Implementing PCIe Reset Sequence in SmartFusion2 and IGLOO2 Devices - Libero SoC v11.6

Table of Contents

Superseded

Purpose

This application note describes how to implement the peripheral component interconnect express (PCIe) reset sequence for the SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) and IGLOO®2 FPGA devices using the CoreABC standalone peripheral initialization flow.

Introduction

The SmartFusion2 and IGLOO2 devices integrate a fourth-generation flash-based FPGA fabric and high-performance communication interfaces on a single chip. The high-speed serial interface (SERDESIF) provides a fully hardened PCIe endpoint implementation and is compliant to the PCIe base specification revision 2.0 and 1.1. For more information on SERDESIF, refer to the *UG0447: SmartFusion2 and IGLOO2 High Speed Serial Interfaces User Guide*.

The PCIe specification describes two reset generation mechanisms:

- **Fundamental reset:** Signaled through an auxiliary side-band signal PERSTn (PCIe reset, active low).
- **In-band reset:** Initiated by the host by setting a specific bit in the training sequence (hot-reset, link enabled or disabled).

PCIe reset causes end point device state machines, hardware logic, port states, and configuration registers (except for the sticky registers) to initialize the default conditions.

© 2015 Microsemi Corporation

During a host initiated PCIe reset process, SERDES PCIe endpoint reset must be generated in a proper sequence and the endpoint device must be reinitialized correctly. If the PCIe endpoint is not reset properly, this may cause corrupt data to be passed through the PCIe link.

Special Consideration for L2

This section describes the following scenarios important for a PCIe endpoint to reset:

- PCIe endpoint connects to the PERSTn
- PCIe endpoint does not connect to the PERSTn

When the PERSTn signal is connected to the endpoint, the rootport uses this signal for endpoint exit. The L2 state prior to the root port access of an endpoint must release itself from L2 without the PERSTn connection. The PERSTn signal is not carried on the long cable or fiber lengths when PCIe is used over distances. This self reset requirement and implementation is discussed in this application note.

This application note also describes the recommended reset sequence for the SmartFusion2 and IGLOO2 PCIe designs. The existing PCIe control plane demo reference design is modified and used to implement and describe the recommended PCIe reset sequence.

References

The following reference documents complement and help in understanding the relevant Microsemi® SmartFusion2 and IGLOO2 FPGA devices flows and features:

- *UG0447: SmartFusion2 and IGLOO2 High Speed Serial Interfaces User Guide*
- *SmartFusion2 Standalone Peripheral Initialization User Guide*
- *UG0456: SmartFusion2 SoC FPGA PCIe Control Plane Demo User Guide*
- *IGLOO2 Standalone Peripheral Initialization User Guide*
- *TU0509: Implementing PCIe Control Plane Design in IGLOO2 FPGA Tutorial*

PCIe Control Plane Demo Design Requirements

Table 1 lists the design requirements.

Table 1 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Security Evaluation Kit: • 12 V adapter (provided along with the kit) • FlashPro4 programmer (provided along with the kit)	Rev D or later
IGLOO2 Evaluation Kit: • 12 V adapter (provided along with the kit) • FlashPro4 programmer (provided along with the kit)	Rev D or later
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero® System-on-Chip (SoC)	v11.6
SoftConsole	v3.4 SP1
Host PC Drivers (provided along with the design files)	-
PCIe Demo application	-

Superseded

Design Description

The implementation of a PCIe reset sequence, which supports the host reset involves detection of PCIe reset using the FPGA fabric logic and generating the reset for endpoint block.

PCIe reset sequence is device dependent and based on whether PERSTn is used in the design. As a result, the reference designs are categorized into the following four use models:

- "M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices with PERSTn" on page 3
- "M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices without PERSTn" section on page 10
- "M2S060 / M2S090 / M2GL060 / M2GL090 Devices with PERSTn" section on page 13
- "M2S060 / M2S090 / M2GL060 / M2GL090 Devices without PERSTn" section on page 19
- "M2S/M2GL 060/090 Device Dual PCIe with PERSTn" on page 21
- "M2S/M2GL/060/090 Device Dual PCIe without PERSTn" on page 27

Note: The System Builder flow (Libero v11.6) does not implement the PCIe/SERDES reset sequence automatically as described in this application note. To implement the PCIe reset sequence, the SERDES standalone peripheral initialization methodology must be followed. Refer to the following documents, for more information about peripheral initialization methodology:

- *SmartFusion2 Standalone Peripheral Initialization User Guide*
- *IGLOO2 Standalone Peripheral Initialization User Guide*

Implementation

The implementation details for each of the above four scenarios are described as follows:

This application note uses M2GL010 Evaluation Kit and the M2S090 Security Evaluation Kit to implement the PCIe reset sequence. These two examples provide both the methods for implementing the PCIe reset sequence.

Note: This application note uses customized M2S090 Kit to implement the PCIe reset sequence for dual PCIe controller.

M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices with PERSTn

Figure 1 on page 4 shows the implementation of PCIe reset sequence for M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 devices with PERSTn. The PCIe reset detection logic detects the PCIe reset and resets the SERDES and CoreABC through the CoreResetP. CoreABC is used to initialize the SERDES (PCIe) through the APB interface. In addition, the APB_MUX logic is implemented to support access to the microcontroller subsystem (MSS) or high-performance memory subsystem (HPMS). By using the APB MUX logic, the user can retain access through SmartFusion2 M3 or utilizing the SmartDebug SERDES utility, which is available for both SmartFusion2 and IGLOO2 devices.

Note: The method to reset the PCIe controller resets the entire controller including the configuration space. This clears any of the "sticky" bits contained in these registers. These "sticky" bits are only present when using PCIe controller in Gen2. If any of these sticky bits are important for the application software on the host, the information must be gathered before the HotReset event is initiated.

Superseded

The PCIe reset detection logic and SERDES reset generation is explained as follows:

PCIe Reset Detection

Detect the entry of the PCIe endpoint to the HOT_RESET state by monitoring the LTSSM[4:0] bits and then call the signal as hot_reset_n_ltssm. The link training and status state machine (LTSSM) bits are available as CoreConfigP output signal PRDATA[30:26].

Generate hot_reset_n as the following: hot_reset_n = PERSTn & hot_reset_n_ltssm.

Connect the PERST_N of SERDES_INIT directly to PERSTn on board.

SERDES (PCIe) Reset Generation

Use the pulse shaping logic to generate the reset pulse from the hot_reset_n and then reset the SERDES core (PMA reset, PCIE controller reset and APB reset), release the SERDES resets and reinitialize the SERDES using CoreABC.

Figure 1 shows the M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 devices PCIe reset generation with PERSTn.

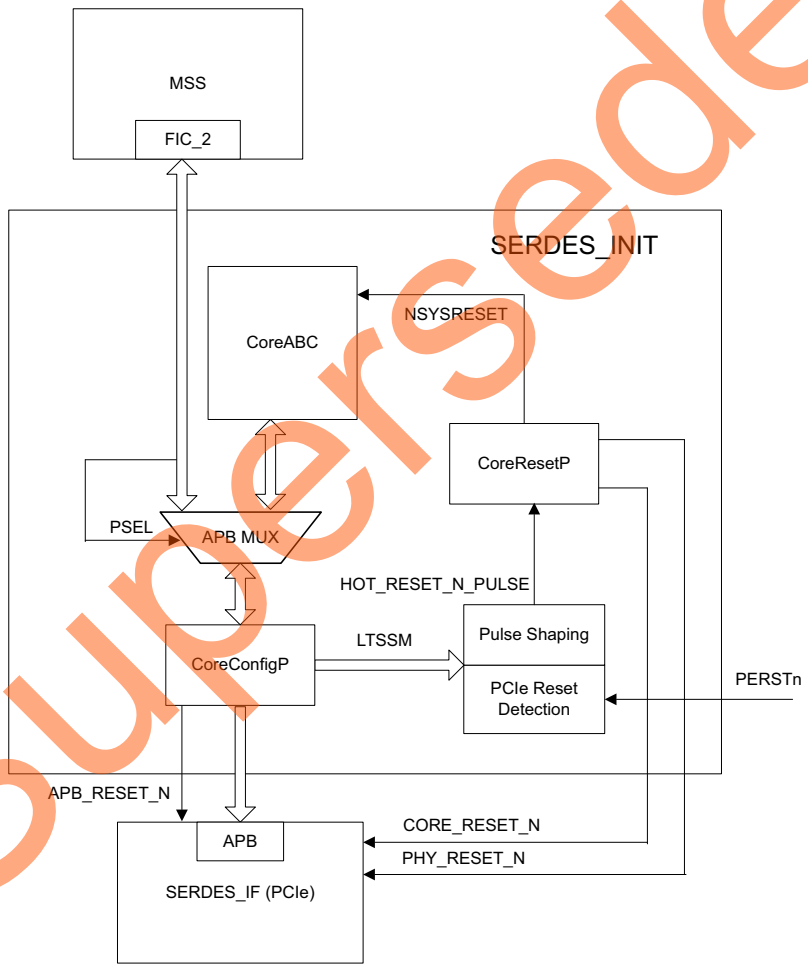


***Figure 1* • M2S/M2GL010/025/050/150 Device PCIe Reset Generation with PERSTn**

Superseded

Implementation using M2GL010 Evaluation Kit

The PCIe control plane demo design for the M2GL010 Evaluation Kit is created using the CoreABC standalone peripheral initialization method and the HOTRESET logic is implemented to detect the PCIe reset. The top level SmartDesign with the SERDES_INIT and SERDES_IF blocks, are shown in Figure 2. The SERDES_INIT is a SmartDesign block to initialize the SERDES and to generate the SERDES (PCIe) resets.

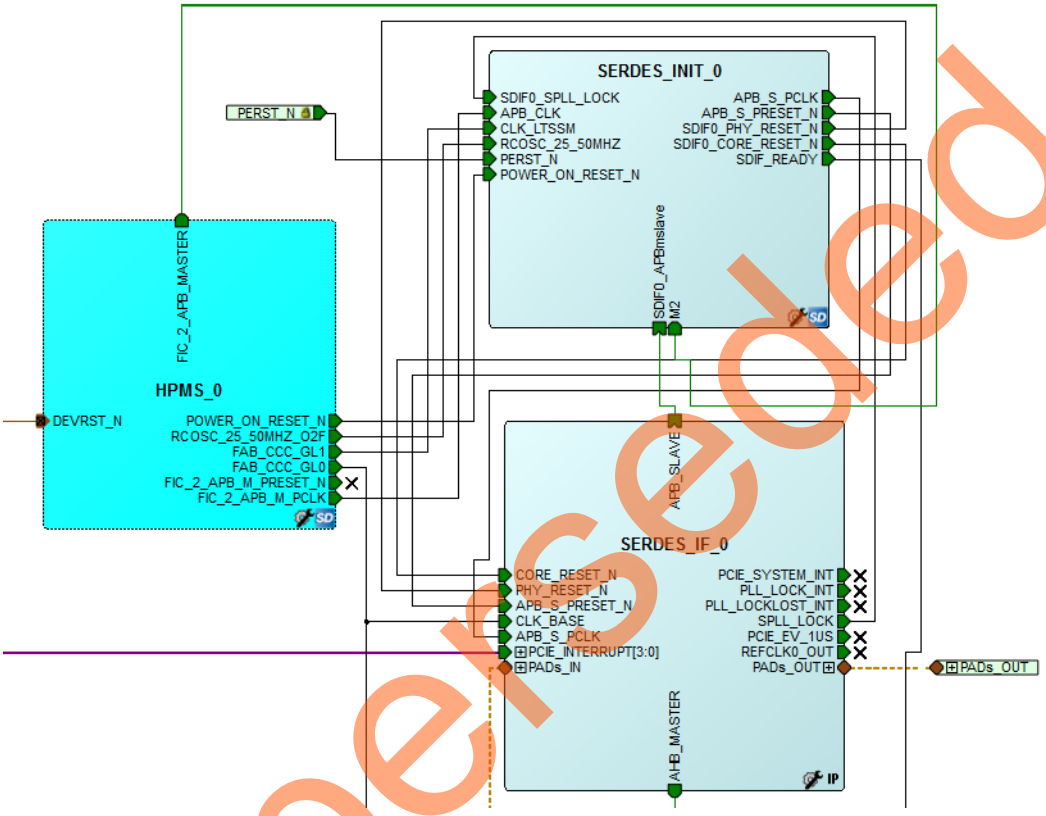


***Figure 2* • Top-Level Connection**

The SERDES_INIT has HOTRESET, CoreResetP, CoreConfigP, and CoreABC blocks, as shown in Figure 3 on page 6. The HOTRESET logic monitors the LTSSM (PRDATA [30:26]), PERSTn signals, and generates the HOT_RESET_N_PULSE. It is connected to the CoreResetP (FAB_RESET_N), which generates resets to SERDES (PCIe) and performs re-initialization of SERDES (PCIe) through CoreABC and CoreConfigP.

Superseded

Figure 3 shows the SERDES INIT SmartDesign.

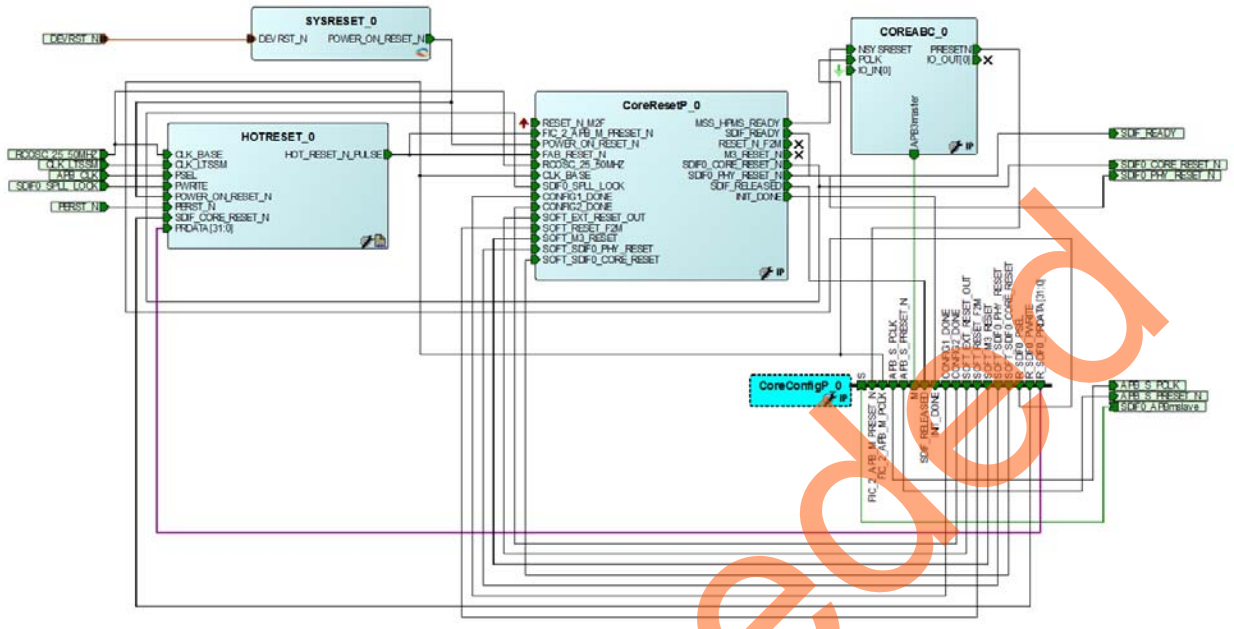


***Figure 3* • SERDES INIT SmartDesign**

Superseded

Figure 4 shows the CoreResetP configured for SERDES_IF_0 block.

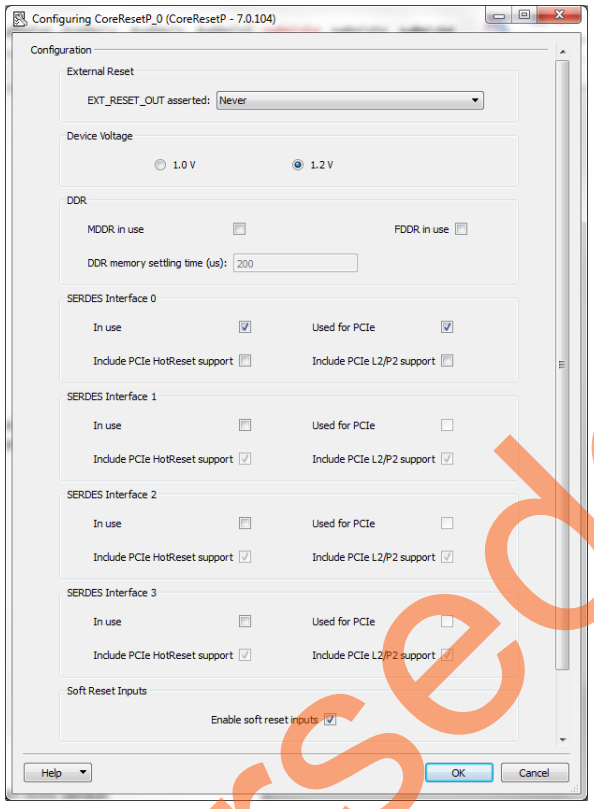


Figure 4 • **CoreResetP Configuration**

Figure 5 shows the CoreConfigP configured for SERDES_IF_0 block.

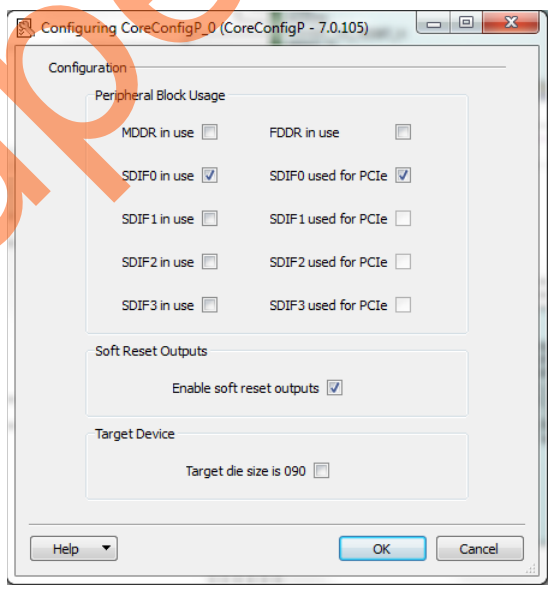


Figure 5 • **CoreConfigP Configuration**

Superseded

CoreABC is configured, as shown in Figure 6.

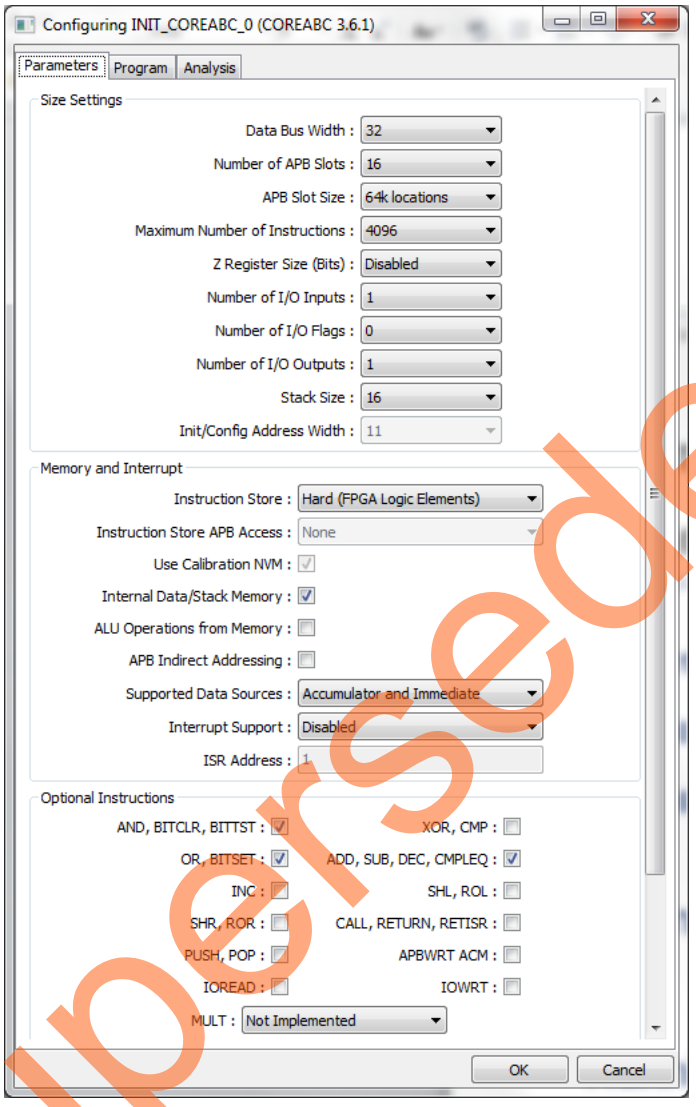


Figure 6 • **CoreABC Configuration**

Superseded

The Libero generates the `SERDESIF_0_init_abc.txt` file at **`<pjt_location>\PCIE_DEMO\component\work\PCIE_DEMO\SERDES_IF_0`** with CoreABC code to initialize the SERDES (PCIe). This code is used for CoreABC program as shown in Figure 7.

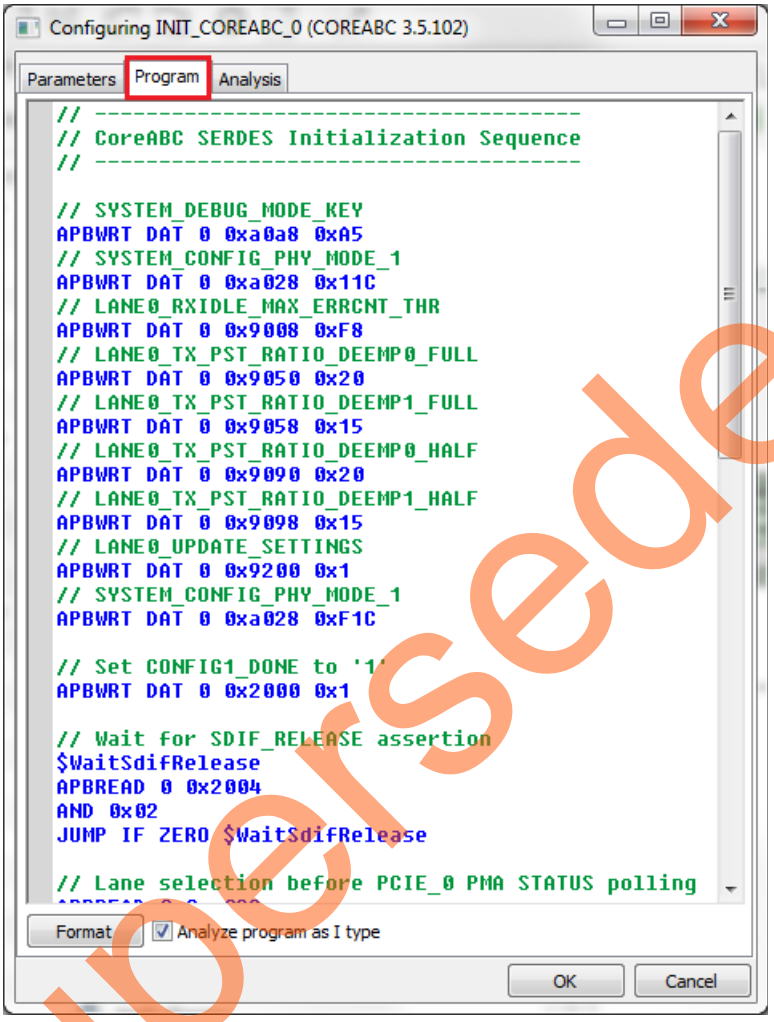


Figure 7 • **CoreABC Code**

Superseded

M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices without PERSTn

Figure 8 shows the implementation of PCIe reset sequence for M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 devices without PERSTn.

PCIe Reset Detection

Detect the entry of the PCIe endpoint to the HOT_RESET and L2 state by monitoring the LTSSM[4:0] bits and call the signals as hot_reset_n_ltssm and l2_detected_n. The LTSSM bits are available as APB signal PRDATA[30:26].

Generate hot_reset_n as the following: hot_reset_n = l2_detected_n & hot_reset_n_ltssm.

SERDES (PCIe) Reset Generation

Use pulse shaping logic to generate the reset pulse from the hot_reset_n and reset the SERDES core (PMA reset, PCIE controller reset,

and APB reset), release the SERDES resets and reinitialize the SERDES using CoreABC.

Note: The method to reset the PCIe controller resets the entire controller including the configuration space. This clears any of the "sticky" bits contained in these registers. These "sticky" bits are only present when using PCIe controller in Gen2. If any of these sticky bits are important for the application software on the host, the information should be gathered before the HotReset event is initiated.

Superseded

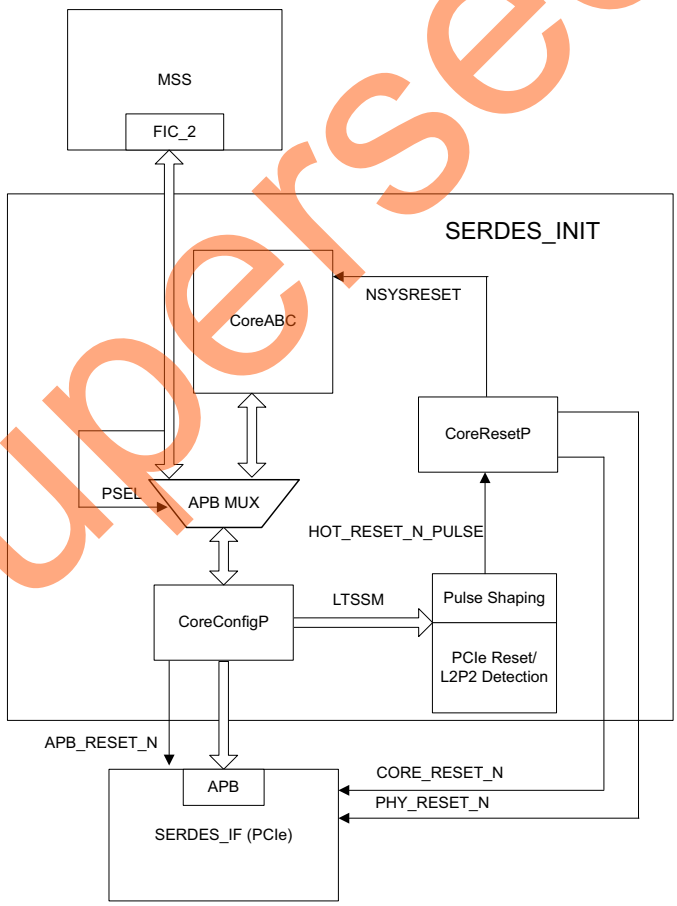


Figure 8 • **M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices PCIe Reset Generation without PERSTn**

Implementation using M2GL010 Evaluation Kit

The PCIe control plane demo design for M2GL010 Evaluation Kit is created using the CoreABC standalone peripheral initialization method and the HOTRESET logic is implemented to detect the PCIe reset. The top-level SmartDesign with the SERDES_INIT and SERDES_IF blocks is shown in Figure 9. The SERDES_INIT is implemented to detect the PCIe reset without monitoring the PERSTn and to generate the SERDES (PCIe) resets.

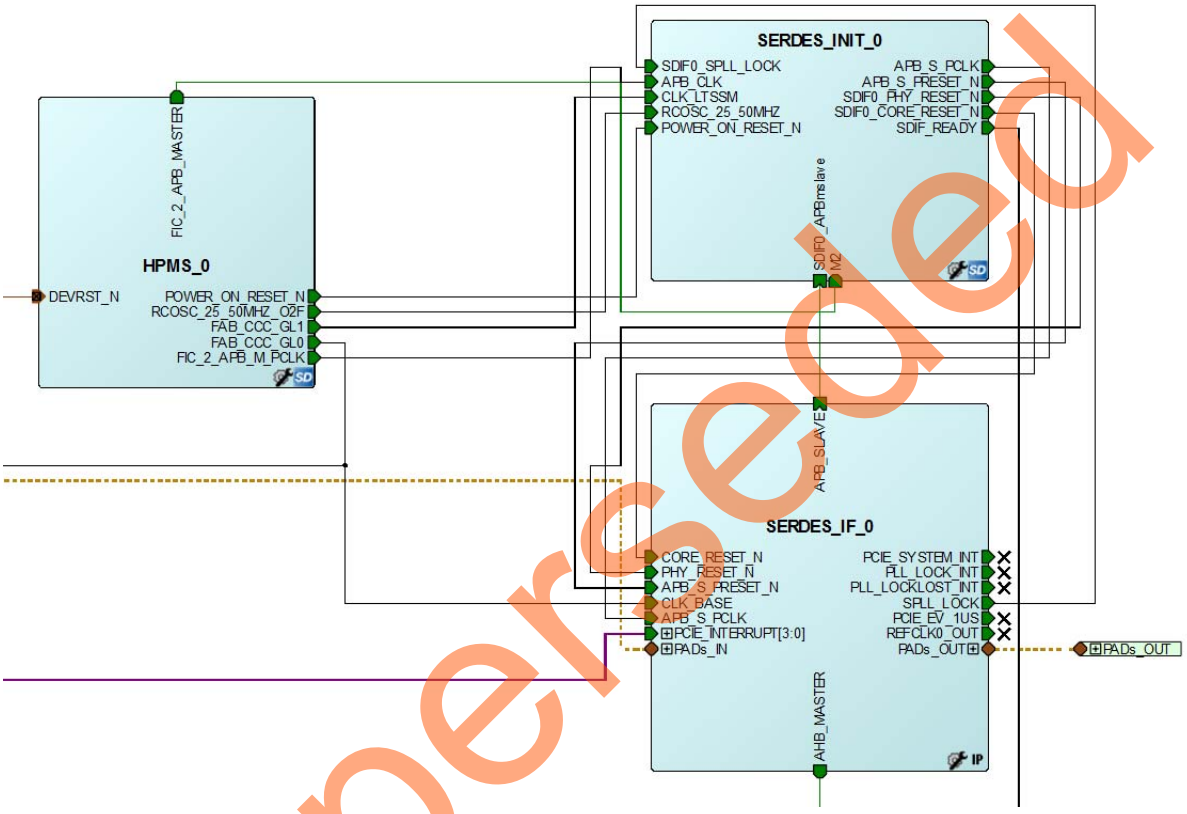


Figure 9 • **Top-Level Design**

The SERDES_INIT has CoreABC, CoreResetP, CoreConfigP, and HOTRESET blocks, as shown in Figure 10 on page 12. The HOTRESET logic monitors the LTSSM (PRDATA [30:26]) signals for hot reset state and L2 state and generates the HOT_RESET_N_PULSE signal. It is connected to the CoreResetP (FAB_RESET_N), which generates resets to SERDES (PCIe) and performs re-initialization of SERDES through CoreABC and CoreConfigP. The configuration of CoreResetP, CoreConfigP, and CoreABC is same as described in "M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices with PERSTn" section on page 3.

Superseded

Figure 10 shows the SERDES INIT SmartDesign.

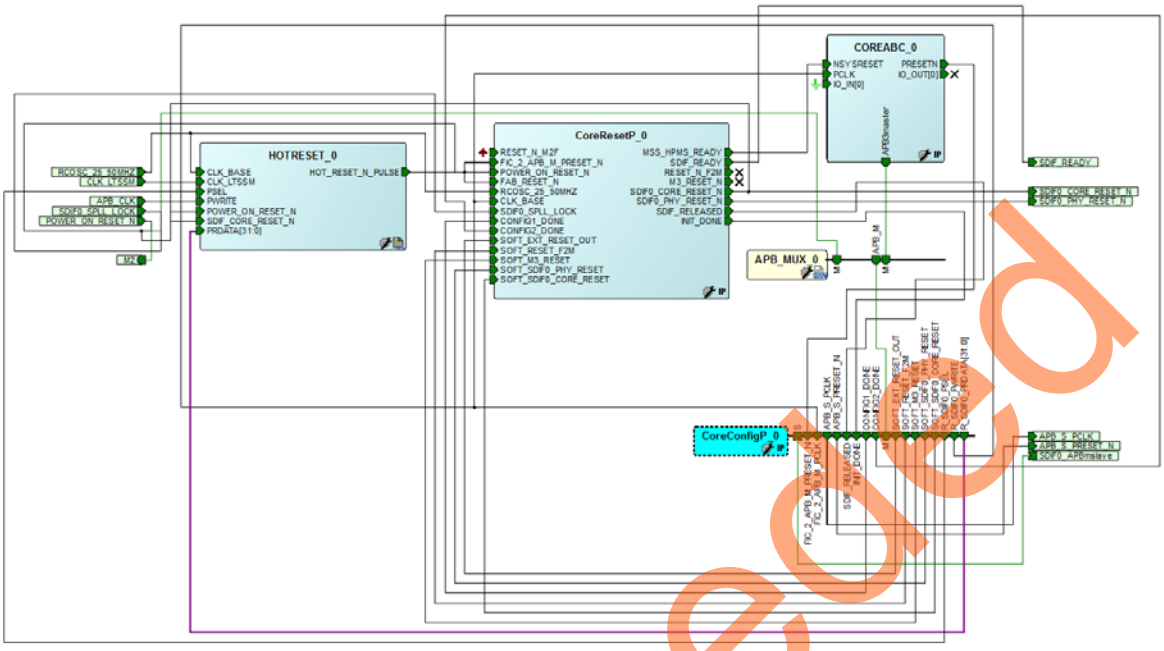


***Figure 10* • SERDES INIT SmartDesign**

Superseded

M2S060 / M2S090 / M2GL060 / M2GL090 Devices with PERSTn

Figure 11 shows the implementation of PCIe reset sequence for M2S060 / M2S090 / M2GL060 / M2GL090 devices with PERSTn. The PCIe reset detection logic detects the PCIe reset and triggers the CoreABC to reset the SERDES Core and AXI using the soft reset. In addition, the APB_MUX logic is implemented to support access to the MSS/HPMS. By using the APB MUX, the user can retain access through the SmartFusion2 M3 or utilizing the SmartDebug SERDES utility, which is available for both SmartFusion2 and IGLOO2 devices.

PCIe Reset Detection

The SERDES_IF2 (PCIe) has LTSSM[5] and L2P2_ACTIVE signals. The LTSSM[5] indicates hot-reset, data link up or L2 exit. The L2P2_ACTIVE indicates that PCIe is in L2 state.

Generate the hot_reset_n as the following: hot_reset_n = (! L2P2_ACTIVE) & (! LTSSM[5]).

Connect the SERDES PCIe PERST_N directly to PERSTn on board.

SERDES (PCIe) Reset Generation

Use pulse shaping logic to generate the pulse from the hot_reset_n and generate interrupt to CoreABC. The CoreABC interrupt routine resets the SERDES PCIE controller reset and AXI reset using the soft reset register.

Superseded

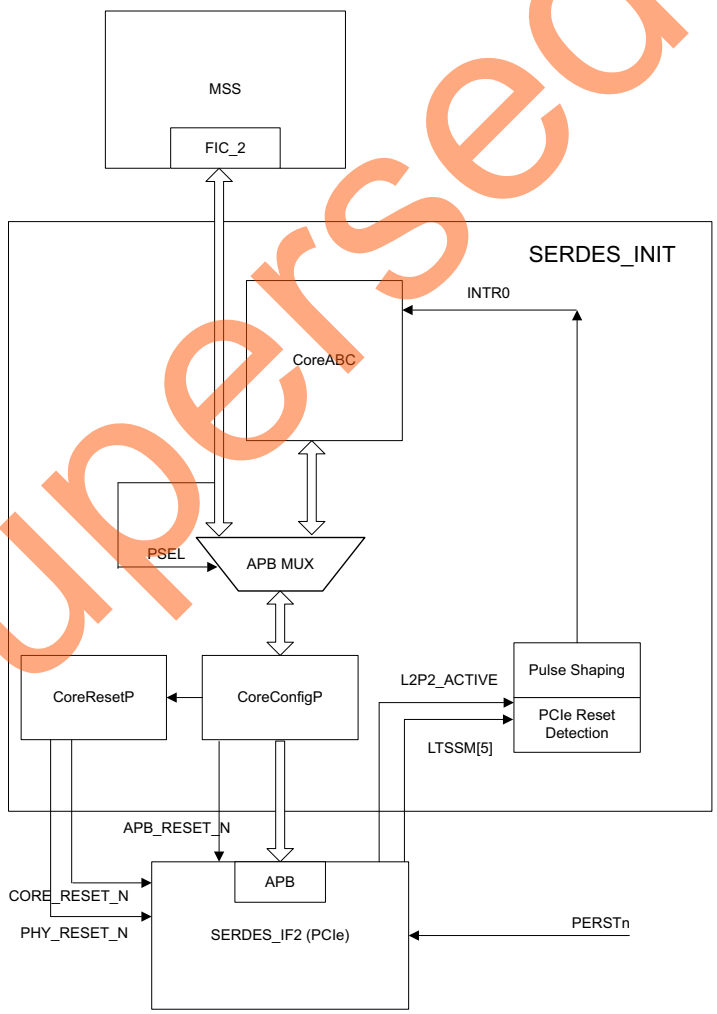


***Figure 11* • M2S060 / M2S090 / M2GL060 / M2GL090 Devices PCIe Reset Generation with PERSTn**

Implementation using M2S090 Security Evaluation Kit

The PCIe control plane demo design for the M2S090 Security Evaluation Kit is created using CoreABC standalone peripheral initialization method and the HOTRESET logic is implemented to detect the PCIe reset. The top-level SmartDesign with the SERDES_INIT and SERDES_IF blocks is shown in Figure 12. The SERDES_INIT is a SmartDesign block to initialize the SERDES and to generate the SERDES (PCIe) resets. The SERDES_IF_2 PERST_N signal is directly connected to the PERSTn on the board.

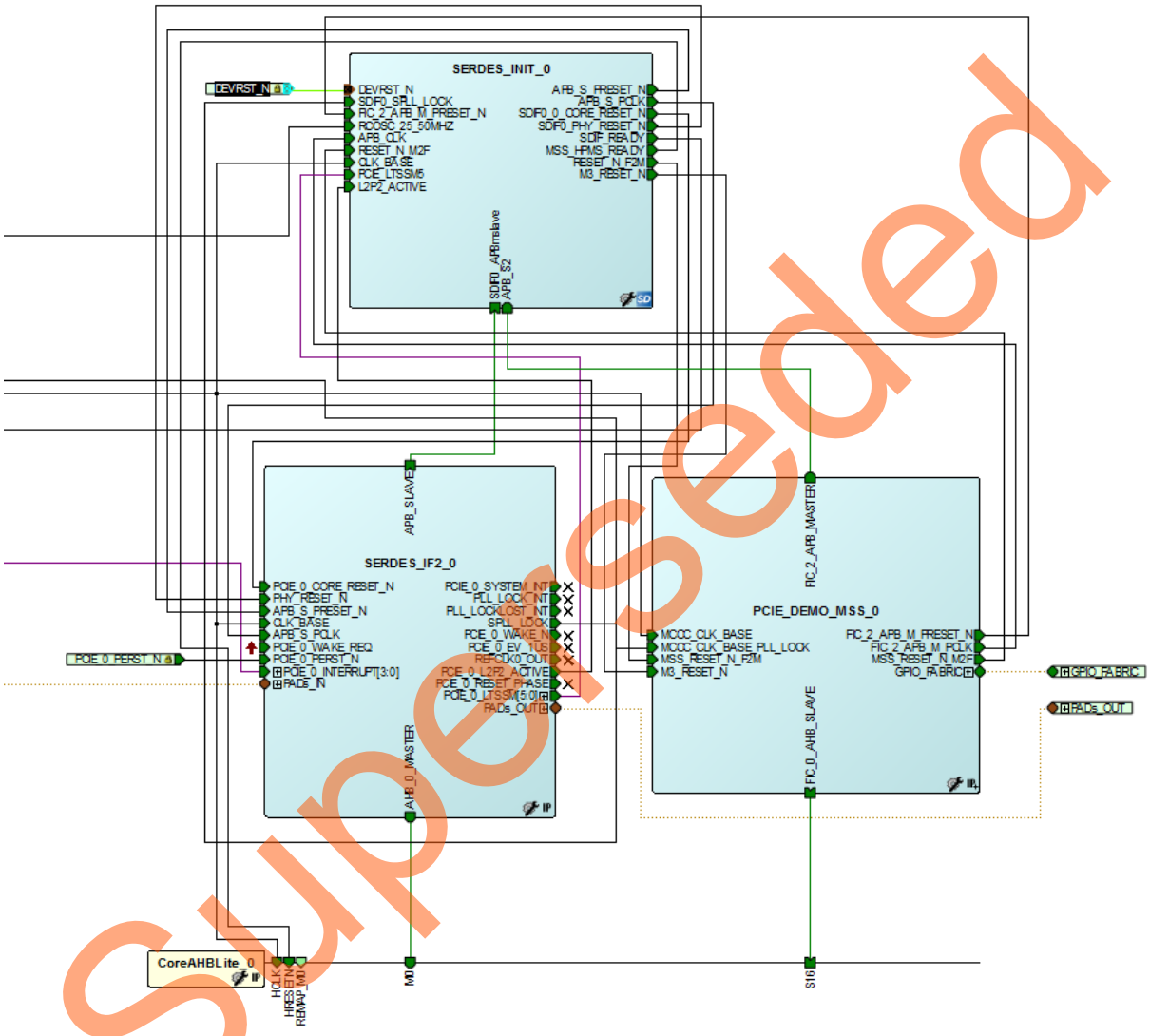


***Figure 12* • Top Level Design**

The SERDES_INIT SmartDesign has CoreABC, CoreResetP, CoreConfigP, APB_MUX, and HOTRESET blocks, as shown in Figure 13 on page 15. The HOTRESET logic monitors the LTSSM[5], L2P2_ACTIVE signals and generates the INTR0 signal. It is connected to the INTREQ of CoreABC and CoreABC interrupt routine issues SERDES (PCIe) Core and the AXI soft resets, when INTREQ goes low.

Superseded

Figure 13 shows the SERDES INIT SmartDesign.

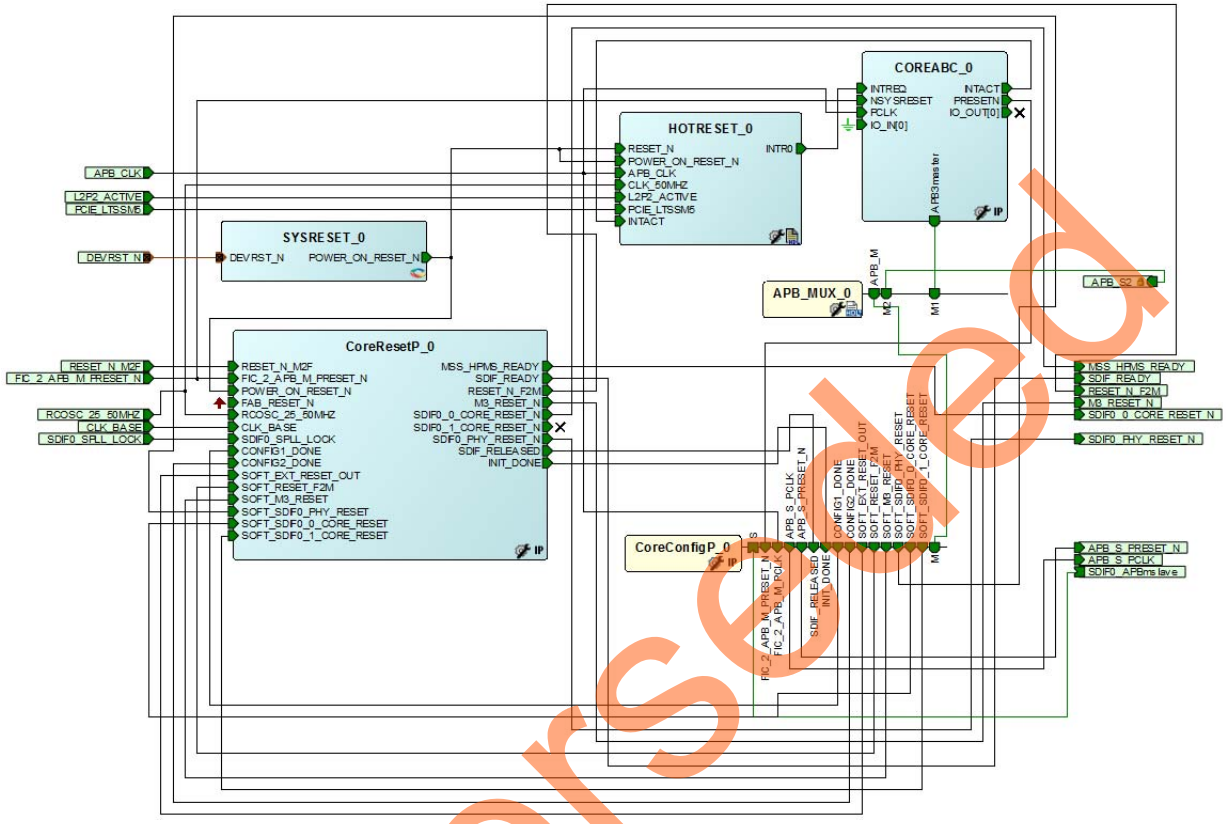


***Figure 13* • SERDES INIT SmartDesign**

Superseded

CoreResetP is configured to generate resets for SERDES_IF2_0, as shown in Figure 14.

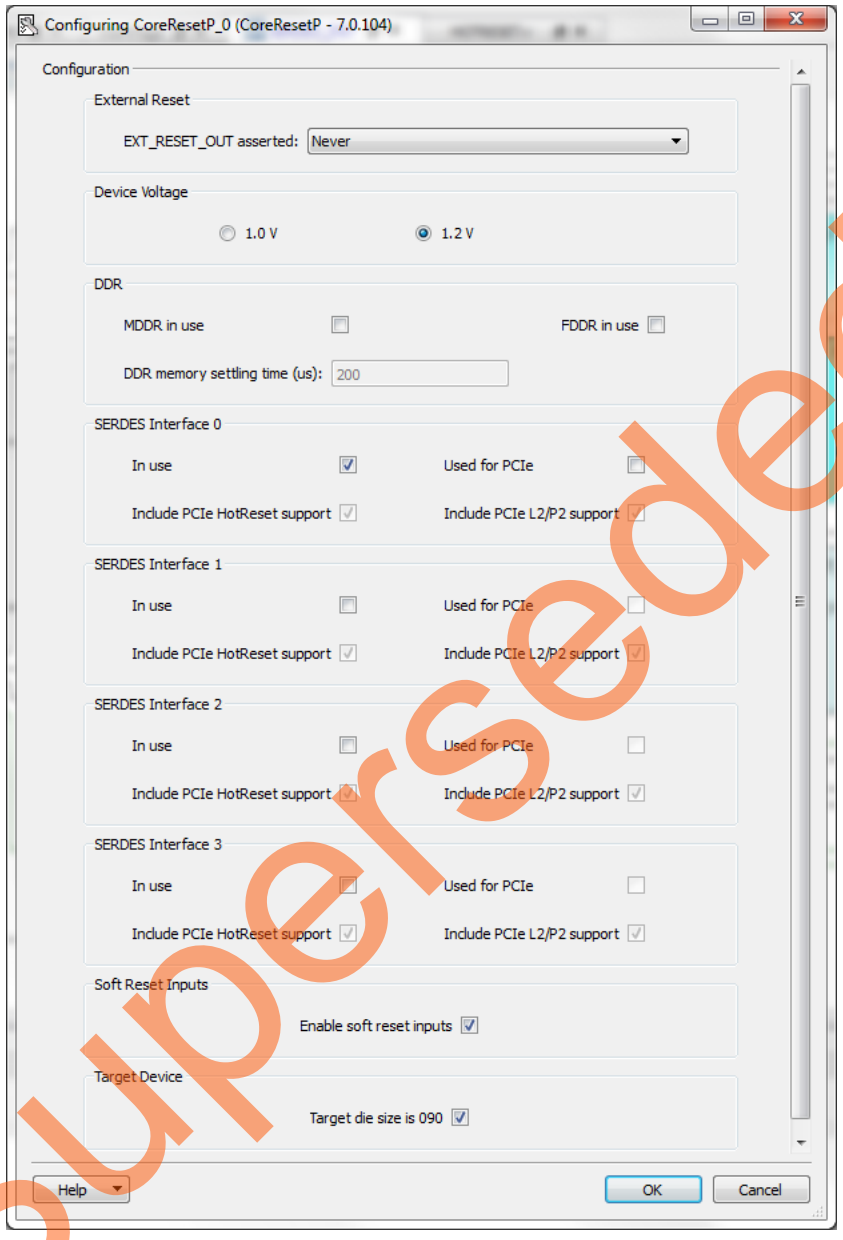


Figure 14 • **CoreResetP Configuration**

Superseded

CoreConfigP is configured for the SERDES_IF_0 block, as shown in Figure 15.

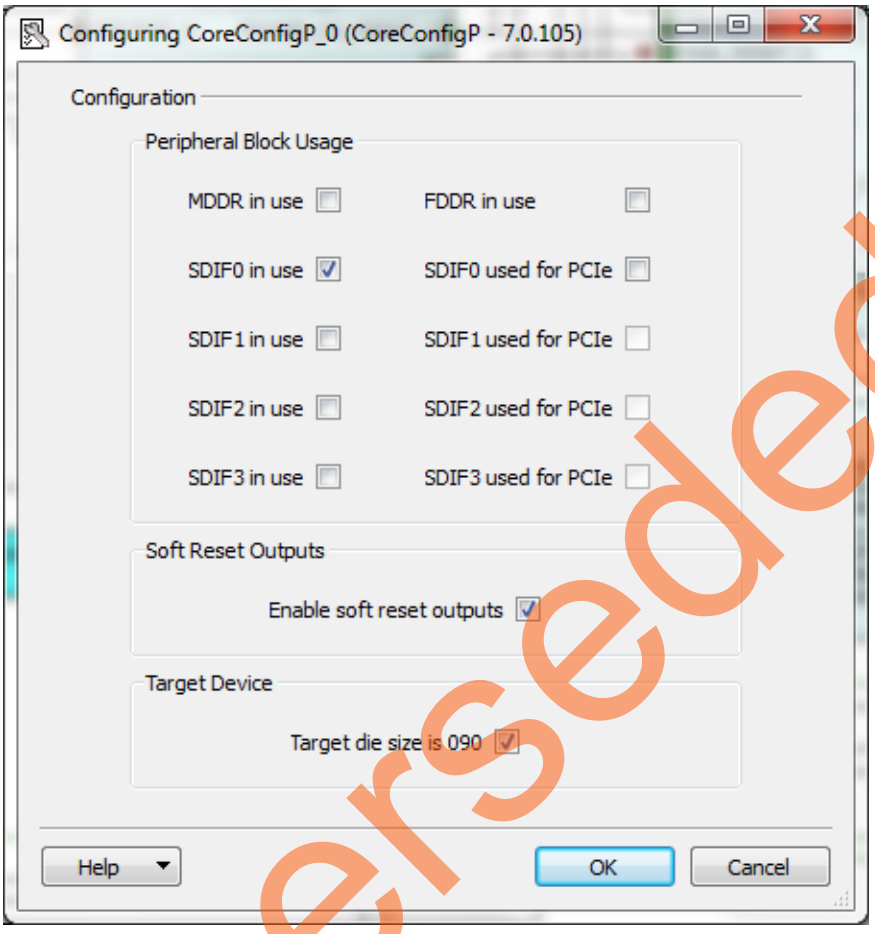


Figure 15 • **CoreConfigP Configuration**

The CoreABC configuration, is as shown in Figure 16 on page 18. Libero generates `SERDESIF_0_init_abc.txt` file at **`<pjt_location>\PCIE_DEMO\component\work\PCIE_DEMO\SERDES_IF2_0`** with CoreABC code to initialize the SERDES (PCIe). This code is modified to reset the SERDES core and AXI, using the SERDES soft reset register on the CoreABC active low interrupt.

Superseded

Figure 16 shows the CoreABC Configuration.

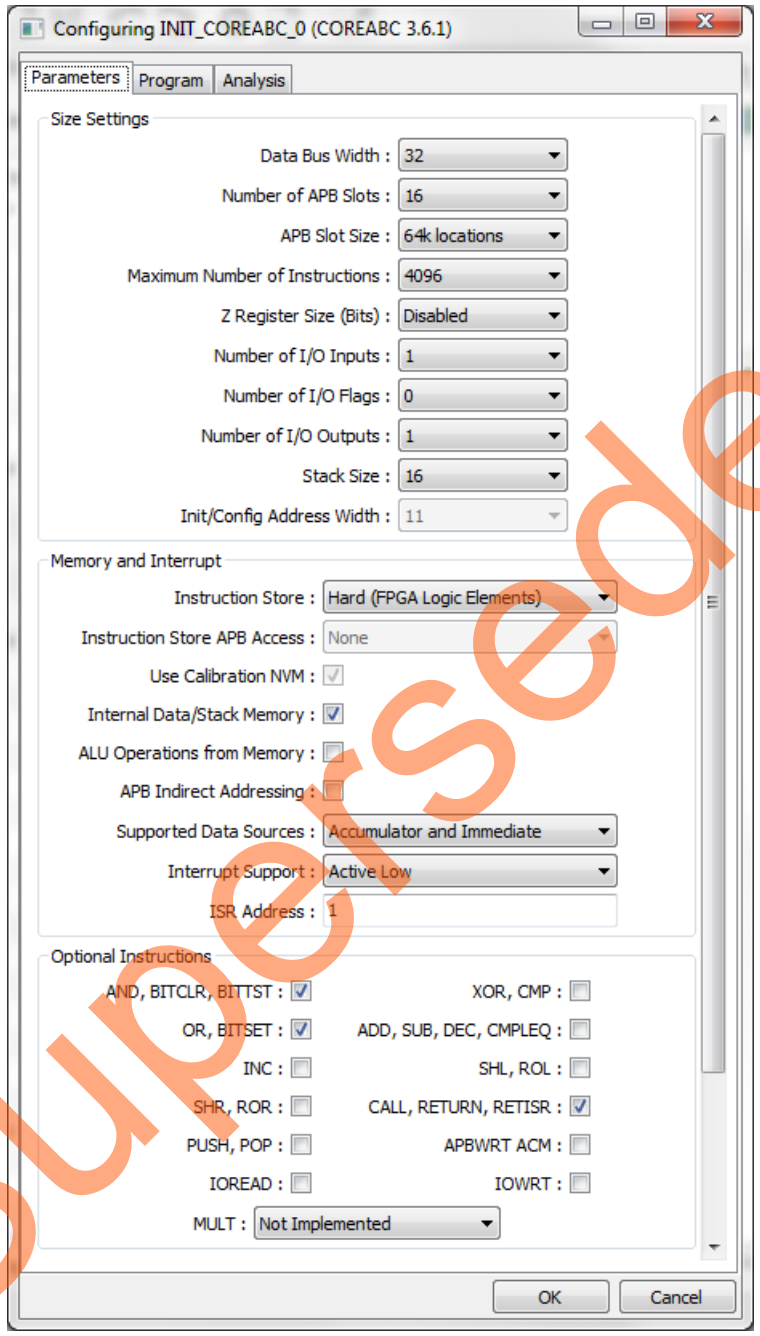


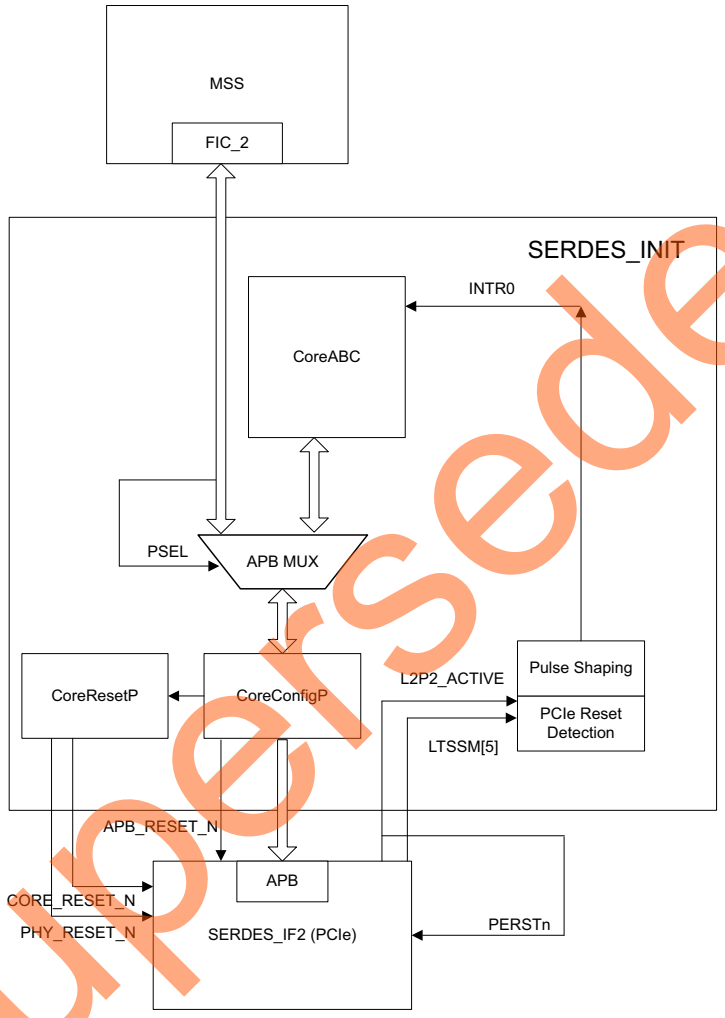
***Figure 16* • CoreABC Configuration**

Superseded

M2S060 / M2S090 / M2GL060 / M2GL090 Devices without PERSTn

Figure 17 shows the implementation of the PCIe reset sequence for M2S060 / M2S090 / M2GL060 / M2GL090 devices without PERSTn.

Note: The SERDES_IF_2 PERST_N signal is directly connected to the SERDES_IF_2 L2P2_ACTIVE signal instead of PERSTn.

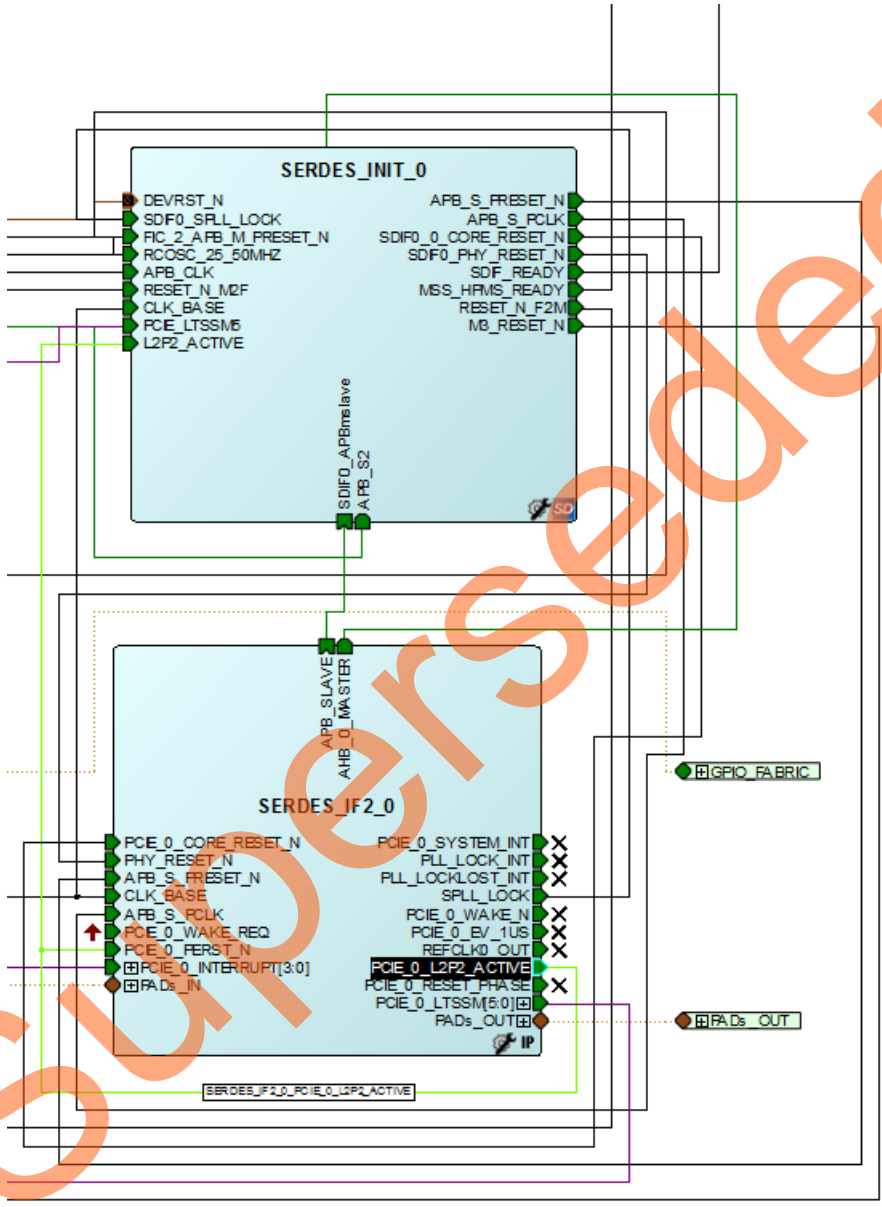


Superseded

***Figure 17* • M2S060 / M2S090 / M2GL060 / M2GL090 Devices PCIe Reset Generation without PERSTn**

Implementation using M2S090 Security Evaluation Kit

The PCIe control plane demo design for M2S090 Security Evaluation Kit is created using the CoreABC standalone peripheral initialization method and the HOTRESET logic is implemented to detect the PCIe reset. The top-level SmartDesign with the SERDES_INIT and SERDES_IF blocks is shown in Figure 18.



Superseded

***Figure 18* • Top-Level Design**

M2S/M2GL 060/090 Device Dual PCIe with PERSTn

Figure 19 shows the implementation of PCIe reset sequence for M2S/M2GL060/090 devices dual PCIe with PERSTn. The PCIe reset detection logic detects the PCIe reset and triggers the CoreABC to reset the SERDES PCIE controller0/controller1 and AXI interface using soft reset. In addition, the APB_MUX logic is implemented to support the SERDES APB register access from the MSS/HPMS as well. Using the APB MUX the user can retain access through SmartFusion2 M3 or utilizing the SmartDebug SERDES utility available for both SmartFusion2 and IGLOO2 devices.

PCIe Reset Detection

SERDES_IF2 (PCIe) has PCIE_0_LTSSM[5] and PCIE_0_L2P2_ACTIVE signals for PCIE_0 core and PCIE_1_LTSSM[5] and PCIE_1_L2P2_ACTIVE signals for PCIE_1 core. LTSSM[5] indicates hot-reset, data link up, or L2 exit. L2P2_ACTIVE indicates that PCIe is in L2 state.

Generate hot_reset_n as: hot_reset_n = (!PCIE_x_L2P2_ACTIVE) & (! PCIE_x_ LTSSM[5]).

Connect the SERDES PCIe PERST_N to PERSTn on the board.

SERDES (PCIe) Reset Generation

Use pulse shaping logic to generate pulse from hot_reset_n and generate interrupt to CoreABC. The CoreABC interrupt routine resets the SERDES PCIE controller0/controller1 reset and AXI interface reset using the soft reset register.

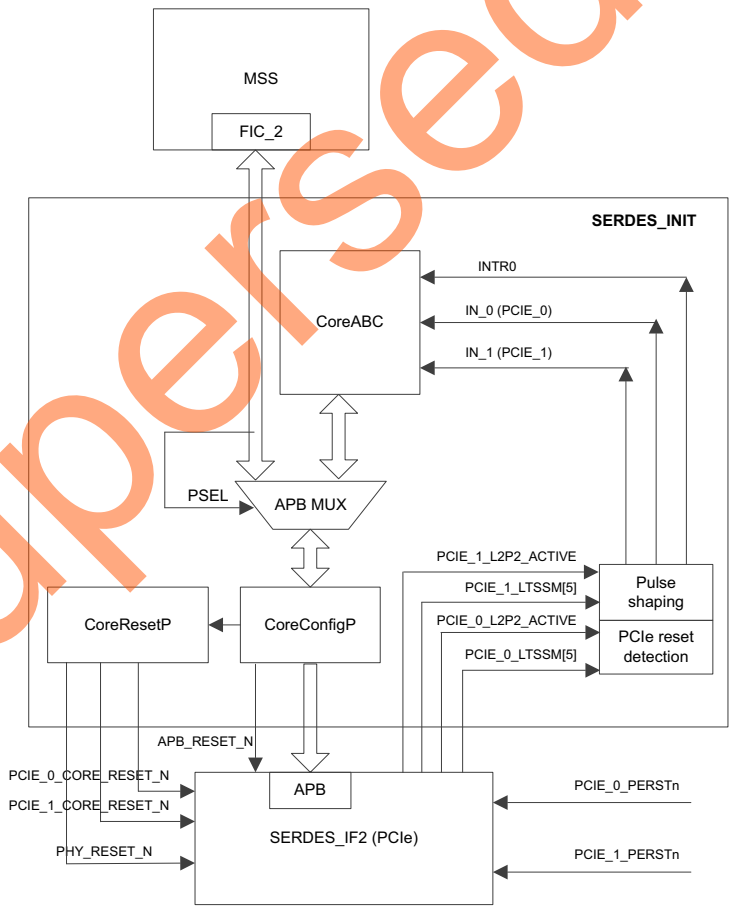


***Figure 19* • M2S/M2GL 060/090 Device PCIe Reset Generation with PERSTn**

Superseded

Implementation Using M2S090 FG676 Device

The PCIe control plane demo design for M2S090 is created using CoreABC standalone peripheral initialization method and HOTRESET logic is implemented to detect the PCIe reset. The top-level SmartDesign with the SERDES_INIT and SERDES_IF blocks is shown in Figure 20. SERDES_INIT is a SmartDesign block used to initialize the SERDES and to generate the SERDES (PCIe) resets. The SERDES_IF_2 PERST_N signals are connected to the PCIe PERSTn signals on the board.

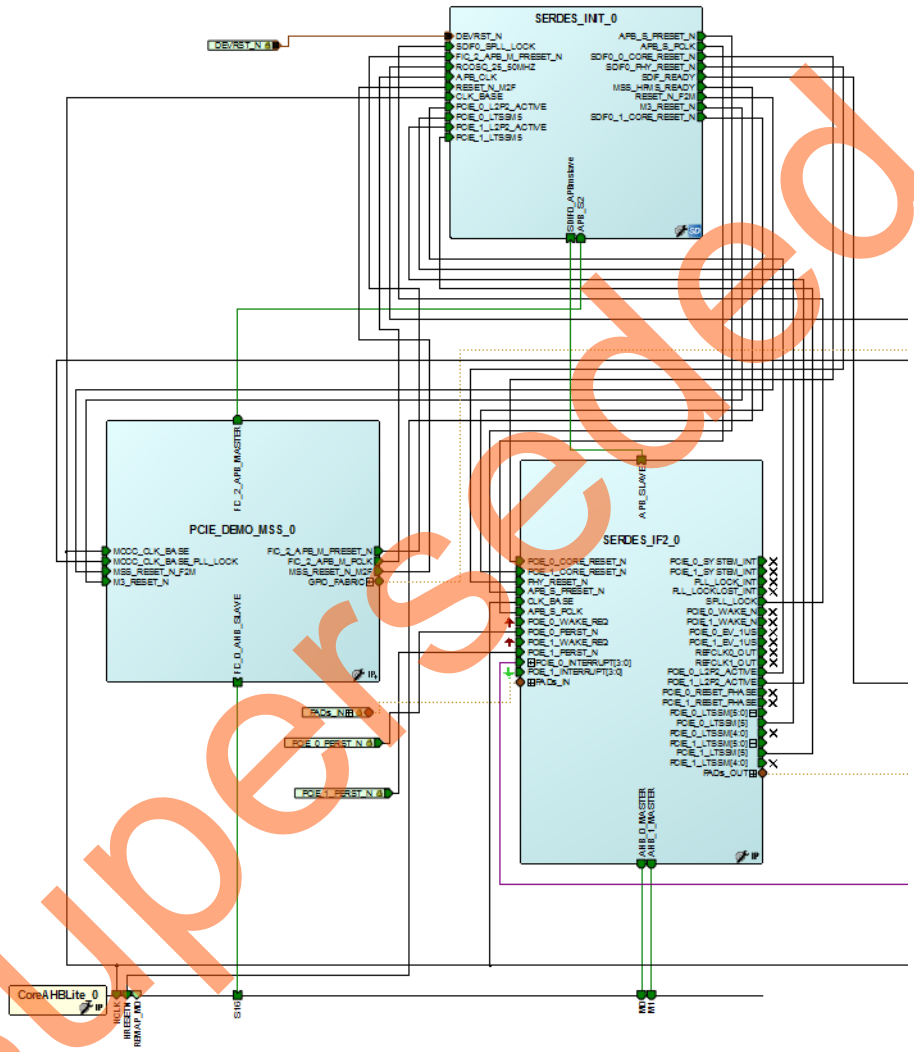


Figure 20 • **Top-Level Design**

Superseded

The SERDES_INIT SmartDesign has CoreABC, CoreResetP, CoreConfigP, APB_MUX, and HOTRESET blocks as shown in Figure 21. The HOTRESET logic monitors the LTSSM[5] and L2P2_ACTIVE signals and generates the INTR0 signal. Two HOTRESET blocks are used to detect the PCIe resets for PCIe controller0/controller1. The INTR0 signals from HOTRESET blocks are used to generate INTREQ of CoreABC. CoreABC determines the PCIe controller0/controller1 HOTRESET events using the PCIE_INTR signals from the HOTRESET blocks and interrupts routine issues SERDES (PCIe) Core0/Core1 and AXI interface soft resets when INTREQ goes low.

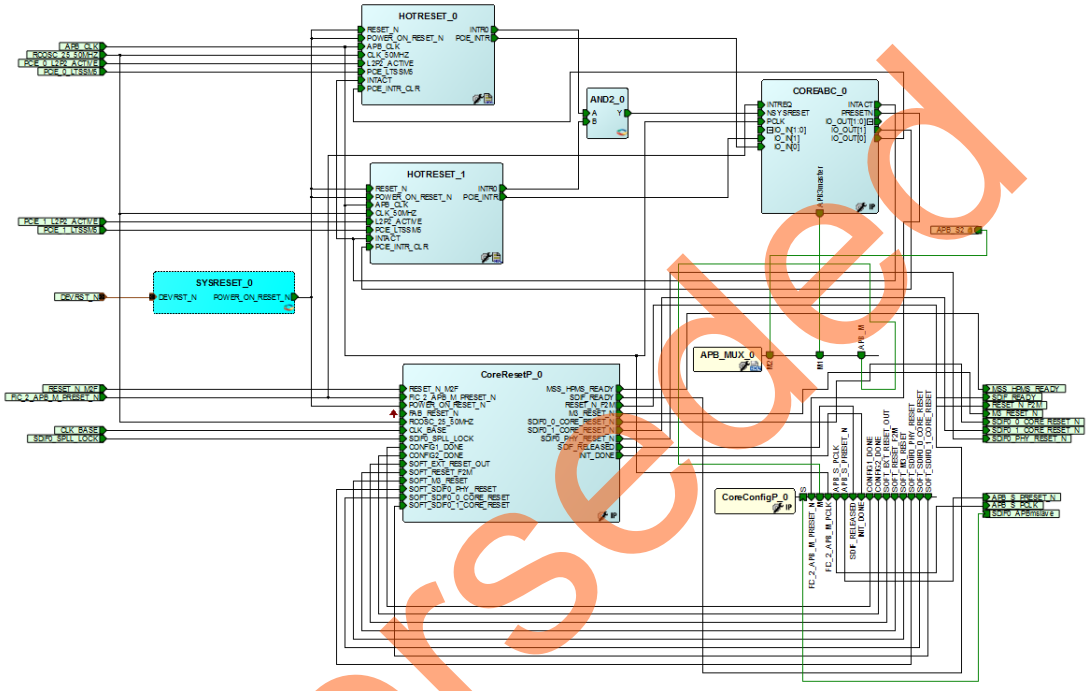


Figure 21 • **SERDES_INIT SmartDesign**

Superseded

CoreResetP is configured to generate resets for SERDES_IF2_0, as shown in Figure 22.

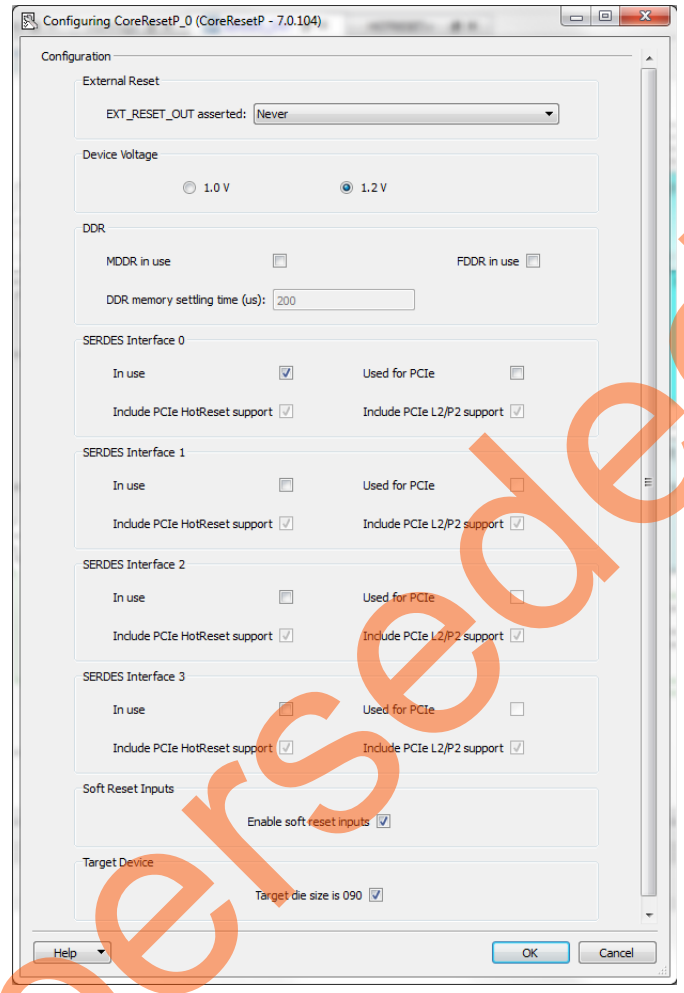


Figure 22 • **CoreResetP Configuration**

Superseded

CoreConfigP is configured for SERDES_IF_0 block, as shown in Figure 23.

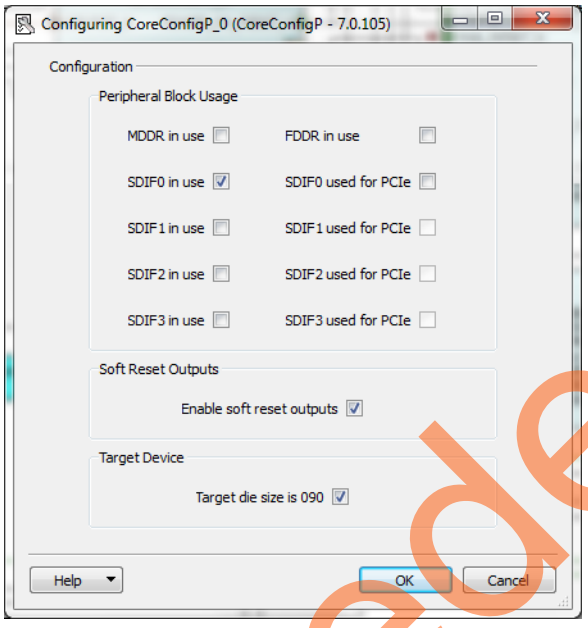


Figure 23 • **CoreConfigP Configuration**

Superseded

CoreABC configuration is shown in Figure 24. Libero generates the `SERDESIF_0_init_abc.txt` file at *<pjt_location>\PCIE_DEMO\component\work\PCIE_DEMO\SERDES_IF2_0* with CoreABC code to initialize the SERDES (PCIe). The code is modified to reset the SERDES core0/core1 and AXI, using the SERDES soft reset register on CoreABC active low interrupt.

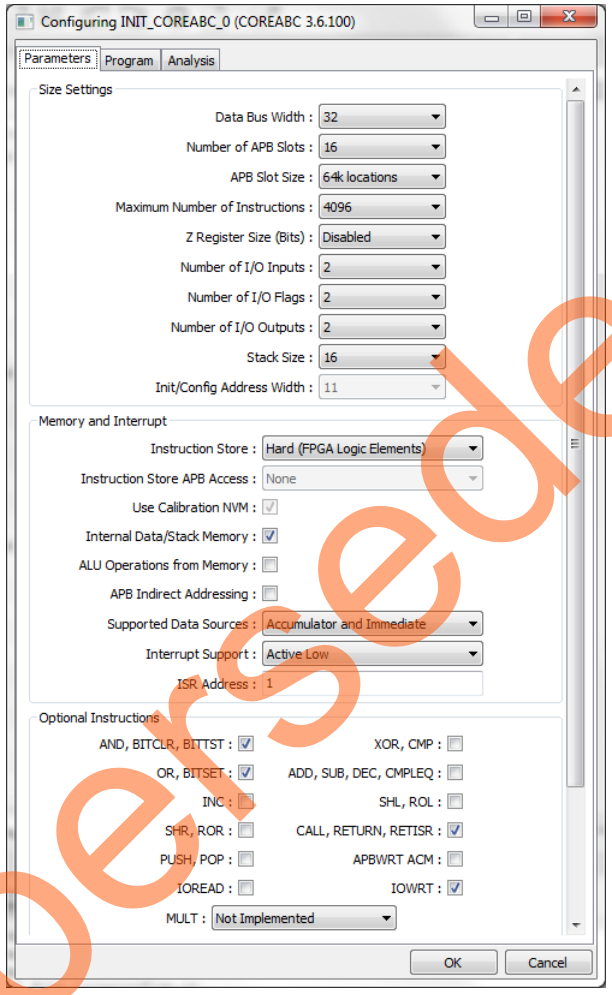


***Figure 24* • CoreABC Configuration**

Superseded

M2S/M2GL/060/090 Device Dual PCIe without PERSTn

Figure 25 shows the implementation of PCIe reset sequence for the M2S/M2GL060/090 devices without PERSTn. The SERDES_IF_2 PERST_N signal is connected to the SERDES_IF_2 L2P2_ACTIVE signal.

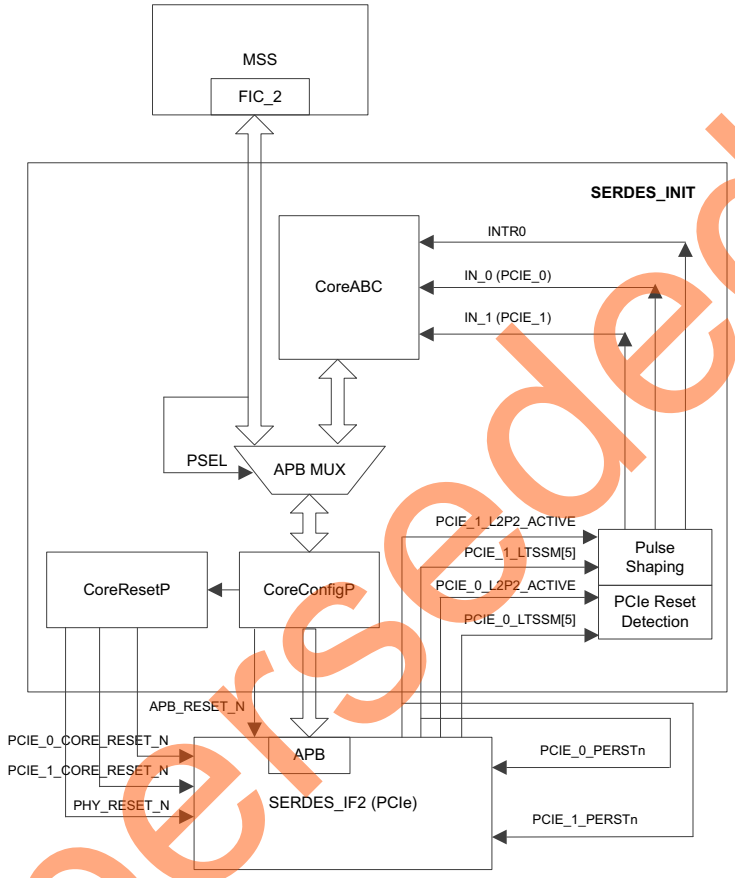


Figure 25 • **M2S090/M2GL090 Device PCIe Reset Generation without PERSTn**

Superseded

Implementation using M2S090 FG676 Device

The PCIe control plane demo design for M2S090 is created using CoreABC standalone peripheral initialization method and HOTRESET logic is implemented to detect the PCIe reset. The top-level SmartDesign with the SERDES_INIT and SERDES_IF blocks is shown in Figure 26.

***Figure 26* • Top-Level Design**

Superseded

Running the Design

For more information about setting up the board and the steps to run the design, refer to “Connecting the Evaluation Kit to the Host PC” and “Running the Design” sections, in the *UG0456: SmartFusion2 SoC FPGA PCIe Control Plane Demo User Guide*.

Testing the PCIe Reset

The reference designs provided with this application note implements a PCIe Control Plane demo. Use the PI flow and CoreABC using the recommended PCIe reset methodology. To validate the PCIe reset flow, the following test is performed by the user on their host system.

The following steps describe how to test the PCIe reset feature:

1. Click **Exit** to quit the PCIe Demo application.
2. Restart (do not shut down) the host PC or put the host PC in Hibernate/Sleep mode (this issues PERSTn to the PCIe endpoint).
3. After the host PC restarts or comes out of the sleep mode, check the Device Manager and ensure that the endpoint device is detected correctly.
4. Run the PCIe Demo application by following steps 2 through 13. The reference design must run for any number of restarts or coming out of Hibernate or Sleep mode without causing any system hang.

Notes:

1. *The Laptop’s PCIe adapter cards do not have PERSTn signal.*
2. *In some cases, while exiting from the sleep mode, the host PC may not recognize the device. This may be due to L0S and L1 entry and exit latencies of the host PC not matching with SmartFusion2 and IGLOO2 PCIe device latencies. This is dependent on specific motherboard configuration. If this occurs, the device needs to be reset to be detected.*

Conclusion

This application note describes the recommended implementation of PCIe reset sequence for the SmartFusion2 and IGLOO2 devices using the standalone peripheral initialization flow. The existing PCIe control plane demo design has been used to illustrate the PCIe reset sequence for the SmartFusion2 Security Evaluation Kit and IGLOO2 Evaluation Kit.

Superseded

Appendix: Design and Programming Files

Download the design files from the Microsemi SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_ac437_liberov11p6_df

The design file consists of Libero SoC Verilog project, SoftConsole software project, and programming files (*.stp) for the SmartFusion2 Security Evaluation Kit board and IGLOO2 Evaluation Kit board. Refer to the `Readme.txt` file included in the design file for the directory structure and description.

Download the programming files from the Microsemi SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_ac437_liberov11p6_pf

The programming file consists of STAPL programming file (*.stp) for the SmartFusion2 Security Evaluation Kit board and IGLOO2 Evaluation Kit board.

Superseded

List of Changes

The following table shows important changes made in this document for each revision.

Revision	Changes	Page
Revision 2 (November 2015)	Updated the document for Libero SoC v11.6 software release (SAR 71573 and SAR 71968).	N/A
Revision 1 (March 2015)	Initial release.	N/A

Superseded

Superseded

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace 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; security technologies and scalable anti-tamper products; Ethernet Solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,600 employees globally. Learn more at **www.microsemi.com**.

Microsemi Corporate 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

E-mail: sales.support@microsemi.com

© 2015 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are 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.