLK0	CCC-NE1 (1014, 254)	SERDES_IF2_0_EPCS_1_RX_CLK	ROUTED	1

Local Clock Nets to RGB Connections

Port Name	Pin Number	I/O Function	From	From Location	Net Name	Fanout	RGB Location	Local Fanout
1 -	-	-	SYSCLK/PROC_IF/reset_ctl_1/reset_bits_RNIP_LVN[0]:Y	(229, 36)	SYSCLK/PROC_IF/reset_ctl_1/reset_bits_RNIP_LVN[0]	37	(219, 3)	2

3.3.4 Global Net Clock Jitter

The RTG4 global network is hardened against Single Event Transients (SETs). The global network employs glitch filtering and physical triplication to mitigate radiation induced clock glitches. The glitch filtering slows the internal clock edge rates to boost SET immunity. However, a consequence of this radiation mitigation is reduced noise tolerance of the global network.

Disturbances in the RTG4 Power Delivery Network (PDN) can contribute to an increase in global network clock jitter.

There are two distinct contributors to increased clock jitter:

- The PDN's inability to meet large transient current demands under extreme design toggle rates. This can occur regardless of the clock frequencies in the user design.
- Design operation with a large amount of logic toggling simultaneously at the PDN resonance frequency, which typically ranges from 10 MHz to 40 MHz, depending on the specific board design.

Depending on the aggressor clock frequency, either one of these two effects at a time can contribute to increased global net clock jitter. The aggressor clock domain can induce jitter on unrelated victim clock domains. Therefore, the victim clock domains must also consider global net clock jitter.

To address the first contributor above, the RTG4 core voltage supply (VDD) must always be maintained within $\pm 5\%$ of 1.2 V at the device package pins, during design operation. The operating voltage specification includes the regulator DC variation plus any power supply ripple over the customer design frequencies, as measured at the device package pins. For information about operating voltage specification, see *DS0131: RTG4 FPGA Datasheet*. The VDD ripple during design operation can be measured for specific designs and boards to ensure that it conforms to the datasheet specifications.

Typically, designs operate with a continuous noise floor and a relatively low simultaneous Flip-Flop (FF) toggle rate, which helps to spread the current demand over time. However, designs with abnormally high toggle rates, greater than 15% of the total device Flip-Flops toggling simultaneously in the same direction, increase the instantaneous current demand on the VDD supply. Furthermore, if that large quantity of simultaneously switching FFs periodically starts and stops with a low enable rate, it can cause a large transient current demand on the PDN as the logic is re-enabled. Depending on the amount of fabric logic that starts toggling simultaneously when the enable is asserted, the transient current demand on the PDN can cause the VDD to dip below the datasheet limits. To avoid extra global net clock jitter, ensure that the VDD is always maintained within the datasheet limits during design operation.

Regarding the second contributor to global net clock jitter, it is related to the percentage of the 151,824 FFs in the device that toggle simultaneously, in the same direction, at a given clock edge, when that clock domain operates within the 10 MHz to 40 MHz range. In typical designs, the average FF toggle rate per clock domain does not exceed ~25%. Specific boards and designs might require higher toggle rates, but support for that level of operation should be confirmed by design specific VDD variation and clock jitter measurements taken on the user board with the design operating at its maximum switching activity. To help quantify the FF toggle rate for a given clock domain, consider the following example:

If a clock domain drives 30% of the 151,824 total available FFs in the RTG4 fabric, and if the average FF toggle rate within that domain is 25%, it equates to only a 7.5% effective toggle rate in the device (assuming no other clock domains are switching simultaneously).

For information about maximum period jitter specifications on the RTG4 Global Network for various FF toggle rates within the device, see *DS0131: RTG4 FPGA Datasheet*. This specification on RTG4 global net clock jitter is a worst-case maximum period jitter that applies across the supported RTG4 silicon process, voltage, and temperature range. The period jitter is useful for analysis at the boundary of the device, such as an interfaces to other devices using common clocking topology instead of source-synchronous clocking. Similarly, period jitter is useful for static timing analysis of inter-clock domain paths. For Static Timing Analysis (STA) of FPGA fabric internal paths, such as register-to-register paths in a single clock domain, it is more realistic to use cycle-to-cycle jitter. Using period jitter during STA of internal reg-to-reg paths in a single clock domain is overly conservative and can create artificial timing violations.

There are several characteristics to consider regarding this PDN disturbance induced RTG4 global net clock jitter:

- These PDN disturbances impact the entire FPGA fabric including both the clock and data paths. This avoids potential skew issues that can arise when clock and data slow down independently.
- All FFs within a clock domain see the same delayed/advanced clock edge for a given cycle.
- Data transfers within a single clock domain do not fail due to this added clock jitter.
- Silicon characterization of victim logic clocked by a victim global clock surrounded by a jitter stress design shows:
 - The maximum frequency achieved in hardware still exceeds the maximum performance reported by SmartTime Static Timing Analysis software.
 - Zero hold time failures are created in silicon when using the shortest (intra-cluster) data path or a data path spanning multiple fabric rows.
 - Global net clock jitter on FPGA fabric data path does not induce SerDes data bit errors at 2.5Gbps (STD speed grade) or 3.125Gbps (-1 speed grade)

Apply the following guidelines when accounting for clock jitter:

Table 6 • Effective Device Flip-Flop Toggle Rate¹

Effective Flip-Flop Toggle Rate (% of device FFs)	CCC PLL Used?	Jitter to use in STA
15% or Less	Yes	Whichever is larger of global net clock Jitter or CCC/PLL output clock jitter, as per DS0131: RTG4 FPGA Datasheet .
	No	Whichever is larger of global net clock Jitter or input buffer clock jitter, as per DS0131: RTG4 FPGA Datasheet .
Greater than 15%	Yes	Typical designs are expected to have less than 30% device FF utilization per clock domain, and an average FF toggle rate less than 25% within the domain for an Effective Device Flip-Flop Toggle Rate of <7.5%. For designs with >15% Effective FF Toggle percentage, measure the global net clock jitter from an output pin using the oscilloscope clock jitter measurement package with the design operating at maximum switching activity. Internal fabric reg-to-reg paths can use jitter data from the RTG4 FPGA Datasheet , with the Root Sum Squares (RSS) method to add the contribution from the input I/O buffer or PLL to the global net clock jitter. External interfaces should account for the measured period jitter during STA.
	No	

1. Product of FF percentage used of total FFs in the device and Average FF Toggle Rate for a given clock domain.

As an example of the methodology used to calculate the clock jitter constraints with each clock domain in the RTG4 design, consider the RTG4 design summarized in the following table, for a -1 speed grade RTG4 device:

Table 7 • Estimation of Worst-Case Clock Jitter for a -1 Speed Grade RTG4 Design

Clock Domain	Frequency (MHz)	Clock Path to FF	Sequential Loads	Estimated FF Toggle Rate within Domain – Parameter A	% Utilization of Total Device FFs – Parameter B	Effective FF Toggle Rate (% of total device FF) – A% of B	Inter-clock relationship	Worst-Case Clock Jitter from domain (ps) pk-to-pk ¹	Worst-Case Clock Jitter to use in STA (ps) pk-to-pk
Clk_0	156.25	CLKBUF to GB	40,000	30%	26.4%	7.9%	Async	524.4	525
Clk_1	125	CLKBUF to GB	38,000	30%	25%	7.5%	Async	500	525
Clk_2	20	CLKBUF to FCCC PLL to GB	15,000	25%	10%	2.5%	Sync with Clock_3 Async to others	1000 + 1.45 * Input_Jitter	1000 + 1.45 * Input_Jitter
Clk_3	80	CLKBUF to FCCC PLL to GB	15,000	15%	10%	1.5%	Sync with Clock_2 Async to others	250 + 1.45 * Input_Jitter	Max of {525, OR 250 + 1.45 * Input_Jitter}
Clk_4	25	LVDS33 CLKBUF to GB (VID ≥ VICM)	10,000	18%	6.6%	1.2%	Async	1600	1600

1. For information about Fabric PLL Output Clock Jitter Specification, Maximum MSIO Input Buffer Jitter Added to Input Clocks Directly Driving Globals, and RTG4 Global Network - Max Period Jitter, see *RTG4 FPGA Datasheet*, revision B or later.

The design described in [Table 7](#), page 21 provides an example showing the impact of various jitter sources on each individual clock domain and on victim clock domains that might otherwise have less clock jitter than the aggressor clock domain. The key factor in this analysis is using a relatively accurate estimate of the average FF toggle rate for the sequential logic in each clock domain. This can be estimated in SmartPower using the Vectorless Analysis technique. Furthermore, performing a gate-level simulation using a comprehensive testbench with realistic design toggle/activity rates can generate a VCD file that can be imported into SmartPower to initialize the toggle rate for all internal nodes. When the estimated FF toggle rate within each domain is available, it can be converted into the effective FF toggle percentage of the total number of device FFs (151,824) by multiplying against the percentage of the total device FF used in that domain. Typical designs do not exceed a 15% effective simultaneous FF toggle rate (in the same direction) out of the 151,824 FFs in the device, but it is design specific, depending on whether it is a control logic or high-speed data path logic. Therefore, the *RTG4 FPGA Datasheet* provides conservative, worst-case global net clock jitter characterization data that should be used carefully, to prevent excessive timing closure difficulty.

For example, Clk_0 has a 30% FF toggle rate within the clock domain, and it uses 40,000 or 26.4% of the 151,824 total FFs available in the device. Therefore, if 30% of the 40,000 FFs switch at each active Clk_0 edge, the effective FF toggle percentage is 12,000 FFs out of 151,824 total, or 7.9%. In other words, this is 30% (Parameter A) of the 26.4% FFs used in the device (Parameter B) which gives $0.3 * 26.4\% = 7.9\%$ effective FF toggle percentage. With that effective FF toggle rate, the Global Network Max Period Jitter for 7.5% FF Toggle percentage on a -1 speed grade specifies 500 ps worst-case period jitter. For information about Global Network Max period Jitter, see *RTG4 FPGA Datasheet*, revision B or later. Since 7.9% is more than the datasheet datapoint of 7.5%, linear interpolation indicates that 524.4 ps

worst-case period jitter can be used. This is written as 525 ps to make it easier when writing clock uncertainty constraints. If it is assumed that this clock enters the RTG4 with an MSIO standard other than a 3.3V differential input, the input buffer jitter added to the clock that directly drives the global network is 128 ps peak-to-peak (which is 2% of the clock period).

When the global net clock jitter is larger than the jitter from the MSIO input buffer, the larger 525 ps clock jitter is used for this domain. Furthermore, the effective FF toggle percentage for Clk_0 makes it an aggressor on the other clock domains in the design, and therefore, the 525 ps clock jitter value for Clk_0 is considered against each clock domain's jitter. The larger value is used for Static Timing Analysis (STA) and Timing Driven Place and Route (TDPR), as per [Table 7](#), page 21. The 525 ps peak-to-peak jitter number is used with the clock uncertainty constraint described in the paragraphs following this design example.

Similar to Clk_0, the derivations of the worst-case clock jitter for each domain is shown below:

Clk_1:

30% of the 38,000 FFs toggle at every clock cycle, that is, 11,400 FFs out of 151,824 FFs total, or 7.5% of the total RTG4 FFs toggle at each active clock edge. Reviewing the [RTG4 FPGA Datasheet](#) table for Global Network – Max Period Jitter, the max period clock jitter is 500 ps. The MSIO input buffer jitter of 160 ps pk-to-pk (± 80 ps) is ignored in this case due to the larger global net clock jitter. Note that this clock domain's global net jitter is a victim of the Clk_0 aggressor clock, and for STA purposes, the larger Clk_0 clock jitter should be used (525 ps peak-to-peak).

Clk_2:

25% of the 15,000 FFs toggle every clock cycle, that is, 3,750 FFs or 2.5% of the total RTG4 FFs toggle at each active clock edge. If you also consider that Clk_3 is synchronous to Clk_2, the Clk_3 domain switches at the same edge as the FFs clocked by Clk_2, the 3,750 FFs from Clk_2 and 2,250 FFs from Clk_1 mean that 6,000 FFs out of 151,824 device FFs switch simultaneously, or less than 4% of the device FFs. Therefore, the CCC/PLL output jitter dominates the Clk_2 clock jitter, and the STA must use 2% of the clock period plus $1.45 \times \text{Input Reference Clock Jitter}$. This is $1000 \text{ ps} + 1.45 \times \text{Input_Jitter}$. The Input_Jitter term is added to model the worst-case PLL jitter transfer where the PLL adds jitter on top of the jitter present on the incoming reference clock. This is a worst-case value to cover the wide range of input clock frequencies and jitter frequencies over the supported operating voltage and temperature ranges. When the input reference clock is $>100 \text{ MHz}$ and the jitter frequency is $< 1 \text{ MHz}$, the PLL typically does not add jitter to the input reference clock. However, it can be difficult to model the jitter frequency of the clock jitter on the input reference clock, thus the maximum worst-case PLL contribution to the input jitter is specified in the [RTG4 FPGA Datasheet](#). For this clock domain, even if the input jitter from the external clock is assumed to be 0 ps, the 1000 ps clock jitter calculated from 2% of the clock period is larger than aggressor clock domain Clk_0 global net period jitter, and thus [Table 7](#), page 21 uses the PLL output jitter as the value for STA.

Clk_3:

The effective device FF toggle percentage from this clock domain is low and thus the global net clock jitter due to switching activity is not considered. Similar to Clk_2, the clock jitter to use for STA is the CCC/PLL output clock jitter of 2% of the clock period plus $1.45 \times \text{Input Reference Clock Jitter}$, or $250 \text{ ps} + 1.45 \times \text{Input_Jitter}$. Depending on the value of the input jitter on the external reference clock, the clock jitter to use for STA is the maximum of either 525 ps from aggressor clock Clk_0, or $250 \text{ ps} + 1.45 \times \text{Input_Jitter}$, whichever is larger.

Clk_4:

This clock uses LVDS33 to input the external reference clock and directly drive the global network. Note that the interface is designed such that the input differential voltage (VID) is greater than or equal to the input common mode voltage (VICM), or $\text{VID} \geq \text{VICM}$. This means that clock jitter to use for STA is derived from the [RTG4 FPGA Datasheet](#) specification on the max MSIO input buffer jitter added to a clock signal directly driving globals. The value is calculated to be 1600 ps worst-case period jitter, based on 4% of the clock period. The effective device FF toggle percentage from this clock domain is low and thus the global net clock jitter due to switching activity is not considered. Furthermore, the Clk_0 aggressor clock jitter is smaller than the input buffer clock jitter, due to the large Clk_4 period, and thus does not play a role in analyzing this domain's clock jitter.

When putting this example into practice, remember that there are some cases where the RTG4 global network jitter can be omitted from the FPGA design Static Timing Analysis. Using the measured characteristics described above for the PDN disturbance induced RTG4 global net clock jitter, a clock domain where the data transfers are confined to a single clock domain within the FPGA does not exhibit setup or hold failures even in the presence of a jitter aggressor clock domain. However, when that logic contains clock domain crossings inside the FPGA, or interfaces to devices outside the FPGA using a common clock, the clock jitter must be included in STA.

A peak-to-peak period jitter value of 300ps from the [DS0131: RTG4 FPGA Datasheet](#) can be used in Libero SoC v12.0 or later using the `set_clock_uncertainty` constraint. For example, to set simple uncertainty in one clock domain, use the SDC constraint:

```
set_clock_uncertainty 0.3 [get_clocks {sys_clk}]
```

Clock uncertainty constraints can also be specified between clock domains as shown in [UG0679: Timing Constraints Editor User's Guide](#).

Note: By default, the `set_clock_uncertainty` constraint applies to both setup and hold checks. The constraint supports `-setup` and `-hold` options to allow selective application to either setup or hold checks. For a single clock domain, Microchip testing has shown that the RTG4 global net clock jitter described above does not create hold violations during fast data path testing. Therefore, the simple clock uncertainty constraint set for reg-to-reg paths in a single domain should be applied for setup checks only.

Starting with Libero SoC v12.2, the Timing Driven Place and Route tool also applies uncertainty constraints during maximum delay optimization and minimum delay repair (MDR).

In Libero SoC v11.9 or earlier, a 300ps jitter value can be viewed as +/-150ps and used with the clock source latency constraint. For example, use the following SDC constraints in Libero SoC v11.9 or earlier:

```
set_clock_latency -source -early -0.150 { RTG4FCCC_0/GL0 }
set_clock_latency -source -late 0.150 { RTG4FCCC_0/GL0 }
```

If a user design does not meet the VDD and FF toggle rate criteria described in the preceding paragraphs and in the [DS0131: RTG4 FPGA Datasheet](#), the global net clock jitter can be addressed in one of the two ways:

- Modify the design to reduce the maximum simultaneous FF toggle rate and to meet the datasheet VDD tolerance specification.
OR
- Measure the global clock jitter directly (route global net to output pin) with the user design running in the fabric and then specify clock uncertainty SDC constraints to account for the measured clock jitter.

There are a few additional decisions to consider when trying to optimize clock jitter with RTG4 designs:

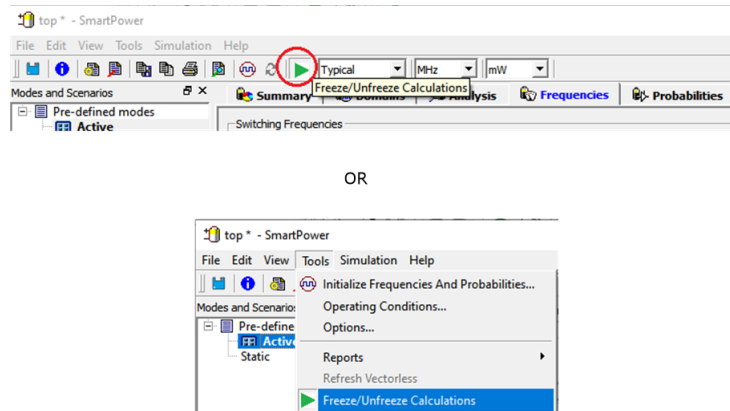
- When using the internal RC Oscillator as a clock source, ensure that it drives a CCC block in the same device corner (SouthEast - SE), as described in the [How to Use On-Chip Oscillator](#) section. Additionally, review the internal RC Oscillator jitter specifications in the [DS0131: RTG4 FPGA Datasheet](#) and ensure that a large enough PLL Lock Window is selected when using the internal RC Oscillator as the reference clock to an RTG4 PLL. The PLL Lock Window setting is described in [Lock Generation Circuit](#) section.
- When selecting RTG4 input I/O standards for incoming CCC PLL and SerDes reference clocks, consider that RTG4 LVDS25 input buffers have better input buffer jitter characteristics at cold temperature compared to the LVDS33/LVPECL33 input buffers, as per the [DS0131: RTG4 FPGA Datasheet](#). See [AN3216: Reference Clocks for RTG4 SerDes REFCLK Inputs and Interface Circuits](#) for more information about selecting and connecting reference clocks.
- Whenever possible, ensure that clock signals are routed using global buffers and global nets. Libero SoC's post-Place & Route Global Net Report issues a warning for all clock signals that use non-global, regular routing resources.

3.4 SmartPower Usage Tips

When using SmartPower to estimate the FF Toggle Rate for a clock domain within the design, here are some helpful tips to update the switching frequency annotation from the default setting of "Fixed Values"

Right-click **Verify Power** > **Open Interactively** in the Libero SoC **Design Flow** pane, the SmartPower window appears. By default, the SmartPower window will be set to have its panes frozen so that the user can switch between tabs quickly, without constantly re-calculating and re-loading the results. Click Play or Pause in the tool bar to freeze or unfreeze the calculation as shown in the following figure.

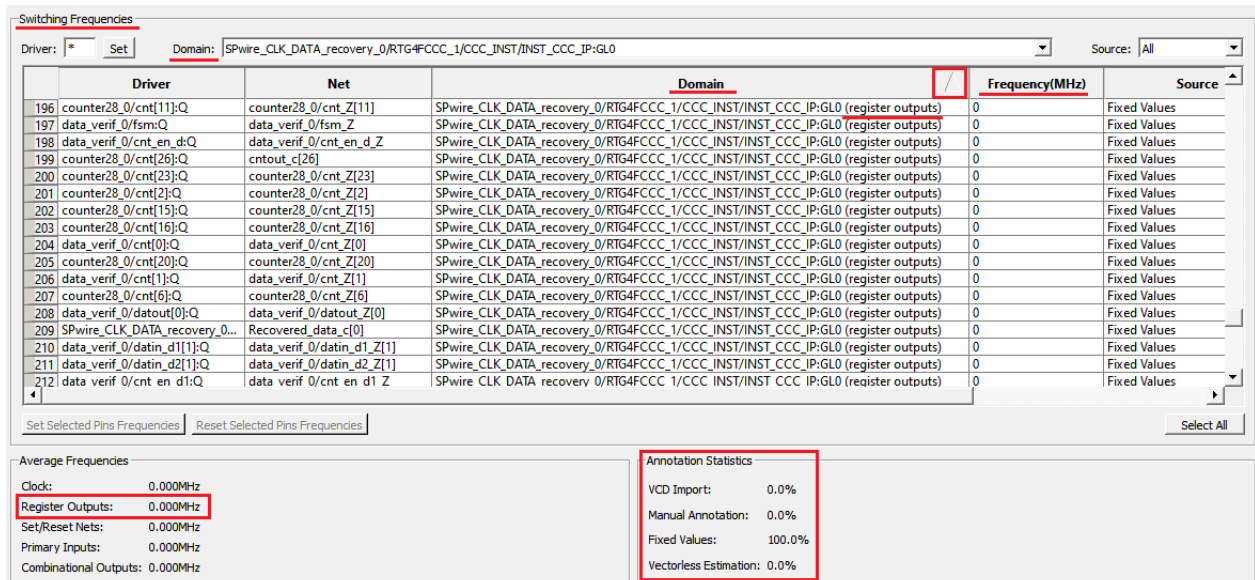
Figure 10 • SmartPower Window



In the **Frequencies** tab of SmartPower, select the clock domain to analyze, and sort the outputs by type (combinatorial output, register output, set/reset, etc), and view the switching frequency being used for power estimation and the source of the toggle rate. By default, the estimation uses 100% Fixed Values for toggle rates. To realistically estimate the register switching frequencies, un-freeze the calculations and either use:

- Vectorless Estimation from the **Tools** Menu -> **Initialize Frequencies and Probabilities...** option
- OR
- VCD File from a comprehensive gate-level simulation from the **Simulation** Menu -> **Import VCD File...**

Figure 11 • Switching Frequencies



When the register outputs are annotated with more realistic switching frequencies, the annotated switching frequency relative to the clock frequency can be used to estimate the toggle rate of the FFs within that clock domain. For example, for a 100 MHz clock, if the register outputs has a 100% toggle rate at each rising edge, the switching frequency would be 50 MHz, or the clock domain frequency divided by 2. Therefore, to determine the toggle rate from the annotated switching frequency for the register outputs, use the formula $\text{Toggle Rate} = (\text{Switching Frequency} / \text{Clock Frequency}) \times 2 \times 100$. This provides the average FF Toggle Rate within the clock domain, and it must be further converted to the effective FF toggle rate within the total available FFs in the RTG4 device, as showed in the example calculations above.

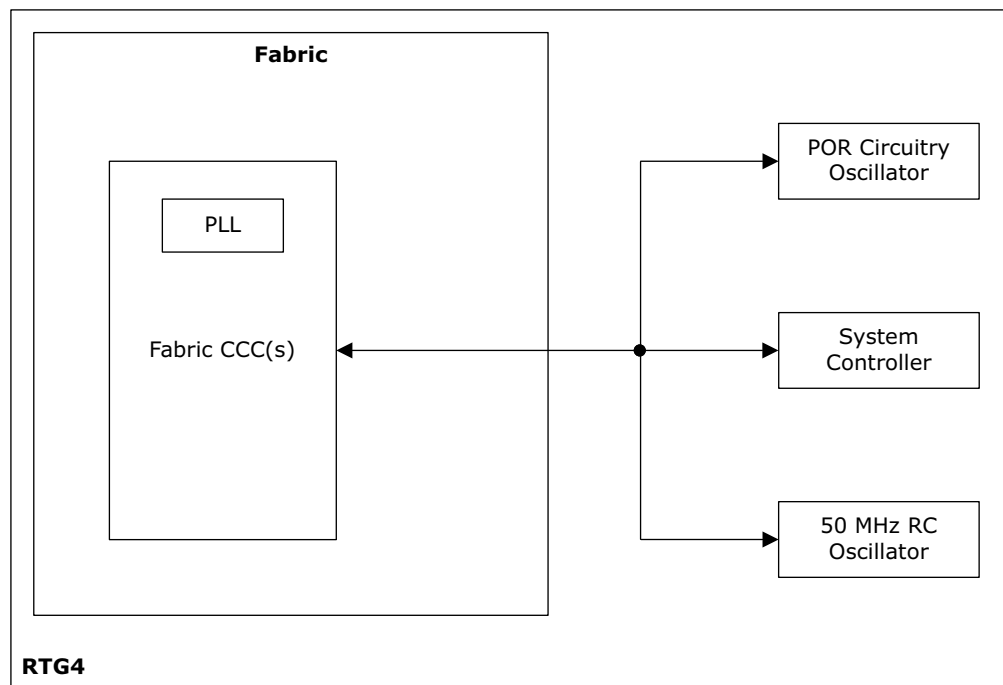
4 On-Chip Oscillator

4.1 Introduction

The RTG4 devices have one radiation-hardened 50 MHz RC oscillator for generating free-running clock. The following figure shows clock sourcing capabilities of the on-chip oscillator.

The on-chip oscillator supplies clocks to the Fabric CCCs, System Controller, and POR circuitry. This chapter describes the on-chip oscillator's clock sourcing capabilities. Refer to the "On-Chip Oscillator" section in the *DS0131: RTG4 FPGA Datasheet* for the electrical characteristics of on-chip oscillator.

Figure 12 • RTG4 On-Chip Oscillator Clock Sourcing Capabilities



4.2 Functional Description

The 50 MHz RC oscillator generates a nominal 50 MHz digital clock signal. It is powered by the device core supply VDD and does not require external components for operation.

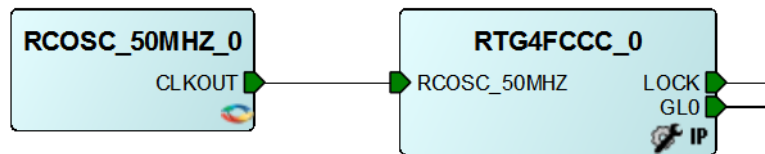
As shown in the preceding figure, the on-chip oscillator outputs are connected to the fabric CCCs, and hard IP blocks (System Controller and POR Circuitry) through dedicated hardwired connections (not routed through FPGA fabric). The 50 MHz Oscillator cannot drive the fabric logic directly. The 50 MHz RC oscillator can be used as reference clock to the CCC and then the CCC output drives fabric logic. The on-chip oscillator can be used by instantiating the Chip Oscillator macro in the Libero SoC software to drive fabric CCCs. It is not required to instantiate the oscillator macro for System Controller operations because it has a dedicated hardwired connection from the 50 MHz RC Oscillator. Refer to *UG0576: RTG4 FPGA System Controller User Guide* for more information. Unused on-chip oscillator is always running even if it is not instantiated and it cannot be disabled.

4.3 How to Use On-Chip Oscillator

The 50 MHz RC oscillator macro is available in the IP Catalog of Libero SoC design software. The oscillator can be used by instantiating the macro in the RTL or in SmartDesign. To instantiate into a SmartDesign tool, drag-and-drop the macro from the IP Catalog into the SmartDesign canvas. For instance, if a fabric CCC is configured to use the 50 MHz oscillator, the input port exposed on the CCC for the 50 MHz oscillator—RCOSC_50 MHz—can only be driven by the output port—CLKOUT—of the 50 MHz oscillator macro as shown in the following figure.

Note: The 50 MHz RC oscillator is located in the lower right corner of the device (SE). For the best overall system jitter from an RC Oscillator generated clock, the CCC block that it drives should be in the same corner of the device.

Figure 13 • 50 MHz RC Oscillator and Fabric CCC Connectivity



5 Fabric Clock Conditioning Circuitry

5.1 Introduction

The RTG4 devices have eight hardened fabric CCCs. Each fabric CCC enables flexible clocking schemes for the logic implemented in the FPGA fabric and can provide the base clock for on-chip hard IP blocks—FDDR and SerDes blocks. Each fabric CCC is completely hardened with the exception of the reference and feedback dividers, RFDIV and FBDIV, delay line, and clock inputs coming from the fabric that are not hardened. CCC dedicated I/O input is not hardened if it is DDRIO. It can be routed through SpaceWire glitch-filter inside CCC to make it SET hardened. Up to two fabric clock inputs can be glitch filtered and delivered as a hardened signal onto the hardened global clock network. Any SET pulses on the delay lines are filtered by the PLL. Each fabric CCC includes a radiation hardened dedicated triple redundant PLL and can generate clock signals of different frequency and phase. The associated PLL can be bypassed, if it is not required. This chapter describes the fabric CCC features and configuration.

5.1.1 Features

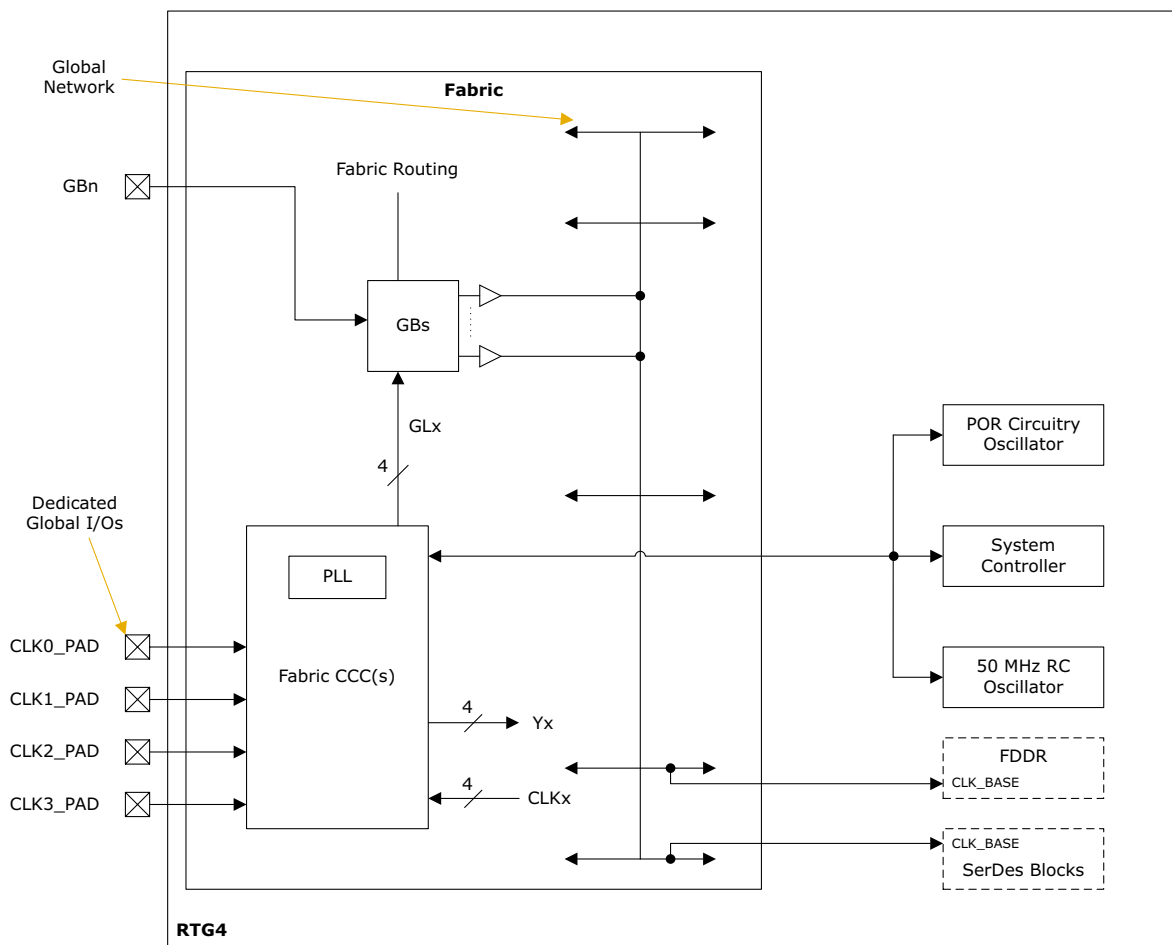
Each fabric CCC supports the following features:

- Internal and external clock sources
- Generation of four different clocks with a maximum frequency up to 400 MHz
 - Drives global network and/or the local routing network
- Automatic output resynchronization after PLL lock
- Configurable phase error window for PLL lock assertion
- Configurable PLL lock delay
- Clock delay or clock advancement
- Clock phase shifting
- Clock inversion
- Clock gating
- SpaceWire for RX clock generation and recovery

5.2 System-Level Block Diagram

The following figure shows the system-level block diagram of the fabric CCC. It shows the inputs and outputs for one fabric CCC; each fabric CCC has a similar set of inputs and outputs.

Figure 14 • Fabric CCC System-Level Block Diagram

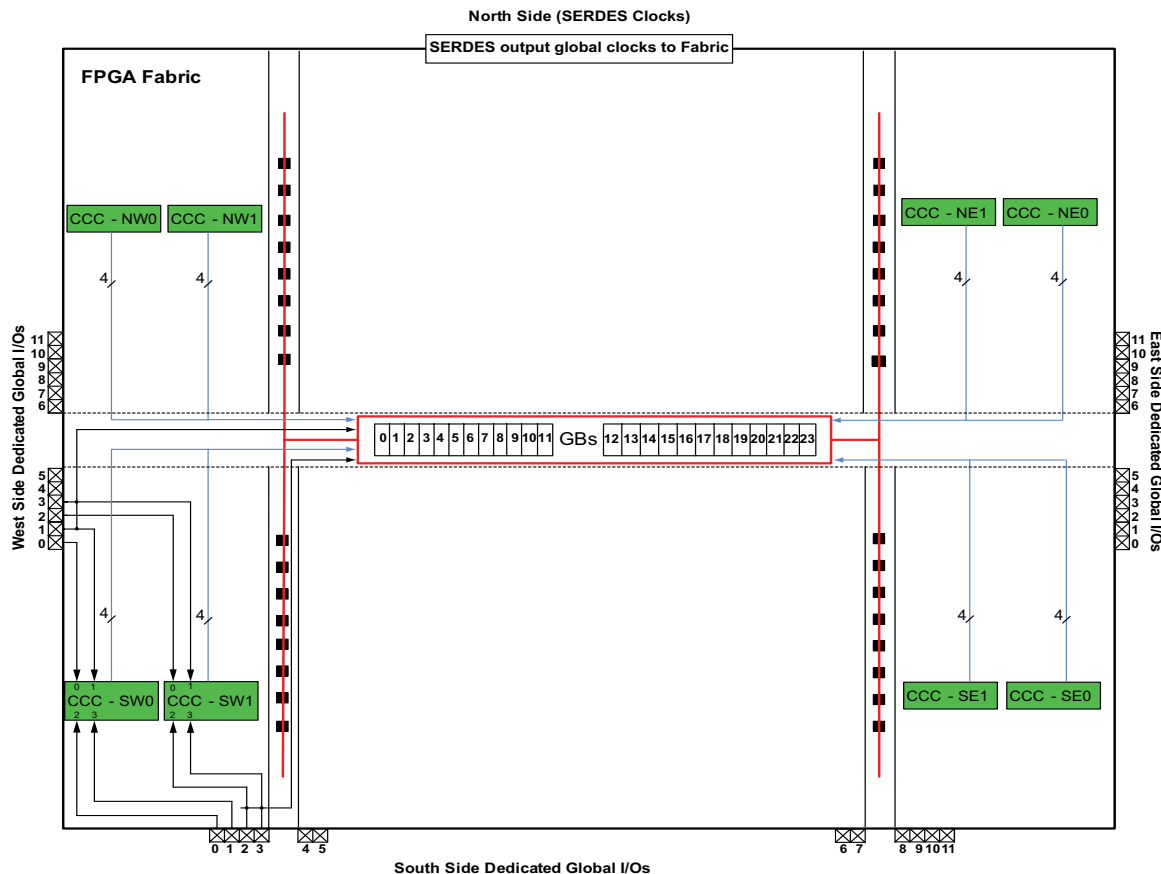


5.3 Fabric CCC Locations

The following figure shows the locations of fabric CCCs in the FPGA fabric of the RTG4 devices. The fabric CCCs are labeled according to their locations in the FPGA fabric floor plan. For instance, the fabric CCCs located in the northeast corner are labeled as CCC-NE0 and CCC-NE1. All the available fabric CCCs are associated with a radiation-hardened dedicated triple redundant PLL. Refer to the [Fabric PLL Circuitry](#), page 37 for more information on PLLs.

Fabric CCC global clock outputs are connected to the GBs.

Figure 15 • Fabric CCC Locations in the RTG4 Devices



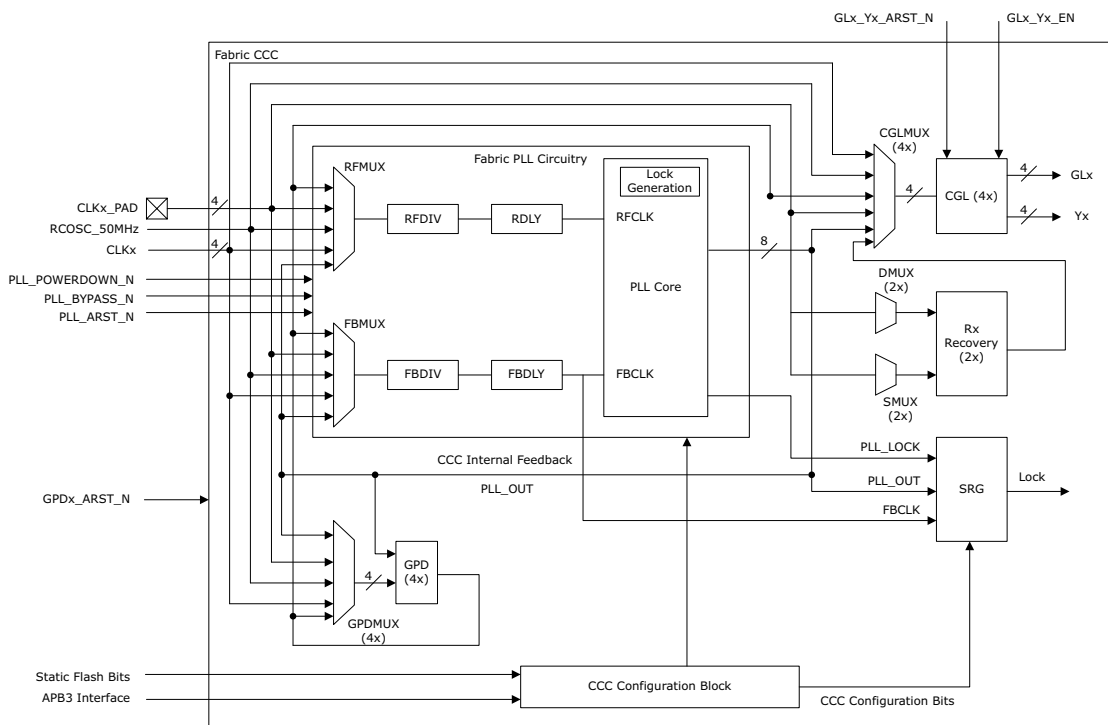
5.4 Functional Description

The following figure shows the functional block diagram of the fabric CCC. Each fabric CCC block consists of the following components:

- Fabric PLL Circuitry
- General purpose mux (GPMUX)
- General purpose divider (GPD)
- Synchronous reset generator (SRG)
- Clock gating logic mux (CGLMUX)
- SpaceWire RX Recovery DMUX/SMUX

The fabric CCC features are statically configured by the CCC macro, which is available in the IP Catalog of the Libero design software. Fabric CCCs can also be configured dynamically through an APB3 bus interface. Refer to the [Fabric CCC Configuration](#), page 49 for more information.

The following sections describe fabric CCC output clocks, input clock sources, and each component of the fabric CCC. [Table 8](#), page 31 the fabric CCC ports and their descriptions. Access and visibility to these ports are controlled by the user configuration.

Figure 16 • Fabric CCC Block Diagram**Table 8 • Fabric CCC Port Description**

Port Name	Direction	Polarity	Description
CLK0_PAD	Input		Input clock when using Dedicated Input Pad 0 configured as single ended I/O.
CLK0_[PADP and PADN]	Input		Input clock when using Dedicated Input Pad 0 configured as differential I/O. P side and N side.
CLK1_PAD	Input		Input clock when using Dedicated Input Pad 1 configured as single ended I/O.
CLK1_[PADP and PADN]	Input		Input clock when using Dedicated Input Pad 1 configured as differential I/O. P side and N side.
CLK2_PAD	Input		Input clock when using Dedicated Input Pad 2 configured as single ended I/O.
CLK2_[PADP and PADN]	Input		Input clock when using Dedicated Input Pad 2 configured as differential I/O. P side and N side.
CLK3_PAD	Input		Input clock when using Dedicated Input Pad 3 configured as single ended I/O.
CLK3_[PADP and PADN]	Input		Input clock when using Dedicated Input Pad 3 configured as differential I/O. P side and N side.
CLK0_SPWR_STROBE_[PADP and PADN]	Input		Input Clock when Dedicated pad 0 is used as input to Strobe of RX block in differential mode.
CLK1_SPWR_DATA_[PADP and PADN]	Input		Input Clock when Dedicated Pad 1 is used as input to Data of RX block in differential mode.

Table 8 • Fabric CCC Port Description (continued)

Port Name	Direction	Polarity	Description
CLK2_SPWR_STROBE_[PADP and PADN]	Input		Input Clock when Dedicated pad 2 is used as input to Strobe of RX block in differential mode.
CLK3_SPWR_DATA_[PADP and PADN]	Input		Input Clock when Dedicated Pad 3 is used as input to Data of RX block in differential mode
CLK0_SPWR_STROBE_PAD	Input		Input Clock when Dedicated pad 0 is used as input to Strobe of RX block configured as single ended I/O.
CLK1_SPWR_DATA_PAD	Input		Input Clock when Dedicated pad 1 is used as input to Data of RX block configured as single ended I/O.
CLK2_SPWR_STROBE_PAD	Input		Input Clock when Dedicated pad 2 is used as input to Strobe of RX block configured as single ended I/O.
CLK3_SPWR_DATA_PAD	Input		Input Clock when Dedicated pad 3 is used as input to Data of RX block configured as single ended I/O.
CLK0	Input		Input clock from FPGA core when using FPGA Fabric Input 0 .
CLK1	Input		Input clock from FPGA core when using FPGA Fabric Input 1 .
CLK2	Input		Input clock from FPGA core when using FPGA Fabric Input 2 .
CLK3	Input		Input clock from FPGA core when using FPGA Fabric Input 3 .
RCOSC_50 MHZ	Input		Input clock when using 50 MHz Oscillator .
GL0	Output		Generated clock driving FPGA fabric global network 0.
GL1	Output		Generated clock driving FPGA fabric global network 1.
GL2	Output		Generated clock driving FPGA fabric global network 2.
GL3	Output		Generated clock driving FPGA fabric global network 3.
GL0_Y0_EN	Input	High	Enable signal for the clock driving FPGA fabric global network 0 (GL0) and fabric local routing (Y0).
GL1_Y1_EN	Input	High	Enable signal for the clock driving FPGA fabric global network 1 (GL1) and fabric local routing (Y1).
GL2_Y2_EN	Input	High	Enable signal for the clock driving FPGA fabric global network 2 (GL2) and fabric local routing (Y2).
GL3_Y3_EN	Input	High	Enable signal for the clock driving FPGA fabric global network 3 (GL3) and fabric local routing (Y3).
GL0_Y0_ARST_N	Input	Low	Asynchronous reset which resets the Clock Gating Logic (CGL) output signals GL0/Y0. Refer to Clock Gating Logic (CGL) , page 44.
GL1_Y1_ARST_N	Input	Low	Asynchronous reset which resets the Clock Gating Logic (CGL) output signals GL1/Y1. Refer to Clock Gating Logic (CGL) , page 44.
GL2_Y2_ARST_N	Input	Low	Asynchronous reset which resets the Clock Gating Logic (CGL) output signals GL2/Y2. Refer to Clock Gating Logic (CGL) , page 44.
GL3_Y3_ARST_N	Input	Low	Asynchronous reset which resets the Clock Gating Logic (CGL) output signals GL3/Y3. Refer to Clock Gating Logic (CGL) , page 44.

Table 8 • Fabric CCC Port Description (continued)

Port Name	Direction	Polarity	Description
Y0	Output		Generated clock driving FPGA fabric local routing resource.
Y1	Output		Generated clock driving FPGA fabric local routing resource.
Y2	Output		Generated clock driving FPGA fabric local routing resource.
Y3	Output		Generated clock driving FPGA fabric local routing resource.
RX0_DATA_PORT	Output		Output port for RX0 Data. Same signal that is driving the RX0 SpaceWire clock recovery Data input.
RX1_DATA_PORT	Output		Output port for RX1 Data. Same signal that is driving the RX1 SpaceWire clock recovery Data input.
LOCK	Output	High	PLL Lock indicator signal. This signal is asserted (lock) High.
PLL_BYPASS_N	Input	Low	Powers-down the PLL core and bypasses it such that PLL_OUT tracks reference clock. This has higher priority than reset.
PLL_ARST_N	Input	Low	Powers-down the PLL core. With the auto-reset logic, PLL_ARST_N puts the PLL into reset until the 1 μ s delay counter finishes.
PLL_POWERDOWN_N	Input	Low	Powers-down the PLL for the lowest quiescent current and the PLL outputs are Low. This has higher priority than reset and bypass.
GPD0_ARST_N	Input	Low	General purpose divider: GPD0 asynchronous reset signal. The GPD0 is held in reset when 0.
GPD1_ARST_N	Input	Low	General purpose divider: GPD1 asynchronous reset signal. The GPD0 is held in reset when 0.
GPD2_ARST_N	Input	Low	General purpose divider: GPD2 asynchronous reset signal. The GPD0 is held in reset when 0.
GPD3_ARST_N	Input	Low	General purpose divider: GPD3 asynchronous reset signal. The GPD0 is held in reset when 0.
Fabric CCC Configuration Bus Interface Signals – APB3 Bus Interface			
APB_S_PCLK	Input		APB3 interface clock signal.
APB_S_PRESET_N	Input	Low	APB3 interface active low reset signal.
APB_S_PADDR[7:2]	Input		APB3 interface address bus. This port is used to address fabric CCC internal registers.
APB_S_PSEL	Input		APB3 interface slave select signal.
APB_S_PENABLE	Input	High	APB3 interface strobe signal to indicate the second cycle of an APB3 transfer.
APB_S_PWRITE	Input		APB3 interface read/write control signal. When High, this signal indicates an APB3 write access and when Low, a read access.
APB_S_PWDATA[7:0]	Input		APB3 interface write data bus.
APB_S_PRDATA[7:0]	Output		APB3 interface read data bus.

Table 8 • Fabric CCC Port Description (continued)

Port Name	Direction	Polarity	Description
APB_S_PREADY	Output		APB3 interface ready indication output to master.
APB_S_PSLVERR	Output		APB3 interface error indication signal.

5.4.1 Fabric CCC Output Clocks

Each fabric CCC generates up to four different global clocks (GL0, GL1, GL2, and GL3) and four core clocks (Y0, Y1, Y2, and Y3) with a maximum frequency up to 400 MHz. The generated global clocks drive the FPGA fabric global networks and core clocks drive the local routing resources in the FPGA fabric. Fabric CCC clock outputs (Yx) can be used to drive internal logic without using the global network resources. As this clock output goes through the FPGA fabric, there is a delay associated with the fabric routing delay. Core clocks are useful when global network resources must be conserved and utilized for other timing-critical paths. When using core clocks, timing analysis must be performed to ensure that there are no skew issues on these paths. The frequencies and phases of the core clocks are the same as those of the associated global clocks. The global buffers associated with the global clock outputs (GLx) are available to the user logic when the global clock outputs are not enabled. The fabric CCCs can be connected to each other or cascaded through the FPGA fabric.

Each of the four output clocks (GLx/Yx) can be driven by the following sources:

- PLL outputs
- General purpose dividers (GPDs) outputs
- Dedicated global I/Os
- On-chip oscillator
- FPGA fabric internal clock signals
- SpaceWire RX Recovery

5.4.2 Fabric CCC Clock Sources

The following clock sources can be used to drive the fabric CCC clock input as well as the fabric CCC outputs (GLx/Yx) directly:

- 50 MHz RC Oscillator
- Dedicated Global I/Os
- FPGA Fabric Clock Sources

The input clock frequency range for the fabric CCCs depends on the PLL usage for the output clocks generation.

- When the PLL is used, the PLL reference clock frequency must be between 10 MHz and 200 MHz
- When the PLL is bypassed, the fabric CCC input clock frequency can be up to 400 MHz

5.4.2.1 On-Chip Oscillator

On-chip radiation hardened oscillator, 50 MHz RC oscillator, has direct access to the fabric CCC(s) through hardwired connections. To use the on-chip oscillator as a clock source, the Chip Oscillator macro must be instantiated in the design and its output connected to the associated fabric CCC input. Refer to the *On-Chip Oscillator*, page 26 for more information.

5.4.2.2 Dedicated Global I/Os

The dedicated global I/Os are the primary source for bringing external clock inputs into the RTG4 device. Some of the dedicated global I/Os have direct access to the GBs, whereas others have to go through either fabric or CCC to reach GBs.

Each fabric CCC has four dedicated global I/Os (CCC_xyz_CLKIw, w = 0 to 3) as inputs. These dedicated global I/Os have direct access to the fabric CCCs and are not routed through the FPGA fabric. Each fabric CCC output (GLx/Yx), as well as the PLL reference clock, can be driven from any one of the four dedicated global I/Os. The following table lists the dedicated global I/Os associated with each fabric CCC for the RT4G150 CG1657 package. As listed in the RT4G150 CG1657 package Pin Function column of the following table, some dedicated global I/Os are shared with the hard IP blocks such as

FDDR. The shared dedicated global I/Os are not available in the fabric CCCs when the associated hard IP blocks are enabled and configured to use the shared dedicated global I/Os.

For other RTG4 devices, refer to the [DS0130: RTG4 FPGA Pin Descriptions](#) and the associated pin outs to identify the dedicated global I/Os associated with fabric CCCs.

Refer to [Dedicated Global I/Os](#), page 8 for more information on dedicated global I/Os.

Table 9 • Dedicated Global I/Os Connections to the Fabric CCC/PLLs

CCC Location	PAD	Dedicated Global I/O Pin Name	RT4G150 CG1657 Pin Number	RT4G150 CG1657 Pin Function
CCC-NE1	CLK0_PAD	CCC_NE1_CLKI0	T39	MSIOD58PB1/GB12_23/CCC_NE1_CLKI0/SPWR_NE1_0_RX_STROBE_P
	CLK1_PAD	CCC_NE1_CLKI1	AA34	MSIOD59PB1/GB12_23/CCC_NE1_CLKI1/SPWR_NE1_0_RX_DATA_P
	CLK2_PAD	CCC_NE1_CLKI2	AB41	DDRIO94PB0/FDDR_E_ADDR15/GB12_23/CCC_NE1_CLKI2/SPWR_NE1_1_RX_STROBE_P
	CLK3_PAD	CCC_NE1_CLKI3	AC39	DDRIO95PB0/FDDR_E_ADDR13/GB12_23/CCC_NE1_CLKI3/SPWR_NE1_1_RX_DATA_P
CCC-NE0	CLK0_PAD	CCC_NE0_CLKI0	M40	MSIOD43PB1/GB12_23/CCC_NE0_CLKI0/SPWR_NE0_0_RX_STROBE_P
	CLK1_PAD	CCC_NE0_CLKI1	V31	MSIOD44PB1/GB12_23/CCC_NE0_CLKI1/SPWR_NE0_0_RX_DATA_P
	CLK2_PAD	CCC_NE0_CLKI2	AA39	MSIOD73PB1/GB12_23/CCC_NE0_CLKI2/SPWR_NE0_1_RX_STROBE_P
	CLK3_PAD	CCC_NE0_CLKI3	AB37	MSIOD74PB1/GB12_23/CCC_NE0_CLKI3/SPWR_NE0_1_RX_DATA_P
CCC-SW1	CLK0_PAD	CCC_SW1_CLKI0	K5	MSIOD219PB7/GB0_11/CCC_SW1_CLKI0/SPWR_SW1_0_RX_STROBE_P
	CLK1_PAD	CCC_SW1_CLKI1	N8	MSIOD218PB7/GB0_11/CCC_SW1_CLKI1/SPWR_SW1_0_RX_DATA_P
	CLK2_PAD	CCC_SW1_CLKI2	C10	MSIO265PB6/GB5/CCC_SW1_CLKI2/SPWR_SW1_1_RX_STROBE_P
	CLK3_PAD	CCC_SW1_CLKI3	K15	MSIO266PB6/GB7/CCC_SW1_CLKI3/SPWR_SW1_1_RX_DATA_P
CCC-SW0	CLK0_PAD	CCC_SW0_CLKI0	G3	MSIOD228PB7/GB0_11/CCC_SW0_CLKI0/SPWR_SW0_0_RX_STROBE_P
	CLK1_PAD	CCC_SW0_CLKI1	N10	MSIOD227PB7/GB0_11/CCC_SW0_CLKI1/SPWR_SW0_0_RX_DATA_P
	CLK2_PAD	CCC_SW0_CLKI2	E6	MSIO253PB6/GB1/CCC_SW0_CLKI2/SPWR_SW0_1_RX_STROBE_P
	CLK3_PAD	CCC_SW0_CLKI3	H10	MSIO254PB6/GB3/CCC_SW0_CLKI3/SPWR_SW0_1_RX_DATA_P

Table 9 • Dedicated Global I/Os Connections to the Fabric CCC/PLLs (continued)

CCC Location	PAD	Dedicated Global I/O Pin Name	RT4G150 CG1657 Pin Number	RT4G150 CG1657 Pin Function
CCC-NW1	CLK0_PAD	CCC_NW1_CLKI0	M2	MSIOD198PB8/GB0_11/CCC_NW1_CLKI0/SPWR_NW1_0_RX_STROBE_P
	CLK1_PAD	CCC_NW1_CLKI1	V11	MSIOD197PB8/GB0_11/CCC_NW1_CLKI1/SPWR_NW1_0_RX_DATA_P
	CLK2_PAD	CCC_NW1_CLKI2	AA3	MSIOD168PB8/GB0_11/CCC_NW1_CLKI2/SPWR_NW1_1_RX_STROBE_P
	CLK3_PAD	CCC_NW1_CLKI3	AB5	MSIOD167PB8/GB0_11/CCC_NW1_CLKI3/SPWR_NW1_1_RX_DATA_P
CCC-NW0	CLK0_PAD	CCC_NW0_CLKI0	T3	MSIOD183PB8/GB0_11/CCC_NW0_CLKI0/SPWR_NW0_0_RX_STROBE_P
	CLK1_PAD	CCC_NW0_CLKI1	AA8	MSIOD182PB8/GB0_11/CCC_NW0_CLKI1/SPWR_NW0_0_RX_DATA_P
	CLK2_PAD	CCC_NW0_CLKI2	AB1	DDRIO147PB9/FDDR_W_ADDR15/GB0_11/CCC_NW0_CLKI2/SPWR_NW0_1_RX_STROBE_P
	CLK3_PAD	CCC_NW0_CLKI3	AC3	DDRIO146PB9/FDDR_W_ADDR13/GB0_11/CCC_NW0_CLKI3/SPWR_NW0_1_RX_DATA_P
CCC-SE0	CLK0_PAD	CCC_SE0_CLKI0	K37	MSIOD22PB2/GB12_23/CCC_SE0_CLKI0/SPWR_SE0_0_RX_STROBE_P
	CLK1_PAD	CCC_SE0_CLKI1	N34	MSIOD23PB2/GB12_23/CCC_SE0_CLKI1/SPWR_SE0_0_RX_DATA_P
	CLK2_PAD	CCC_SE0_CLKI2	F28	MSIO334PB4/GB17/CCC_SE0_CLKI2/SPWR_SE0_1_RX_STROBE_P
	CLK3_PAD	CCC_SE0_CLKI3	F27	MSIO335PB4/GB19/CCC_SE0_CLKI3/SPWR_SE0_1_RX_DATA_P
CCC-SE1	CLK0_PAD	CCC_SE1_CLKI0	G39	MSIOD13PB2/GB12_23/CCC_SE1_CLKI0/SPWR_SE1_0_RX_STROBE_P
	CLK1_PAD	CCC_SE1_CLKI1	N32	MSIOD14PB2/GB12_23/CCC_SE1_CLKI1/SPWR_SE1_0_RX_DATA_P
	CLK2_PAD	CCC_SE1_CLKI2	D33	MSIO346PB4/GB21/CCC_SE1_CLKI2/SPWR_SE1_1_RX_STROBE_P
	CLK3_PAD	CCC_SE1_CLKI3	G29	MSIO347PB4/GB23/CCC_SE1_CLKI3/SPWR_SE1_1_RX_DATA_P

5.4.2.2.1 CCC/PLL Restrictions for the CQ352 Package Devices

The following are the restrictions for the CQ352 package devices:

- Only the CCCs on the south have dedicated input connections. The CCCs on the north can only have fabric input connections. Each of the south CCCs have two dedicated inputs.
- Each of the south CCCs is connected to only the second pair of SpaceWire I/Os—dedicated input pads 2 and 3.

5.4.2.3 FPGA Fabric Clock Sources

FPGA fabric clock sources are clocks coming through the FPGA fabric routed nets. Each fabric CCC has four clock inputs (CLKx, x = 0 to 3) from the FPGA fabric. Each fabric CCC output (GLx/Yx), as well as the reference clock, can be driven from any one of the four fabric clock inputs.

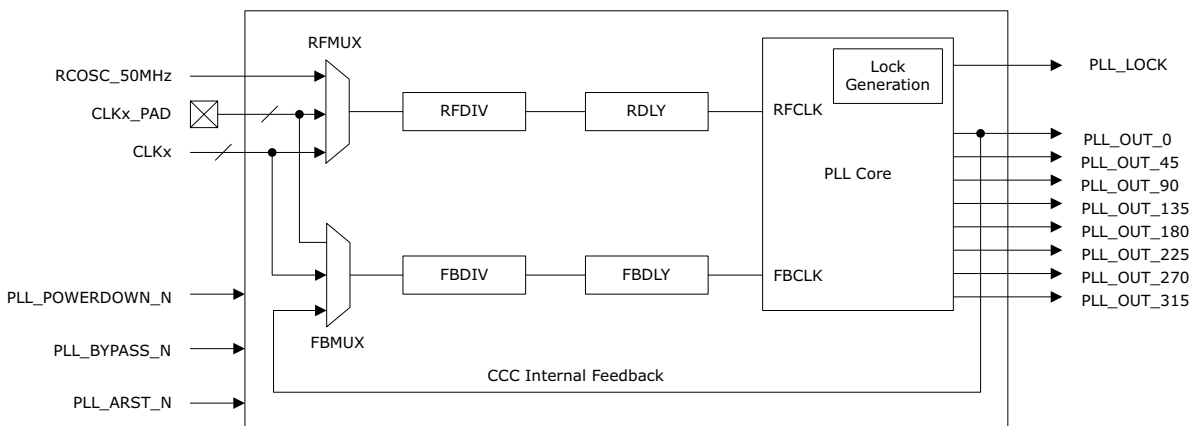
It is also possible to assign the fabric clock input to any of the regular I/O pins. This option provides the flexibility of selecting any regular I/O location but introduces additional delay due to FPGA fabric routing.

5.4.3 Fabric PLL Circuitry

Each fabric CCC includes components for operating the associated triple redundant PLL. Clocks requiring frequency synthesis and phase adjustments can utilize the PLL before connecting to the global or local routing networks. The following figure shows the block diagram of the fabric PLL circuitry. Fabric PLL circuitry includes the following components:

- PLL Core
- RFMUX
- FBMUX
- RFDIV and FBDIV
- RDLY and FBDLY

Figure 17 • Fabric PLL Circuitry



5.4.3.1 PLL Core

The RTG4 devices have eight radiation tolerant triple redundant PLLs. The PLL core can be configured in three different feedback modes:

- **PLL internal feedback:** When the PLL internal feedback mode is used, the PLL is automatically configured as a triple redundant PLL and thus radiation hardened. The CCC output clocks are not in sync with reference clock when the PLL internal feedback mode is selected. The CCC output clocks can be resynchronized between each other after the PLL is locked.
- **PLL external feedback:** The PLL external feedback can either be CCC internal feedback or external to the CCC. When the PLL configured to use external feedback mode, the PLL is in non-triplicated mode and only one PLL is being used. The CCC output clocks are in sync with reference clock when the CCC external feedback mode is selected. In PLL internal and CCC internal mode, the output clocks are not in sync with the reference clock.

Note: Each PLL is radiation hardened but does not provide the radiation tolerance as the three working together (triplicated mode). When in single PLL mode, radiation hits causes temporary clock glitches but recovers quickly. Radiation hits to the VCO may cause various types of responses from complete shutdown of the PLL, which is rare; large frequency variation that may or may not eventually recover without a reset, to small glitches or jitter on the clock.

All the PLLs can accept a reference clock input ranging from 10 MHz to 200 MHz. The minimum output frequency of the PLL is 20 MHz. Each PLL includes a phase shifter and the PLL output clock is available in eight phases with a 45° phase difference (0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°). The fabric PLL features are configured using static flash bits set by the CCC macro available in the Libero SoC design software.

The supplies required to power the PLL is VDDPLL, which must be connected to 3.3 V. To achieve a reasonable level of long term jitter performance, it is vital to deliver a low noise power supply to the PLL. Typically, an R-C or R-L-C filter is used with the C being composed of multiple devices to achieve a wide spectrum of noise absorption.

Note: The unused PLLs are configured to be held in power-down mode to consume the lowest quiescent current.

See *AC439 & AC453: RTG4 FPGAs Board Design and Layout Guidelines Application Note* for power-up and power-down sequence requirements of VDDPLL. For applications which cannot meet the guideline of ensuring VDDPLL is not the last supply to ramp up, see *How to Use Fabric CCC(s)*, page 60.

5.4.3.2 RFMUX

The reference clock multiplexer (RFMUX) allows selecting the PLL reference clock from the available fabric CCC clock sources, refer [Figure 17](#), page 37. The RFMUX can invert the clock inputs. As the PLL aligns reference and feedback clocks on the rising clock edge, the inversion feature allows aligning them on the falling reference and feedback clocks. The RFMUX is not a glitchless MUX and any glitches will be filtered by the PLL.

5.4.3.3 FBMUX

The feedback clock multiplexer (FBMUX) allows selecting the PLL feedback clock source from an internal or an external feedback loop, depending on the system requirements, refer [Figure 17](#), page 37. The internal feedback loop is internal to the PLL (not shown in [Figure 17](#), page 37 as it is internal to the PLL core). The external feedback loop is from one of the CCC global clocks (GLx) through the CCC internal routing (see [Figure 17](#), page 37), FPGA fabric or externally to the chip through dedicated global I/O pads. Minimum jitter and phase error can be achieved, when the PLL internal feedback loop is used. When PLL internal feedback is used, the programmable delay and GPD synchronization are not available.

The FBMUX can invert the clock inputs. As the PLL aligns reference and feedback clocks on the rising clock edge, the inversion feature allows aligning them on the falling reference and feedback clocks. The FBMUX is not a glitchless MUX and any glitches will be filtered by the PLL.

5.4.3.4 RFDIV and FBDIV

The PLL reference clock divider (RFDIV) is a 6-bit programmable divider. The feedback clock divider (FBDIV) is a 6-bit programmable frequency divider. RFDIV and FBDIV generate PLL reference clock (RFCLK) and feedback clock (FBCLK). The function of RFDIV and FBDIV is described in EQ 1 and EQ 2:

$$\text{RFCLK} = (\text{Input clock frequency}) / (\text{RFDIV [5:0]} + 1)$$

EQ 1

$$\text{FBCLK} = (\text{Input clock frequency}) / (\text{FBDIV [5:0]} + 1)$$

EQ 2

The visual CCC macro configurator derives the necessary divider values based on the input frequency, desired output frequencies, and the selected feedback loop. These dividers are configured statically by flash bits when the FPGA fabric is programmed.

5.4.3.5 RDLY and FBDLY

Each fabric CCC has a programmable delay element in the PLL reference clock path (RDLY) and feedback loop path (FBDLY), refer [Figure 17](#), page 37. For PLLs, adding delay in the reference clock path enables clock delay and adding delay in the feedback clock path enables clock advancement with respect to the PLL reference clock. A typical application for clock advancement is to advance the on-chip clock to cancel the global network delay and align the internal clock to an external clock on the board.

Each fabric CCC can be configured with a delay in the reference path or feedback path. The delay time of the programmable delay elements can be varied between 0 and 6.4 ns with a 100 ps (typical) step size. For information about CCC/PLL specifications, see *RTG4 FPGA Datasheet*. The programmable delay elements are bypassed when the PLL delay is set to zero or the PLL internal feedback loop is used.

5.4.3.6 Lock Generation Circuit

A lock signal, LOCK, is provided to indicate that the PLL has locked onto the incoming signal. The lock signal asserts High to indicate that the frequency and phase lock is achieved. Note that if there is no LOCK, the PLL outputs will still toggle. The PLL output will not toggle in case of reset or POWERDOWN as shown in the following table. For CCC, the outputs control dependency on the LOCK is dependent on the state of the GPDs. Refer to the [GPD Operating Modes](#), page 43 for more details. The precision of the lock discrimination can be adjusted using the lock window controls.

The lock window represents the phase error window for lock assertion as a fraction of the divided reference clock period. The lock window can be adjusted between 1500 parts per million (ppm) and 16000 ppm. To avoid unnecessary toggling of LOCK signal, use the default Lock window setting which is 6000 ppm. If toggling of LOCK signal is observed, increase lock window setting to higher value. Lock window can be configured in RTG4 Clock Conditioning Circuit (CCC) configurator. The integration of the lock period can be adjusted using the lock counter. The lock counter or lock delay indicates the number of reference clock cycles to wait after the PLL is locked before asserting the lock signal. The lock delay is useful for avoiding false toggling of the lock signal. The lock counter can be configured between 32 and 1048576 cycles in multiples of 2. Refer to [Table 29](#), page 54 for more details.

The RTG4 fabric PLL uses a Phase and Frequency Detector (PFD) to measure the phase error between the divided reference clock and the divided feedback clock inputs to the PLL. When the measured phase error is within the parts per million lock window, the internal lock counter starts counting reference clock cycles until it reaches the selected lock count value. This allows additional time for the PLL output to further settle before asserting the lock output. The LOCKCNT value is pre-set to a value (1024 cycles) that allows the PLL to reach its normal output jitter, given a typical reference clock, while minimizing the PLL lock assertion time.

The lock window setting also determines the amount of phase error occurred at the PLL output before a PLL unlock event occurs. There are various scenarios which can cause the PLL to unlock, such as increased reference clock jitter, power supply noise, and specific PLL sensitivities to radiation effects or operating temperature increases, such as those described in Customer Notifications [PCN18009.7](#) and [CN19009](#). In these cases, selecting a larger lock window setting means that the PLL output exhibits a larger phase offset relative to the input reference clock prior to the unlock event.

Some PLL applications, such as source synchronous designs, or clock insertion delay cancellation, require a small phase offset between the input reference clock and the PLL output. In other cases, the application might only require the CCC outputs that are phase aligned with each other, and do not care whether the outputs are phase aligned to the input reference clock.

To minimize the phase offset, the following formula can be used to determine the minimum PLL lock window setting for a given PFD frequency:

Min LOCKWIN Setting (ppm) = MIN(ROUNDUP-TO-NEXT-SETTING (650 + 52.34*PFD-Rate-in-MHz), 6000) ppm

The PFD frequency is the divided input clock frequency sent to the PLL. It can be obtained from the Libero SoC CCC configuration GUI (Advanced Setting tab) where the input reference divider value is shown.

PFD Frequency = Input Reference Clock Frequency / Reference Divider value

For example, at a 25 MHz PFD rate, the minimum Lock Window Setting is:

MIN(ROUNDUP-TO-NEXT-SETTING (650 + 52.34*25), 6000) ppm
 = MIN (ROUNDUP-TO-NEXT-SETTING (1958.5), 6000)
 = MIN (2000, 6000) ppm
 = 2000 ppm

This means that the Lock Window setting should be at least 2000 ppm to maintain approximately a 5° phase offset between the divided reference clock and the feedback clock at the PLL PFD input, even prior to an unlock event.

For applications that do not require phase alignment between the input reference clock and the PLL output, the minimum value calculated for larger lock window settings can be used.

5.4.3.7 Fabric PLL Control Signals

A reset control signal, PLL_ARST_N, is provided to power-down the PLL core and asynchronously reset all its internal digital blocks. When the PLL_ARST_N signal is asserted Low, all the PLL outputs are driven Low. A bypass signal, PLL_BYPASS_N, is provided which powers-down the PLL core and bypasses it such that the PLL output tracks the reference clock. All the output clocks have the same phase in PLL Bypass mode. The PLL_BYPASS_N signal has precedence over the PLL_ARST_N signal.

A fabric PLL can be held in Power-down mode for the lowest quiescent current by asserting its power-down control signal, PLL_POWERDOWN_N. When the PLL_POWERDOWN_N signal is asserted Low, the PLL powers-down and its outputs are held Low. The PLL_POWERDOWN_N signal has precedence over all other control signals. The following table lists the control signal values for PLL output and power states. Each PLL has its own control signals available to drive from the user logic in the FPGA fabric.

Table 10 • Control Signals for PLL Output and Power State

PLL_POWERDOWN_N	PLL_BYPASS_N	PLL_ARST_N	PLL_OUT	PLL POWERED
0	X	X	Low	OFF
1	0	0	Low ¹	OFF ¹
1	0	1	Tracks RFCLK input	OFF
1	1	0	Low ¹	OFF ¹
1	1	1	Normal Operation	ON

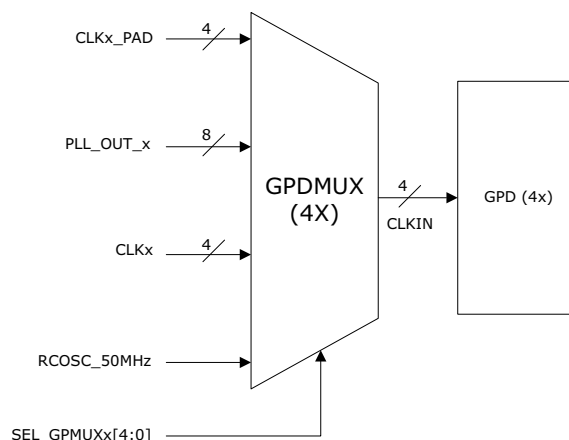
1. PLL_ARST_N powers-down the PLL core. With the auto-reset logic, PLL_ARST_N puts the PLL into reset, but only until the 1 μ s delay counter finishes. After this, the PLL comes out of reset even if PLL_ARST_N is asserted.

5.4.4 GPMUX

The clock source selection for generating the output clocks through general purpose divider (GPD) is performed using a general purpose multiplexer (GPMUX), refer to the preceding table. The output of the GPMUX is the clock input to the GPD. Each GPD has its own GPMUX that selects the clock source independently for each GPD. Each GPMUX selects the clock source from the fabric CCC clock sources (external dedicated I/Os, RC Oscillator, fabric clock) or PLL outputs, as shown in the following figure. The fabric PLL is bypassed, if none of the fabric CCC outputs uses PLL outputs.

Each GPMUX has a feature to invert the output of the GPMUX that can be controlled by writing to the appropriate fabric CCC configuration registers (FCCC_GPMUXx_CR, x = 0 to 3) through APB3 interface. The GPMUX is not a glitchfree MUX. The CGL to filter out glitches before sending the signal on the GLx/Yx output. Refer to the [Clock Gating Logic \(CGL\)](#), page 44 for more information on CGL. Refer to [Fabric CCC Configuration](#), page 49 for more information on fabric CCC configurations.

Figure 18 • GPMUX Input Clock Sources



5.4.5 GPD

GPDs divide the frequency of the source clock selected by the GPMUX by a factor of 1 to 255 to generate the desired fabric CCC output frequencies (EQ 3 and EQ 4). Each fabric CCC has four GPDs. The output of the GPDs can be used as the source for any of the four fabric CCC outputs. For instance, GPD0 can be used on a path to the GL1 output. The CCC macro configurator configures the GPD values to achieve the closest possible match to the desired output frequencies, based on the input frequency and the selected feedback loop.

$$\text{CLKOUT} = \text{CLKIN} / \text{GPDx_DIV}[7:0], \text{ where, GPDx_DIV}[7:0] = 1 \text{ to } 255$$

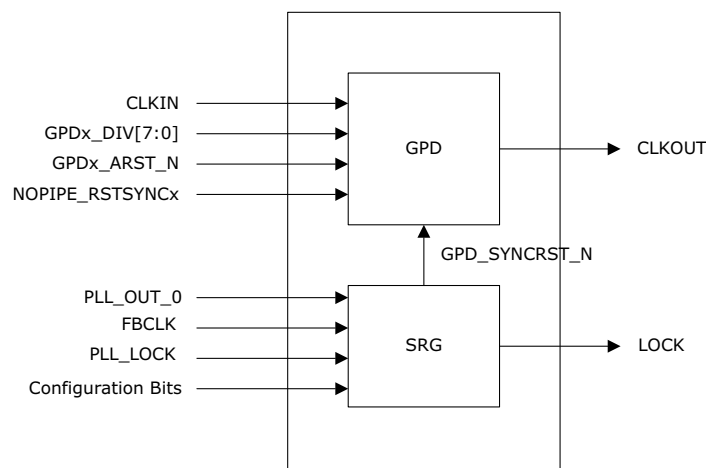
EQ 3

$$\text{CLKOUT} = \text{CLKIN}, \text{ for GPDx_DIV}[7:0] = 0$$

EQ 4

The following figure shows the GPD block diagram.

Figure 19 • GPD Block Diagram



The GPD behavior is outlined as follows:

- The duty cycle of the GPD output clock (CLKOUT) is 50%, if the duty cycle of the input clock (CLKIN) is 50%.
- The rising edge of CLKOUT always occurs as a result of a rising edge inputs on CLKIN.
- The falling edge of CLKOUT always occurs as a result of a rising edge on CLKIN, if the divider value is an even number ≥ 2 .
- The falling edge of CLKOUT always occurs as a result of a falling edge on CLKIN, if the divider value is an odd number or if divisor = 0.

The GPDs divide the phase of the non-zero phase PLL output by the division value. In other words, as shown in EQ 5, the actual phase shift is not the same as the PLL phase shift selected, if the GPD division value is not one.

$$\text{Actual Phase Shift} = (\text{Selected PLL output phase shift}) / (\text{GPD division value})$$

EQ 5

Each GPD has an asynchronous reset (GPD_ARST_N) and a synchronous reset (GPD_SYNCRST_N) for resetting the GPD, refer to the preceding figure. Both the resets are active Low. The GPD output clock (CLKOUT) is reset to Low when a reset signal is asserted. The asynchronous reset comes from the FPGA fabric and synchronous reset comes from the synchronous reset generator (SRG).

Note: When the GPD's in a CCC are used without using the PLL, then the GPD outputs can power-up to unpredictable output phase relative to the input clock. Hence, it is recommended to keep the GPDs in reset using GPD_ARST_N until the input clock becomes stable.

5.4.6 SRG

The SRG block sends synchronous reset signals to the GPDs when the PLL locks or on request by the System Controller. The purpose of the SRG block is to synchronize (or re-align) the outputs of the GPDs with the PLL input clock after the PLL lock is achieved. This block also produces a LOCK signal which is sent out to the FPGA fabric.

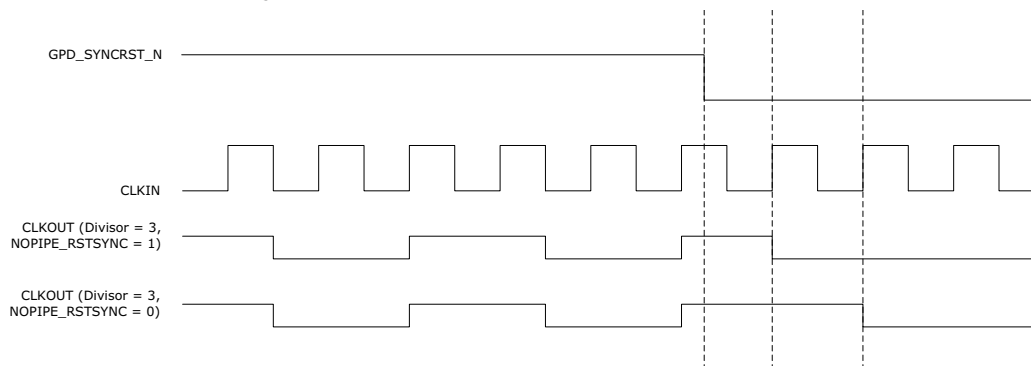
The SRG has an internal reset pipeline stage to get the correct phase alignment at the GPD output while using non-zero phase PLL output as the GPD input. The CCC macro configurator automatically sets the reset pipeline stage control signal based on the GPD input selection.

Whenever GPD_SYNCRST_N changes from High to Low:

- CLKOUT is reset to Low at the next rising edge of CLKIN, if the internal reset pipeline stage is not enabled.
- CLKOUT is reset to Low at the second rising edge of CLKIN, if the internal reset pipeline stage is enabled.

The following figure shows the GPD output clock behavior for synchronous reset assertion.

Figure 20 • Assertion of GPD Synchronous Reset – GPD_SYNCRST_N

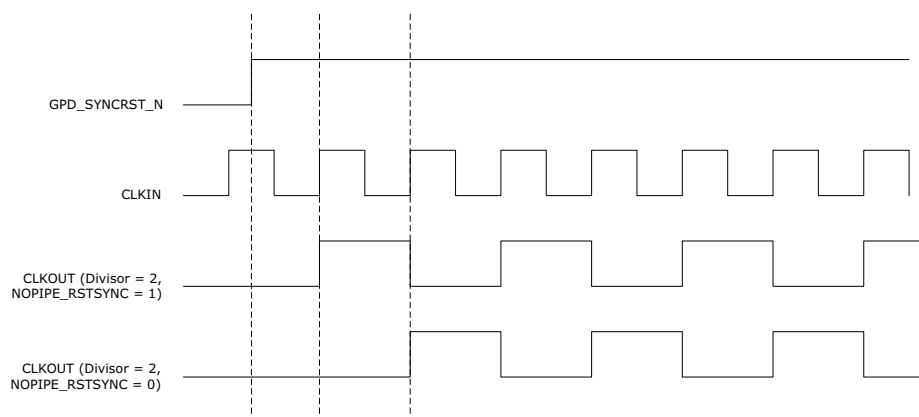


When releasing GPD_SYNCRST_N from Low to High:

- CLKOUT is clocked to High at the next rising edge of CLKIN, if the reset pipeline stage is not enabled.
- CLKOUT is clocked to High at the second rising edge of CLKIN, if the internal reset pipeline stage is enabled.

The following figure shows the GPD output clock behavior for synchronous reset release.

Figure 21 • Releasing of GPD Synchronous Reset – GPD_SYNCRST_N



5.4.6.1 GPD Operating Modes

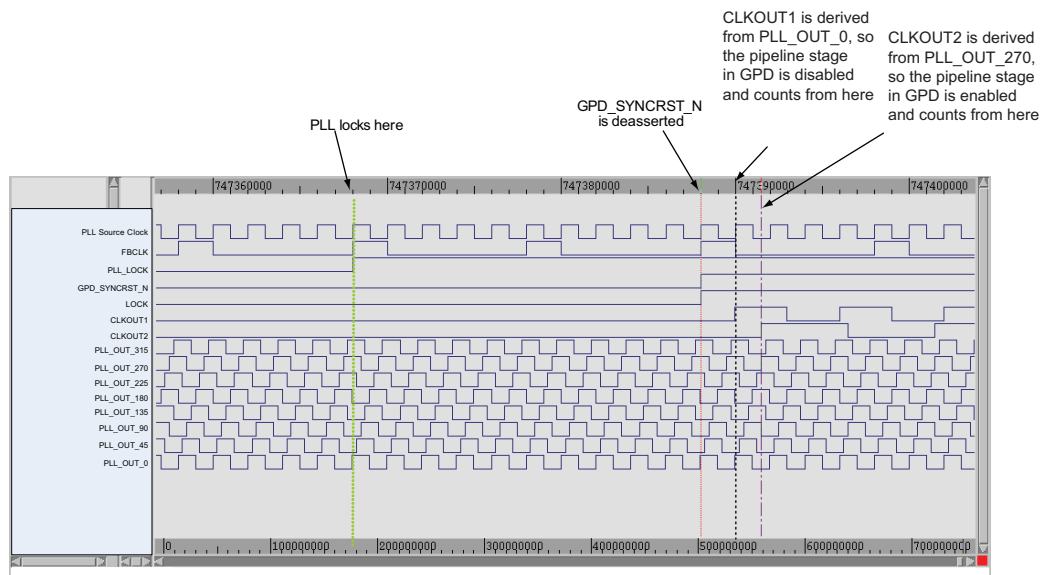
The GPDs support the following operating modes. For the GPDs that are driven by the PLL output clock, Microsemi recommends using either the first or second mode described as follows:

1. GPDs are held in reset after power-up, release and synchronize with the PLL reference clock after the PLL is locked:
 - If enabled, GPDs outputs are held in reset Low after power-up. Hence, the output(s) (GLx/Yx) driven by the GPDs are held Low (or High if inverted) after power-up.
 - After the PLL lock is achieved, GPD synchronous reset is released synchronously with the second FBCLK rising edge.
 - Does not apply to the output(s)/GPD(s) on the PLL external feedback path, if any.
 - Does not apply for the output(s)/GPD(s) not driven by one of the PLL 8 phases output.
 - If you select PLL Internal feedback then outputs will be resynchronized between each other after PLL locks, but will not be resynchronized with the PLL reference clock.

The following figure shows the GPDs outputs (CLKOUT1 and CLKOUT2) synchronization operation for this mode.

- The PLL source clock goes to PLL through RFDIV and the RFDIV division value is set to 4 (divided by 5).
- The FBDIV division value is set to 4 (divided by 5).
- CLKOUT1 is generated by a GPD whose input is PLL_OUT_0 and the GPD divider value is set to 3.
- CLKOUT2 is generated by another GPD whose input is PLL_OUT_270 and the GPD divider value is set to 5.

Figure 22 • GPD_SYNCRST_N Released after PLL Locks



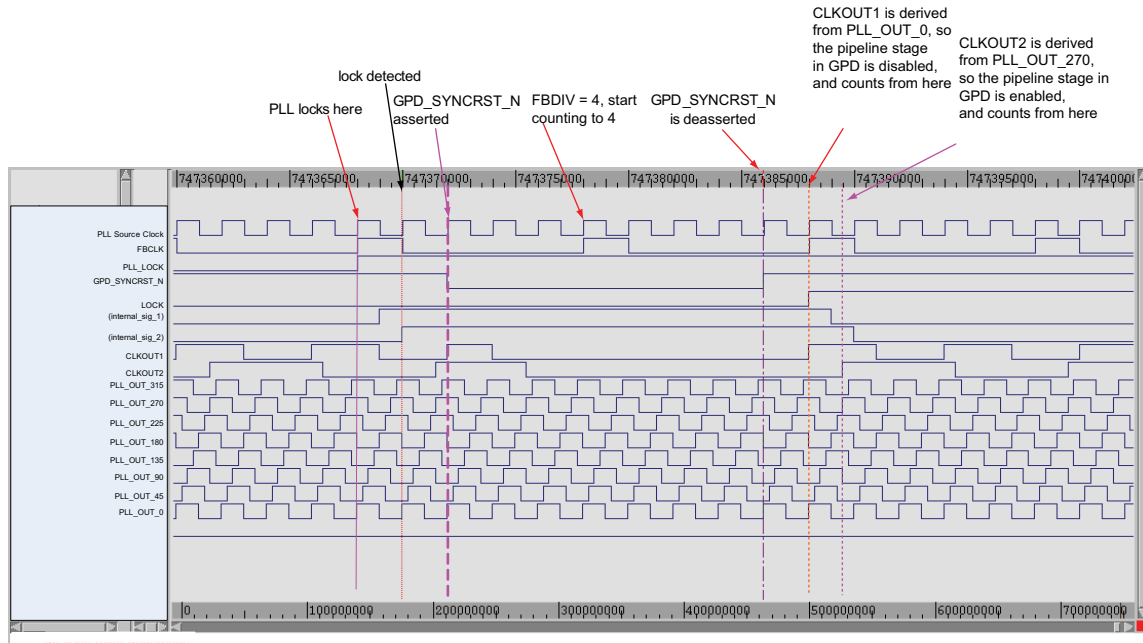
2. GPDs operate after power-up and resynchronize with the PLL reference clock after the PLL is locked:
 - If enabled, GPDs operate after power-up. Therefore, the outputs (GLx/Yx) driven by the GPDs are operational after power-up.
 - When the PLL is locked and detected by the rising edge of PLL_OUT_0, the SRG asserts the GPD_SYNCRST_N signal after completion of one PLL_OUT_0 clock cycle.
 - After GPD_SYNCRST_N is asserted, the SRG waits for the first rising edge of FBCLK to trigger the counting of (M - 1) clock edges of PLL_OUT_0 clock, where $M = \text{FBDIV}[5:0] + 1$.
 - The SRG releases GPD_SYNCRST_N on the (M - 1)th rising edge of PLL_OUT_0 clock to let GPD start counting on the Mth clock edge of PLL_OUT_0 clock. In other words, the release of GPD_SYNCRST_N restarts the GPDs on the PLL output clock edge, which correspond to the 2nd rising edge of FBCLK after the lock changes state from 0 to 1.
 - Does not apply to the output(s)/GPD(s) on the PLL external feedback path, if any.
 - Does not apply for the output(s)/GPD(s) not driven by one of the PLL 8 phases output.

- If you select PLL Internal feedback then outputs will be resynchronized between each other after PLL locks, but will not be resynchronized with the PLL reference clock.

The following figure shows the GPDs outputs (CLKOUT1 and CLKOUT2) synchronization operation for this mode.

- The PLL source clock goes to PLL through RFDIV and the RFDIV division value is set to 4 (divided by 5).
- The FBDIV division value is set to 4 (divided by 5).
- CLKOUT1 is generated by a GPD whose input is PLL_OUT_0 and the GPD divider value is set to 3.
- CLKOUT2 is generated by another GPD whose input is PLL_OUT_270 and the GPD divider value is set to 5.

Figure 23 • GPD Output Resynchronization after PLL Locks



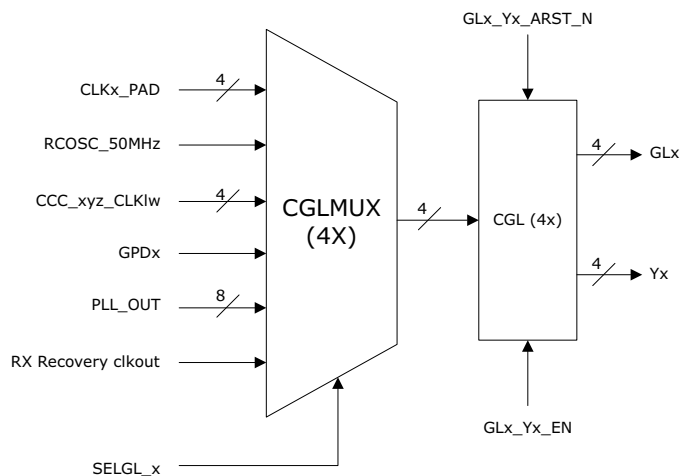
3. GPDs operate after power-up, no automatic resynchronization:
 - If enabled, GPDs operate after power-up. Hence the outputs (GLx/Yx) driven by the GPDs are operational after power-up.
 - There is no automatic resynchronization after the PLL lock.
 - All GPD(s) directly connected to one of the 11 input clocks (bypassing the PLL) are configured in this mode.

5.4.7 Clock Gating Logic (CGL)

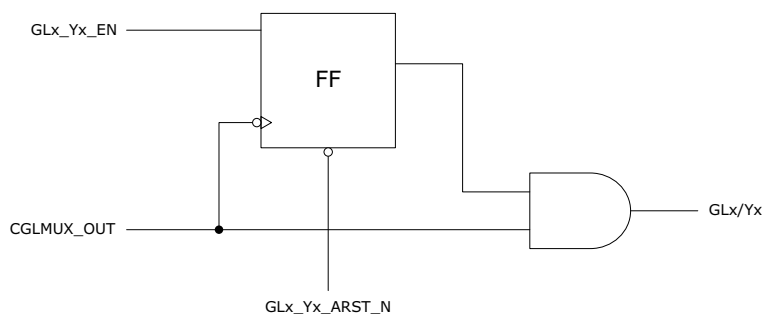
The clock source selection for generating the output clocks through clock gating logic (CGL) is performed using a clock gating logic multiplexer (CGLMUX), refer [Figure 16](#), page 31. The output of the CGLMUX is the clock input to the CGL. Each CGL has its own CGLMUX that allows to select the clock source independently for each CGL. Each CGLMUX selects the clock source from the fabric CCC clock sources (dedicated global I/Os, RC Oscillator, fabric clock), GPDx, SpaceWire RX recovery block, or PLL outputs, as shown in the following figure. The fabric PLL is bypassed, if none of the fabric CCC outputs uses PLL outputs. The CGLMUX is not a glitchfree MUX. You can use CGL to ensure glitchfree switching by de-asserting GLx_Yx_EN signal before changing the SELGLx signal. De-asserting GLx_Yx_EN signal stops the output clocks and drives low.

GLx_Yx_ARST_N is the asynchronous active low reset to the CGL and it forces the GLx/Yx to low. GLx_Yx_ARST_N must be forced low upon power-up. If GLx_Yx_ARST_N is asynchronous to the input clock, it may cause narrow pulse on GLx/Yx. During normal operation, use GLx_Yx_EN if GLx/Yx needs to be low.

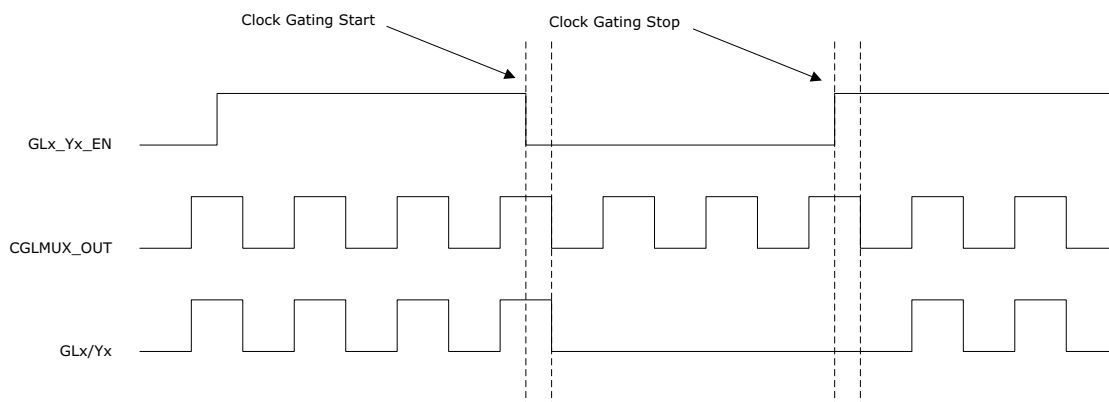
Note: If any clock on the device is to be stopped, it must be stopped low.

Figure 24 • CGLMUX Block Diagram

Clock Gating Logic (CGL) circuit is implemented using negative edge triggered flip-flop as shown in the following figure.

Figure 25 • CGL Circuit

The following figure shows the waveform for the CGL operation.

Figure 26 • CGL Operation Waveform

Each CGLMUX has a feature to invert the output of the CGLMUX that can be controlled by writing to appropriate fabric CCC configuration registers (FCCC_CGLMUXx_CR, x = 0 to 3) through APB3 interface. Refer to [Fabric CCC Configuration](#), page 49 for more information about fabric CCC configuration.

5.4.8 RX Clock Recovery (SpaceWire)

The RX Clock Recovery block is dedicated for SpaceWire application. SpaceWire uses two signal lines, Data and Strobe, generally driven by LVDS inputs. The RX Clock Recovery block generates RX clock based on the data and strobe inputs, which are from external I/Os as shown in the following figure. Either data signal line or strobe signal line changes its logical value in one clock cycle, not both of them. This allows for easy clock recovery with a good jitter tolerance by XORing the two signals line values to generate the clock to be used in the fabric as shown in the following figure. The generated RX Clock is radiation hardened and drives the hardened global clock network. The data signal will also be available to the fabric to perform SpaceWire protocol as shown in [Figure 28](#), page 46. It also includes de-glitching circuit to prevent any unwanted narrow clock pulse at output. The de-glitching circuit is used to filter either SET or system-level glitches. There are two blocks of RX Clock Recovery in each CCC block. For more information regarding SpaceWire Clock and Data Recovery in RTG4 FPGAs, see the [AC444: Implementing SpaceWire Clock and Data Recovery in RTG4 FPGA Application Note](#).

Each RX Clock Recovery block has its own DMUX and SMUX to provide two inputs for data and strobe as shown in the following figure. The Data and Strobe are selected from the dedicated CLKx_PAD input pads.

Figure 27 • SpaceWire Rx Clock Recovery Block Diagram

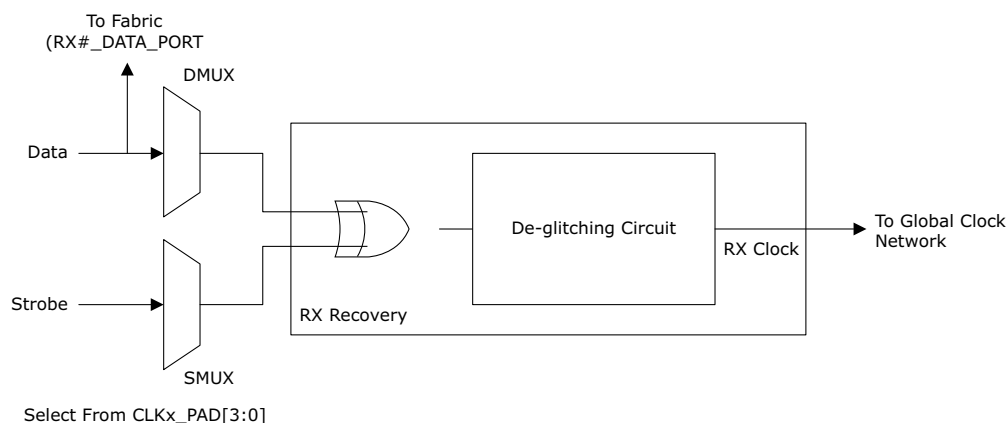
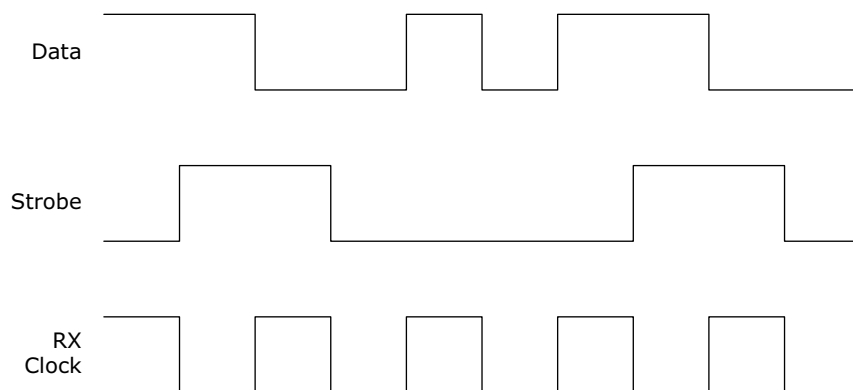


Figure 28 • SpaceWire Rx Clock Recovery Waveform



There is a total of 16 Data/Strobe I/O pairs that can be used for SpaceWire. Four of these 16 are DDRIOs. Refer to the [DS0130: RTG4 FPGA Pin Descriptions](#) for more information on the SpaceWire pins definition. Note that the DDRIO receivers are not hardened and when used for SpaceWire, there is limitation on the maximum achievable performance. Refer to the [DS0131: RTG4 FPGA Datasheet](#) for more details. Any clock glitches can be mitigated by the glitch filter within the SpaceWire circuit in each CCC.

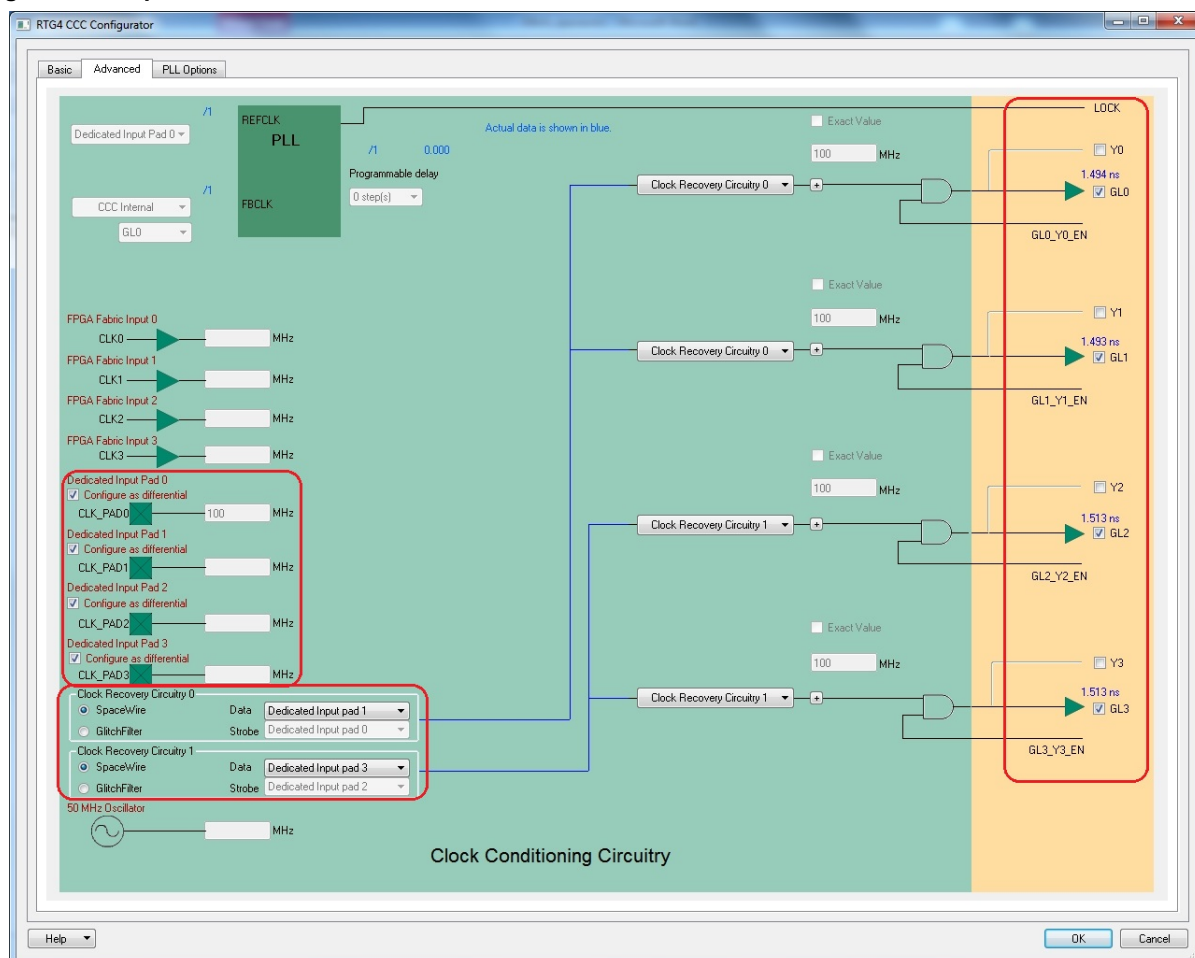
5.5 How to Use SpaceWire Mode and Glitch Filter Mode using RX Clock Recovery Block

The RX Clock Recovery Block generates an RX clock that can drive any one of the output clocks. The Clock Recovery Circuitry Blocks also include a de-glitching circuit to prevent any undesirable narrow clock pulse at output. This de-glitching circuit is always enabled in the clock recovery block. It will filter out a glitch of ~600ps or smaller. The delay value for this filter is not editable by the user, but rather it is PVT compensated between 600ps (typical) to 1ns (max) delay. The 600ps delay in the glitch filter is the same value used by the SLE SET Filter. The flip-flop SET filter is optional and is off by default. Use the following flow to configure the clock driven by the RX Clock Recovery.

1. Select the number of desired output clocks up to four different global clocks (GL0, GL1, GL2, and GL3) and four core clocks (Y0, Y1, Y2, and Y3).
2. For each selected output clock, select the desired reference (input) clock from which the output will be derived. It can be either from Recovery Circuit 0 or 1.
3. For SpaceWire mode, select the Clock Pad for the data input. The clock pad for the strobe input is automatically selected by the CCC configurator based on the clock pad selected for the data input as shown in the following figure.

Note: The dedicated input pad can be configured as either single ended or differential. Refer to the **RTG4 Clock Conditioning Circuit with PLL Configuration User Guide** for more information on the RX Clock Recovery options.

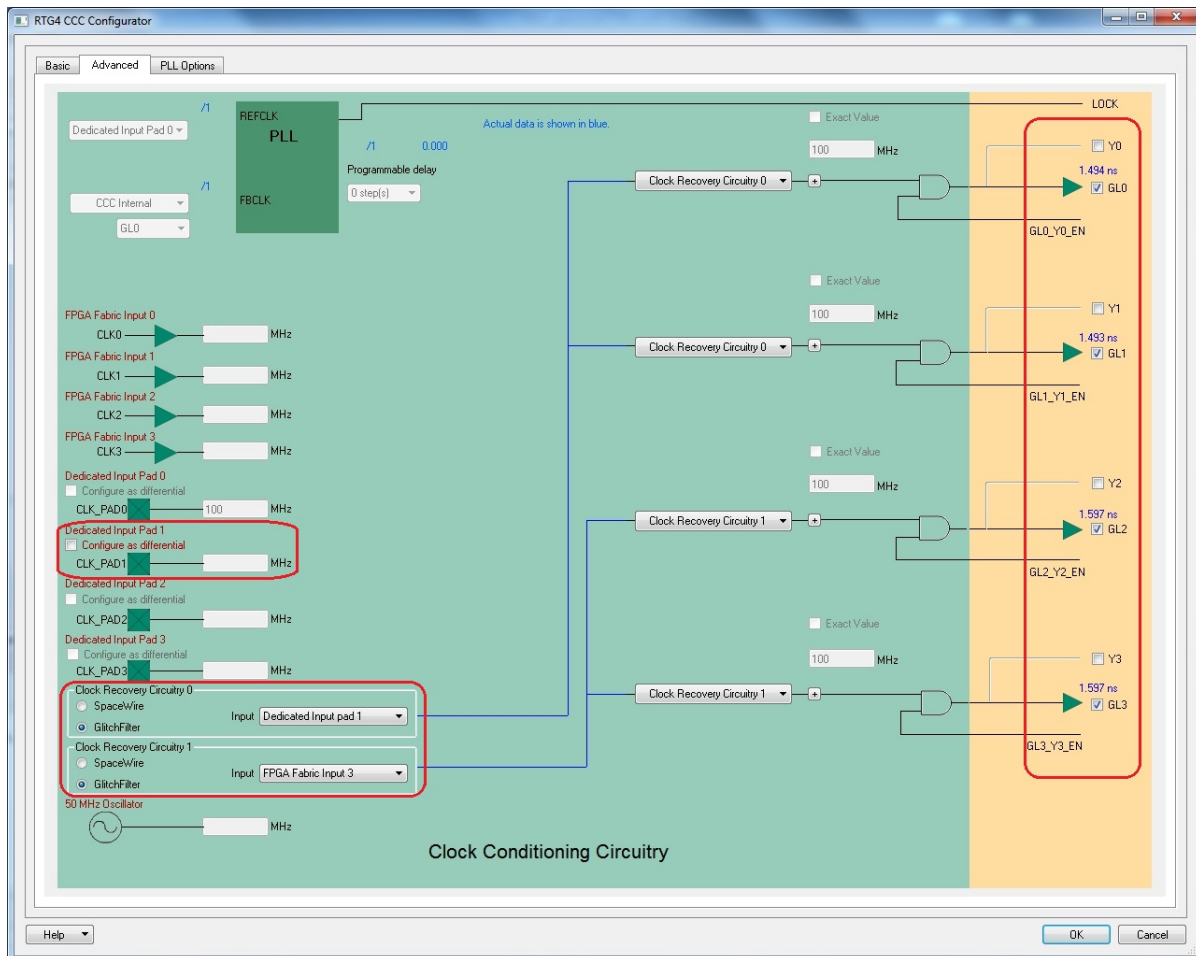
Figure 29 • SpaceWire



- For using Glitch Filter alone, select the Glitch Filter mode option in the configurator. The clock coming from dedicated input pad or FPGA fabric can be de-glitched using the glitch filter circuit.

Note: For RTG4 ES (Engineering Sample) devices, the Glitch Filtering mode is not available.

Figure 30 • Glitch Filter



5.6 Guidelines for using CCC with Simultaneous Switching Registers

The combination of high register utilization and toggling rate results high PLL jitter and subsequently the PLL may come out of lock. Use the following guidelines for stable PLL operation:

- Use external clock as PLL reference clock, wherever possible for stable PLL operation.
- PLL lock window must be set to maximum value.
- Use the CCC/PLL (CCC_SE0 or CCC_SE1) closest to the on-chip oscillator, if the reference clock has to come from on-chip oscillator. In case of farthest CCC/PLL is used in conjunction with on-chip oscillator, the PLL behaves as follows:
 - For 100% toggle rate: Up to 75% register utilization, the PLL Lock is stable and Jitter is within the specification.
 - For 98% register utilization: Up to 12.5% toggle rate, the PLL Lock is stable and Jitter is within the specification.

5.7 Fabric CCC Configuration

Fabric CCCs are configurable statically through flash configuration bits set in the programming bitstream or dynamically through an APB3 bus interface. The flash configuration bits are the configuration bits associated with programmed flash switches. The fabric CCC flash configuration bits provide the default operating state of the fabric CCC.

5.7.1 Fabric CCC Static Configuration

The CCC macro available in the IP Catalog of the Libero SoC design software provides a visual configuration wizard for a quick and easy way to configure the fabric CCC with desired the settings. The configuration registers are then initialized with values from static flash bits during power-up. All the registers can be re-configured dynamically (write operation) through APB3 interface.

5.7.2 Fabric CCC Dynamic Configuration

Each fabric CCC has an APB3 bus interface for dynamic configuration of the fabric CCC parameters without reprogramming the device. The fabric CCC configuration is controlled by radiation hardened volatile configuration registers, which are loaded with values from the flash configuration bits at power-up. An APB3 master can change the fabric CCC configuration by writing to the appropriate registers. Refer to [Fabric CCC Configuration Registers](#), page 49 for more information about fabric CCC control registers and their bit definitions.

Note: The APB slave interface is shared between two CCCs, which are present in each corner. Use CCCAPB macro, if the two CCCs from same corner needs to be dynamically configured.

For more information CCC Dynamic Configuration, see [AC458: RTG4 CCC Dynamic Configuration Application Note](#).

5.8 Fabric CCC Configuration Registers

The following table shows the fabric CCC configuration registers.

Table 11 • Fabric CCC Register Map

Register Name	Address	Description
Reserved	0x00	Reserved
FCCC_RFMUX_CR	0x01	RFMUX configuration register, see Table 12 , page 51
FCCC_RFDIV_CR	0x02	RFDIV configuration register, see Table 13 , page 51
FCCC_FBMUX_CR	0x03	FBMUX configuration register, see Table 14 , page 51
FCCC_FBDIV_CR	0x04	FBDIV configuration register, see Table 15 , page 51
Reserved	0x05	Reserved
Reserved	0x06	Reserved
FCCC_CGLMUX0_CR	0x07	CGLMUX0 configuration register, see Table 16 , page 51
Reserved	0x08	Reserved
FCCC_CGLMUX1_CR	0x09	CGLMUX1 configuration register, see Table 17 , page 52
Reserved	0x0A	Reserved
FCCC_CGLMUX2_CR	0x0B	CGLMUX2 configuration register, see Table 18 , page 52
Reserved	0x0C	Reserved
FCCC_CGLMUX3_CR	0x0D	CGLMUX3 configuration register, see Table 19 , page 52
FCCC_GPMUX0_CR	0x0E	GPMUX0 configuration register, see Table 20 , page 52
FCCC_GPMUX1_CR	0x0F	GPMUX1 configuration register, see Table 21 , page 53
FCCC_GPMUX2_CR	0x10	GPMUX2 configuration register, see Table 22 , page 53

Table 11 • Fabric CCC Register Map (continued)

Register Name	Address	Description
FCCC_GPMUX3_CR	0x11	GPMUX3 configuration register, see Table 23 , page 53
FCCC_GPD0_CR	0x12	GPD0 configuration register, see Table 24 , page 53
FCCC_GPD1_CR	0x13	GPD1 configuration register, see Table 25 , page 54
FCCC_GPD2_CR	0x14	GPD2 configuration register, see Table 26 , page 54
FCCC_GPD3_CR	0x15	GPD3 configuration register, see Table 27 , page 54
FCCC_PLL_CR0	0x16	PLL lock window configuration register, see Table 28 , page 54
FCCC_PLL_CR1	0x17	PLL lock counter configuration and lock status register, see Table 29 , page 54
FCCC_PLL_CR2	0x18	Write one pulse for reload of flash bits, see Table 30 , page 55
Reserved	0x19	Reserved
Reserved	0x1A	Reserved
FCCC_PLL_CR3	0x1B	PLL internal or external feedback path selection register, see Table 31 , page 55
FCCC_GPDS_SYNC_CR	0x1C	GPDs outputs realignment request configuration register, see Table 32 , page 55
FCCC_PLL_CR4	0x1D	PLL internal output divider configuration register, see Table 33 , page 55
Reserved	0x1E	Reserved
FCCC_PLL_CR5	0x1F	PLL internal reference clock divider configuration register, see Table 34 , page 55
FCCC_PLL_CR6	0x20	PLL internal feedback clock divider configuration register, see Table 35 , page 56
FCCC_PLL_CR7	0x21	PLL loop filter range configuration register, see Table 36 , page 56
FCCC_GPD0_SYNC_CR	0x22	GPD0 operating mode configuration register, see Table 37 , page 56
FCCC_GPD1_SYNC_CR	0x23	GPD1 operating mode configuration register, see Table 38 , page 57
FCCC_GPD2_SYNC_CR	0x24	GPD2 operating mode configuration register, see Table 39 , page 57
FCCC_GPD3_SYNC_CR	0x25	GPD3 operating mode configuration register, see Table 40 , page 58
FCCC_PDLY_CR	0x26	Programmable delay elements configuration register, see Table 41 , page 58
Reserved	0x27	Reserved
Reserved	0x28	Reserved
Reserved	0x29	Reserved
Reserved	0x2A	Reserved
FCCC_RX_RECOVERY_CR	0x2B	Rx clock recovery MUX select configuration register, see Table 42 , page 58

5.8.1 Fabric CCC Configuration Registers Bit Definitions

The following tables show the bit definitions of fabric CCC configuration registers.

Table 12 • FCCC_RFMUX_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
5	INVRF	R/W	When INVRF = 1, inverts the output of the RFMUX.
[4:0]	SELRF	R/W	RFMUX select lines for reference clock selection. Refer to Table 43 , page 59 for reference clock selection.

Table 13 • FCCC_RFDIV_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
[5:0]	RFDIV	R/W	RFDIV division value. $RFCLK = (\text{Input clock frequency}) / (RFDIV[5:0] + 1)$

Table 14 • FCCC_FBMUX_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
5	INVFB	R/W	When INVFB = 1, inverts the output of the FBMUX.
[4:0]	SELFB	R/W	FBMUX selects lines for feedback clock selection. Refer to Table 43 , page 59 for feedback clock selection.

Table 15 • FCCC_FBDIV_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
[5:0]	FBDIV	R/W	FBDIV division value $FBCLK = (\text{Input clock frequency}) / (FBDIV[5:0] + 1)$

Table 16 • FCCC_CGLMUX0_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
5	INVGL_0	R/W	CGLMUX0 output inversion 1: Inverts the CGLMUX0 output. 0: Does not invert the CGLMUX0 output.
[4:0]	SELGL_0	R/W	CGLMUX0 clock source selection. Refer to Table 43 , page 59 for clock source selection.

Table 17 • FCCC_CGLMUX1_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
5	INVGL_1	R/W	CGLMUX1 output inversion 1: Inverts the CGLMUX1 output. 0: Does not invert the CGLMUX1 output.
[4:0]	SELGL_1	R/W	CGLMUX1 clock source selection. Refer to Table 43 , page 59 for clock source selection.

Table 18 • FCCC_CGLMUX2_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
5	INVGL_2	R/W	CGLMUX2 output inversion 1: Inverts the CGLMUX2 output. 0: Does not invert the CGLMUX2 output.
[4:0]	SELGL_2	R/W	CGLMUX2 clock source selection. Refer to Table 43 , page 59 for clock source selection.

Table 19 • FCCC_CGLMUX3_CR

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
5	INVGL_3	R/W	CGLMUX3 output inversion 1: Inverts the CGLMUX3 output. 0: Does not invert the CGLMUX3 output.
[4:0]	SELGL_3	R/W	CGLMUX3 clock source selection Refer to Table 43 , page 59 for clock source selection.

Table 20 • FCCC_GPMUX0_CR

Bit Number	Field Name	R/W	Description
7	RESERVED	–	Reserved
6	NOPIPE_SYNCRST0	R/W	1: Disable pipe-line stage for GPD0 and synchronous reset is disabled. 0: Enable pipe-line stage.
5	INV_GPMUX0	R/W	1: Invert the output of the GPMUX0. 0: Disable the clock inversion of GPMUX0. The default value of this bit is 0.
[4:0]	SEL_GPMUX0	R/W	Select lines for the GPMUX0. Refer to Table 43 , page 59 for GPMUX0 output selection.

Table 21 • FCCC_GPMUX1_CR

Bit Number	Field Name	R/W	Description
7	RESERVED	–	Reserved
6	NOPIPE_SYNCRST1	R/W	1: Disable pipe-line stage for GPD1 and synchronous reset is disabled. 0: Enable pipe-line stage.
5	INV_GPMUX1	R/W	1: Invert the output of the GPMUX1. 0: Disable the clock inversion of GPMUX1. The default value of this bit is 0.
[4:0]	SEL_GPMUX1	R/W	Select lines for the GPMUX1. Refer to Table 43, page 59 for GPMUX1 output selection.

Table 22 • FCCC_GPMUX2_CR

Bit Number	Field Name	R/W	Description
7	RESERVED	–	Reserved
6	NOPIPE_SYNCRST2	R/W	1: Disable pipe-line stage for GPD2 and synchronous reset is disabled. 0: Enable pipe-line stage.
5	INV_GPMUX2	R/W	1: Invert the output of the GPMUX2. 0: Disable the clock inversion of GPMUX2. The default value of this bit is 0.
[4:0]	SEL_GPMUX2	R/W	Select lines for the GPMUX2. Refer to Table 43, page 59 for GPMUX2 output selection.

Table 23 • FCCC_GPMUX3_CR

Bit Number	Field Name	R/W	Description
7	RESERVED	–	Reserved
6	NOPIPE_SYNCRST3	R/W	1: Disable pipe-line stage for GPD3 and synchronous reset is disabled. 0: Enable pipe-line stage.
5	INV_GPMUX3	R/W	1: Invert the output of the GPMUX3. 0: Disable the clock inversion of GPMUX3. The default value of this bit is 0.
[4:0]	SEL_GPMUX3	R/W	Select lines for the GPMUX3. Refer to Table 43, page 59 for GPMUX3 output selection.

Table 24 • FCCC_GPD0_CR

Bit Number	Field Name	R/W	Description
[7:0]	GPD0_DIV	R/W	GPD0 division value. CLKOUT = CLKIN / GPD0_DIV[7:0], for GPD0_DIV[7:0] = 1 to 255 CLKOUT = CLKIN, for GPD0_DIV[7:0] = 0

Table 25 • FCCC_GPD1_CR

Bit Number	Field Name	R/W	Description
[7:0]	GPD1_DIV	R/W	GPD1 division value. CLKOUT = CLKIN / GPD1_DIV [7:0], for GPD1_DIV [7:0] = 1 to 255 CLKOUT = CLKIN, for GPD1_DIV [7:0] = 0

Table 26 • FCCC_GPD2_CR

Bit Number	Field Name	R/W	Description
[7:0]	GPD2_DIV	R/W	GPD2 division value. CLKOUT = CLKIN / GPD2_DIV [7:0], for GPD2_DIV [7:0] = 1 to 255 CLKOUT = CLKIN, for GPD2_DIV [7:0] = 0

Table 27 • FCCC_GPD3_CR

Bit Number	Field Name	R/W	Description
[7:0]	GPD3_DIV	R/W	GPD3 division value. CLKOUT = CLKIN / GPD3_DIV [7:0], for GPD3_DIV [7:0] = 1 to 255 CLKOUT = CLKIN, for GPD3_DIV [7:0] = 0

Table 28 • FCCC_PLL_CR0

Bit Number	Field Name	R/W	Description
[7:3]	RESERVED	–	Reserved
[2:0]	LOCKWIN	R/W	PLL lock window parameter setting. 000 = Reserved, 100 = 3000ppm 001 = Reserved, 101 = 4000ppm 010 = 1500ppm, 110 = 6000ppm 011 = 2000ppm, 111 = 16000ppm See Lock Generation Circuit , page 39 to determine the minimum Lock Window setting for a given CCC configuration.

Table 29 • FCCC_PLL_CR1

Bit Number	Field Name	R/W	Description
[7:5]	RESERVED	–	Reserved
4	LOCK	R	Indicates PLL is locked 1: PLL is locked. 0: PLL is not locked.
[3:0]	LOCKCNT	R/W	PLL lock delay counter setting. Indicates the number of reference clock cycles before the PLL Lock is asserted from Lock being detected. Lock Delay = $(2^{(LOCKCNT[3:0]+5)})$. For example, 0000=32,...,1111=1048576

Table 30 • FCCC_PLL_CR2

Bit Number	Field Name	R/W	Description
[7:1]	RESERVED	–	Reserved
0	RLD_STATIC_CONFIG	W	Writing 1 to this register reloads the CCC/PLL static configurations that are specified using the CCC/PLL software configurator. The load is performed synchronously to the register clock domain fixed at one clock cycle then will revert to low.

Table 31 • FCCC_PLL_CR3

Bit Number	Field Name	R/W	Description
[7:1]	RESERVED	–	Reserved
0	FSE	R/W	Selects between internal and external feedback 1: PLL internal feedback path selected. 0: External feedback path selected.

Table 32 • FCCC_GPDS_SYNC_CR

Bit Number	Field Name	R/W	Description
[7:1]	RESERVED	–	Reserved
0	SW_RESYNC_G PD	R/W	To request realignment of GPDs output. A pulse of 1 to 0 requests the realignment of GPDs output, and then the GPDs is resynchronized by resetting.

Table 33 • FCCC_PLL_CR4

Bit Number	Field Name	R/W	Description
[7:3]	RESERVED	–	Reserved
[2:0]	DIVQ	R/W	PLLs internal output divider division value. Valid divider values are 1, 2, 4, 8, 16, and 32: 000 = ÷1 001 = ÷2 010 = ÷4 011 = ÷8 100 = ÷16 101 = ÷32

Table 34 • FCCC_PLL_CR5

Bit Number	Field Name	R/W	Description
[7:6]	RESERVED	–	Reserved
[5:0]	DIVR	R/W	PLLs internal reference divider division value, which is equal to DIVR[5:0] binary value + 1. For example, 000000 = ÷1, ..., 111111 = ÷64

Table 35 • FCCC_PLL_CR6

Bit Number	Field Name	R/W	Description
[7:0]	DIVF	R/W	PLLs internal feedback divider division value, which is equal to DIVF[7:0] binary value + 1. For example, 00000000 = ÷1, ..., 11111111 = ÷256 The feedback divider basically causes the PLL reference clock to be multiplies up to a higher frequency internal to the PLL.

Table 36 • FCCC_PLL_CR7

Bit Number	Field Name	R/W	Description
[7:4]	RESERVED	–	Reserved
[2:0]	RANGE	R/W	PLL filter range. Used to set the PLL filter range as follow: 000 = BYPASS 001 = 10 - 16.8 MHz 010 = 16.8 - 26.8 MHz 011 = 26.8 - 43 MHz 100 = 43 - 69 MHz 101 = 69 - 110 MHz 110 = 110 - 175 MHz 111 = 175 - 200 MHz

Table 37 • FCCC_GPD0_SYNC_CR

Bit Number	Field Name	R/W	Description
[7:3]	RESERVED	–	Reserved
2	GPD_MODE_N0	R/W	GPD0 operating mode: 1: GPD0 is held in reset after power-up, released, and synchronized with the PLL reference clock after the PLL is locked. 0: GPD0 operates after power-up and resynchronizes with the PLL reference clock after the PLL is locked.
1	SRESET_GENEN0	R/W	1: Enable GPD0 synchronous reset generation when SW_RESYNC_GPD is set to 1. 0: GPD0 does not get reset when SW_RESYNC_GPD is set to 0.
0	RESET_GENEN0	R/W	Enable GPD0 synchronous reset generation when Lock changes from Low to High: 1: Enable GPD0 synchronous reset generation on PLL lock. 0: Disable GPD0 synchronous reset generation. GPD0 synchronous reset remains High after power-up, if synchronous reset generation is disabled.

Table 38 • FCCC_GPD1_SYNC_CR

Bit Number	Field Name	R/W	Description
[7:3]	RESERVED	–	Reserved
2	GPD_MODE_N1	R/W	GPD1 operating mode: 1: GPD1 is held in reset after power-up, released, and synchronized with the PLL reference clock after the PLL is locked. 0: GPD1 operates after power-up and resynchronizes with the PLL reference clock after the PLL is locked.
1	SRESET_GENEN1	R/W	1: Enable GPD1 synchronous reset generation when SW_RESYNC_GPD is set to 1. 0: GPD1 does not get reset when SW_RESYNC_GPD is set to 0.
0	RESET_GENEN1	R/W	Enable GPD1 synchronous reset generation when Lock changes from Low to High: 1: Enable GPD1 synchronous reset generation on PLL lock. 0: Disable GPD1 synchronous reset generation. GPD1 synchronous reset remains High after power-up, if synchronous reset generation is disabled.

Table 39 • FCCC_GPD2_SYNC_CR

Bit Number	Field Name	R/W	Description
[7:3]	RESERVED	–	Reserved
2	GPD_MODE_N2	R/W	GPD2 operating mode: 1: GPD2 is held in reset after power-up, released, and synchronized with the PLL reference clock after the PLL is locked. 0: GPD2 operates after power-up and resynchronizes with the PLL reference clock after the PLL is locked.
1	SRESET_GENEN2	R/W	1: Enable GPD2 synchronous reset generation when SW_RESYNC_GPD is set to 1. 0: GPD2 does not get reset when SW_RESYNC_GPD is set to 0.
0	RESET_GENEN2	R/W	Enable GPD2 synchronous reset generation when Lock changes from Low to High: 1: Enable GPD2 synchronous reset generation on PLL lock. 0: Disable GPD2 synchronous reset generation. GPD2 synchronous reset remains High after power-up, if synchronous reset generation is disabled.

Table 40 • FCCC_GPD3_SYNC_CR

Bit Number	Field Name	R/W	Description
[7:3]	RESERVED	–	Reserved
2	GPD_MODE_N3	R/W	GPD3 operating mode: 1: GPD3 is held in reset after power-up, released, and synchronized with the PLL reference clock after the PLL is locked. 0: GPD3 operates after power-up and resynchronizes with the PLL reference clock after the PLL is locked.
1	SRESET_GENEN3	R/W	1: Enable GPD3 synchronous reset generation when SW_RESYNC_GPD is set to 1. 0: GPD3 does not get reset when SW_RESYNC_GPD is set to 0.
0	RESET_GENEN3	R/W	Enable GPD3 synchronous reset generation when Lock changes from Low to High: 1: Enable GPD3 synchronous reset generation on PLL lock. 0: Disable GPD3 synchronous reset generation. GPD3 synchronous reset remains High after power-up, if synchronous reset generation is disabled.

Table 41 • FCCC_PDLY_CR

Bit Number	Field Name	R/W	Description
7	RESERVED	–	Reserved
6	RF_DLINE	R/W	If RF_DLINE = 1, delay setting SEL_PLL_DLINE[5:0] is applied to the delay line in the PLL reference clock path (RDLY). The feedback clock path delay line (FBDLY) is set to 0. This enables clock-delay. If RF_DLINE = 0, delay setting SEL_PLL_DLINE[5:0] is applied to the delay line in the PLL feedback clock path (FBDLY). The reference clock path delay line (RDLY) is set to 0. This enables clock-acceleration.
[5:0]	SEL_PLL_DLINE	R/W	Programmable delay lines setting. If set to 6'b000000, the clock delay lines on both reference and feedback paths are bypassed.

Table 42 • FCCC_RX_RECOVERY_CR

Bit Number	Field Name	R/W	Description
[7:4]	SEL_RX1	R/W	MUX select on DMUX and SMUX for RxClock Recovery1.
[3:0]	SEL_RX0	R/W	MUX select on DMUX and SMUX for RxClock Recovery0.

5.8.2 Fabric CCCs Multiplexers Selection Control

Different desired output phases can be configured dynamically by controlling the different mux selection inputs as shown in the following figure. For example, the GPMUXx (x = 0 to 3) selection is dynamically configured through the FCCC_GPMUXx register which has SEL_GPMUXx as select line that can be used to select different output phases. Refer to the [Fabric CCC Configuration Registers](#), page 49 for more information on the different fabric CCC Configuration registers.

Table 43 • Fabric CCCs Multiplexers Selection Control

MUX Selection Input	GPMUXx Output, select port is SEL_GPMUXx (x = 0 to 3)	CGLMUXx Output, select port is SELGL_x (x = 0 to 3)	RFMUX Output, select port is SELRF	FBMUX Output, select port is SELFB
0	PLL_OUT_315	PLL_OUT_315	PLL_OUT_315	PLL_OUT_315
1	PLL_OUT_270	PLL_OUT_270	PLL_OUT_270	PLL_OUT_270
2	PLL_OUT_225	PLL_OUT_225	PLL_OUT_225	PLL_OUT_225
3	PLL_OUT_180	PLL_OUT_180	PLL_OUT_180	PLL_OUT_180
4	PLL_OUT_135	PLL_OUT_135	PLL_OUT_135	PLL_OUT_135
5	PLL_OUT_90	PLL_OUT_90	PLL_OUT_90	PLL_OUT_90
6	PLL_OUT_45	PLL_OUT_45	PLL_OUT_45	PLL_OUT_45
7	PLL_OUT_0	PLL_OUT_0	PLL_OUT_0	PLL_OUT_0
8	Reserved	GPD0	Reserved	Reserved
9	Reserved	GPD1	Reserved	Reserved
10	Reserved	GPD2	Reserved	Reserved
11	Reserved	GPD3	Reserved	Reserved
12	logic-0	logic-0	logic-0	logic-0
13	logic-0	logic-0	logic-0	logic-0
14	logic-1	logic-1	logic-1	logic-1
15	logic-1	logic-1	logic-1	logic-1
16	CCC_xyz_CLKI0	CCC_xyz_CLKI0	CCC_xyz_CLKI0	CCC_xyz_CLKI0
17	CCC_xyz_CLKI1	CCC_xyz_CLKI1	CCC_xyz_CLKI1	CCC_xyz_CLKI1
18	CCC_xyz_CLKI2	CCC_xyz_CLKI2	CCC_xyz_CLKI2	CCC_xyz_CLKI2
19	CCC_xyz_CLKI3	CCC_xyz_CLKI3	CCC_xyz_CLKI3	CCC_xyz_CLKI3
20	CLK0	CLK0	CLK0	CLK0
21	CLK1	CLK1	CLK1	CLK1
22	CLK2	CLK2	CLK2	CLK2
23	CLK3	CLK3	CLK3	CLK3
24	logic-0	logic-0	logic-0	logic-0
25	logic-0	Reserved	logic-0	logic-0
26	logic-0	Reserved	logic-0	logic-0
27	RCOSC_50 MHz	RCOSC_50 MHz	RCOSC_50 MHz	RCOSC_50 MHz
28	logic-0	logic-0	logic-0	logic-0
29	logic-0	logic-0	logic-0	logic-0
30	logic-1	logic-1	logic-1	logic-1
31	logic-1	logic-1	logic-1	logic-1

5.9 How to Use Fabric CCC(s)

The Libero SoC IP Catalog now includes two RTG4 Fabric CCC configurator cores:

- RTG4 Clock Conditioning Circuit (RTG4FCCC)
- RTG4 FCCC with Enhanced PLL Calibration

The following sections describes the configuration of the standard Fabric CCC core as well as the FCCC core with enhanced PLL calibration.

The standard CCC macro configurator enables configuration of the CCC/PLL blocks available in RTG4 devices. The CCC macro must be instantiated from Libero IP Catalog into the design in order to use a fabric CCC. Multiple CCC macros must be instantiated in order to use more than one fabric CCC in a design. The FCCC with Enhanced PLL Calibration configurator core is used to remove the PLL lock stability dependence on the operating temperature during normal operation, per **CN19009**. This enhanced PLL calibration CCC core configures two CCC/PLL blocks at a time, per device corner, as shown in [Figure 13](#), page 27. Therefore, the user must note the CCC location and CCC input reference clock input pin when configuring CCC_0 and CCC_1 per core instance. The enhanced PLL calibration CCC core also includes fabric logic wrapped around the two dynamic CCC instances per core, to perform PLL calibration upon power-up or PLL reset/powerdown input de-assertion.

After reviewing the usage details for the standard FCCC with PLL below, see [Using RTG4 FCCC with Enhanced PLL Calibration](#), page 66 for the usage details. The CCC configurator includes the following tabs:

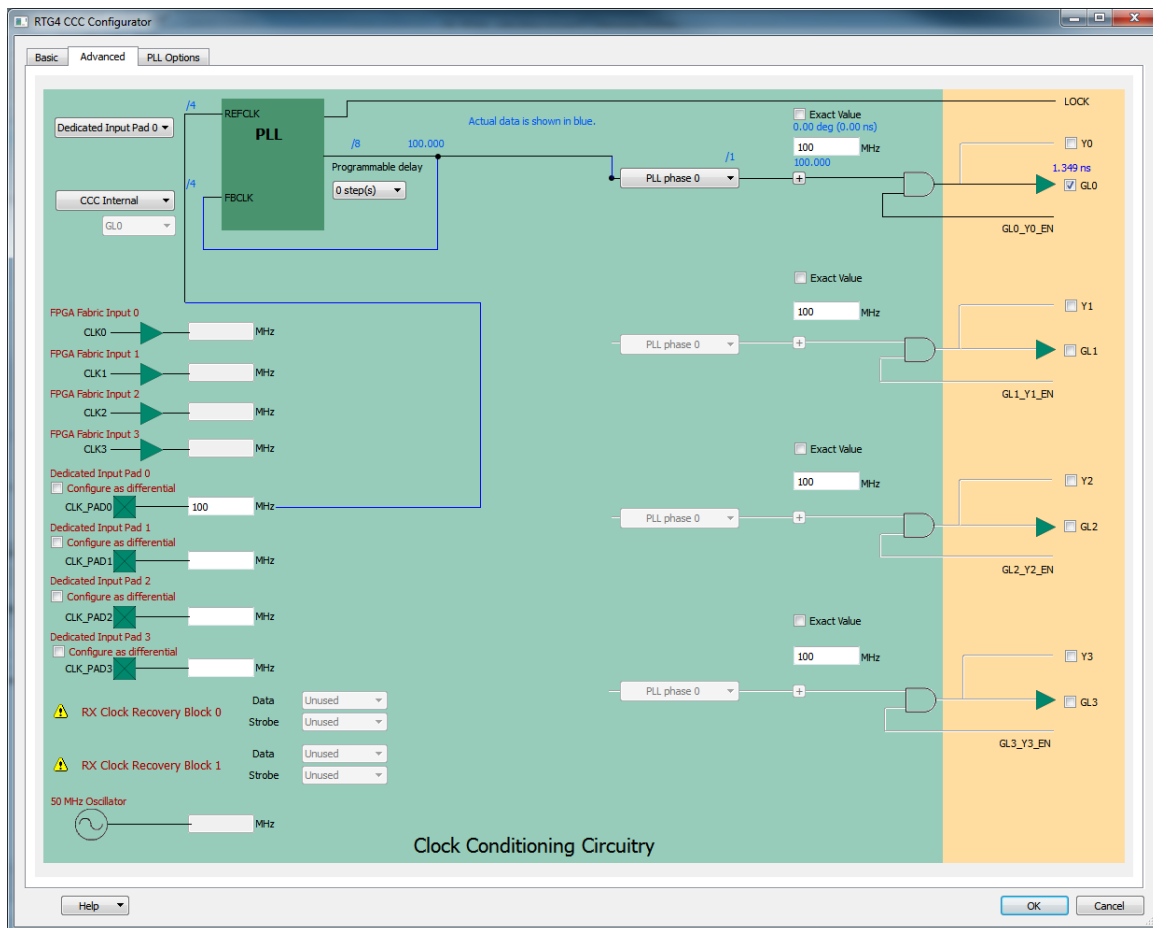
- Basic
- Advanced / PLL Options

5.9.1 Basic

Use the Basic tab to set basic clock configurations such as the reference clock source and frequency, number of desired output clocks (up to four), and the required output frequency for each selected output clock. In Basic Options configuration, the PLL is always used to generate the output frequency. The feedback of the PLL is internal to the CCC. Refer to the **RTG4 Clock Conditioning Circuit with PLL Configuration User Guide** for more information on the CCC/PLL Basic Configurations options.

5.9.2 Advanced Options

The **Advanced** tab inherits settings from the basic tab as shown in the following figure. Modifying a parameter in the Advanced tab that cannot be reflected in the Basic tab results in a warning in the Basic tab.

Figure 31 • Advanced Options

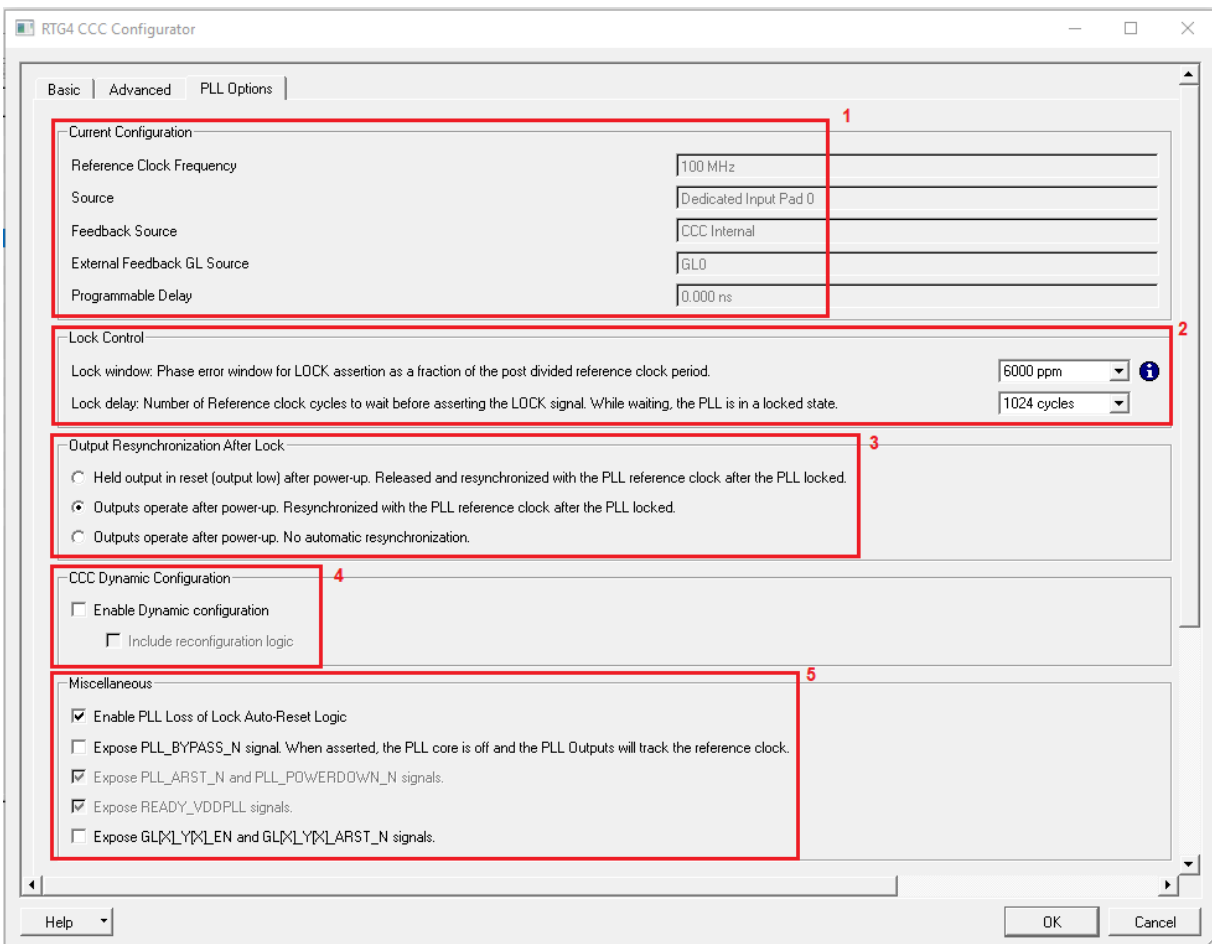
For advanced use-cases, Microsemi recommends using the following flow to configure the clock generated by the CCC (flow proceeds from right to left in the GUI):

1. Select the number of desired output clocks (up to four).
2. For each selected output clock, set the required output frequency.
3. For each selected output clock, select the desired reference (input) clock from which the output will be derived. It can be either:
 - One of CCC input clocks (PLL bypass mode).
 - One of 8 PLL output phases.
 - One of the two Clock Recovery Circuitry (0 or 1)
4. If required:
 - Select the PLL reference clock source and frequency
 - Select the PLL feedback source
5. Enter the frequency of each selected source clock(s) (either as the PLL reference or direct source for the output). The configurator uses those frequencies to compute the division factor of the PLL reference and feedback dividers as well as the GPD dividers.

5.9.3 PLL Options

The following figure shows the PLL options in RTG4 CCC configurator.

Figure 32 • PLL Options



1. PLL Current configuration: The Current configuration section of the PLL Options tab summarizes the PLL current configuration according to the selections made in the Basic tab and Advanced tab of the CCC configurator.

2. PLL Lock Control: Set the PLL lock control parameters, Lock window, and Lock delay, as appropriate for the design.
 - Lock Window—enables configuration of the maximum phase error allowed for the PLL to indicate that it has been locked. The lock window is expressed as ppm of the RFCLK frequency.
 - Lock Delay—enables configuration of the number of RFCLK clock cycles by which the lock is delayed after the PLL has reached the lock condition.

Note: If toggling of LOCK signal is observed, increase the LOCK window setting to higher value.

Note: See [Lock Generation Circuit](#), page 39 to determine the minimum Lock Window setting for a given CCC configuration.

3. Output Resynchronization configuration: Each fabric CCC contains four GPDs. The GPDs source and division settings are automatically configured based on the frequency requirement specified in the CCC configurator. Microsemi recommends to resynchronize the GPDs after the PLL lock is achieved.
4. CCC Dynamic configuration: APB slave interface can be exposed to fabric. For more information see [Fabric CCC Configuration](#), page 49.
5. Miscellaneous Options: The following control signals can be exposed to the FPGA fabric.

Enable PLL Loss of Lock Auto Reset Logic – This option is enabled by default whenever the PLL is used. When enabled, the options to "Expose PLL_ARST_N and PLL_POWERDOWN_N signals" and "Expose READY_VDDPLL signal" are enabled and grayed-out. Disabling this option will result in a Warning Exclamation Icon with a tool-tip message that states: Warning: The insertion of PLL loss of lock auto-reset logic has been disabled. Auto-reset is helpful in the event the PLL lock is lost during operation. For more information, review Customer Notifications: **19009** and **18009.7** on the Microsemi website to understand the consequences of disabling this option.

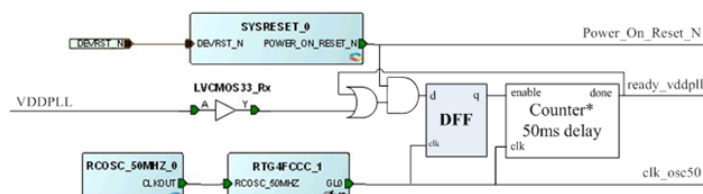
Expose PLL_BYPASS_N signal – When selected, the input signal PLL_BYPASS_N is exposed to the FPGA Fabric. When this signal is asserted, the PLL core is turned off and the PLL outputs track the reference clock. This signal is active low.

Expose PLL_ARST_N and PLL_POWERDOWN_N signals – When selected, the signals PLL_ARST_N and PLL_POWERDOWN_N are exposed to the FPGA Fabric. When PLL_ARST_N is asserted, the PLL core is turned off and the PLL outputs are low. With the auto-reset logic, PLL_ARST_N puts the PLL into reset until the 1 μ s delay counter finishes. After this, the PLL comes out of reset even if PLL_ARST_N is asserted. When PLL_POWERDOWN_N is asserted, PLL is off and is in the lowest power consumption mode. The PLL outputs are low. Both signals are active low. When PLL Internal Feedback mode is selected, both PLL_ARST_N and PLL_POWERDOWN_N signals are exposed.

Expose READY_VDDPLL signals – When selected, the READY_VDDPLL signal is exposed to the FPGA Fabric. When READY_VDDPLL is low, the PLL core is turned off and the PLL outputs are low. Tie READY_VDDPLL to high if you are certain that VDDPLL will not be the last supply to ramp up, per the [DS0131: RTG4 FPGA Datasheet](#) and [AC439 & AC453: RTG4 FPGAs Board Design and Layout Guidelines Application Note](#) recommendations. When PLL Internal Feedback mode is selected, READY_VDDPLL is exposed.

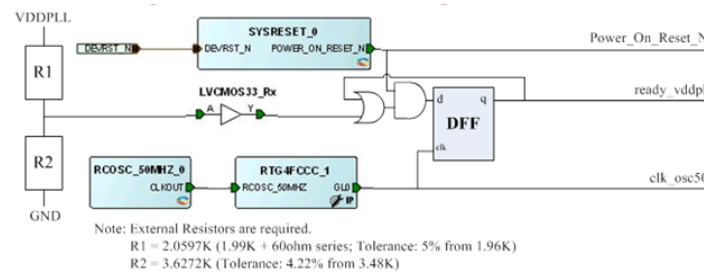
For applications which cannot meet the guideline of ensuring VDDPLL is not the last supply to ramp up, select this option to expose READY_VDDPLL and drive the input from one of the two suggested circuits described:

Method – 1 (No external resistors required)



* counter needs to be hardened. No internal generated clock or reset signal. Reset or set should be through data path.
When "enable" = 0, "done" = 0. First stage FF data input and counter output should be gated by "enable" signal.
When "enable" = 1, "done" = 1 when counter reaches to full count.

Method – 2 (External resistor required)



The two methods shown are provided in order to allow for the following trade-offs, though both implementations will allow the PLL to operate properly for all power sequences and power supply ramp rates:

- **Method - 1:** Does not require any external resistors on the board, but requires an extra 50 mS counter on the device.
- **Method - 2:** Requires external resistors on the board, but allows for improved PLL Lock times with typical VDDPLL ramp rates if VDDPLL is ramped later than VDD, VPP and VDDI (for the bank that has the READY_VDDPLL pin in it) as it does not include the 50 mS counter.

Expose GL[X]_Y[X]_EN and GL[X]_Y[X]_ARST_N signals—When selected, the GL[X]_Y[X]_EN and GL[X]_Y[X]_ARST_N signals are exposed to the FPGA fabric. By default, these signals are not exposed.

Expose GPD[X]_ARST_N signals for all used GPDs—This option is only available when the input clock to the GPD(s) is not sourced from the PLL. For example, a CCC output configured to use an input clock source coming from a Dedicated Input, a Fabric Input, or an internal oscillator will cause this option to appear and be enabled. When selected, the GPD[X]_ARST_N signals are exposed for all used GPDs. For CCC output configurations, which use the PLL output, this option is not listed in the PLL Options tab because the GPD reset signal is used inside the CCC to achieve the desired "Output Resynchronization After Lock" setting. This input has a built-in glitch filter to mitigate the impact of Single Event Transients on this signal.

These control signals are de-asserted by the configurator when they are not exposed to the FPGA fabric.

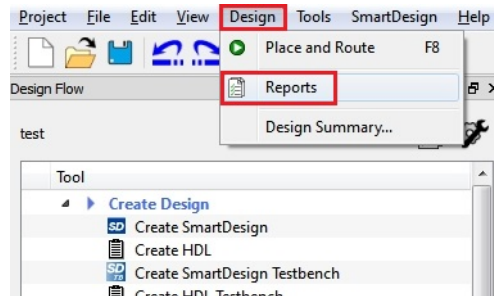
The configurator automatically tries to compute a configuration that meets the frequency requirements and all the internal CCC/PLL constraints. If the configurator is unable to find an exact solution for all requirements, it finds a configuration that globally minimizes the error between the required and actual frequency. Refer to the **RTG4 Clock Conditioning Circuit with PLL Configuration User Guide** for more information on the CCC/PLL Advanced Configurations and PLL options.

5.9.4 PLL/CCC Configuration Register Report

This section helps to view the PLL/CCC configuration registers report. The PLL/CCC configuration register values get affected when the PLL/CCC configurator is run and modify the values and settings in the configurator. If the user generates the SmartDesign then PLL/CCC configuration report is generated. The following steps describes how to generate the PLL/CCC configuration register report:

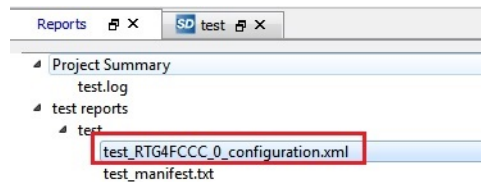
1. Click the **Design** tab and then click **Reports**.

Figure 33 • Design Window



2. In the **Reports** tab, double-click <Project name>_RTG4CCC_x_configuration.xml file.

Figure 34 • PLL/CCC Configuration Report File



The following figure shows a sample CCC registers configuration report.

Figure 35 • CCC Registers Configuration Report

Project Summary

test.log

test reports

test

test_RTG4FCCC_0_configuration.xml

test_manifest.txt

CCC Registers Configured Value

Register Name	Address	Value	Description
Reserved	0x00	0x00	Reserved
FCCC_RFMUX_CR	0x01	0x10	RFMUX configuration register
FCCC_RFDIV_CR	0x02	0x03	RFDIV configuration register
FCCC_FBMUX_CR	0x03	0x07	FBMUX configuration register
FCCC_FBDIV_CR	0x04	0x03	FBDIV configuration register
Reserved	0x05	0x00	Reserved
Reserved	0x06	0x00	Reserved
FCCC_CGLMUX0_CR	0x07	0x07	CGLMUX0 configuration register
Reserved	0x08	0x00	Reserved
FCCC_CGLMUX1_CR	0x09	0x18	CGLMUX1 configuration register
Reserved	0x0a	0x00	Reserved
FCCC_CGLMUX2_CR	0x0b	0x18	CGLMUX2 configuration register
Reserved	0x0c	0x00	Reserved
FCCC_CGLMUX3_CR	0x0d	0x18	CGLMUX3 configuration register
FCCC_GPMUX0_CR	0x0e	0x4c	GPMUX0 configuration register
FCCC_GPMUX1_CR	0x0f	0x4c	GPMUX1 configuration register
FCCC_GPMUX2_CR	0x10	0x4c	GPMUX2 configuration register
FCCC_GPMUX3_CR	0x11	0x4c	GPMUX3 configuration register
FCCC_GPD0_CR	0x12	0x01	GPD0 configuration register
FCCC_GPD1_CR	0x13	0x01	GPD1 configuration register
FCCC_GPD2_CR	0x14	0x01	GPD2 configuration register
FCCC_GPD3_CR	0x15	0x01	GPD3 configuration register
FCCC_PLL_CR0	0x16	0x06	PLL lock window configuration register
FCCC_PLL_CR1	0x17	0x05	PLL lock counter configuration and lock status register
FCCC_PLL_CR2	0x18	0x00	Write 1 pulse for reload of flash bits
Reserved	0x19	0x00	Reserved
Reserved	0x1a	0x00	Reserved
FCCC_PLL_CR3	0x1b	0x00	PLL internal or external feedback path selection register
FCCC_GPDS_SYNC_CR	0x1c	0x00	GPDs outputs realignment request configuration register
FCCC_PLL_CR4	0x1d	0x03	PLL's internal output divider configuration register
Reserved	0x1e	0x00	Reserved
FCCC_PLL_CR5	0x1f	0x00	PLL's internal reference clock divider configuration register

5.9.5 Using RTG4 FCCC with Enhanced PLL Calibration

Enhanced PLL calibration is a functional extension to the RTG4 FCCC core described in [How to Use Fabric CCC\(s\)](#), page 60. Enhanced PLL calibration provides lock stability that is not dependent on the operating junction temperature. Continued usage of the PLL instantiated through the standard RTG4 FCCC core means that the PLL lock stability will be dependent on the junction temperature rise during operation, as discussed in [CN19009](#). Designers are encouraged to review the CN. They should use the Temperature Rise Window calculator spreadsheet available on the RTG4 website to confirm the supported temperature rise window for each PLL used in their design.

Starting with Libero SoC v11.9 SP5 and v12.3, designers have the choice to continue designing with the standard RTG4 FCCC core, described in the previous section, or migrate to the RTG4 FCCC with Enhanced PLL calibration, also known as RTG4FCCCECALIB. The enhanced PLL calibration is only supported for the single PLL mode and is not available for the triplicated PLL, as described in [CN19009B](#). Therefore, the PLL Internal feedback mode is not available in the RTG4 FCCC with Enhanced PLL calibration core.

Migration from the standard RTG4 FCCC core to the RTG4 FCCC with Enhanced PLL Calibration is recommended in the following scenario:

- The non-triplicated RTG4 Fabric PLL is used in the design and the application must support operation at a junction temperature rise window beyond that predicted by the [PLL Temp Rise Window calculator](#) described in [CN19009](#).

Designs using the FCCC in configurations that bypass the PLL do not need to migrate to the enhanced PLL calibration core unless that CCC shares a device corner with a PLL requiring the enhanced PLL calibration core.

Existing designs can be migrated by following the steps described in section 3 of the [Libero SoC v11.9 SP5 Release Notes](#).

Refer to the RTG4 FCCC with Enhanced PLL Calibration Configurator User's Guide ([UG0887](#)) for details on using the configurator GUI.

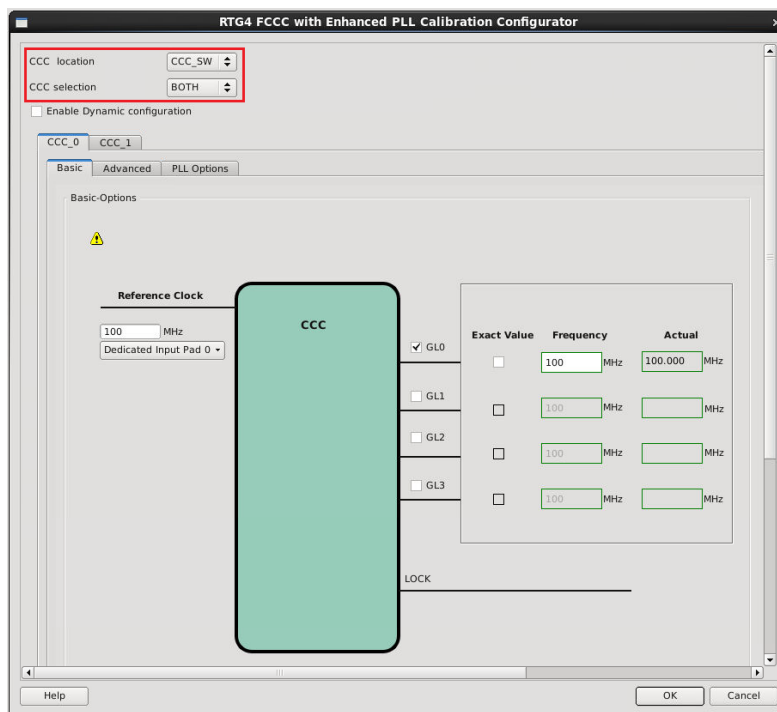
There are two major differences to note when using the enhanced PLL calibration core versus the standard FCCC core:

- The RTG4 FCCC with Enhanced PLL calibration core applies to both CCCs in a device corner.
- The core generates additional fabric logic around the CCC instance(s) to perform PLL calibration.

5.9.5.1 Selecting CCC Location and CCC Instance(s) being Configured

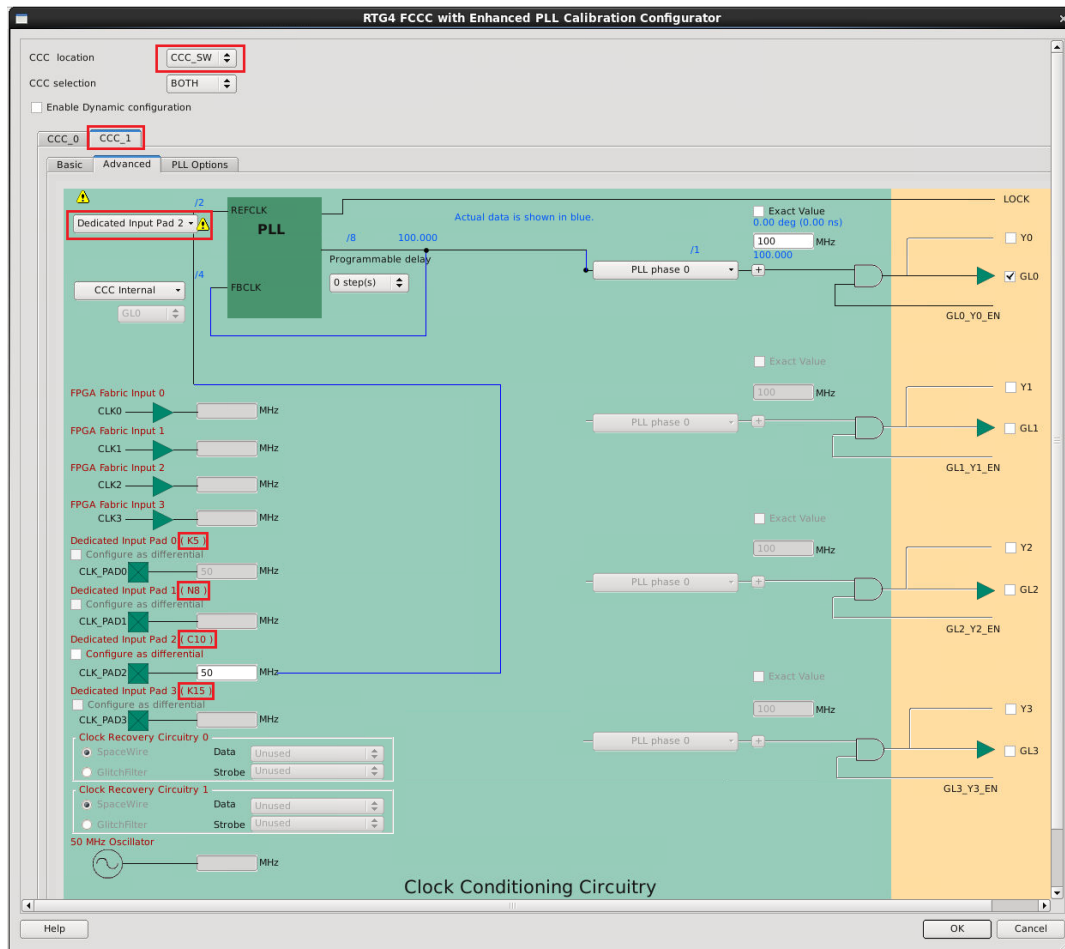
When configuring FCCCs using the enhanced PLL calibration core, the user must be aware of the desired CCC location in the device and the specific CCC instance(s) being used in each corner. As shown in [Figure 15 on page 30](#), RTG4 contains two fabric CCCs in each of the four device corners. These are referred to as CCC_NW, CCC_NE, CCC_SE, and CCC_SW. The specific CCCs in each corner are referred to as CCC_0 and CCC_1. The dedicated CCC clock input I/O pads include the CCC location and CCC # in the pin name, such as the CG1657 package pin C10 whose pin name is MSIO265PB6/GB5/**CCC_SW1_CLKI2**/SPWR_SW1_1_RX_STROBE_P which denotes that it can be used as the dedicated clock input pad 2 (CLKI2) for CCC #1 in the SW device corner (CCC_SW1).

The FPGA designer must be aware of the package pin assignment selected for the design and board layout to ensure that the appropriate CCCs are being configured based on the selected dedicated clock input pins. As shown in the following figure, the first step in configuring the FCCC with Enhanced PLL Calibration is selecting the CCC location (NW, NE, SE, and SW) and which CCCs are being used in that corner (0, 1, or both).

Figure 36 • Selecting CCC Being Instantiated in Enhanced PLL Calibration Core


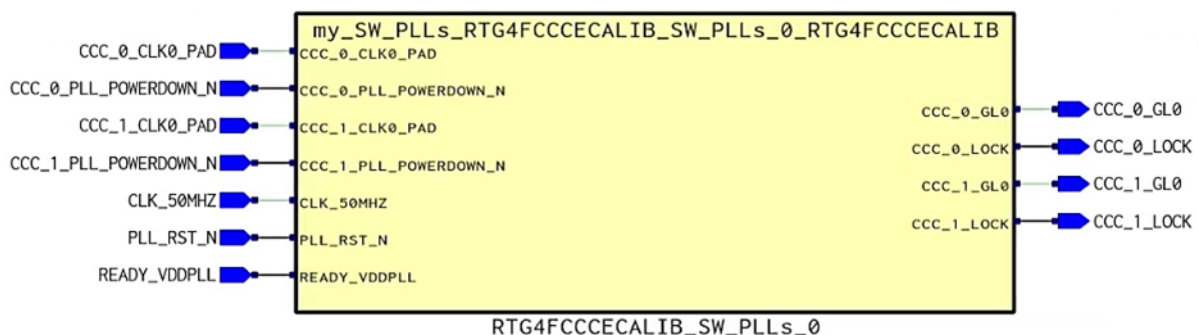
To further ensure the correct selections for CCC location and CCC # in the configurator, the Advanced tab now shows the package pin number for the dedicated input pad selected. Refer to the highlighted text as shown in the following figure.

Figure 37 • Package Pin Number Shown on Advanced Configuration Tab per CCC



After configuring both CCCs in a device corner, the generated RTG4 FCCC with Enhanced PLL calibration component will have ports labeled as CCC_0 and CCC_1 when they apply to a specific CCC in the device corner. The remaining ports that are not labeled with CCC_0/1 apply to the calibration IP in general and affect all CCCs enabled in that instance. An example component is shown in the following figure.

Figure 38 • Graphical Example of Generated RTG4FCCC with Enhanced PLL Calibration



5.9.5.2 Enhanced PLL Calibration Fabric IP

The enhanced PLL calibration core generates soft fabric IP around each of the CCC/PLL instances enabled by the user. The fabric logic is necessary to perform dynamic CCC configuration updates to implement the enhanced PLL calibration feature. At a high-level, the IP dynamically modifies each enabled CCC's input reference divider and feedback divider such that the PLL's internal Voltage Controlled Oscillator (VCO) is forced to run at least 1.5x faster than required by the user configuration. The fabric IP then issues a PLL reset pulse (of at least 1us) and upon reset release, the PLL VCO selects a high gain setting to achieve the faster VCO rate. By temporarily setting the VCO speed higher than the user configuration, it ensures that the VCO gain selected by the PLL's internal start-up process is high enough to maintain PLL lock onto the user desired frequency, regardless of the operating junction temperature rise experienced by the design, within the supported RTG4 operating range of -55C to 125C. This completely mitigates the issue described in RTG4 Customer Notification CN19009. After each enabled PLL has been calibrated to a high VCO gain, the fabric IP then returns the CCC divider settings to the user desired configuration, without applying a further PLL reset, thus preserving the high VCO gain setting. Once the PLL acquires lock onto the user desired clock frequency, the respective CCC_{0,1}_LOCK output will go high indicating that the user design can now rely on the CCC/PLL clock output.

The enhanced PLL calibration process restarts whenever the device is powered-up, DEVRST_N is released, the PLL_RST_N / PLL_POWERDOWN_N are toggled, or if READY_VDDPLL goes low. Furthermore, if the 'Enable Auto-Reset of PLL on Loss of Lock' PLL option is enabled for a given CCC, the de-assertion of the respective CCC_#_LOCK output will restart the PLL calibration process for that respective CCC/PLL only. If the Auto-Reset logic is disabled, the user is responsible for monitoring the CCC_#_LOCK output and deciding how the design will react to an event such as a radiation induced loss of lock.

Note: The generated component has one PLL_RST_N input which means that assertion and release of this input will re-calibrate all enabled CCC/PLLs within that device corner (up to 2) instantiated in the RTG4 FCCC with Enhanced PLL Calibration component. If the user wants to re-calibrate a specific PLL out of the 2 used in the device corner, the CCC_0_PLL_POWERDOWN_N or CCC_1_PLL_POWERDOWN_N input should be toggled instead.

The READY_VDDPLL input must go high before the PLL calibration process will run. Refer to section [PLL Options](#), page 62 for details on the READY_VDDPLL input usage, and two methods that can be used to detect when VDDPLL reaches its nominal voltage, if the user board design cannot guarantee that VDDPLL will not be the last supply to ramp up.

There is one shared CCC APB interface available per corner to perform dynamic CCC re-configuration of up to 2 CCCs per device corner. This is the primary reason that the RTG4 FCCC with Enhanced PLL Configurator is used to configure up to 2 CCCs at a time. Since there is only one shared CCC APB interface per corner, user designs that perform dynamic CCC configuration during design operation also require the use of this APB interface. Therefore, when the user checks the configurator option to Enable Dynamic Configuration, an APB interface is added to the generated component port list and a fabric-based APB mux logic is generated to allow the enhanced PLL calibration to occur at PLL start-up, then allow user dynamic CCC re-configuration during normal operation. The select signal for the ABP mux logic keeps the CCC APB interface under the control of the enhanced PLL calibration core until the PLL calibration is complete. At this point, the PLL_ELOCK_DONE signal goes high, which switches the APB mux to allow the user APB master to drive the CCC APB interface.

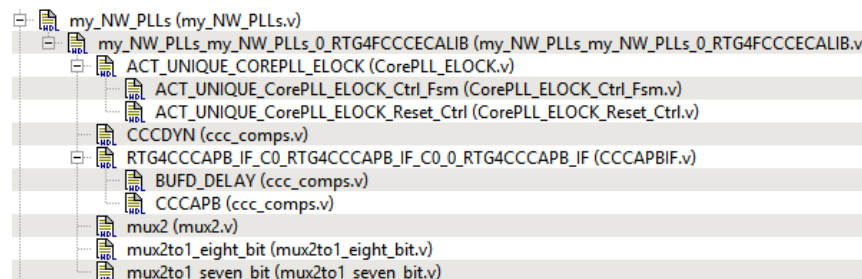
Starting with v2.2 of the RTG4 FCCC with Enhanced PLL Calibration (RTG4FCCCECALIB) core, there have been several updates to improve the robustness of the soft IP generated around the CCC block. See [CN19009C](#) for description of these updates.

The RTG4 FCCC with Enhanced PLL Calibration core instantiates the following components:

- 1 or 2 dynamic CCC instances (CCCDYN) called CCC_INST_0 and CCC_INST_1 for CCC_0 and CCC_1, respectively
- Instantiating the COREPLL_ELOCK IP which is comprised of:
 - PLL Calibration control FSM (CorePLL_ELOCK_Ctrl_Fsm)
 - PLL Reset control logic (CorePLL_ELOCK_Reset_Ctrl)
- CCC APB configuration interface (RTG4CCCAPB_IF) which includes:
 - APB interface buffer logic to ensure correct bus interface timing
 - APB mux logic when the user chooses to Enable Dynamic Configuration
 - PLL_ELOCK_DONE drives the APB mux logic select signal
- Gating logic to suppress the CCC_0/1 PLL lock output signal until after the PLL calibration is complete
 - Status signaled by PLL_ELOCK_DONE which enables CCC_#_LOCK_EN output of the CorePLL_ELOCK IP

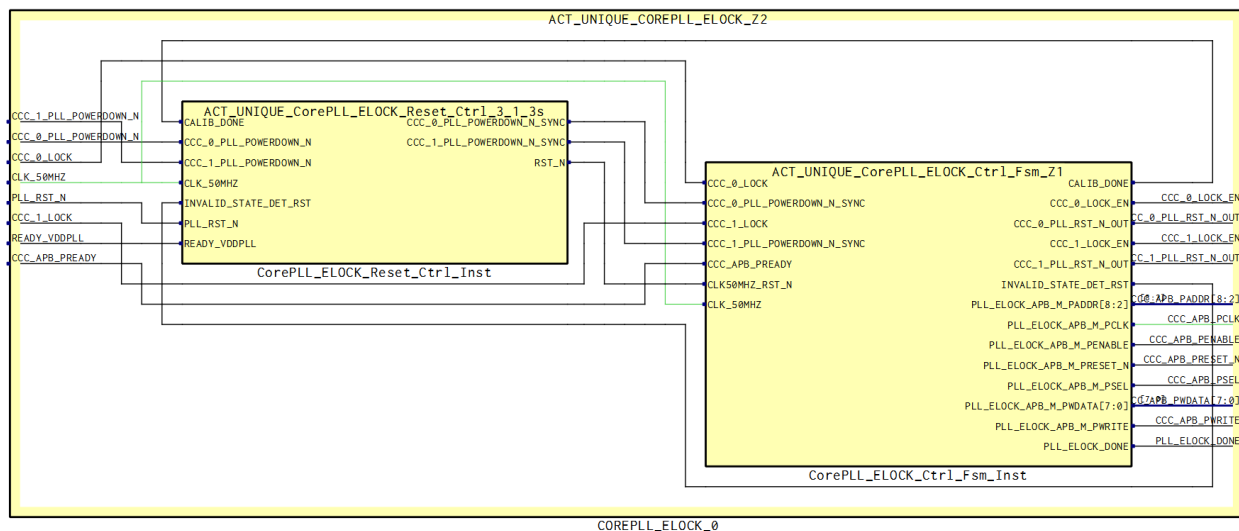
The following figure shows an example of component module hierarchy for an instance of the RTG4 FCCC with Enhanced PLL Calibration with user dynamic configuration enabled:

Figure 39 • HDL Module Hierarchy for RTG4 FCCC with Enhanced PLL Calibration



The following figure shows a block diagram of the COREPLL_ELOCK IP:

Figure 40 • CorePLL_ELOCK Block Diagram



The CorePLL_ELOCK fabric IP applies the following design practices:

- Asynchronous input signals such as CCC_0_LOCK, CCC_1_LOCK, READY_VDDPLL, CCC_0_PLL_POWERDOWN_N, and CCC_1_POWERDOWN_N are synchronized to the CLK_50 MHz input clock before being used to implement the PLL calibration and auto-reset (if enabled).
- Control FSM uses one-hot state encoding and single-bit flip illegal state detection (XNOR), which places the FSM in the IDLE state
- A free running 50 MHz, or slower, clock input is required
 - This clock should not be derived from an RTG4 fabric PLL
 - The input clock can be slower than 50 MHz, but cannot be faster to ensure that the RTG4 PLL_ARST_N minimum pulse width requirement of 1us is met
- The PLL lock output assertion of each CCC is synchronized through a series of 3 fabric flip-flops clocked by the lowest frequency CCC output clock (GL[x]) to produce the final CCC_0/1_LOCK output
- The CCC_#_LOCK_EN output is generated in the CLK_50 MHz domain
 - This output is used to enable the respective CCC lock output to pass through the gating logic, but the CCC_#_LOCK assertion to user logic is driven a flip-flop clocked by the respective PLL's clock output
 - In the **PLL Options** tab for each CCC, if the **Output Resynchronization After Lock** option to "Hold clock outputs in reset (output low) after power-up" is selected, then the CCC_#_LOCK_EN output is also used to drive the respective CCC's GLx_Yx_EN clock output enable signal. As described in [Clock Gating Logic \(CGL\)](#), page 44, the CCC includes Clock Gating Logic (CGL) per [Figure 25](#), page 45. Since, the CCC_#_LOCK_EN output is clocked by the CLK_50 MHz clock input, it is possible that the GLx_Yx_EN clock output enable can arrive asynchronously to the clock being gated, depending on the relationship of the CLK_50 MHz and the CCC input reference clock used to derive the PLL output clock.
 - Since it is typically a don't care about which clock cycle of the PLL output clock is the first cycle being outputted when transitioning from clock-gated to clock-enabled state, the asynchronous relationship of the GLx_Yx_EN and the gated-clock should not be a concern.
 - The GLx_Yx_EN is latched on the falling edge of the PLL output clock and has half a clock period to settle before the next rising edge of the PLL output clock. The high gain of the CGL flip-flop should prevent a metastability concern. However, if the user is concerned, another FF clocked by the CCC reference clock can be manually inserted to synchronize the CCC_#_LOCK_EN signal before it is fed to the CCC GLx_Yx_EN input.

Starting with RTG4FCCCECALIB core version 2.2.xxx, the following core functionality is implemented:

- By default, PLL_RST_N assertion is synchronously captured using CLK_50MHZ, with an added user option to revert asynchronous reset assertion
 - If asynchronous reset assertion is selected, the user must implement an SET mitigated approach to supply the asynchronous reset input signal. For information about creating SET mitigated async resets, such as triplicated logic cones feeding an RGRESET macro, see [AC463: RTG4 Radiation Mitigated Clock and Reset Network Usage Application Note](#).
- The PLL_RST_N reset release is always synchronized to CLK_50MHZ
- The PLL calibration soft IP consumes the PLL_RST_N input synchronously to CLK_50MHZ, eliminating the creation of local asynchronous reset nets
- Each core instance comes with an Netlist Design Constraint (.NDC) file to enable single-event transient (SET) mitigation for all flip-flops in the soft IP
- The calibration FSM LOCK_WAIT state has a 500 μ s timeout counter so that the calibration automatically restarts if the PLL does not acquire lock within the maximum lock acquisition time. For information about maximum lock acquisition time, see [RTG4 FPGA Datasheet](#).
- For configurations where the user enables dynamic CCC configuration, the APB interface clock is driven by the CLK_50MHZ input, requiring that the user's APB host also be synchronized to the CLK_50MHZ clock.

The generated HDL for the RTG4 FCCC with Enhanced PLL Calibration is plain text HDL, which allows for user inspection and application specific customization, as required. In some instances, the user might choose to disable the optional PLL loss of lock auto-reset logic and instead implement a similar scheme more suited to their application. For example, user logic running on a free running clock source (not derived from the RTG4 fabric PLL) can monitor the PLL lock output, and upon loss of lock, put the design

in a safe state and toggle the appropriate PLL_POWERDOWN_N signals to restart the PLL calibration process to re-acquire lock.

If Single Event Transient (SET) mitigation is desired for the generated sequential logic in this core, it is recommended to use the set_mitigation NDC constraint in the Libero SoC Enhanced Constraint Flow to turn on SET mitigation for all instances of the RTG4 FCCC with Enhanced PLL Calibration. Refer to the UG0688: PDC Commands User Guide for more information on the set_mitigation NDC constraint. SET mitigation can also be enabled at a global project level from the Libero Project Menu -> Project Settings... -> Device Settings window. When using RTG4FCCCECALIB v2.2.xxx or later, an NDC file is generated with each core instance, and it can be added to the active set of constraints by clicking Derive Constraints in the Constraints Manager, Netlist Attributes tab.

Note: If the user selects a Lock Window on the PLL Options tab that doesn't meet the Min LOCKWIN Setting formula described in section [Lock Generation Circuit](#) of this user's guide, the RTG4 FCCC component generation will fail with an error message indicating that an illegal PLL Lock Window value was selected for the given divided reference clock input to the PLL. Refer to section [Lock Generation Circuit](#), page 39 to select a suitable Lock Window setting for the desired CCC/PLL frequency configuration.

After generating an RTG4 FCCC with Enhanced PLL Calibration component, it is important to use the Constraints Manager, Timing tab, in the Libero SoC Enhanced Constraints Flow to derive the appropriate timing constraints for each generated component. These derived constraints can be used as-is, or manually merged into the top-level project hierarchy if the current Libero project is not the top-level project. The following is an example set of derived SDC constraints for reference:

```
create_clock -name {CCC_1_CLK2_PAD} -period 10 [ get_ports { CCC_1_CLK2_PAD } ]

create_clock -name {CCC_0_CLK0_PAD} -period 25 [ get_ports { CCC_0_CLK0_PAD } ]

create_generated_clock -name {my_SW_PLLs_0/CCC_INST_1/GL0} -multiply_by 3 -
divide_by 5 -source [ get_pins { my_SW_PLLs_0/CCC_INST_1/CLK2_PAD } ] -phase
0 [ get_pins { my_SW_PLLs_0/CCC_INST_1/GL0 } ]

create_generated_clock -name {my_SW_PLLs_0/CCC_INST_0/GL0} -multiply_by 5 -
divide_by 2 -source [ get_pins { my_SW_PLLs_0/CCC_INST_0/CLK0_PAD } ] -phase
0 [ get_pins { my_SW_PLLs_0/CCC_INST_0/GL0 } ]

set_false_path -from [ get_cells {
my_SW_PLLs_0/COREPLL_ELOCK_0/CorePLL_ELOCK_Ctrl_Fsm_Inst/*CCC_1_LOCK_EN } ] -
to [ get_pins { my_SW_PLLs_0/CCC_1_DFN1_0/D } ]

set_false_path -from [ get_cells {
my_SW_PLLs_0/COREPLL_ELOCK_0/CorePLL_ELOCK_Ctrl_Fsm_Inst/*CCC_0_LOCK_EN } ] -
to [ get_pins { my_SW_PLLs_0/CCC_0_DFN1_0/D } ]
```

Each generated component also includes a <instance_name>configuration.xml report file describing the CCC_0 and CCC_1 configuration selected in the configuration GUI. The report is found in the Libero project's component/work/<component_name>/<instance_name>/ folder. This report also includes the VCO frequency used during the PLL calibration process, for reference. The following figure shows a snippet of the configuration.xml file.

Figure 41 • RTG4 FCCC with Enhanced PLL Calibration Configuration Report**CCC_1 Actual and Calibrated Configuration Summary**

	Normal VCO	High VCO
Reference clock frequency	100 MHz	N/A
GL0/Y0 Output frequency	60 MHz	N/A
GL1/Y1 Output frequency		N/A
GL2/Y2 Output frequency		N/A
GL3/Y3 Output frequency		N/A
External REF Divisor	5	5
Internal REF Divisor	1	1
External FB Divisor	3	5
Internal FB Divisor	1	1
Output Divisor	16	16
VCO frequency	960 MHz	1600 MHz
GPD0 divider	1	N/A
GPD1 divider	1	N/A
GPD2 divider	1	N/A
GPD3 divider	1	N/A
Achieved Output Frequency(GL0/Y0)	60.000 MHz	N/A
Achieved Output Frequency(GL1/Y1)		N/A
Achieved Output Frequency(GL2/Y2)		N/A
Achieved Output Frequency(GL3/Y3)		N/A

When simulating the RTG4 FCCC with Enhanced PLL Calibration, it is important to note the following limitations:

- Simulating Lock Delay (LOCKCNT) values selected in the PLL Options tab is not supported
- Simulating the PLL calibration process is not cycle-accurate and does not provide a specification for the time it takes to calibrate the PLL and acquire PLL lock onto the desired frequency.
 - The PLL calibration process includes a 1uS PLL_ARST_N assertion pulse and a 150uS VCO gain selection delay
 - The RTG4 Fabric PLL has a worst-case PLL lock acquisition time defined in the RTG4 Datasheet ([DS0131](#))

5.9.5.3 Selecting CCC GLx Outputs to Ensure Placement Success

In the standard CCC core, the Libero SoC Place and Route algorithm could swap the GLx CCC output assignments to reach a routable solution to the device Global Buffer (GB) inputs while adhering to the GB connection assignment rules described in [Global Buffer \(GB\)](#), page 11. Since the RTG4 Fabric CCC with Enhanced PLL Calibration core instantiates dynamic CCC instances to perform the PLL calibration process, GLx output swapping is not performed by the placer. This preserves the user GLx clock assignment ordering to support designs that dynamically change the CCC configuration during normal operation. If the GLx output assignment was swapped by the Placer for a dynamic CCC, the user design could unintentionally re-configure the wrong CCC outputs.

Designs which use a high number of Global Nets and CCC GLx outputs could encounter a Placer Failure with the message:

```
Error: No solution was found which could accommodate the global nets (and their drivers).
```

To resolve such an error, see [Global Buffer \(GB\)](#), page 11 to ensure that the GLx outputs selected for all the CCCs in the same vertical half of the die meets the GB connection assignments.

Alternatively, the following rules can be applied proactively by the user when selecting CCC GLx outputs:

GL Assignment Rule #1 - Designs without local async resets (RGRESETs) and without SerDes EPCS Direct Global Outputs:

- For each $x = \{0, 1, 2, 3\}$, the count of GLx across the 4 CCCs on the West locations (CCC_NW* and CCC_SW*) cannot exceed 3.
- For each $x = \{0, 1, 2, 3\}$, the count of GLx across the 4 CCCs on the East locations (CCC_NE* and CCC_SE*) cannot exceed 3.

GL Assignment Rule #2 - Designs containing local async resets (RGRESETs) but no SerDes EPCS Direct Global Outputs:

- The count of GL0 across the 4 CCCs on the West locations (CCC_NW* and CCC_SW*) cannot exceed 2.
- The count of GL0 across the 4 CCCs on the East locations (CCC_NE* and CCC_SE*) cannot exceed 2.
- For each $x = \{1, 2, 3\}$, the count of GLx across the 4 CCCs on the West locations (CCC_NW* and CCC_SW*) cannot exceed 3.
- For each $x = \{1, 2, 3\}$, the count of GLx across the 4 CCCs on the East locations (CCC_NE* and CCC_SE*) cannot exceed 3.

Amendment to GL Assignment Rules #1 and #2 - Designs containing SerDes EPCS Direct Global Outputs

- Reduce the counts of GL0 and GL2 based on the usage of each particular SERDES clock according to [Table 4 on page 13](#).

5.9.5.4 Performing Dynamic CCC Configuration with RTG4FCCCECALIB

There are several considerations to note when using Dynamic CCC Configuration along with the RTG4FCCC with Enhanced PLL Calibration core.

During normal operation, if the RTG4FCCCECALIB instance is configured with Dynamic Configuration enabled, use the APB configuration interface to dynamically update CCC configuration registers. For the RTG4FCCCECALIB core, it is not recommended to update registers that invalidates the PLL calibration applied to the original CCC/PLL frequency configuration. Furthermore, if the RTG4FCCCECALIB instance has PLL Loss of Lock Auto-Reset Logic enabled, consider the consequences of performing a dynamic CCC re-configuration that causes the PLL to lose lock. If performed, it triggers a PLL reset and re-calibration using the original component configuration for FCCC_FBDIV_CR (CCC Feedback Divider), FCCC_PLL_CR4 (PLL Internal Output Divider), and FCCC_PLL_CR6 (PLL Internal Feedback Clock Divider), overwriting any new values dynamically configured in these registers.

Another important consideration when performing user dynamic CCC configuration of an RTG4FCCCECALIB core during normal operation is preventing contention on the APB configuration bus. The APB data and control signals are multiplexed between the PLL calibration soft-IP and the user's APB host logic. During power-up or DEVRST_N release, the PLL calibration soft-IP controls the APB configuration interface, and the user APB host logic must not initiate any transactions during this time. Furthermore, whenever the user APB host logic is not initiating a new transaction, it should drive the APB data and control signals to a safe state so that there is no chance of inadvertent CCC register writes. When the calibration is complete and the corresponding CCC_{0/1}_LOCK signal goes high, the MUX select line, controlled by the PLL_ELOCK_DONE output of COREPLL_ELOCK module, passes dynamic configuration control over to the user APB host interface. After this time, the user APB host can initiate dynamic CCC configuration transactions.

When performing dynamic CCC configuration, the user APB host logic should ensure the corresponding CCC_{0/1}_LOCK output is asserted (to confirm that any PLL calibrations or re-calibrations are complete). In addition, if PLL Loss of Lock Auto-Reset is enabled, or if there are any other design/system level initiators that can assert PLL_RST_N or PLL_POWERDOWN_N, then the APB host logic should check whether the CCC_{0/1}_LOCK output remains asserted during the duration of the APB transaction. If the PLL LOCK de-asserts for any reason during an APB write, the APB host should wait until the PLL re-acquires lock and then read back that CCC register to ensure the APB write completed, and if not, retry the APB transaction. This recommendation assumes that the PLL loss of lock is not initially triggered by the user's APB write to a CCC configuration register that caused the PLL to lose lock.

As mentioned in the preceding sections, and in [CN19009C](#), the RTG4FCCCECALIB core is updated to remove a 2-input MUX on the APB_S_PCLK signal. The updated core re-uses the mandatory CLK_50MHZ input signal to clock both the calibration soft-IP and the dynamic configuration APB interface during user reconfiguration. This means that the user logic controlling the APB configuration bus must also be clocked or be synchronous with the CLK_50MHZ input. Since both the PLL calibration IP and the dynamic configuration interface have a maximum clock frequency of 50MHz, the same free running 50MHz or slower clock can be used for both functions. Furthermore, for added robustness, the user can select a SET mitigated clock input path. For example, use an external free running 50MHz or slower clock entering the device via a GB# device package pin and CLKBUF macro. CLK_50MHZ must not be sourced from a fabric PLL to prevent a circular dependency where the PLL calibration/re-configuration depends on a PLL output being available.

For designs that cannot update to v2.2 or later of the RTG4FCCCECALIB core, the following steps can be used to manually edit an older core version such that the MUX for APB_S_PCLK is removed:

- Generate the desired RTG4FCCCECALIB core in any Libero SoC project with Dynamic Configuration enabled
- Save the generated log and SDC files:
 - <project_folder>/component/work/<core_name>/<core_name>_manifest.txt
 - <project_folder>/component/work/<core_name>/<core_inst_name>/<core_inst_name>configuration.xml
 - <project_folder>/component/work/<core_name>/<core_inst_name>/<core_name>_<core_inst_name>_RTG4FCCCECALIB.sdc
- Import the HDL files into a top-level Libero project as HDL source files:
 - <project_folder>/component/work/<core_name>/<core_inst_name>/<core_name>.v
 - <project_folder>/component/work/<core_name>/<core_inst_name>/<core_name>_<core_inst_name>_RTG4FCCCECALIB.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/CCCAPBIF.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/CorePLL_ELOCK.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/CorePLL_ELOCK_Ctrl_Fsm.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/CorePLL_ELOCK_Reset_Ctrl.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/mux2.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/mux2to1_eight_bit.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/mux2to1_seven_bit.v
 - <project_folder>/component/Actel/SgCore/RTG4FCCCECALIB/2.1.010/ccc_comps.v
- Edit the <core_name>.v top-level file as follows:
 - Comment-out input port APB_S_PCLK in module declaration and port list

```

274 module RTG4FCCCECALIB_C0 (
275     // Inputs
276     APB_S_PADDR,
277     // APB_S_PCLK,
278
279     //-----
280     // Input
281     //-----
282     input [8:2] APB_S_PADDR;
283     //input      APB_S_PCLK;
284     input      APB_S_PENABLE;

```

- Comment-out wire declaration for APB_S_PCLK

```

328 //wire      APB_S_PCLK;
329 wire      APB_S_PRESET_N;

```

- Comment-out APB_S_PCLK in instantiation of lower-level module

```

384 RTG4FCCCECALIB_C0 RTG4FCCCECALIB_C0_0 RTG4FCCCECALIB RTG4FCCCECALIB_C0_0 (
385     // Inputs
386     .APB_S_PSEL      ( APB_S_PSEL ),
387     // .APB_S_PCLK    ( APB_S_PCLK ),
388     .APB_S_PWRITE    ( APB_S_PWRITE ),

```

- Edit the <core_name>_<inst_name>_RTG4FCCCECALIB.v top-level file as follows:
 - Comment-out input port APB_S_PCLK in module declaration and port list

```

5 module RTG4FCCCECALIB_C0_RTG4FCCCECALIB_C0_0_RTG4FCCCECALIB(
6     APB_S_PREADY,
7     APB_S_PSLVERR,
8     APB_S_PSEL,
9     APB_S_PRDATA,
10    APB_S_PWDATA,
11    APB_S_PADDR,
12    //    APB_S_PCLK,
13    APB_S_PWRITE,

32    input  [8:2] APB_S_PADDR;
33    //input  APB_S_PCLK;
34    input  APB_S_PWRITE;

```

- Comment-out the MX2_APB_S_PCLK instance

```

141 //    mux2 MX2_APB_S_PCLK (.A(
142 //        CCC_APB_PCLK_A_COREPLL_ELOCK_0_MX2_APB_S_PCLK_net), .B(
143 //        APB_S_PCLK), .S(
144 //        MX2_7B_APB_S_PADDR_S_COREPLL_ELOCK_0_PLL_ELOCK_DONE_net), .Y(
145 //        APB_S_PCLK_Y_CCC_APB_INST_MX2_APB_S_PCLK_net));

```

- Replace APB_S_PCLK input connection of RTG4CCCAPB_IF_C0... instance CCC_APB_INST from .APB_S_PCLK(APB_S_PCLK_Y_CCC_APB_INST_MX2_APB_S_PCLK_net), to .APB_S_PCLK(CCC_APB_PCLK_A_COREPLL_ELOCK_0_MX2_APB_S_PCLK_net),

```

318 .APB_S_PSLVERR(APB_S_PSLVERR), .APB_S_PSEL(
319     APB_S_PSEL_Y_CCC_APB_INST_MX2_APB_S_PSEL_net), .APB_S_PCLK(
320     CCC_APB_PCLK_A_COREPLL_ELOCK_0_MX2_APB_S_PCLK_net), .APB_S_PRESET_N(
321     APB_S_PRESET_N_Y_CCC_APB_INST_MX2_APB_S_PRESET_N_net),

```

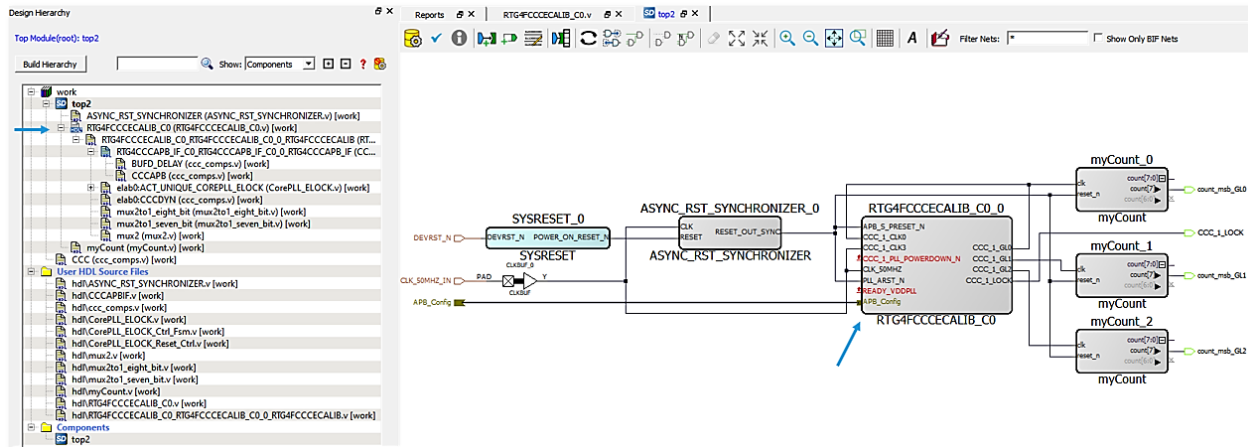
- This takes the forwarded CLK_50MHZ from CorePLL_ELOCK.v's APB_S_PCLK output and uses it to drive the APB configuration interface in that device corner
- If SmartDesign is used to instantiate and connect this core in the top-level design, then create an HDL+ core for the RTG4FCCCECALIB core in the design hierarchy and add an APB slave BIF port as described below:
 - Right-click the **RTG4FCCCECALIB** core in the design hierarchy and select **Create Core from HDL...**
 - In the **Edit Core Definition - Ports and Parameters** window, Click **Add/Edit Bus Interfaces...**
 - In the **Edit Core Definition - Bus Interfaces** window, select **APB Slave**, name the **BIF**, use the prefix **APB_S_** to map the APB signals by name, then map the APB_S_PSELx port to APB_S_PSEL and click **OK**
- If SmartDesign is not used, integrate/instantiate the modified core into the HDL design
- Ensure CLK_50MHZ uses an SET mitigated input path and a global routing resource
- Modify the core's SDC file to match the top-level project's design hierarchy naming level where the core is instantiated and import the SDC into the top-level project's list of timing constraints
 - This step manually derives the constraints since the design no longer contains the generated IP component in the top-level project to enable automatic constraint derivation from the Constraints Manager

Note: After manually modifying the RTG4FCCCECALIB core, any desired changes to the CCC configurator requires generating a new core component in another Libero project, and repeating the manual steps above to apply the modifications and re-importing into the desired top-level project

Note: The original RTG4FCCCECALIB component configuration can be recreated via the Tcl script embedded as comments in the core's top-level HDL file, to enable easy re-creation of the original component configuration and subsequent minor changes

An example SmartDesign canvas with the HDL+ component is shown in the following figure (for a design which has not upgraded to RTG4FCCCECALIB v2.2 or later).

Figure 42 • Example SmartDesign Canvas with the HDL+ Component



5.9.6 Simulation Support

Microsemi Libero SoC design software provides pre-compiled simulation models for the CCC macro to show functional behavior of the fabric CCC. The simulation steps include generating the top-level component, which instantiates CCC macro, performing simulation for verification with the ModelSim tool, and performing static timing analysis with SmartTime in Libero SoC.

5.9.6.1 Use Model 1: Clock Frequency Synthesis

Deriving clocks of various frequencies from a single reference clock is known as frequency synthesis.

This use model configures one of the on-chip oscillators as a clock source to the fabric CCC(s). This use model requires the instantiation of both the Chip Oscillator macro and CCC macro in the design. The

Chip Oscillator macro is used to drive the CCC macro reference clock input. This use model uses the Basic tab of the CCC configurator for CCC configuration.

Refer to the [How to Use On-Chip Oscillator](#), page 27 for more details. It is also possible to use external clock sources through dedicated global I/Os instead of on-chip oscillator.

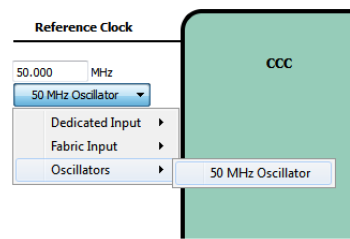
1. Create a SmartDesign instance
2. Access the CCC configurator from the Libero SoC IP Catalog. In the CCC configurator, select the output clocks and enter the desired clock frequencies. These frequencies are used by the CCC configurator to compute the configuration of the CCC dividers and PLL to meet the requirements. The Actual column displays the actual value that the configurator was able to achieve, refer to the following figure.
 - GL0 - 100 MHz
 - GL1 - 150 MHz
 - GL2 - 200 MHz

Figure 43 • Output Clocks Settings

	Exact Value	Frequency	Actual
<input checked="" type="checkbox"/> GL0	<input type="checkbox"/>	100 MHz	100.000 MHz
<input checked="" type="checkbox"/> GL1	<input type="checkbox"/>	150 MHz	150.000 MHz
<input checked="" type="checkbox"/> GL2	<input type="checkbox"/>	200 MHz	200.000 MHz
<input type="checkbox"/> GL3	<input type="checkbox"/>	100 MHz	

3. Select the fabric CCC reference clock input from the drop-down menu. For this use model, select the 50 MHz Oscillator as the clock source, refer to the following figure.

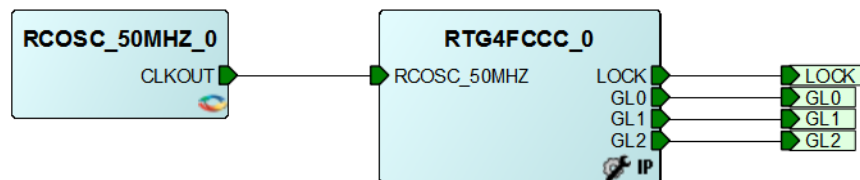
Figure 44 • Fabric CCC Reference Clock Selection



4. Click **OK** to close the CCC configurator.
5. Instantiate into SmartDesign instance the 50 MHz Oscillator macro from the Libero SoC IP Catalog
6. Connect the 50 MHz Oscillator macro output to the CCC macro reference clock input. The following figure shows the final SmartDesign component with the CCC outputs promoted to top-level for simulation.

The generated clocks can then be connected to the fabric logic, external component through I/Os, or as a base clock to the RTG4 hard IP blocks (FDDR and SerDes blocks).

Figure 45 • Fabric CCC and On-chip Oscillator Connectivity



5.9.6.2 Use Model 2: Clock Delay Adjustment

For PLLs, delays in the feedback path effectively advance the PLL output with respect to the reference clock. Adding a delay in the PLL reference clock path enables clock delay and adding a delay in the PLL feedback clock path enables clock advancement. A typical application for clock advancement is to advance the on-chip clock in order to cancel the global network delay and align the internal clock to an external clock on board.

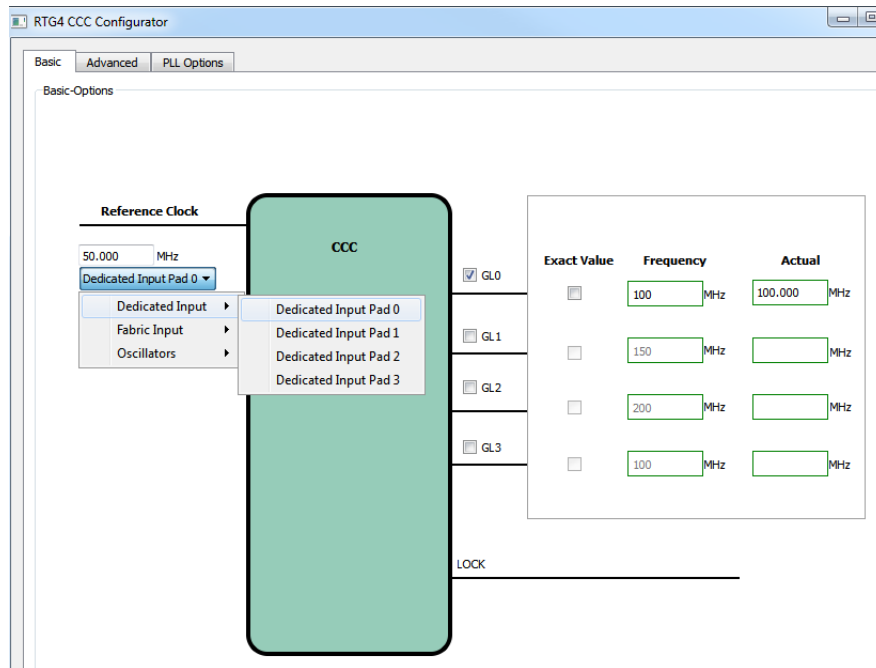
For this use model, perform the fabric CCC basic configuration using the **Basic** tab of the CCC configurator as outlined in the above use model. For clock delay adjustments, configure the fabric CCC programmable delay elements using the **Advanced** tab of the CCC configurator. Selecting a positive delay value enables the delay elements in the reference clock path to delay the output clock with respect to the PLL reference clock. Selecting a negative delay value in the CCC configurator enables the delay element in the feedback clock path for the output clock advancement with respect to the PLL reference clock. Perform the timing analysis using SmartTime for the actual delay between the reference clock and output clocks.

5.9.6.3 Use Model 3: External Clock Source through Dedicated Global I/O

This use model shows how to source the fabric CCC reference clock from an external clock source. Each fabric CCC is associated with four dedicated global I/Os for providing external reference clock. The mapping of fabric CCCs is done by Libero place-and-route software based on the dedicated global

I/O selected. The selection of the dedicated global I/Os is performed through I/O Constraint Editor of Libero SoC design software. Use the following steps to configure the fabric CCC with external reference clock:

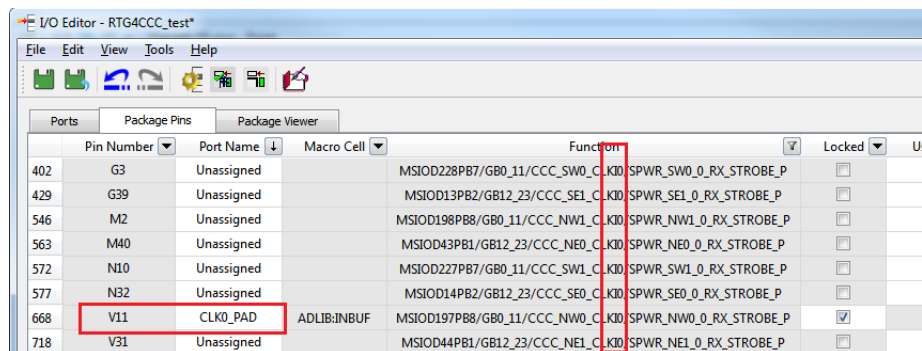
1. Instantiate the CCC macro onto the SmartDesign canvas.
2. In the CCC macro configurator,
 - Select the output clocks and enter the desired clock frequencies.
 - Select one of the four dedicated global input pads as CCC reference clock source and set its frequency, as shown in the following figure.

Figure 46 • Fabric CCC Configuration

3. Connect the generated fabric CCC output clocks to the desired modules in the design.
4. Generate the top-level SmartDesign component and ensure that the top-level component is generated successfully.
5. Perform the Synthesis of the design.
6. Compile the design to perform legality check and basic netlist optimization.
7. Open **I/O Constraint Editor** (in the Libero SoC Design Flow window, **Implement Design > Edit Constraint > I/O Constraints**) to select the specific Input Pad.
8. Perform the pin assignment for the fabric CCC dedicated global input pad. In the package view under **Function** filter for the specific clock function, for example, CLKI0 as shown in Figure 47, page 79.

A dedicated global I/O can be selected corresponding to the dedicated global input pad number (CLKI0, CLKI1, CLKI2, or CLKI3) which is the same as the fabric CCC reference clock source selected in Step 2.

For example, **Dedicated Input Pad 0** is selected as the reference clock source. The pin assignment can be done to a pin, whose pin name is CCC_xyz_CLKI0, where xy represents the CCC location and z represent the CCC number.

Figure 47 • Dedicated Global I/O Selection


Pin Number	Port Name	Macro Cell	Function	Locked	U:
402	G3	Unassigned	MSIOD22PB7/GB0_11/CCC_SW0_CLKI0 SPWR_SW0_0_RX_STROBE_P	<input type="checkbox"/>	
429	G39	Unassigned	MSIOD13PB2/GB12_23/CCC_SE1_CLKI0 SPWR_SE1_0_RX_STROBE_P	<input type="checkbox"/>	
546	M2	Unassigned	MSIOD198PB8/GB0_11/CCC_NW1_CLKI0 SPWR_NW1_0_RX_STROBE_P	<input type="checkbox"/>	
563	M40	Unassigned	MSIOD43PB1/GB12_23/CCC_NE0_CLKI0 SPWR_NE0_0_RX_STROBE_P	<input type="checkbox"/>	
572	N10	Unassigned	MSIOD227PB7/GB0_11/CCC_SW1_CLKI0 SPWR_SW1_0_RX_STROBE_P	<input type="checkbox"/>	
577	N32	Unassigned	MSIOD14PB2/GB12_23/CCC_SE0_CLKI0 SPWR_SE0_0_RX_STROBE_P	<input type="checkbox"/>	
668	V11	CLK0_PAD	ADLIB:INBUF MSIOD197PB8/GB0_11/CCC_NW0_CLKI0 SPWR_NW0_0_RX_STROBE_P	<input checked="" type="checkbox"/>	
718	V31	Unassigned	MSIOD44PB1/GB12_23/CCC_NE1_CLKI0 SPWR_NE1_0_RX_STROBE_P	<input type="checkbox"/>	

9. Commit the changes in the I/O Editor.
10. Run the **Place-and-Route** flow.

5.9.6.4 Use Model 4: PLL in Triple Module Redundant (TMR) Mode

A triple module redundant (TMR) PLL is used when the PLL Internal feedback mode is selected, while a single, non-TMR PLL is used when CCC Internal or External feedback modes are selected in the CCC configurator software. The RTG4 TMR PLL includes three sub-PLLs with a voted lock output to help mitigate single event effects in a radiation environment. When the PLL is configured for triple redundant configuration through the selection of PLL Internal feedback mode, it can experience a loss of lock, which does not automatically self-recover and requires toggling the PLL_ARST_N reset input to regain lock.

In the Libero SoC v11.9, the RTG4 CCC configurator v1.1.226 introduces an auto-reset circuit as shown in the following figure, which monitors the TMR PLL voted lock signal when it is in PLL Internal feedback mode. This fabric circuit automatically issues a reset command to the PLL if loss of lock is detected. The initial auto-reset circuit introduced in Libero SoC v11.9 was updated in Libero SoC v11.9 SP4 to ensure that the user's fabric driven PLL_ARST_N input assertion also starts the 1us delay counter to meet the PLL_ARST_N minimum assertion pulse width. The change means that the fabric PLL_ARST_N input drives the SLE synchronous reset input (SLn) instead of bypassing the counter and directly connecting to the AND gate in front of the PLL. Furthermore, in Libero SoC v11.9 SP4, the auto-reset logic was made available to both the single and triplicated (TMR) fabric PLL, and enabled by default in the RTG4 FCCC configurator. The HDL files associated with the auto-reset circuit can be found in the Libero Design Hierarchy as shown in Figure 49, page 80.

Figure 48 • PLL Auto-Reset Circuit

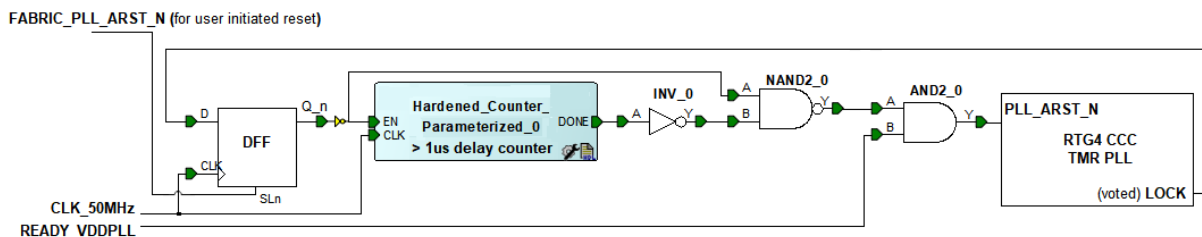
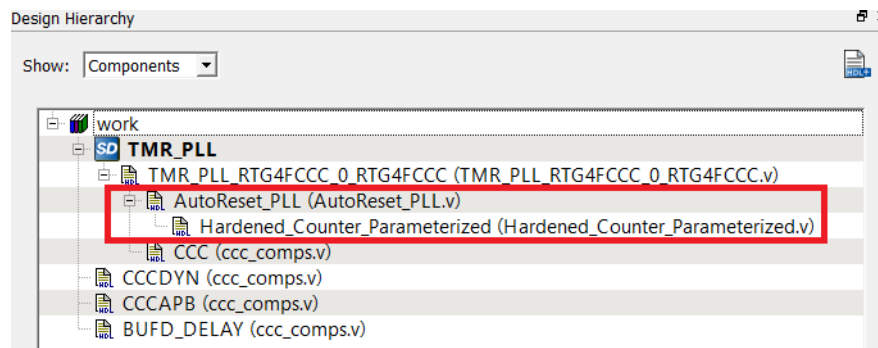


Figure 49 • PLL Auto-Reset Circuit HDL Files



The circuit requires a 50 MHz clock, or slower, that is readily available from the on-chip RC oscillator. You need to instantiate the RCOSC_50 MHz macro and distribute its output through a GLx of any one CCC and eventually connect to the exposed CLK50_MHz input pin of the CCC. The CCC connected to RCOSC_50MHz could even be the same CCC as this one. When PLL Internal Feedback mode is selected, both PLL_ARST_N and PLL_POWERDOWN_N signals are always exposed to provide additional user access to these signals. The auto-reset circuit for the TMR PLL will still function even if the user connects the exposed PLL_ARST_N input to a logic-high because the circuit combines this user input with the status of the voted lock. The input to the 50 MHz oscillator can be directly given through its input pin from an external source or from the GL3 output of the CCC, bypassing the fabric PLL.

If the PLL_ARST_N and PLL_POWERDOWN_N inputs are unused in your design, these additional exposed ports can be tied to logic-high.

Figure 50 • RTG4 FCCC with Enhanced PLL Calibration Configurator

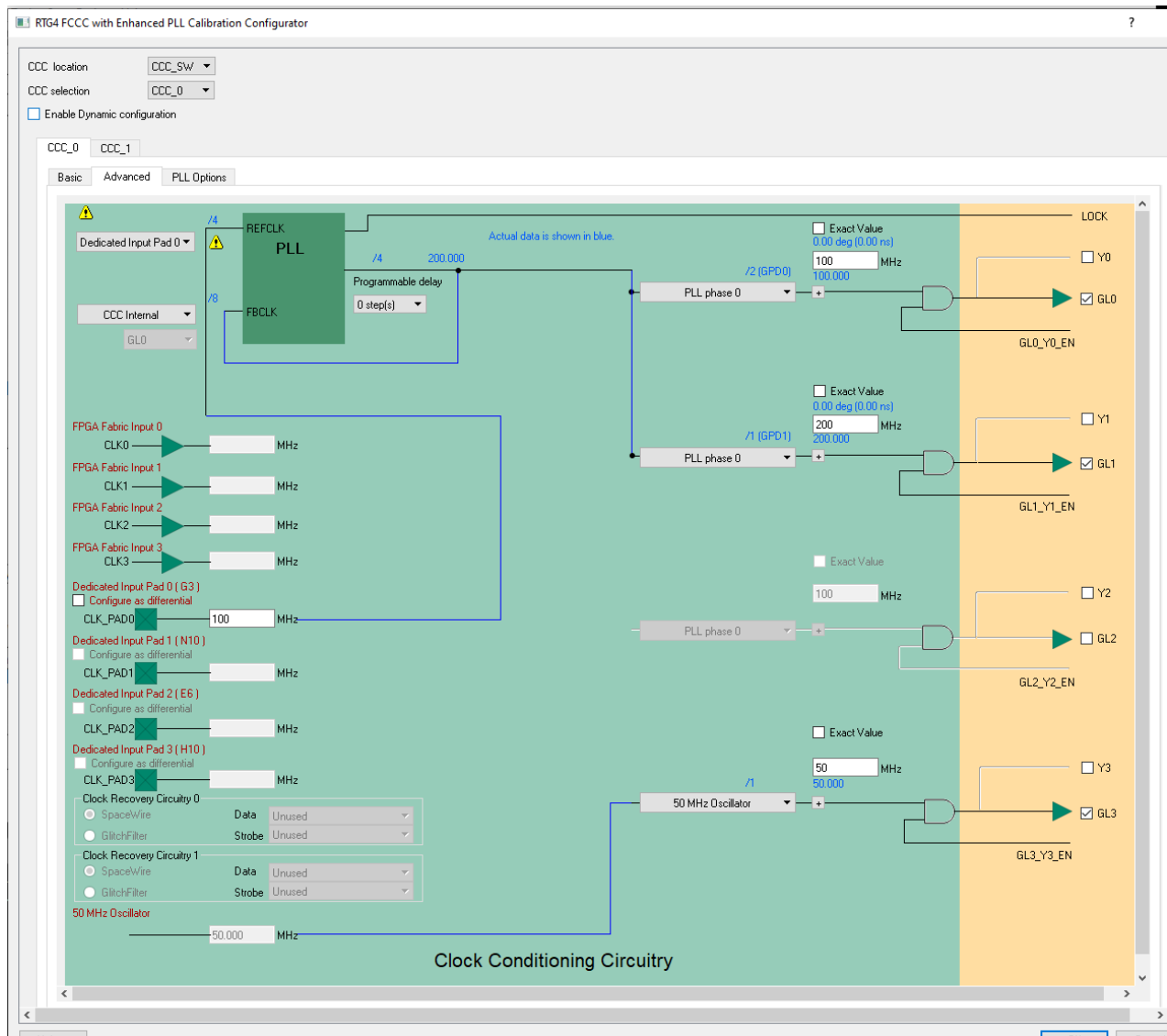


Figure 51 • FCCC Calibration SmartDesign

