AC463
Application Note
RTG4 Radiation Mitigated Clock and Reset Network Usage

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

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.

51900463. 2.0 10/21

Contents

Figures

Tables

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 2.0

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

- Updated section Resets, page 5 to clarify that the optional SET mitigation for SLEs includes the input data path ports D, EN, and SLn and add a high level design recommendation.
- Updated section Radiation Mitigation Designed into RTG4, page 7 to add a link to White Paper WP0191 and link to the RTG4 Radiation Tolerant Feature Summary spreadsheet.
- Updated section Resets, page 13 to add a section covering synchronous resets.
- Updated Table 1, page 8 to add FPGA Fabric, FDDR, and SerDes details.
- Added Figure 16, page 22.
- Added chapter Mitigating Radiation Effects for RTG4 SerDes and FDDR Blocks, page 26.

1.2 Revision 1.0

Revision 1.0 is the first publication of this document published in March 2019.

2 RTG4 Radiation Mitigated Clock and Reset Network Usage

Microsemi RTG4™ radiation tolerant flash-based FPGAs offer significant performance and flexibility advantages to designers working on satellite systems and other equipment required to perform in high radiation environments. RTG4 employs several radiation hardening by design techniques, which help to achieve a high level of total ionizing dose (TID) hardness. RTG4 also provides built-in Single Event Transient (SET) and Single Event Upset (SEU) mitigation in critical areas to enable the creation of high reliability designs.

This application note describes how to use the built-in clock and reset networks and builds upon the material covered in several RTG4 device user's guides such as *UG0574: RTG4 FPGA Fabric User Guide*, *UG0576: RTG4 FPGA System Controller User Guide*, and *UG0586: RTG4 FPGA Clocking Resources User Guide*.

This document describes additional considerations when planning an application's clock and reset architecture to best utilize the built-in SET mitigation provided by the RTG4 dedicated clock and reset routing resources. Individual application requirements determine how designers trade-off the insertion of SET mitigation techniques versus meeting timing performance goals. In general, designs with very high reliability requirements operating in the harshest radiation environments try to apply as many mitigation techniques as possible in the design while trying to meet the timing performance goals. To validate that the design’s radiation tolerance goals are met, some designers perform radiation beam testing of their RTG4 design. In some cases, different versions of the same design are beam tested to see whether the reliability requirements are still met with fewer added mitigation techniques to achieve higher timing performance.

2.1 RTG4 Architecture Overview

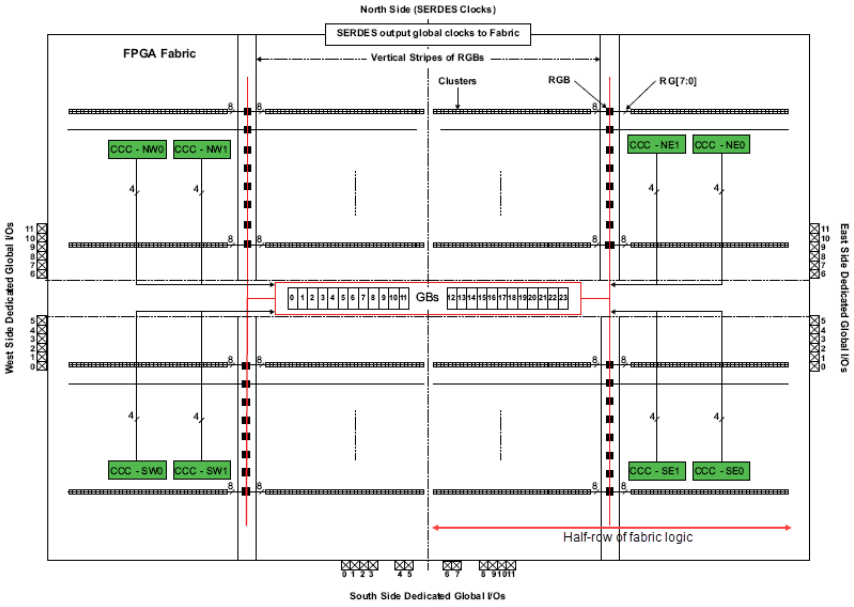
This section provides an overview of the RTG4 clock and reset network architecture to effectively utilize these radiation mitigated resources.

2.1.1 Clocks

RTG4 FPGAs contain low skew global clock networks, which have built-in mitigation against the potential Single Event Effects (SEEs) experienced in space flight applications. The RTG4 clock network is comprised of triplicated metal lines, which drive SET mitigated clock and global signals throughout the FPGA Fabric. Note that the added radiation hardening and the global network architecture adds inherent latency to a signal being routed onto a global resource.

Navigating the built-in fabric resources requires an understanding of the device layout. A top-level view of the device die layout is shown in Figure 1, page 3.

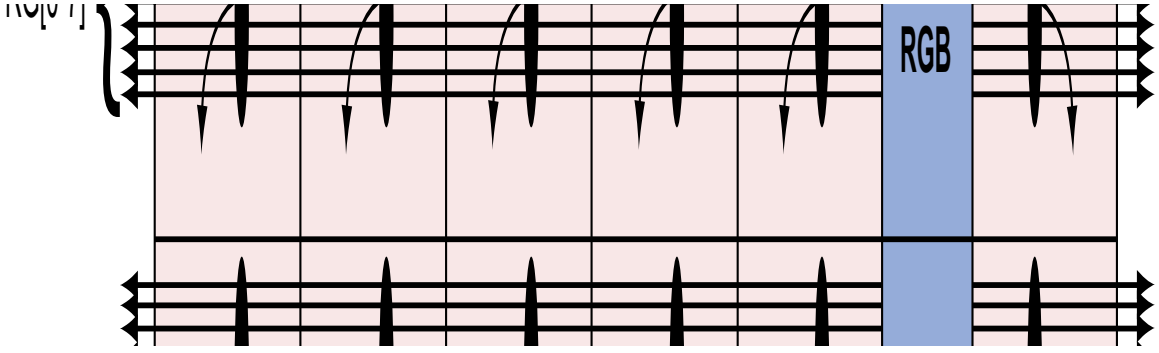
Figure 1 • **RTG4 Global Network**



As shown in Figure 1, the FPGA fabric can be viewed as a rectangle with left and right half-chip areas. There are 24 chip-level global nets, which can reach the entire FPGA fabric. These are driven by the 24 Global Buffers (GBs) depicted in the center of Figure 1. Depending on the user logic being implemented, the global net might only need to span either the left or right half of the fabric, and thus each chip-level global can be split into two half-chip global signals called GBR and GBL. This is automatically done by the Libero SoC Layout software without user intervention. The ability to split a chip-level global means that the device supports a maximum of 48 half-chip global signals. All chip-level and half-chip global signals must pass through the GBs at the center of the device. Furthermore, each vertical half of the FPGA fabric contains a vertical stripe of Row Global Buffers (RGBs) located in the center of that half chip area. The RGBs drive up to eight global signals across the half-row of logic to which they belong. In summary, the chip-level globals in a design pass through GBs and RGBs to reach the Fabric Logic Elements, RAM blocks, and Mathblocks. Complex designs might also employ user defined placement regions, which guides the Placer algorithm in assigning chip-level or half-chip global nets.

Figure 2 shows the RGB driving clocks into the fabric logic For more information, see *UG0586: RTG4 FPGA Clocking Resources User Guide*.

Figure 2 • **Row Global Signals Driving Clusters**



The RTG4 global nets are typically used to create low-skew clock signals to the entire device. There are also cases where very high fanout data nets can be automatically promoted onto a global net, but the default fanout threshold for such promotion is set to 5000. The default promotion thresholds can be modified in Libero SoC by right-clicking the **Synthesize** step in the **Design Flow** tab and selecting **Configure Options...**.The usage of a chip-level global implies that the logic must be able to tolerate the clock insertion delay from the clock source, through the GBs, onto the RGBs and finally to the fabric flip-flops.

If a specific relationship between incoming data and clock must be maintained, the user can employ techniques to compensate for the clock insertion delay, such as using:

- programmable I/O delay elements on MSIO and MSIOD inputs to delay the data path, or
- negative programmable delay values in the RTG4 CCC to advance the clock.

Note: Programmable input buffer delay elements are not available to input signals that use a hardwired input path to a GB, such as the CLKBUF macro or GB## input pin.

It is also possible to promote a fabric signal directly onto an RGB resource using an RCLKINT macro, thus bypassing the GBs. However, this approach requires extra consideration about the radiation tolerance of this promoted row clock, see Clocks, page 9.

The physical location of the RGBs within the fabric half-row relative to components such as I/O Flip-Flops (IOFFs) and SerDes blocks also plays a role in the design planning.

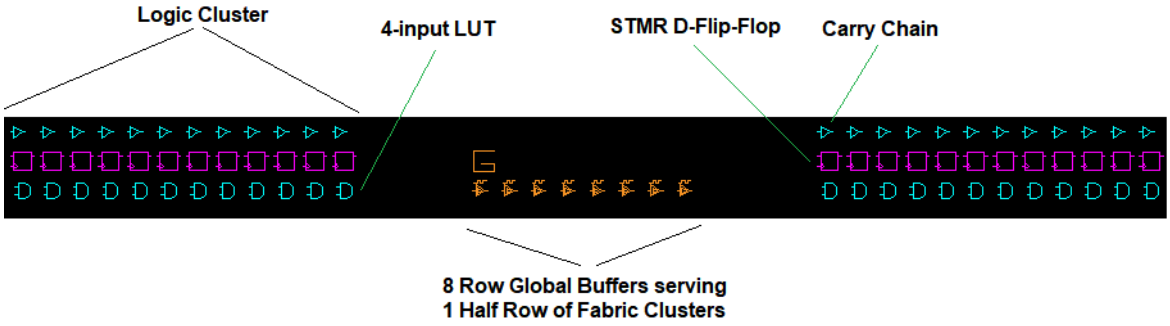
For example:

- SerDes: There are six SerDes blocks horizontally placed across the top of the fabric, with three above each vertical half of the die. Clock inputs from the FPGA fabric to the SerDes blocks, such as the EPCS mode flywheel FIFO read and write clock inputs (EPCS_x_RXFWF_RCLK and EPCS_x_TXFWF_WCLK) must be sourced from the RGBs in the top-row of the fabric. If all the three SerDes blocks within a vertical half of the device are fully utilized, then there are up to 24 flywheel FIFO clock inputs required from the fabric (12 for Rx lanes and 12 for Tx lanes). The user must evaluate whether the serial interface uses a 0 ppm common clock scheme and whether any of the individual lanes within x2 or x4 links can use a shared clocking scheme to ensure that the number of distinct clock inputs sourced from the top half-row of the fabric does not exceed the eight available RGBs.
- IOFFs: I/O banks 4, 5, and 6 run horizontally across the bottom of the device. When using I/O register combining to absorb a fabric flip-flop into the IOFF, the number of distinctly clocked synchronous interfaces that can be assigned to these banks is limited to the number of clocks available from the bottom row of RGBs to the IOFFs. In other words, when using the IOFFs, up to 16 distinctly clocked synchronous I/O interfaces, eight per half row, can be assigned to the horizontal I/O banks. This scenario does not exist for vertical I/O banks—1, 2, 7, and 8—because the banks span across multiple rows. For the vertical banks, there are three IOFFs per each half-row of the fabric.

The fabric logic is organized in rows. Each row contains either full logic clusters or IP interface logic clusters to connect with fabric RAM, DSP blocks, or dedicated circuits such as the SerDes and FDDR blocks. Unlike the full logic cluster, the IP interface logic clusters do no contain the carry-chain functionality. Each cluster is a grouping of 12 logic elements containing 12 SET and SEU Mitigated Triple Module Redundant (STMR) Sequential Logic Elements (SLEs), 12 LUT-4 combinatorial logic cells, and 12 carry-chain elements. Half-rows on the East (right half) of the device contain 65 logic clusters and those on the West (left half) contain 59 clusters. The interface logic clusters are available as user logic whenever the associated RAM or DSP blocks are unused, and the carry-chain functionality is not required.

The reader is encouraged to open the ChipPlanner tool in Libero SoC Constraint Manager, Floor Planner tab, to visually inspect the device organization from a software tool viewpoint, as shown in Figure 3, page 5. Opening ChipPlanner requires a design, even a trivial test design, which has completed the Synthesis/Compile step. Holding the mouse over an element in the Chip Canvas pane opens a tool-tip describing the fabric component. See *UG0574: RTG4 FPGA Fabric User Guide* for more information about ChipPlanner tool.

Figure 3 • **ChipPlanner View of RTG4 Fabric Logic**



Each cluster of 12 flip-flops can only be fed by one global clock signal at a time. In other words, even though there are up to eight RGBs driving each half-row of logic, each cluster in that half-row can accept only one of the eight global signals at a time. The Libero SoC Place and Route tool automatically performs the mapping of user logic to fabric clusters. However, designers creating complex floor-planning constraints need to avoid the creation of placement regions whose boundaries cut through a cluster because it adds placement difficulty if sequential logic from two clock domains competes for access to the same cluster.

2.1.2 Resets

RTG4 FPGAs support designs utilizing asynchronous resets, synchronous resets, or both. RTG4 devices are designed with a single, fracturable, global asynchronous reset resource connected to all flip-flops (FFs) in the device. This single global asynchronous reset resource is called GRESET.

Note: The term GRESET can be referenced in multiple ways, and thus the relevant context must be applied. For example, GRESET is also the name of the RTG4 library primitive used to promote a fabric signal onto the GRESET resource. Similarly, there are GRESET input pins on RTG4, which can directly feed the GRESET resource.

This chip-wide reset resource can be fractured to create smaller, localized asynchronous resets called Row Global Resets (RGRESETs). An RGRESET resource is available per half-row of logic clusters, within the two vertical RGB stripes. The RGRESET macro in the Libero SoC macro library implements the fabric input to the RGRESET driver in each half-row of logic. Up to 206 local RGRESET resources can exist in the device, however, designers are instructed to minimize the total number of asynchronous reset nets in the design to aid with the Place and Route process. Minimizing the number of asynchronous resets is important because each half-row of the device can only be driven by a single asynchronous reset. This implies that user logic being reset by different asynchronous resets must be separated and placed onto different half rows in the fabric. In other words, each additional asynchronous reset "domain" imposes a physical placement constraint on the design. The usage of an RGRESET resource on a given half-row of the fabric reduces the number of RGBs available to drive clocks and globals into half row from eight RGBs to seven RGBs. GRESET and RGRESET resources can only drive the asynchronous reset input (ALn) of the RTG4 flip-flop (SLE). Even when Global Buffers are used to input an asynchronous reset to the device, the Libero software routes the signal from the global resource onto a GRESET or RGRESET resource before the reset signal reaches the SLE ALn input.

RTG4 also supports synchronous reset signals, which can use either regular routing resources or chip-level global routing resources depending on the design and the fanout. Therefore, there are less constraints surrounding the number of synchronous resets allowed in the design.

If the fanout of the synchronous reset ≤ 12, then all the flip-flops using this synchronous reset can fit within one logic cluster. In this scenario, Synplify Pro synthesis tool maps this reset into the flip-flop data input path using a combinatorial gate instead of connecting the reset to the SLn or enable pins of the RTG4 SLE. The reset accomplishes the same function when it gates the D-input to the flip-flop, but this adds delay on the input path and uses non-SET mitigated routing resources for the reset net. Users should watch for this scenario and evaluate the impact to the design requirements. In both cases, enabling the optional SET mitigation for synchronously reset FFs will filter transients on the entire FF data input path including D, EN, and SLn inputs.

When user HDL code includes the reset signal in the sensitivity list of an always block, the conditional reset check using if-else coding causes Synplify to map the reset signal to the asynchronous ALn input of the RTG4 SLE. This behavior occurs regardless of whether the user has synchronized the reset signal at a higher logic level.

If the user HDL does not include the reset signal in the sensitivity list, and the always block depends on the clock, the reset signal will be mapped as a synchronous reset. The synchronous reset is typically routed to the SLE SLn input, except in the scenario described above.

For scenarios where the always block includes both an asynchronous reset and a synchronized reset, Synplify will map the async reset to the ALn input and the sync reset to the SLE EN input.

Therefore, it is important to review the Libero SoC Global Nets Report to understand whether all reset signals have been mapped in the desired manner.

Avoid using the same reset signal both asynchronously and synchronously in different modules because it consumes extra unintended resources and can impact the SET tolerance of the signal. A synchronous reset fed into a different module or soft IP block that uses it asynchronously results in the automatic creation of a localized reset, using an RGRESET macro without any added logic cone triplication for SET mitigation. Similarly, an asynchronous reset fed into another module that uses it synchronously is not only a potential design flaw, but could result in the async reset (GRESET or RGRESET) branching onto a local routing resource and being promoted onto a chip-level global via a CLKINT macro, potentially allowing single-event transients to propagate onto the chip-level global used for the "synchronous" portion of the reset. For more information about creating SET mitigated reset signals, see Resets, page 13.

It is recommended to use active-low reset signals in the RTG4 design because the SLE macro natively uses active-low reset inputs (ALn and SLn). The SLE_RT macro for RTG4 is shown in the *RTG4 Macro Library User's Guide*. If an active-high reset is utilized, it causes the insertion of an inverter macro and can negatively impact the SET tolerance of the reset signal depending on which reset promotion macro, GRESET or RGRESET, follows the inverter in the reset path.

A typical reset practice in high-reliability designs is to create an asynchronously asserted and synchronously de-asserted reset signal. This scheme can allow the design to be placed in the reset state even in the absence of a clock signal. Proper synchronization of the reset release can help avoid metastability issues. An example scheme for creating SET mitigated, asynchronously asserted and synchronously de-asserted reset signals is discussed in Asynchronously Asserted, Synchronously De-Asserted Reset, page 20.

Note: The high-level design recommendation for reset signals in RTG4 is to consolidate as many asynchronously asserted, synchronously de-asserted reset nets as possible into one net distributed via the GRESET resource. Critical logic that must be able to enter the reset state even without a clock present should use this reset signal. Other reset domains whose release cannot be synchronized to the one main asynchronous reset domain should be converted into synchronous resets with the optional SET mitigation enabled on the synchronously reset FF loads. Although converting user logic and IP cores to consume a reset synchronously involves manual effort, the one-time effort can pay off across multiple RTG4 designs/projects. In contrast, trying to create an RTG4 design with many asynchronous reset nets requires dealing with manual logic triplication and RGRESET instantiation. Each additional asynchronous reset net adds implied constraints for the Place-and-Route tool and potentially more difficulty achieving timing closure for every RTG4 design.

During power-up, or after a DEVRST_N release, all the flip-flops in the device automatically preset or reset as part of the device power-on reset (POR) process depending on the static value of the SLE ADn input, without requiring any user intervention. The selection of preset or reset for all user logic FFs (including I/O FFs) that use an asynchronously asserted reset when the POR is asserted depends on whether the user logic has coded the flip-flop's reset state as a logic 0 or logic 1.

User flip-flops coded with synchronous reset or without any reset are also reset during the device POR, but the reset state during POR will default to a logic '0' based on the default tie-off of the ADn input to the SLE. For more information about the POR duration and release timing, see *UG0576: RTG4 FPGA System Controller User Guide*. This includes the flip-flops in the Microsemi hard IP blocks, such as LSRAM, uSRAM, uPROM interface, DSP Mathblock, SerDes PCS, FDDR, and CCC components. The reset state of the flip-flops within Microsemi hard IP blocks is pre-defined by the design of those IP blocks

and is not user configurable. The internal POR circuit propagates this global reset using the POWER_ON_RESET_N signal. See Conceptual Block Diagram of Power-On Reset Generation in *UG0576: RTG4 FPGA System Controller User Guide*.

Note that in the Power-Up to Functional and DEVRST_N to Functional Timing diagrams shown in *UG0576: RTG4 FPGA System Controller User Guide*, the POR release timing is slightly different for user FFs which are connected to the GRESET resource versus those which don't use GRESET, such as FFs using RGRESET, synchronous resets, or no user reset at all. After the POR is released, the FFs will start functioning, unless delayed by the assertion of a user reset. For example, a user design which has coded a FF to use a synchronous reset with a reset state of logic '1' will first be cleared by the device POR. After the internal POR releases, if the user's synchronous reset signal remains asserted, it will preset that FF to logic '1' until the user synchronous reset is released. In other words, the ALn and ADn inputs to the SLE have priority over the SLn and SD inputs. For information about the truth table of the SLE macro, see *RTG4 Macro Library User's Guide*.

Note that the RTG4 DEVRST_N input pin is an important pin to consider when designing the system for radiation tolerance. DEVRST_N provides a means to bring the RTG4 device back to a known state without requiring a power-cycle. This feature could prove useful in a radiation environment. Alternatively, power-cycling the device can accomplish the same behavior, but could also have a bigger impact to the system when power rails are shared. For more information about DEVRST_N, see *DS0130: RTG4 FPGA Pin Descriptions Datasheet* and *AC439 & AC453: RTG4 FPGAs Board Design and Layout Guidelines Application Note*.

2.2 Radiation Mitigation Designed into RTG4

RTG4 radiation tolerant FPGAs use radiation hardening by design techniques throughout the device as described in *WP0191: Mitigation of Radiation Effects in RTG4 Radiation-Tolerant FPGAs White Paper*. The entire device is hardened against Single Event Latchup (SEL). Similarly, all the programmable flash switch elements use a push-pull configuration to achieve Total Ionizing Dose (TID) hardening without exhibiting timing performance degradation across the supported total dose limit. The Sequential Logic Element (SLE) has built-in SET mitigation and Triple Module Redundancy (TMR) for SEU mitigation. SET filtering of the data (D), enable (EN), and synchronous load (SLn) inputs of the SLE can be enabled or disabled on a per-instance basis, depending on the design performance requirements versus radiation tolerance requirements. The per-instance SET mitigation control is performed using **Netlist Design Constraints** (.ndc) files in Libero SoC. Refer to the *Libero SoC PDC Commands User Guide* entry for the "set_mitigation" constraint. Turning on the SLE SET filters will add between 600ps (typical) to 1ns (max) delay to the data entering the SLE, and therefore some designs will choose to enable the SLE SET filters for critical logic blocks only, or those which can tolerate the performance impact. The description above, and respective device radiation test reports, show that RTG4 offers many radiation tolerance improvements over previous flash-based FPGAs such as RTProASIC3EL. Additionally, RTG4 includes SET mitigation above that available in the RTAX-S Antifuse FPGA, since the target RTG4 applications could potentially run faster and thus be more likely to clock-in a radiation-induced glitch.

The global clocking and asynchronous reset resources have built in SET hardening including triplicated drivers and metal lines.

Promotion of fabric routed signals onto hardened global routing resources via CLKINT or RCLKINT could allow SETs to propagate onto the global resource if the design techniques described in the Clocks, page 9 are not implemented. The RTG4 fabric CCC and the internal RC Oscillator also have built-in radiation hardening with the following exceptions:

- Specific CCC Clock inputs to the PLL are not hardened including: fabric clock inputs, the input clock reference divider (RFDIV), the feedback clock input divider (FBDIV), and the delay lines associated with the reference and feedback clock input paths.
    - The PLL has built-in low pass filters, which can help filter out glitches in the input path
- The CCC Y# outputs to local fabric routing are not SET hardened

Furthermore, the Radiation Tolerant PLL in the RTG4 fabric CCC can also be used in a Triple-Redundant configuration by selecting the PLL Internal feedback mode. Using the PLL in triplicated mode can improve radiation upset performance, when coupled with the auto-reset logic (see PLL Lock Output as a

Design Reset Source, page 21). The following table lists the SET and SEU hardening levels for the clock and reset related features.

Table 1 • **SET and SEU Hardening by Design in RTG4 Clock and Reset Resources**

Category	Feature	Hardening Applied
Clock	Global Routing Resources	SET Hardened
	CCC (without using PLL)	Fully SET and SEU Hardened when using dedicated clock inputs and global outputs without using optional delay lines
	Radiation Tolerant PLL	SET and SEU hardening applied, see test reports for upset rates
	SpaceWire Clock Recovery Circuit	Recovers SET Hardened Clock using built-in glitch filter and can drive global clock routing resource
	Internal RC Oscillator	SET Hardened
I/O	MSIO	Input buffers are SET hardened
	MSIOD	Input buffers are SET hardened
	DDRIO	Not SET hardened
Asynchronous Reset	GRESET	SET hardened routing resource plus built-in glitch filter
	RGRESET	SET hardened routing resource with three inputs and majority voting logic, no built-in glitch filter
FPGA Fabric	Flip-Flops included in: • Fabric Sequential Logic Elements (SLE) • Mathblock pipeline registers • LSRAM and uSRAM pipeline registers • I/O Registers (Input, Output, Output Enable)	• SEU hardened via Triple-Module Redundancy (TMR) • Optional SET mitigation available to filter glitches on input data path including data (D), enable (EN) and sync load (SLn) inputs • Called SET-Mitigated Triple Module Redundant Flip-Flop (STMR-FF)
	LSRAM/uSRAM	• Data Interleaving to prevent Multi-Bit Upsets (MBU) • Optional Error Correcting Code (ECC) mode for Single-bit Error Correction, Double-bit Error Detection (SECDED) • Optional SET mitigation
	Mathblock	• SEU hardened • Optional SET mitigation
FDDR	East and West FDDR blocks	• SEU and SET hardening, except DDRIO buffers, as noted above • Optional SECDED mode
SerDes	PCS	SEU and SET hardening
	PHY	SEU hardening, except SerDes block PMA registers, as noted in *RTG4 FPGA High Speed Serial Interfaces User Guide*

Refer to the RTG4 Radiation Tolerant Feature Summary table at the following link for a complete summary of the built-in radiation mitigation and measured upset rates: https://www.microsemi.com/document-portal/doc_download/1245569-rtg4-radiation-tolerant-features-summary.

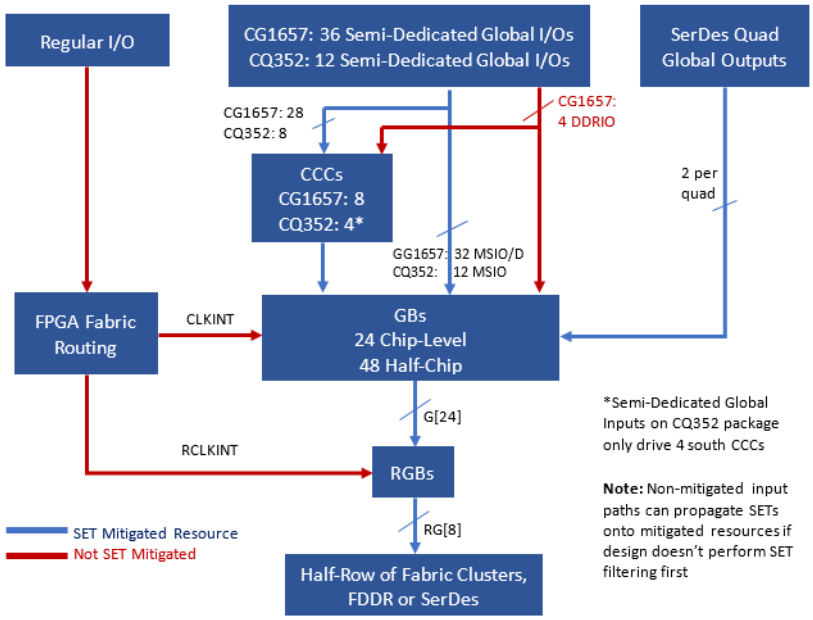
2.3 Selecting Input Sources

The following section describes the various input sources for clock and resets, and their radiation tolerance. This description enables the designer to effectively utilize the RTG4 built-in radiation mitigation to create hardened clock and reset networks.

2.3.1 Clocks

The following figure shows the various paths that an input clock can take to reach user logic within a fabric cluster or to reach the inputs of hard IP blocks such as the FDDR memory controllers or SerDes quads.

Figure 4 • **Global Clock Input Options on RTG4**



As shown in the preceding figure, there are several ways to bring in an external clock into the RTG4. Some paths are entirely SET mitigated by design, while others require additional user steps to add mitigation. The I/O pin nomenclature is described in *DS0130: RTG4 FPGA Pin Descriptions Datasheet*.

- The input naming shows that some semi-dedicated pins have several possible functions, such as:
- Regular user I/O denoted by bank type, pin number, differential pair P or N member, and bank number. For example, MSIOD58PB1.
  - MSIO and MSIOD pins have built-in SET mitigation, while DDRIO pins do not.
  - If regular inputs drive signals onto regular fabric routing, then the path is no longer SET mitigated because regular routing nets are not internally triplicated.
  - The combination of MSIO/MSIOD and global routing will allow an input signal to remain SET mitigated.
- Dedicated global input pin, GB## which directly drives the internal global network via a hardwired connection from the input buffer to the Global Buffer (GB). This is an SET mitigated path.
- CCC input pin denoted by the CCC location and CLK input number, such as CCC_NE1_CLKI0. These dedicated CCC input paths are SET mitigated when they come from an MSIO or MSIOD input pin.
- SpaceWire Data and Strobe input pairs associated with the two CCC clock recovery blocks within each fabric CCC. The output of the clock recovery block can be directly fed to a GB via a GL# output of the CCC. The SpaceWire clock recovery blocks have a built-in glitch filter to prevent glitches from being propagated onto the global network.

Each SerDes quad can directly feed two signals onto GB global routing resources using SET mitigated paths, as shown in Figure 4, page 9. This is useful when routing the recovered RX clock for one of the four lanes to the fabric receive data path logic and the SerDes EPCS RX flywheel FIFO fabric interface. Similarly, the SerDes reference clock for the link, or the TX clock for one of the four lanes, can be routed to the transmit data path logic and SerDes EPCS TX flywheel FIFO fabric interface. Designs using multi-lane wide links in common clock systems with 0 ppm offset between transmitter and receiver clocks can potentially minimize the number of fabric RGBs required to drive the EPCS flywheel FIFO clock inputs by sharing clocks across multiple lanes in the quad.

As shown in Figure 4, page 9, the RT4G150 in the CG1657 package supports 32 CCC input pins, which can provide an external reference clock to the RTG4 fabric clock conditioning circuits (CCCs). Of these 32 CCC input pins, there are 28 pins located on the MSIO and MSIOD banks and four are located on the DDRIO banks. The MSIO and MSIOD input buffers are triplicated to add built-in SET mitigation by design. This allows the CCC input path to be SET mitigated. In contrast, the DDRIO CCC inputs require additional mitigation, such as glitch filtering, to add SET mitigation on the clock input path.

The RT4G150 in the CQ352 package supports eight CCC input pins, which drive the four south CCCs in the device. All eight of the CCC input pins are MSIO inputs with built-in triplication.

The following table lists the CG1657 input reference clock pin selection for the North-East 1 CCC (CCC_NE1). To make use of the built-in SET mitigation, select either pin T39 Clock Input 0 (CLKI0) or pin AA34 Clock Input 1 (CLKI1) because these pins use MSIOD inputs.

Table 2 • **CCC_NE1 Reference Clock Input Pin Selection**

CCC Location	CCC Input	Dedicated CCC I/O Pin Name	RT4G150-CG1657 Pin #	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

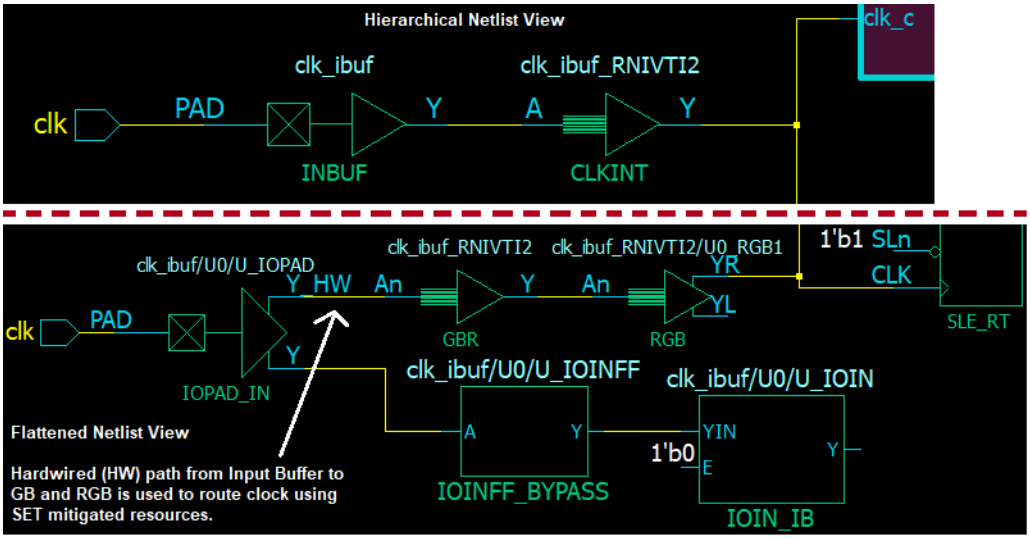
The following RTG4 fabric paths are not SET mitigated:

- Regular routing resources
- Combinatorial logic cells (4-input LUTs)

The designer must consider if the clocks routed on regular routing nets or pass through any combinatorial logic, such as an inverter, on its way to the CLK inputs of the SLE D-Flip-Flop. In some designs, there might not be a complete SET mitigated path available for all the clocks required, but understanding which clocks are SET mitigated and which are not allows the designer to complete an analysis of the radiation tolerance of the design and make conscious trade-offs.

The following figure shows an example where the hierarchical view of the design appears to bring an external clock into the RTG4 using a regular I/O, and fabric routing to a CLKINT macro to promote the clock onto a GB resource. To ensure a fully SET mitigated clock path, the input clock signal is connected to a GB# capable input pin, such as MSIO347PB4/GB23/CCC_SE1_CLKI3/SPWR_SE1_1_RX_DATA_P. This allows the routing for this clock input to use the hardwired path to the GB resource. This can also be enforced by manually instantiating a CLKBUF macro in the top-level design and assigning the clock input signal to the PAD port of the CLKBUF. The designer can confirm the desired implementation has been achieved by reviewing the Global Nets Report produced after the Layout step in Libero SoC, and by reviewing the post-Layout flattened netlist view in Netlist Viewer. The output of the CLKBUF can be used as an SET mitigated global clock within the user design.

Figure 5 • **Using Hardwired Paths for External Clock Input**



Global Net Report

Microsemi Corporation - Microsemi Libero Software Release v11.9

I/O to GB Connections

Port Name	Pin Number	I/O Function	From	To	Net Name	Net Type	Fanout
clk	G29	MSIO347PB4/GB23/CCC_SE1_CLKI3/ SPWR_SE1_1_RX_DATA_P	clk_ibuf/U0/U_IOPAD:Y_HW	GBR[23]	clk_ibuf	HARDWIRED	1

To maintain an SET hardened path for routing the external input clock to the hardened global network, the CCC outputs selected must be the GL0, GL1, GL2, or GL3 outputs instead of the Y0, Y1, Y2 or Y3 outputs. The GL# outputs are hardwired to the RTG4 GB resources. The Y# outputs drive regular fabric routing resources, which are not SET mitigated.

From a radiation tolerance point of view, consider whether the distribution of the global clock utilizes the RTG4 fabric PLLs within the CCC. The RTG4 fabric PLL supports usage in two distinct modes:

1. Single PLL mode, via CCC Internal or External feedback configuration.
2. Triple-module redundant (TMR) mode, when configured for PLL internal feedback, in which three PLLs work together to improve the radiation tolerance of the PLL and generate a voted lock signal.

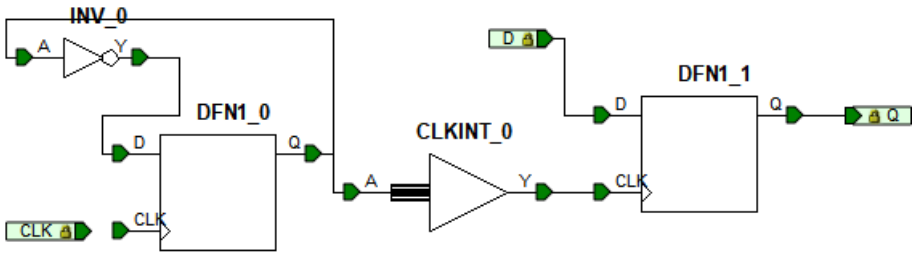
Figure 5 shows that clocks sourced from regular I/O or fabric logic can be promoted onto the RTG4 SET mitigated global network or row-global network via a CLKINT or RCLKINT macro but will not be SET mitigated. These clocks can be SET mitigated by applying SET filtering on the signal prior to promotion onto a global resource. A fabric signal being promoted onto a hardened GB or RGB, via CLKINT or RCLKINT respectively, can potentially allow an SET to propagate onto the internally triplicated global network. To avoid this scenario, employ SET filtering on the signal prior to promotion onto a global resource.

To add SET filtering on a fabric routed user clock signal, replace the CLKINT with a CCC glitch filter found in an unused SpaceWire clock recovery block before promotion onto a global routing resource. When the clock recovery blocks are unused for SpaceWire clock recovery, the CCC glitch filter is available to filter glitches or SETs on user input signals to the CCC. The filtered signal is then assigned to one of the CCC GL# outputs to drive a GB resource. Take note of the relative location of the CCC selected and the fabric clock source to avoid adding extra routing delay. The CCC SpaceWire glitch filter requires a falling edge on the input signal to latch-in the asserted output enable signal before the filtered signal passes through to the CCC GLx output. Therefore, the glitch filter works well for clock signals since they are guaranteed to have a falling edge. In contrast, the CCC glitch filter might not be suitable for static signals, such as an active-low reset input. Such a reset is typically active (low) during device/design startup, and then transitions from low to high when the reset is released. In this case, a falling edge does not occur and the signal is not guaranteed to pass through the CCC clock gating logic

onto the GLx output. A fabric-based SET filter would be more suitable for an active-low reset input (see Resets, page 13).

Figure 6 shows the replacement of a CLKINT by a CCC glitch filter to promote a fabric clock onto a global resource while adding SET mitigation. In the original circuit, the CLKINT is driven by a fabric signal via local regular routing, which could allow an SET to propagate onto the SET mitigated GB resource via the CLKINT. Libero SoC also generates a warning in the Log Window and in the Global Nets Report generated during the Layout process.

Figure 6 • **Replacing non-SET Mitigated Fabric Clock Promotion with CCC Glitch Filter Global Promotion**

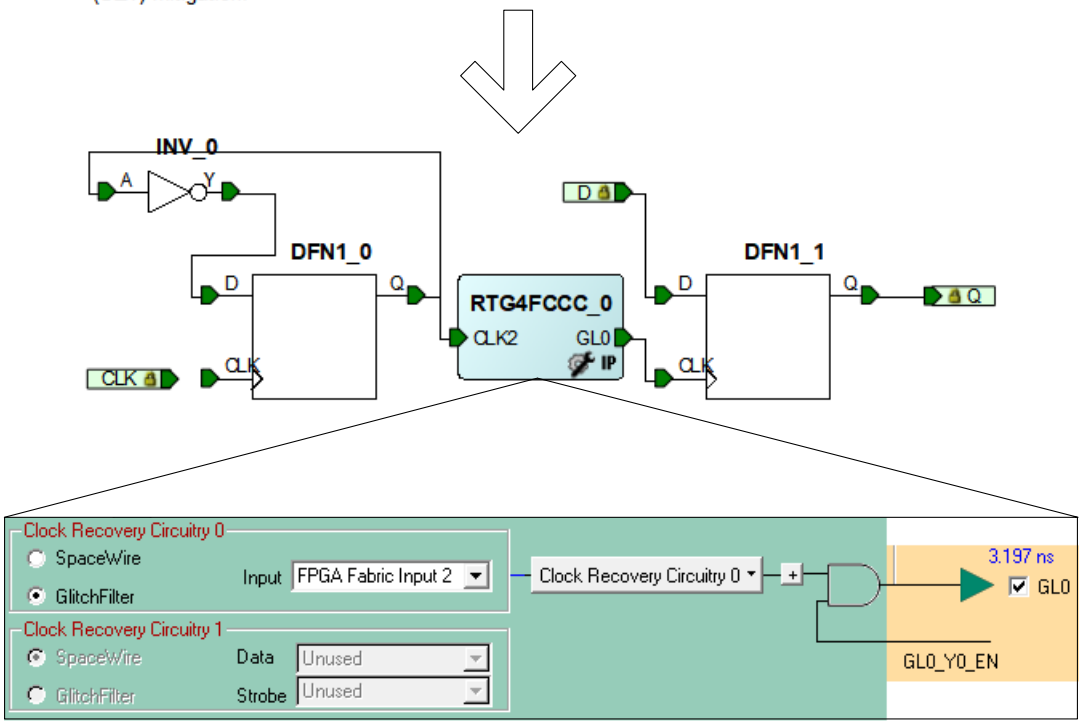


Warnings

The following clocks or resets are driven by fabric-generated or local nets and are not single event transient (SET) mitigated:

	From	From Location	I/O Bank	To	To Location	Net Name	Fanout
1	DFN1_0:Q	(156, 244)	-	CLKINT_0:An	GBL[20]	DFN1_0_Q	2

Microsemi recommends the use of dedicated global clocks with built-in single event transient (SET) mitigation.



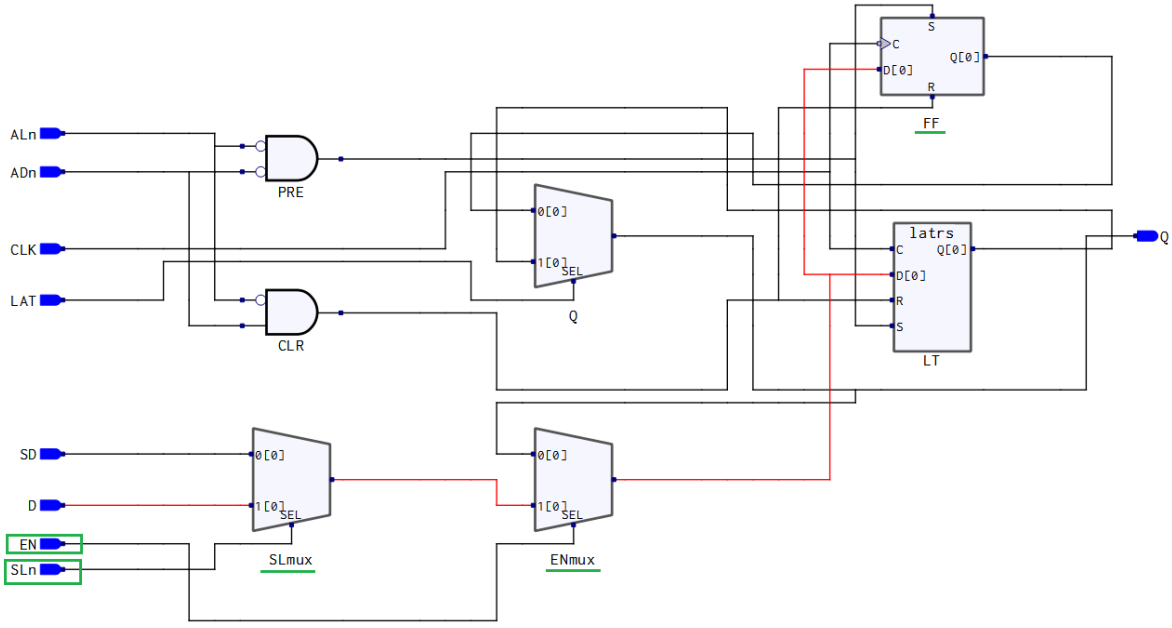
2.3.2 Resets

2.3.2.1 Synchronous Resets

In RTG4, the SET Mitigated Triple Module Redundant Flip-Flop (STMR-FF) can be used with SET mitigation enabled to filter out Single-Event Transients (SETs) on the flip-flop's input data path. The SLE input data path includes the data (D), enable (EN), and synchronous load (SLn) inputs, as shown in Figure 7 from the Synplify Pro technology view of the RTG4 SLE macro. Therefore, critical logic that uses synchronous resets can be SET mitigated by simply enabling SET mitigation on those FFs. This provides the flexibility to route synchronous resets using regular FPGA fabric routing because SLn glitches will be filtered by the RTG4 SLE with SET mitigation enabled. If the fanout of the synchronous reset net exceeds the global promotion threshold, the synchronous reset will be promoted to a global routing resource.

If the user does not enable SET mitigation on the synchronously reset FFs, it's possible for a SET on a regular routing resource to trigger an accidental reset. When these resets are synchronous to a clock domain, a SET would have to occur during the setup or hold time of the active clock edge to cause an error. As a result, these resets are more tolerant to Single-Event Transients than asynchronous resets. However, as the clock frequency of the design increases, this argument loses significance because the clock edges are more frequent and closer together. If this is a concern, it can be addressed by enabling the STMR-FF SET mitigation option for the instances of concern.

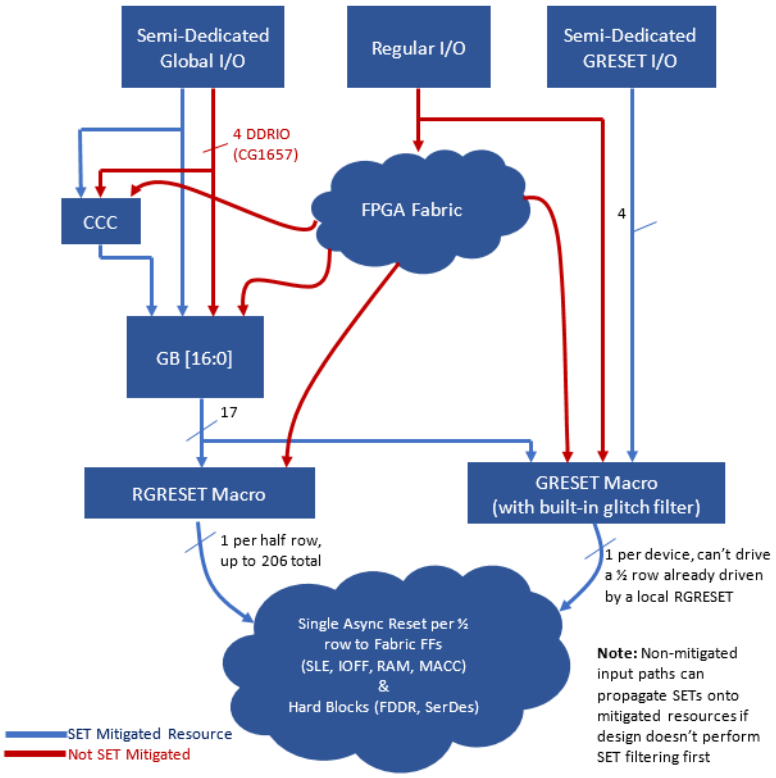
Figure 7 • **Graphical Representation of SLE_RT Macro from Synplify Pro**



2.3.2.2 Asynchronous Resets

This section discusses Asynchronous resets. For Synchronous reset information, see the Synchronous Resets, page 13. Figure 8 shows the various ways to route an asynchronous reset signal into RTG4. If the external asynchronous reset path enters the device via an SET mitigated path and remains on a mitigated path all the way to the GRESET or RGRESET macros, then the reset signal will also be SET mitigated by design. If there are non-mitigated paths involved in the asynchronous reset routing, then extra considerations must be made to fully utilize the SET mitigation built into RTG4.

Figure 8 • **Possible Paths for Asynchronous Reset Routing**



The GRESET macro contains a built-in SET filter, which provides additional flexibility when designing the asynchronous reset network. GRESET can be fed by a regular input buffer, or a fabric generated signal, and still provide SET mitigation on the signal driving the internal GRESET global asynchronous routing resource.

In contrast, the RGRESET macro does not have a built-in SET filter to block the passage of transients onto the triplicated RGRESET routing resource. Therefore, the RGRESET macro provides three input ports to triplicate the logic cone feeding the RGRESET inputs so that a majority vote can be performed when promoting the signal onto a row global reset resource. By triplicating the logic cone driving the RGRESET, a transient on a single branch of the logic cone does not propagate onto the SET mitigated RGRESET routing resource. Triplicating the logic cone feeding the RGRESET requires manual user planning because the Libero design flow does not perform automatic triplication of the user logic. When a GB resource is used to drive an RGRESET, manual triplication is not required because the GBs are internally triplicated routing resources.

The implication is that low fanout asynchronous resets which are created as part of IP cores, or lower level design blocks, could potentially be automatically promoted onto an RGRESET network without triplication of the logic cone generating the reset signal. Review the Compile and Layout reports generated by Libero SoC to ensure all design resets are well understood.

When reviewing the Libero SoC Global Net Report, the FF load of smaller RGRESET networks is an important consideration. When Libero SoC automatically inserts RGRESET macros for smaller asynchronous reset nets, in some cases, a global buffer (GBR or GBL) is inserted to promote the reset signal onto a global net before reaching half-row RGRESET drivers. It occurs when the loads of the asynchronous reset net cannot easily fit on one half-row of fabric logic. In contrast, if the asynchronous reset load after the RGRESET easily fits into one half-row of logic clusters, the GB is not necessary. Understanding this point is required when manually instantiating RGRESET macros fed by manually triplicated logic cones. If timing violations are observed on this asynchronous reset net, the user should determine how to proceed, such as using one of the examples listed below and validating that it meets the project's radiation tolerance goals, potentially via radiation beam testing.

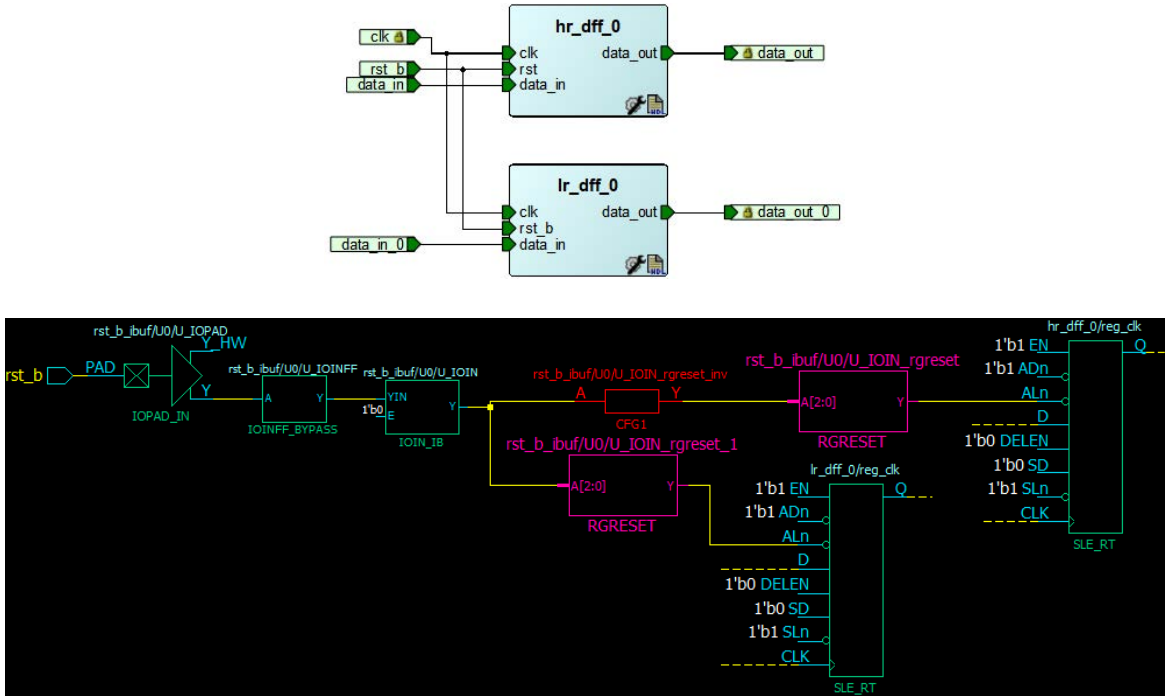
- In one approach, an async-on/sync-off reset synchronizer with FFs that have SET mitigation enabled could be placed close to the RTG4 global buffers in the center of the chip to minimize the length of non-SET mitigated regular routing used between the reset synchronizer output FF and the GB/CLKINT inserted/instantiated before the auto-inserted RGRESET is added by Libero SoC during Place and Route.
- In another approach, the user might consider resolving timing violations on this asynchronous reset net by triplicating the reset logic cone and manually instantiating the RGRESET macro, but with the addition of a GB (CLKINT) instance before each of the three RGRESET inputs to prevent skew on the reset signal as it is distributed across multiple half-rows of logic clusters. This idea will quickly consume global nets and should be avoided, if possible.

Instead of performing this detailed and manual RGRESET analysis for every asynchronous reset net in every RTG4 design, the high-level design recommendation for RTG4 is to consolidate as many asynchronously asserted reset nets as possible into one net distributed via the GRESET resource (as shown in Figure 11, page 17 and Figure 14, page 20). Other smaller reset domains whose release cannot be synchronized to the one main asynchronous reset domain should be converted into synchronous resets with optional SET mitigation enabled on the synchronously reset FF loads. Although converting user logic and IP cores to consume a reset synchronously involves manual effort, the one-time effort can pay off across multiple RTG4 designs/projects versus dealing with manual logic triplication and RGRESET instantiation for every RTG4 design.

As another example, an asynchronous reset can be used as an active-low signal by most of the design, but used as active-high by a sub-module. If the reset polarity is consistent throughout the design, the highest fanout active-low reset is automatically promoted onto the SET filtered GRESET resource by the Libero Place and Route tool. However, in this scenario where the reset is used as both active-low and active-high, the resulting layout uses two RGRESET resources without any logic cone triplication to add SET mitigation, and with an added inverter delay before the reset edges reach the active-high reset flip-flops.

Figure 9, page 16 shows an example where an active-low reset "rst_b" is sent to both active-low and active-high reset logic. The hr_dff_0 requires an inversion of the rst_b to create an active-high reset, and thus the mapped netlist view of this design shows an added inverter CFG1 and fabric routing to create the active-high reset.

Figure 9 • **Active-Low RST_B Signal Used as Active-High Reset on HR_DFF_0**



When triplicating the logic cone driving an RGRESET macro, if the logic cone generating the reset signal contains a flip-flop, then start the logic triplication at the instance of the last FF stage before the RGRESET macro with the SLE SET filter enabled. Enabling the SET filter and triplicating the logic cone from the last FF stage onward filters transients prior to the FF.

Figure 10, page 17 shows an example of logic cone triplication where STMR SLEs with SET filters enabled are part of the logic cone. For more information about how to enable SLE SET filters by instance, see the set_mitigation NDC command documented in the *Libero SOC PDC Commands User's Guide*.

Figure 10 • **Connecting RGRESET Macro to Triplicated Logic Cone**

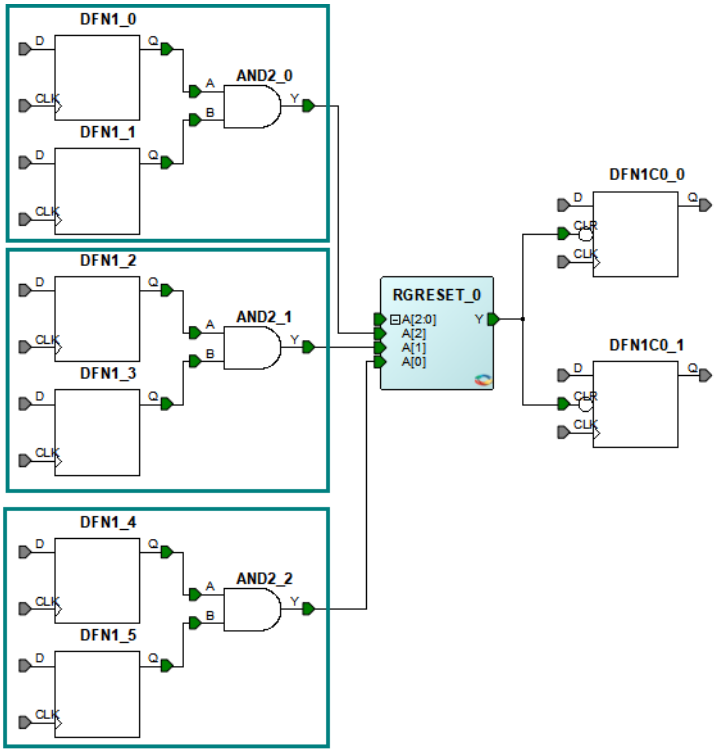
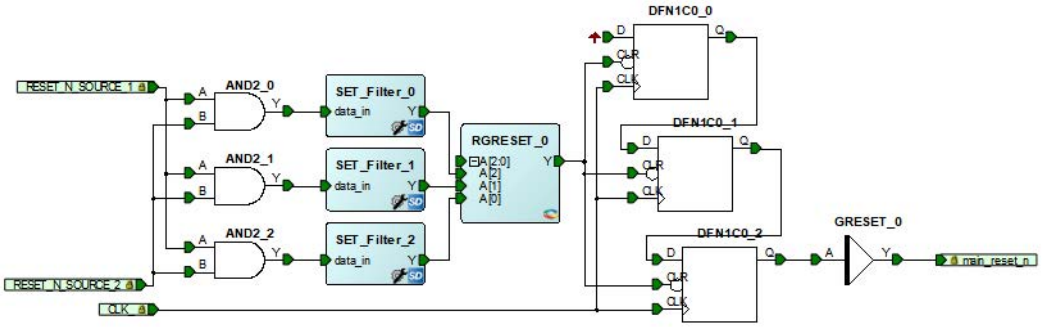


Figure 11 • **Triplicated Logic Cone Connected to RGRESET Macro**



There are scenarios where logic cone triplication is not possible or suitable, and in those cases, fabric based SET filters might be required to achieve the user's desired level of SET mitigation. If the logic cone feeding the RGRESET is combinatorial, then the entire path should be triplicated or user SET filtering should be added in the path. For example, Figure 11 shows two active-low reset sources being combined to generate an active-low asynchronous reset to three synchronizer flip-flops. If the two reset sources are components which cannot be triplicated, then it is difficult to add SET mitigation to the reset being promoted onto the RGRESET resource. For example, the lock output from the fabric PLL and the POWER_ON_RESET signal from the SYSRESET macro are always routed via regular fabric routing resources, therefore, they are not SET mitigated by design. Furthermore, If async reset sources do not have any STMR SLEs in the path, it can imply triplicating the logic cone all the way back to the input pins sourcing the signal, thus, using three device input pins for each reset source, which is not likely to occur in a typical design. As another example, if one of the reset sources is the POWER_ON_RESET signal from the SYSRESET macro, then it is not possible to triplicate that logic cone since there is only one SYSRESET macro. It is also impractical to triplicate the logic cone if one reset source is the lock output of a fabric PLL. Therefore, adding fabric-based SET filtering can create three SET filtered branches of the input signal feeding the RGRESET. If an SET occurs on one of the reset sources, it reaches all three branches of the signal, and if the transient pulse width is shorter than the delay window created by the fabric SET filter, it will prevent propagation onto the RGRESET resource. If an SET occurs on the fabric

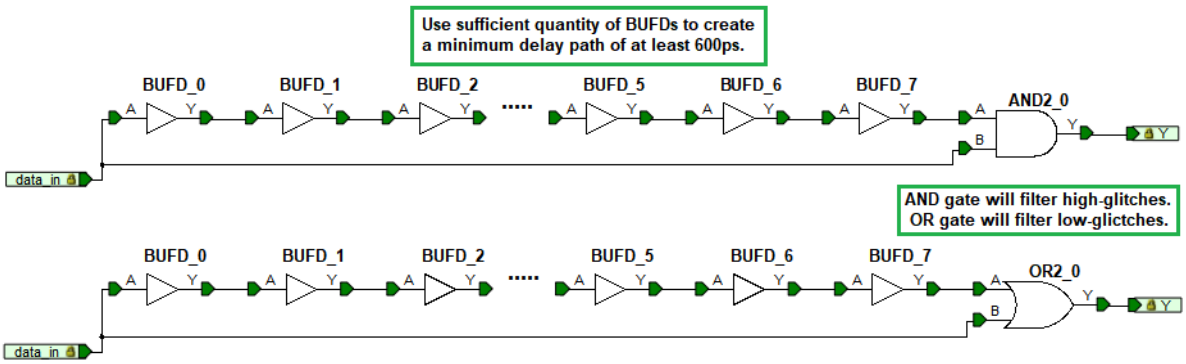
routing on one of the distinct signal branches feeding the RGRESET, then the other two inputs to the RGRESET will over-vote the SET.

Figure 12 shows an example of a fabric-based SET filter made of BUFD macros, which are not optimized away during Synthesis and Compile. The fabric-based SET filter compares the input signal with a delayed version of itself and if they agree, pass the signal to the output. The choice of gate used at the end of the filter depends on whether the application is concerned with filtering out high or low pulses. An AND gate filters out a high pulse when the data is low. An OR gate filters out a low pulse when the data is high. For an active-low reset signal, if the application cannot tolerate a high glitch, which causes the temporary and premature release of the asserted reset, then an AND gate can be used. On the other hand, if the goal is to prevent a low glitch from unnecessarily resetting the system, then an OR gate can be used.

The delay path created by the fabric-based SET filter should be approximately 600 ps in min-delay analysis to align with the minimum delay used by the built-in SET filters on the RTG4 STMR SLE. The user can also create an inclusive placement region around the SET filter to better control the delay between iterative Layout operations.

Starting with Libero SoC v12.0, the user can employ a primitive macro called BUFD_DELAY to generate the delayed version of the signal to be filtered. The BUFD_DELAY macro has a longer cell delay than a single BUFD buffer. The BUFD_DELAY buffer delay is approximately 0.4ns at worst-case military temperature conditions. This means that fewer BUFD_DELAYs would be required to create a fabric-based SET filter. For more information, See *RTG4 Macro Library User's Guide* accompanying Libero SoC v12.0 or later.

Figure 12 • **Fabric-Based SET Filter Using BUFD Macros**



Some designs might require a fabric-based SET filter that filters-out both high and low glitches, instead of the filters depicted in Figure 12. In that case, the final AND/OR gate can be replaced with a "guard gate" implementing a fabric-based 3-input majority voter (MAJ3), similar to that implemented in the RGRESET macro.

The 3-input majority voter implements the function:

$$Y = (A0 \times A1) + (A0 \times A2) + (A1 \times A2)$$

The three inputs to the MAJ3 voter would be the following, as shown in Figure 13, page 19:

- The original signal being filtered.
- The BUFD/BUFD_DELAY delayed version of the signal.
- The output of the MAJ3 voter fed-back.

In steady state, all three inputs agree and the output does not change.

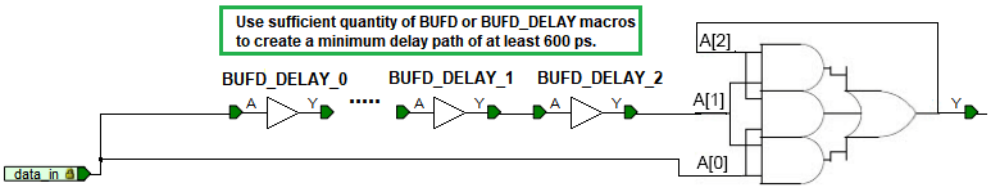
During a signal transition, input 1 changes first. Since only 1 input is different, it is outvoted, and the output of the MAJ3 does not change. Next, the delayed version of the input signal changes, meaning that the two transitioned inputs now agree, and they outvote the third input, which was being fed back from the MAJ3 output. The gate changes state, and the output now agrees with the inputs.

When a glitch occurs on the first input, such that its pulse-width is smaller than the delay used on the second input, even though the first input changes, it is outvoted by the delayed and fed-back signal values. Since the glitch duration is shorter than the filter delay, the first input changes back to agreement

before the delayed input changes. When the delayed input changes, it also is outvoted and the glitch does not propagate through the guard-gate. Therefore, the output doesn't change and the transient is filtered out.

The following figure shows an example of the fabric-based SET filter implementing a MAJ3 voter guard gate.

Figure 13 • **Fabric SET Filter**



The following module shows an example of the code required to create a fabric-based SET filter with a 3-input majority voter guard-gate:

```
module Fab_SET_Filter_MAJ3( A, Y );


    parameter Length = 3;


    input A;
    output Y;


    wire delayed_signal[Length-1:0];
    wire delayed_output;


    genvar i;
    generate
        for (i = 0; i < Length; i = i + 1)
        begin : buf_chain
            if (i == 0)
            begin
            BUFD_DELAY BUF_start ( .A(A), .Y(delayed_signal[i]) );
            end
            else if (i < Length - 1)
            begin
          BUFD_DELAY myBUFD ( .A(delayed_signal[i-1]), .Y(delayed_signal[i])
);
            end
            else if (i == Length - 1)
            begin
           BUFD_DELAY BUF_last ( .A(delayed_signal[i-1]), .Y(delayed_output)
);
```

```
            end
        end
    endgenerate

    assign Y = (A & delayed_output) | (A & Y) | (delayed_output & Y);

endmodule
```

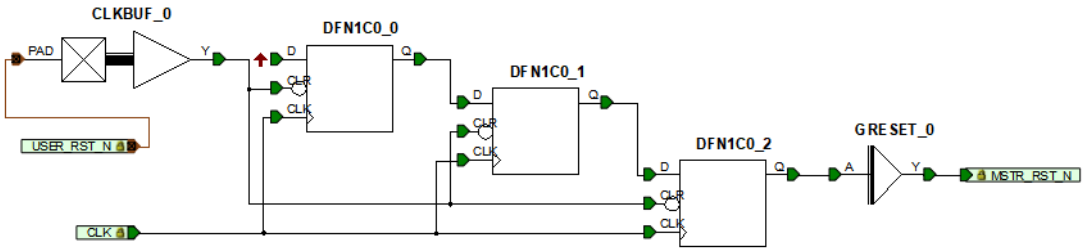
2.3.3 Asynchronously Asserted, Synchronously De-Asserted Reset

Many high reliability FPGA designs utilize a reset, which can be asynchronously asserted (async-on) and synchronously de-asserted (sync-off). The reset release is typically synchronized to the respective clock domain of the flip-flops it affects. An increase in the number of asynchronous reset nets can increase the challenge associated with placing and routing the design, which in turn can decrease the designs timing performance. To address this, designers are encouraged to consolidate asynchronous resets as much as possible. For example, if there are several synchronous clock domains, then consider synchronizing the reset release to the slowest of these clocks and use that reset for all the flip-flops in all the synchronous clock domains.

Typically, the highest fanout asynchronous reset in the design uses the GRESET routing resource and low fanout asynchronous resets use RGRESET resources. Therefore, when creating an async-on, sync-off reset signal, an RGRESET should be used to clear the synchronizer flip-flops. It is also recommended to confirm that the same half-row of the fabric is used for the placement of the synchronizer flip-fops and the RGRESET macro, such as those used in Figure 11, page 17.

During the design of the async-on, sync-off reset signal, consider which reset sources will be used to assert the reset and which SET mitigated paths are available. As shown in Figure 14, if the only async reset source is an external logic reset signal, an MSIO/MSIOD input pin with the GB## functionality can be used to input the external reset signal to an RGRESET resource using hardwired SET mitigated paths. The GRESET macro distributes the asynchronously asserted, synchronously de-asserted SET mitigated reset to the user logic.

Figure 14 • **Async-On, Sync-Off Reset Logic Using Single External Reset Source**



The design in Figure 14 can be extended if there are multiple asynchronous resets required within the overall RTG4 design, due to the number of asynchronous clock domains. The highest fanout reset can use the GRESET resource as shown in Figure 14. The other, lower fanout, internal reset signals can use RGRESET resources instead of GRESET. These additional resets can still use the external user reset signal coming in on a GB## pin, but the actual reset synchronizer circuit must be triplicated, with SLE SET filters enabled, so that three copies of the synchronized reset drive the RGRESET macro fabric inputs. This adds SET mitigation so that an SET isn't propagated onto the RGRESET resource. The triplication isn't required for the signal routed to GRESET because the GRESET macro has a built-in glitch filter.

On the other hand, if there are multiple reset sources which need to be combined and used to assert the async reset to the synchronizer flip-flops, then the user should evaluate if there is a way to add SET mitigation to the combined reset signal. One example is shown in Figure 11, page 17 by using a fabric based SET filter.

2.3.4 SYSRESET Macro Usage

After discussing the RTG4 asynchronous reset architecture and ways to create SET mitigated resets in a user design, another point to consider is the usage of the SYSRESET macro. This macro provides the user design with direct access to the POWER_ON_RESET_N signal which is asserted during power-up or after a DEVRST_N assertion. RTG4 has a built-in power-on/DEVRST_N reset which is distributed to all flip-flops in the device, regardless of the SYSRESET macro usage. Although many RTG4 demo designs use DEVRST_N and SYSRESET as a logic reset for the user design, this is not required. DEVRST_N is a chip-level reset, similar to a power-cycle, to bring an unresponsive system back to a known state. Designers who wish to implement a user logic reset can instead use one of the reset sources shown in Figure 6, page 12, such as an I/O pin. The SYSRESET macro usage is optional and often not required. Note that the output of the SYSRESET macro, POWER_ON_RESET_N, is a fabric signal which is routed using regular local routing resources. This means that user action is required to add SET mitigation to this reset source. One use-model where the SYSRESET macro is useful would be to keep user flip-flops which are synchronously reset held in the reset state longer than they would be by default. As mentioned earlier, user flip-flops which aren't driven by the GRESET resource will be released from reset to user control sooner than the flip-flops which are connected to the GRESET resource. According to the RTG4 System Controller User's Guide, the time delta between the non-GRESET flip-flops entering user control versus the POWER_ON_RESET de-assertion is 6.64 us. If the POWER_ON_RESET_N output of the SYSRESET macro is combined with the other user reset sources driving these non-GRESET flip-flops, they will see the same reset assertion duration as the GRESET connected flip-flops. This idea is also dependent on the design specific timing for the other reset sources. For example, if the user reset sources are all asserted longer than the POWER_ON_RESET signal, and de-asserted in some coordinated manner, perhaps aligned with respect to each clock domain, then the usage of the SYSRESET macro is not necessary. Another use model for SYSRESET is for user logic coded for a synchronous preset to logic 1. In this case, these flip flops will be initially cleared by the hardware embedded power-on reset. However, when they are released to user control, the SYSRESET macro output will drive them to preset. When combining multiple reset sources together, apply the techniques mentioned above to add SET mitigation, where possible.

2.3.5 PLL Lock Output as a Design Reset Source

The lock output from a CCC's PLL is often utilized as a reset source, combined with other user reset signals, to create the user logic reset for a design. This practice is employed to ensure the design is put into a safe or known state when there is potential degradation on the quality of the PLL output clock, rather than allowing potentially erroneous design operation. Again, this combined reset signal should be analyzed to see if any of the SET mitigation techniques discussed above can be applied to enhance the SET robustness of the reset signal.

The RTG4 fabric PLLs have been tested for radiation effects and the reports can be found on the *Microsemi Radiation and Reliability Data* web page.

The following list summarizes the PLL lock upset rates when using an externally sourced reference clock, in the presence of heavy ions under GEO solar min conditions:

- Single PLL can experience a self-recoverable loss of lock every 33 years.
- Single PLL can experience a loss of lock requiring PLL reset every 6,800 years.
- TMR-PLL can experience a loss of lock requiring PLL reset every 145 years.

The heavy-ion test result shows that the TMR-PLL is less likely to lose lock than the single PLL, but when the TMR-PLL loses lock, it requires a PLL reset toggle to re-acquire lock.

The radiation test reports also show that during proton testing, no loss of lock events have been detected. This test result is typically applicable to Low Earth Orbit (LEO) or sub-orbital space flight applications.

Depending on the user design requirements, the TMR-PLL upset rates might be sufficiently low such that it does not pose a concern to the design operation. However, as a precaution, Libero SoC v11.9 and later, includes updates that requires the usage of the updated RTG4 Fabric CCC configurator v1.1.226. This CCC configurator version will insert auto-reset logic for any CCC configured to use the PLL Internal feedback mode. Figure 15, page 22 shows a graphical representation of the auto-reset logic inserted around an RTG4 fabric CCC instance configured to use the TMR-PLL in Libero SoC v11.9. Starting with

Libero SoC v11.9 SP4, the auto-reset logic was updated slightly so that the user's FABRIC_PLL_ARST_N input assertion does not bypass the 1 us delay counter, as shown in Figure 16.

Figure 15 • **Auto-Reset Logic Built-In to RTG4 CCC with TMR PLL in Libero SoC v11.9**

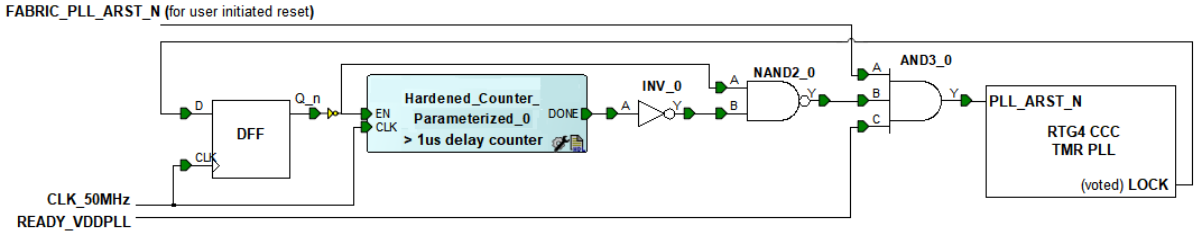
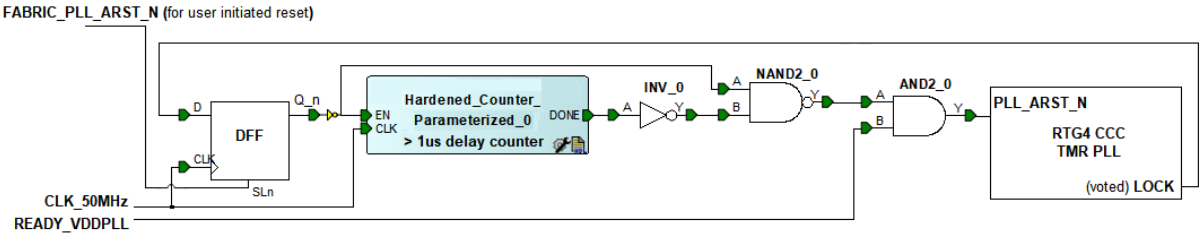


Figure 16 • **Auto-Reset Logic for RTG4 CCC in Libero SoC v11.9 SP4 Onwards**



The auto-reset logic monitors the voted lock output from the TMR-PLL and if it transitions LOW during normal operation, it asserts the PLL_ARST_N for at least 1 us, followed by reset release to allow the TMR-PLL to re-acquire lock onto the CCC input reference clock. The circuit requires a free running clock input of 50 MHz or less. With a 50 MHz clock input, the hardened delay counter asserts the PLL_ARST_N for ~1.2 us to satisfy the 1 us PLL_ARST_N minimum pulse width requirement. The RTG4 internal 50 MHz RC oscillator can be used to provide this clock input if the user does not have an external free running clock source running at 50 MHz or less. A slower clock input increases the time the TMR PLL is held in reset during an auto-recovery event.

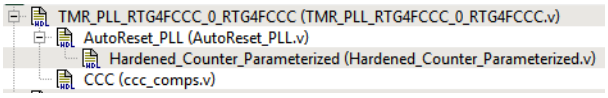
The preceding circuit also includes an extra reset input to allow user initiated dynamic reset signals coming from the FPGA fabric to reach the PLL_ARST_N input. Usage of the fabric reset input must meet the PLL_ARST_N minimum pulse width requirement listed in the *DS0131: RTG4 FPGA Datasheet*.

The READY_VDDPLL input signal allows board designs—which cannot meet the RTG4 power-up requirement—to ramp-up VDDPLL to the minimum recommended voltage level before the last power supply starts ramping up. It can be used to keep the PLL in reset until the VDDPLL supply reaches nominal conditions. If the board design meets the *DS0131: RTG4 FPGA Datasheet* recommendation by not allowing VDDPLL to be the last supply to ramp-up, then this input can be tied HIGH. Otherwise, see the *UG0586: RTG4 FPGA Clocking Resources User Guide* for information about circuits that can be used to drive the READY_VDDPLL input based on the VDDPLL expected ramp time or monitored voltage level.

Note that the internal RC oscillator output cannot directly drive the FPGA fabric, and must be routed onto a global through a CCC output. If there are spare outputs on the CCC implementing the TMR PLL, then the same CCC can be used with 1 GLx output sourcing its reference clock from the RC oscillator. The GLx output can then drive the CLK_50MHz input of the same CCC, and any others CCCs in the design, which require the 50 MHz RC oscillator clock. To minimize the effect of fabric switching noise on the RC oscillator output clock jitter, it is recommended to connect it to a CCC that is physically close to the RC Oscillator on the device die.

Designs containing a CCC using the PLL in triple redundant mode are shown in the design hierarchy with additional HDL source files added to implement the auto-reset feature. The following figure shows the HDL files.

Figure 17 • **HDL Files Generated with TMR PLL**



In some applications, an unpredictable PLL reset due to a radiation induced loss of lock event cannot be tolerated, even if the chance of the event occurring within the design's planned operating lifetime is very low. Therefore, the user can augment the built-in auto-recovery option by monitoring the three single PLL lock outputs and scheduling a TMR PLL reset event when only one of the three single PLLs loses lock. With this scheme, the detection of one PLL losing lock can be treated as an interrupt/error flag to the application, allowing time to complete any critical processing before the user design asserts the PLL_ARST_N input to the TMR PLL. The individual single PLL lock outputs are available in the CCC macro and the user can pull those to the top-level of the CCC instance in the design. The following figure highlights the voted lock and the individual single PLL lock outputs.

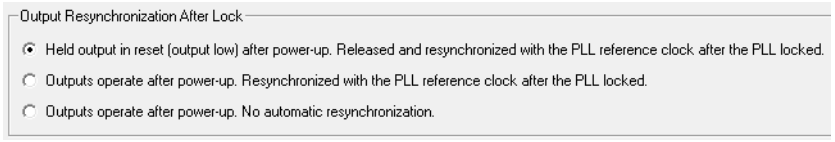
Figure 18 • **Modified CCC HDL Wrapper Exposing Single PLL Lock Outputs**

```
module TMR_PLL_RTG4FCCC_0_RTG4FCCC(
        GL0_Y0_ARST_N,
        GL0_Y0_EN,
        VOTED_LOCK,
        LOCK_0,
        LOCK_1,
        LOCK_2,
        CLK0_PAD,
        GL0,
        PLL_ARST_N,
        READY_VDDPLL,
        CLK_50MHZ,
        PLL_POWERDOWN_N
    );
input  GL0_Y0_ARST_N;
input  GL0_Y0_EN;
output VOTED_LOCK, LOCK_0, LOCK_1, LOCK_2;
input  CLK0_PAD;
output GL0;
input  PLL_ARST_N;
input  READY_VDDPLL;
input  CLK_50MHZ;
input  PLL_POWERDOWN_N;

    wire gnd_net, vcc_net, CLK0_PAD_net, GL0_net, pll_reset_net;

    AutoReset_PLL AutoReset_PLL_0 (.CLK_50MHZ(CLK_50MHZ), .PLL_ARST_N(
        PLL_ARST_N), .READY_VDDPLL(READY_VDDPLL), .VOTED_LOCK(LOCK),
        .PLL_RESET(pll_reset_net));
    VCC vcc_inst (.Y(vcc_net));
    GND gnd_inst (.Y(gnd_net));
    CLKINT GL0_INST (.A(GL0_net), .Y(GL0));
    CCC #( .INIT(175'h08810124C90000AB80404041664C991D86182011C940), .VCOFREQUENCY(660.000)
        , .CORE_VERSION("1.1.226") )  CCC_INST (.BUSY(), .APB_S_PREADY(
        ), .APB_S_PSLVERR(), .Y0(), .Y1(), .Y2(), .Y3(), .LOCK(VOTED_LOCK),
        .PLL_LOCK({LOCK_0, LOCK_1, LOCK_2}), .APB_S_PSEL(gnd_net), .CLK0(
        vcc_net), .CLK1(vcc_net), .CLK2(vcc_net), .CLK3(vcc_net),
        .GL0_Y0_EN(GL0_Y0_EN), .GL1_Y1_EN(vcc_net), .GL2_Y2_EN(vcc_net),
        .GL3_Y3_EN(vcc_net), .PLL_BYPASS_N(vcc_net), .PLL_POWERDOWN_N(
        PLL_POWERDOWN_N), .PLL_ARST_N(pll_reset_net), .GPD0_ARST_N(
        vcc_net), .GPD1_ARST_N(vcc_net), .GPD2_ARST_N(vcc_net),
        .GPD3_ARST_N(vcc_net), .GL0_Y0_ARST_N(GL0_Y0_ARST_N), .GL1_Y1_ARST_N(
        vcc_net), .GL2_Y2_ARST_N(vcc_net), .GL3_Y3_ARST_N(vcc_net),
        .CLK0_PAD(CLK0_PAD_net), .CLK1_PAD(gnd_net), .CLK2_PAD(gnd_net)
        , .CLK3_PAD(gnd_net), .RCOSC_50MHZ(gnd_net), .GL0(GL0_net),
        .GL1(), .GL2(), .GL3(), .DEL_CLK_REF(), .APB_S_PRDATA({nc3,
        nc4, nc5, nc6, nc7, nc8, nc9, nc10}), .APB_S_PWDATA({vcc_net,
        vcc_net, vcc_net, vcc_net, vcc_net, vcc_net, vcc_net, vcc_net})
        , .APB_S_PADDR({vcc_net, vcc_net, vcc_net, vcc_net, vcc_net,
        vcc_net}), .APB_S_PCLK(vcc_net), .APB_S_PWRITE(vcc_net),
        .APB_S_PENABLE(vcc_net), .APB_S_PRESET_N(gnd_net));
    INBUF CLK0_PAD_INST (.PAD(CLK0_PAD), .Y(CLK0_PAD_net));

endmodule
```

The user can configure the PLL outputs to remain low until the PLL is locked using the PLL Options tab of the CCC configurator, as shown in the following figure. This ensures that the CCC clock outputs derived from the PLL output remains low until the PLL locks. This might be useful for an application where the PLL resets due to a loss of lock event. When the PLL feedback mode is set to PLL Internal to enable the

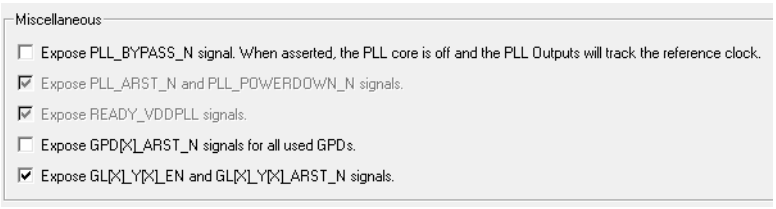
TMR PLL, the CCC outputs derived from the PLL resynchronizes with each other, but not with the input reference clock due to the difference in feedback clock path between PLL Internal feedback versus CCC Internal feedback.

Figure 19 • **Configuring CCC/PLL Output to Remain LOW Until Locked**



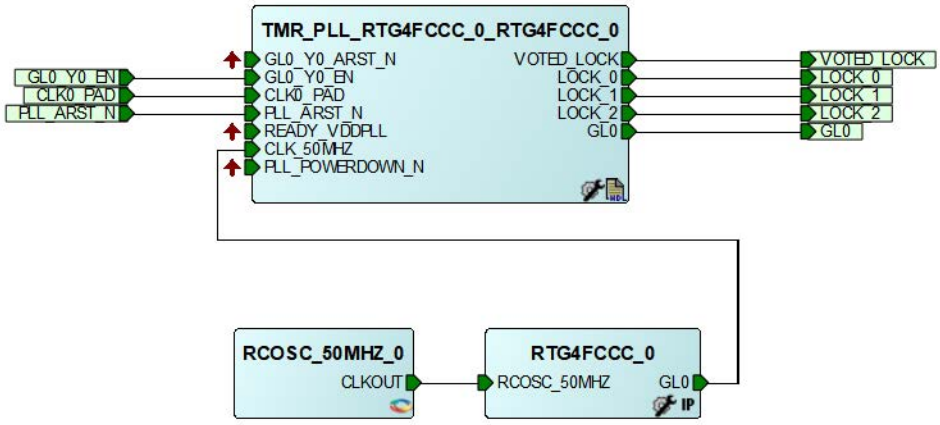
As an alternative to setting the PLL options to hold the clock output low until the PLL locks, the user can employ the GLx_Yx_EN output enable signal to gate-off the corresponding CCC GLx global clock output to ensure that the clock remains low until the PLL resets and all three lock outputs are re-acquired. Since the PLL_ARST_N input is an asynchronous reset to the PLL, the GLx_Yx_EN can bring the clock outputs low in a controlled manner, once the application is ready to asynchronously reset the PLL due to the de-assertion of one of the individual single PLL lock outputs. The GLx output enable signals can be exposed in the PLL Options tab of the CCC configurator as shown in the following figure.

Figure 20 • **Exposing GL[X]_EN Input through RTG4 CCC Configurator PLL Options Tab**



In the following figure, a separate CCC is used to source the 50 MHz clock from the internal RC oscillator. However, if the original CCC implementing the TMR PLL is physically close to the RC oscillator, and has a spare output, then a single CCC can be used with the corresponding GLx output for the 50 MHz wrapped around to the CLK_50MHz input.

Figure 21 • **Modified CCC Wrapper Exposing Individual PLL Lock Signals**



Furthermore, application specific logic can be connected to the modified CCC module, which performs the following:

- Monitor LOCK_0, LOCK_1, and LOCK_2 outputs
- Assert a system interrupt or error flag if any one single PLL loses lock
- Wait for the system to reach an operation point where it can tolerate a PLL reset
  - This step might include putting the design into a safe state
- Synchronously de-assert all the GLx_Yx_EN inputs for all PLL derived CCC outputs
- Assert the PLL_ARST_N input for at least 1us before releasing the PLL from reset
- Wait for VOTED_LOCK output to assert
- Assert all GLx_Yx_EN inputs for all PLL derived CCC outputs to continue with normal design operation

The user logic used to monitor a given PLL should be clocked from a source which is not derived from the PLL being monitored.

2.4 Global Resource Assignment during Libero SoC Automated Tool Flow

The allocation strategy for global networks during the Libero SoC tool flow uses the following priority order:

1. User instantiated global macros
   - CLKINT, CLKINT_PRESERVE, RCLKINT
   - CLKBUF, CLKBUF_DIFF, CLKBIBUF
   - GRESET, RGRESET
2. Promotion of very high fanout nets via CLKINT is weighed against the Synthesis and Compile threshold values
   - Minimum Number of Clock Pins driven by the high fanout net (default is 2)
   - Minimum Number of Asynchronous Pins driven by the high fanout net (default is 12)
   - Minimum Number of Data Pins driven by the high fanout net (default is 5000)
   - Buffer Nets with fanout which exceed the max fanout guide (default is 10,000)
   - Automatic Promotion of high fanout can be prevented in Synthesis using the syn_noclockbuf directive on a specific net
3. Clock nets are put on the low-skew global networks to minimize clock skew and hold-time violations.
   - After Synthesis, Libero SoC runs a Compile process, which can demote some chip-level global nets to row-global nets, if they meet the criteria set for demotion thresholds. This helps to better utilize the chip-level global resources (GBs, GBRs, or GBLs), while creating localized row-globals using rows of RGBs. This includes the replacement of CLKINT macros with RCLKINT macros. Similarly, non-clock, data nets can be demoted to regular routing if their fanout is below the default threshold of 5000.
4. Asynchronous reset/set nets are assigned to GRESET or RGRESET resources:
   - If there are no manually instantiated GRESET macros in the design, the highest fanout asynchronous reset/set net is assigned to the GRESET resource.
   - The remaining async reset/set nets driven by input buffers using hardwired connections to CLKINT, such as GB## or CLKBUF macros, will have the CLKINT replaced with a hardwired connection from GB to RGRESET macro.
   - Local async reset/set nets driven by fabric routing will be automatically assigned to an RGRESET macro, using only one input of the triplicated inputs. A warning will be generated in the Global Net Report during Layout. It is recommended that the designer triplicate the logic cone, see Resets, page 13.

Once the layout is complete, the user is strongly encouraged to review the Global Nets report to understand how the design clocks and resets have been assigned to device resources.

3 Mitigating Radiation Effects for RTG4 SerDes and FDDR Blocks

RTG4 devices contain hard logic blocks, implementing specific functions, such as RAM and DSP blocks, and specific interfaces, such as DDR Memory and High-Speed SerDes transceivers. These blocks have been enhanced with radiation mitigation techniques where possible. In some cases, radiation mitigation techniques were not added inside the hard block to preserve the block level timing performance. In those cases, application-level mitigation techniques are required to mitigate and handle scenarios such as Single-Event Functional Interrupts (SEFIs). This chapter discusses the high-level considerations required when using the SerDes and FDDR blocks in a radiation environment.

The RTG4 DDR Memory Controller blocks, called FDDR East and West, are described in the *UG0573: RTG4 DDR Memory Controller User Guide*.

The RTG4 SerDes blocks are described in the *UG0567: RTG4 High-Speed Serial Interfaces User Guide*. This document describes the SerDes in all applicable modes, including PCIe, XAUI, and External PCS (EPCS) modes.

The user guides linked above combined with Table 1, page 8 of this application note and the *RTG4 Radiation Tolerant Feature Summary* table provide details about the radiation tolerance mitigations that have been built-in to the RTG4 FDDR and SerDes blocks and their measured upset rates.

The following sections describe, at a high level, the types of application-level techniques that designers can employ to further mitigate radiation effects on the RTG4 FDDR and SerDes blocks. This section is included in the RTG4 Radiation Mitigated Clock and Reset Network Usage application note because the selection and design of clock and resets for the FDDR and SerDes blocks can affect how well these interfaces perform in a radiation environment.

The user should evaluate the documented radiation performance of the FDDR and SerDes blocks against their application-specific operating requirements to decide what additional design-level mitigations should be applied and hardware verified, potentially via radiation beam testing.

3.1 FDDR

Typical RTG4 designs that employ the FDDR blocks will use the FDDR IP catalog core version that includes the automatic initialization subsystem. In this case, the user can modify the design to monitor the FDDR subsystem status and take action in the event of radiation induced upset. It might require customization of the default initialization subsystem to expose additional FDDR Controller status signals and add user instruction sequences to the CoreABC program.

The RTG4 FDDR controllers offer several status indicators that the user design can monitor to detect radiation induced issues requiring intervention to restore functionality to the DDR memory subsystem. For example, there are interrupt signals that can be exposed to the user design that includes the following:

- PLL_LOCK_INT - Asserts when FPLL lock is achieved
- PLL_LOCKLOST_INT - Asserts when FPLL lock is lost
- ECC_INT - Asserts when ECC errors are detected
- IO_CALIB_INT - Asserts when DDR I/O calibration is complete
- FIC_INT - Asserts when there is a transaction error on the Fabric Interface Controller (FIC) port between the FPGA fabric and the FDDR controller
- APB_S_PSLVERR - Asserts when there is an error condition on an APB transfer to the fabric initiator

Similarly, there is an FPLL_LOCK output port from the RTG4FDDRC macro that can be monitored for radiation induced FPLL loss of lock events.

Furthermore, if the CLK_BASE and CLK_BASE_PLL_LOCK is sourced from a fabric CCC/PLL, a radiation induced loss of lock on the FCCC could also trigger interruption to the DDR_FIC_CLK, the FDDR_CLK, and FPLL_LOCK outputs. Thus, monitoring the FCCC outputs that provide the CLK_BASE

to the FDDR can detect a potential cause of an FDDR interruption and allow the design to take a pre-defined action.

When an error condition is detected, the user can decide on the course of action. For example: Toggling the INIT_RESET_N input to the FDDR with Initialization core will reset the APB configuration interface, reset and recalibrate the FPLL, cause the FDDR to enter soft reset (which is similar to asserting CORE_RESET_N), re-write the FDDR configuration registers, and subsequently perform external DRAM memory reset and initialization. This is a disruptive type of reset, but it ensures that the FDDR subsystem is restarted so that the functionality can be restored.

For some applications, the estimated orbital upset rates calculated in the *RTG4 FDDR controller Radiation Test Report* might be sufficiently low to tolerate an infrequent DDR subsystem reset. In other cases, a customer application might require more complex mitigation and recovery procedures that they can manually implement by modifying the default FDDR with Initialization core and subsequently testing via simulation and hardware validation, including radiation beam testing. Such modifications might include:

- Periodic scrubbing of the FDDR configuration registers that govern the FDDR controller and DRAM access timing to ensure a coherent configuration register set is maintained.
- Ensuring that the REG_DDRC_SELFREF_EN bit is always enabled so that when the fabric design stops issuing new commands/transactions, the DDR memory is placed in self-refresh mode.
- Upon a loss of CLK_BASE_PLL_LOCK or FPLL_LOCK, halt all transactions to the FDDR controller (so the DDR memory can enter self-refresh mode), wait for the FCCC lock to assert (if the FCCC has auto-reset enabled), and subsequently for the FPLL lock to assert. If the FPLL lock does not re-assert after CLK_BASE_PLL_LOCK returns, then re-run the subset of the CoreABC initialization code that resets and re-calibrates the FPLL. Once the FPLL lock re-asserts, resume transactions to the FDDR controller.

Note: The user should confirm that these high-level ideas allow a design to meet its application-specific operating requirements. It involves application-specific hardware validation testing, potentially including radiation beam testing.

3.2 SerDes

The most straightforward operational guidance for mitigating a SEFI on an active RTG4 SerDes link is to detect a SEFI and reset/re-initialize the lane/link, which includes resetting the RTG4 SerDes and its initialization subsystem (via the INIT_RESET_N input), then allowing the CoreABC initialization program to re-run. The RTG4 High-Speed Serial Interfaces User Guide and the RTG4 High Speed Serial Interface Configuration User Guides found on the RTG4 Documents website provide details on using the Libero SoC IP Catalog cores with automatic initialization as well as the manual construction of the initialization subsystem using CoreABC. Detecting a SEFI on an active link is a design-specific, protocol-specific task that the user must implement. The last-resort link reset reaction described above can be complemented by using a robust serial protocol coupled with continuous monitoring and recovery actions to minimize the number of times a lane/link must be re-initialized.

Systems using the SerDes in PCIe and XAUI mode also use the embedded SPLL to manage the clock domain data transfer between the FPGA Fabric AHB/AXI interface and the PCIe/XAUI PCS blocks in the SerDes. Monitoring the SPLL lock can be used as an error condition alerting the design to take action.

In general, detecting and dealing with a SerDes SEFI requires that the serial protocol in use includes protocol-level data integrity provisions (that is packet-level CRCs, link-state monitoring, redundant lanes, dynamic link width scaling, etc). For example, the SpaceFibre protocol aims to minimize radiation effects and provide reliable communication with quality of service capabilities. It includes error detection, isolation, and automatic error recovery from faults in the link, such as bit-flips, burst errors, and loss of lock.

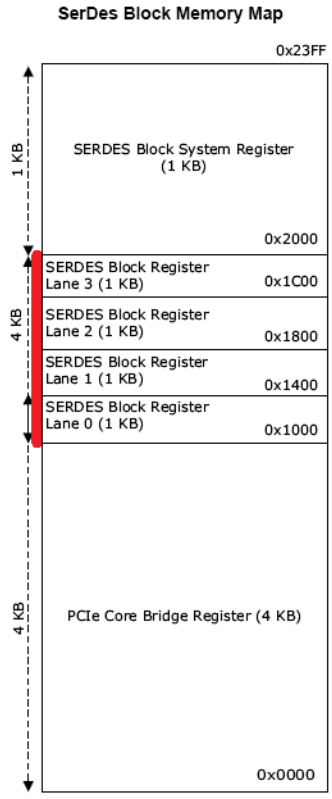
Once an issue is detected on a lane/link, the issue must be classified, and the appropriate response is taken:

- Corrupted frames can be automatically resent by the serial protocol and classified as a retry event.
- A burst of errors in a short period could be classified as a lane disconnection requiring the lane to be reinitialized.
- Similarly, loss of CDR lock or de-assertion of EPCS_#_RX_VAL could be classified as a lane disconnection requiring the lane to be reinitialized via a SERDES_LANE#_SOFTRESET request or EPCS_#_RESET_N (or CORE_RESET_N for XAUI/PCIe) assertion.
- A persistent failure of a SerDes lane that prevents the lane from linking even after a SerDes PMA reset is issued could be classified as a SEFI requiring that the lane's SerDes Block Registers be re-loaded with the correct PMA configuration before reinitializing the link.

The user design must be able to safely bring down the broken lane(s) within the link, re-initialize the affected SerDes lane(s), and re-acquire the link on the affected lane(s). Some protocols have a redundant lane to handle one lane going down. Some protocols can dynamically scale down the link width to preserve the connection while re-initializing a lane with an error.

The SerDes Block System Registers in RTG4 SerDes interfaces are SEU protected. However, the SerDes Block Registers within the PMA block for each lane are not SEU immune, as highlighted in red in Figure 22. Therefore, it is important to review the SerDes Lane Block Registers to understand when the registers can be written (that is some require the PHY to be in reset or the PMA to be ready), and which registers are shadow registers that are only shifted into the PMA when the UPDATE_SETTINGS register is programmed. During operation, maintaining a coherent set of register values for each lane's SerDes Block Registers is important to recover from a SEFI by putting the respective lane's PHY into reset, re-writing the lane's SerDes Block Register values, and writing a '1' to the lanes' UPDATE_SETTINGS register.

Figure 22 • **RTG4 SerDes Block Memory Map**



The user also needs to be cognizant of the recovered clock sharing schemes used for multi-lane RX links, such that the recovered clock from one lane does not impact other lanes. For example, resetting the lane whose RX_CLK is used for the RX FWF RCLK across multiple lanes will disrupt the operation of all dependent lanes.

Similarly, issuing a SerDes PMA update impacts a pair of lanes at a time. Selecting primary and redundant lanes requires keeping lane pair 0/1 isolated from lane pair 2/3 to prevent a redundant lane from being impacted if the primary lane's RX clock goes away or if the PMA for specific lanes must be re-initialized.

If a common reference clock is used across all lanes and between both devices in the link, a user might broadcast the SerDes REFCLK out of the SerDes GLOBAL_OUT_# to use that free-running REFCLK across all lanes TX and RX Fly Wheel FIFO (FWF) clocks and fabric PCS logic (for SerDes EPCS mode). It also helps to manage the utilization of the RTG4 RGBs required to connect to SerDes input clocks (which must belong to the fabric row of RGBs closest to the SerDes).

Another important consideration is the source of reset signals and clocks selected to control the SerDes initialization subsystem and the user's custom monitoring and recovery logic. Minimizing the number of upset sources on a particular interface could reduce the statistical likelihood of an upset. Review the RTG4 PLL and RC Oscillator radiation test reports to determine if the upset rates are acceptable for the specific project requirements. If not, an external clock source, such as the SerDes REF CLK, can be used and divided down to the required frequencies by using the RTG4 FCCC General Purpose Dividers (GPDs) with the PLL bypassed. For example, one of the required clocks is a 50 MHz or slower clock for the SerDes APB configuration interface and initialization subsystem using CoreABC. Another benefit of bypassing the PLL in the FCCC GPD scheme proposed above is that it avoids relying on the PLL lock output as a reset source that could inadvertently reset the SerDes block if radiation induced PLL upset occurs.

Again, the mitigation schemes employed in the user's RTG4 design depend on the application-specific reliability requirements and serial protocol's built-in capabilities for fault detection and recovery. The system behavior in a harsh environment might require beam-testing if existing radiation test reports cannot be leveraged.

In 2021, STAR-Dundee conducted a SpaceFibre and SpaceWire radiation test campaign on RTG4. Readers are advised to review the test report, when available, to understand the test design configuration, built-in mitigation, and resulting upset rates.

3.3 Conclusion

The RTG4 Radiation Tolerant FPGA includes built-in features to enable the FPGA designer to create high reliability designs for harsh environments. Understanding the clock and reset resources of the device, and which paths have built-in Single-Event Effect (SEE) mitigation, is critical to planning the pin assignment and global signal routing for a given design.