

AC437
Application Note
Implementing PCIe Reset Sequence in SmartFusion2
and IGLOO2 Devices



a  MICROCHIP company

Microsemi Headquarters

One Enterprise, Aliso Viejo,
CA 92656 USA

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

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

Sales: +1 (949) 380-6136

Fax: +1 (949) 215-4996

Email: sales.support@microsemi.com

www.microsemi.com

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

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

About Microsemi

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

Contents

1	Revision History	1
1.1	Revision 5.0	1
1.2	Revision 4.0	1
1.3	Revision 3.0	1
1.4	Revision 2.0	1
1.5	Revision 1.0	1
2	Implementing PCIe Reset Sequence in SmartFusion2 and IGLOO2 Devices	2
2.1	Purpose	2
2.2	Introduction	2
2.2.1	Special Consideration for L2	2
2.3	References	3
2.4	PCIe Control Plane Demo Design Requirements	3
2.5	Prerequisites	4
2.6	Design Description	4
2.7	Implementation	4
2.7.1	M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices with PERSTn	5
2.7.2	M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices without PERSTn	12
2.7.3	M2S060 / M2S090 / M2GL060 / M2GL090 Devices with PERSTn	15
2.7.4	M2S060 / M2S090 / M2GL060 / M2GL090 Devices without PERSTn	22
2.7.5	M2S/M2GL 060/090 Device Dual PCIe with PERSTn	23
2.7.6	M2S/M2GL 060/090 Device Dual PCIe without PERSTn	30
2.8	Running the Design	32
2.8.1	Testing the PCIe Reset	33
2.9	Conclusion	33
3	Appendix: Programming the Device Using FlashPro Express	34

Figures

Figure 1	M2S/M2GL010/025/050/150 Device PCIe Reset Generation with PERSTn	6
Figure 2	Top-Level Connection	7
Figure 3	SERDES INIT SmartDesign	8
Figure 4	CoreResetP Configuration	9
Figure 5	CoreConfigP Configuration	10
Figure 6	CoreABC Configuration	11
Figure 7	CoreABC Code	12
Figure 8	M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices PCIe Reset Generation without PERSTn	13
Figure 9	Top-Level Design	14
Figure 10	SERDES INIT SmartDesign	15
Figure 11	M2S060 / M2S090 / M2GL060 / M2GL090 Devices PCIe Reset Generation with PERSTn	16
Figure 12	Top Level Design	17
Figure 13	SERDES INIT SmartDesign	18
Figure 14	CoreResetP Configuration	19
Figure 15	CoreConfigP Configuration	20
Figure 16	CoreABC Configuration	21
Figure 17	M2S060 / M2S090 / M2GL060 / M2GL090 Devices PCIe Reset Generation without PERSTn	22
Figure 18	Top-Level Design	23
Figure 19	M2S/M2GL 060/090 Device PCIe Reset Generation with PERSTn	24
Figure 20	Top-Level Design	25
Figure 21	SERDES_INIT SmartDesign	26
Figure 22	CoreResetP Configuration	27
Figure 23	CoreConfigP Configuration	28
Figure 24	CoreABC Configuration	29
Figure 25	M2S/M2GL 060/090 Device Dual PCIe without PERSTn	30
Figure 26	Top-Level Design	31
Figure 27	IGLOO2 Evaluation Kit Board	32
Figure 28	SmartFusion2 Security Evaluation Kit Board	33
Figure 29	FlashPro Express Job Project	34
Figure 30	New Job Project from FlashPro Express Job	35
Figure 31	Programming the Device	35
Figure 32	FlashPro Express—RUN PASSED	36

Tables

Table 1	Design Requirements	3
---------	---------------------------	---

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 5.0

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

- Updated the document for Libero SoC v12.6.
- Removed the references to Libero version numbers.

1.2 Revision 4.0

The following is a summary of the changes made in revision 4.0 of this document.

- Information about L2 state was updated. Refer [Special Consideration for L2](#), page 2.
- Information about SERDES_IF_2_L2P2_ACTIVE signal was added. Refer [M2S060 / M2S090 / M2GL060 / M2GL090 Devices without PERSTn](#), page 22 and [M2S/M2GL 060/090 Device Dual PCIe without PERSTn](#), page 30.
- Information about [M2S060 / M2S090 / M2GL060 / M2GL090 Devices with PERSTn](#), page 15 and [M2S/M2GL 060/090 Device Dual PCIe with PERSTn](#), page 23 was updated.
- Updated the document for Libero SoC v11.8 SP3 software release.

1.3 Revision 3.0

Updated the document for Libero SoC v11.7 software release.

1.4 Revision 2.0

Updated the document for Libero SoC v11.6 software release.

1.5 Revision 1.0

Revision 1.0 is the first publication of this document.

2 Implementing PCIe Reset Sequence in SmartFusion2 and IGLOO2 Devices

2.1 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.

2.2 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 about 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 endpoint device state machines, hardware logic, port states, and configuration registers (except for the sticky registers) to initialize the default conditions.

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.

2.2.1 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 root port uses this signal for an endpoint to exit from L2 state. Some of the applications do not use PERSTn signal (when PCIe is used over distances through long cable or fiber lengths). In such applications, the L2 state before the root port access of an endpoint must release itself from L2 without the PERSTn connection. 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.

2.3 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

2.4 PCIe Control Plane Demo Design Requirements

The following table lists the hardware and software required for this application.

Table 1 • Design Requirements

Requirement	Version
Operating System	64 bit Windows 7 and 10
Hardware	
SmartFusion2 Security Evaluation Kit and IGLOO2 Rev D or later Evaluation Kit:	
• 12 V adapter (provided along with the kit) - FlashPro4 programmer (provided along with the kit)	
Software	
FlashPro Express	Refer to the <code>readme.txt</code> file provided in the design files for the software versions used with this reference design.
Libero [®] System-on-Chip (SoC)	
SoftConsole	
Host PC Drivers	Refer <i>TU0456: SmartFusion2 SoC FPGA PCIe Control Plane Tutorial</i>
GUI executable	The GUI executable is available at GUI folder of <i>TU0456: SmartFusion2 SoC FPGA PCIe Control Plane Tutorial</i> design file.

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.5 Prerequisites

Before you begin:

1. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location.

<https://www.microsemi.com/product-directory/design-resources/1750-libero-soc>

2. For demo design files download link:

http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_ac437_df

The design file consists of the Libero SoC Verilog project, SoftConsole software project, and programming files (*.job) 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.

2.6 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, page 5](#)
- [M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 Devices without PERSTn, page 12](#)
- [M2S060 / M2S090 / M2GL060 / M2GL090 Devices with PERSTn, page 15](#)
- [M2S060 / M2S090 / M2GL060 / M2GL090 Devices without PERSTn, page 22](#)
- [M2S/M2GL 060/090 Device Dual PCIe with PERSTn, page 23](#)
- [M2S/M2GL 060/090 Device Dual PCIe without PERSTn, page 30](#)

Note: The System Builder flow 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. For more information about peripheral initialization methodology, refer the following documents:

- [SmartFusion2 Standalone Peripheral Initialization User Guide](#)
- [IGLOO2 Standalone Peripheral Initialization User Guide](#)

2.7 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.

Note: For the PCIe reset sequence implementation, the CoreABC clock must be more than 1 MHz.

Note: The PCIe reset sequence described in this application note requires the following register bit settings, which are default settings in Libero:

- `cfg_r_l2_p2_enable = 1'b1`
- `enable_perstn_support = 1'b0`

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

Figure 1, page 6 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 ARM Cortex-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. 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.

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

2.7.1.1 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].

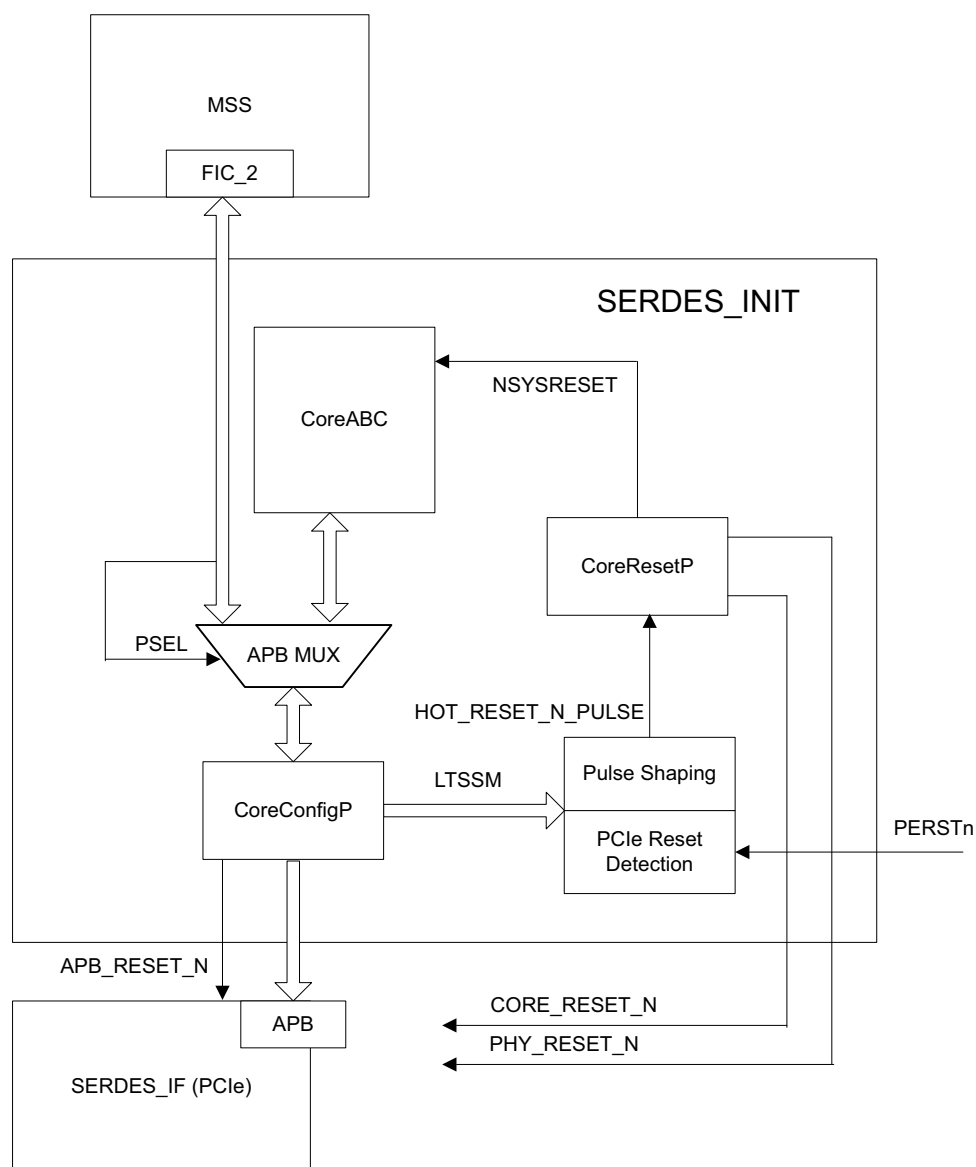
Generate hot_reset_n as the following: hot_reset_n = PERSTn and hot_reset_n_ltssm.

Connect the PERST_N of SERDES_INIT directly to PERSTn on board.

2.7.1.2 SERDES (PCIe) Reset Generation

Use the pulse shaping logic to generate the reset pulse from the hot_reset_n and connect it to the CoreResetP asynchronous reset input to reset the SERDES core (PMA reset, PCIE controller reset, and APB reset). Release the SERDES resets and reinitialize the SERDES using CoreABC.

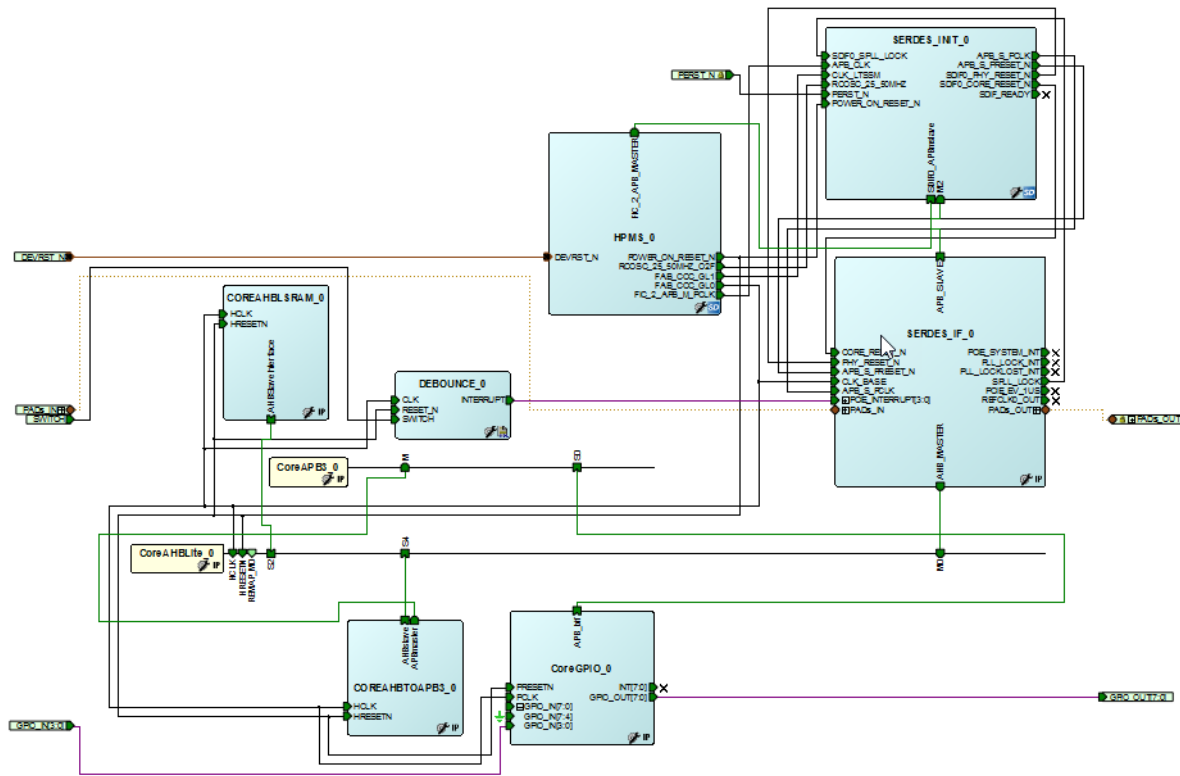
The following figure shows the M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150 devices PCIe reset generation with PERSTn.

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


2.7.1.3 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 the following figure. The SERDES_INIT is a SmartDesign block to initialize the SERDES and to generate the SERDES (PCIe) resets.

Figure 2 • Top-Level Connection



The following figure shows the SERDES INIT SmartDesign.

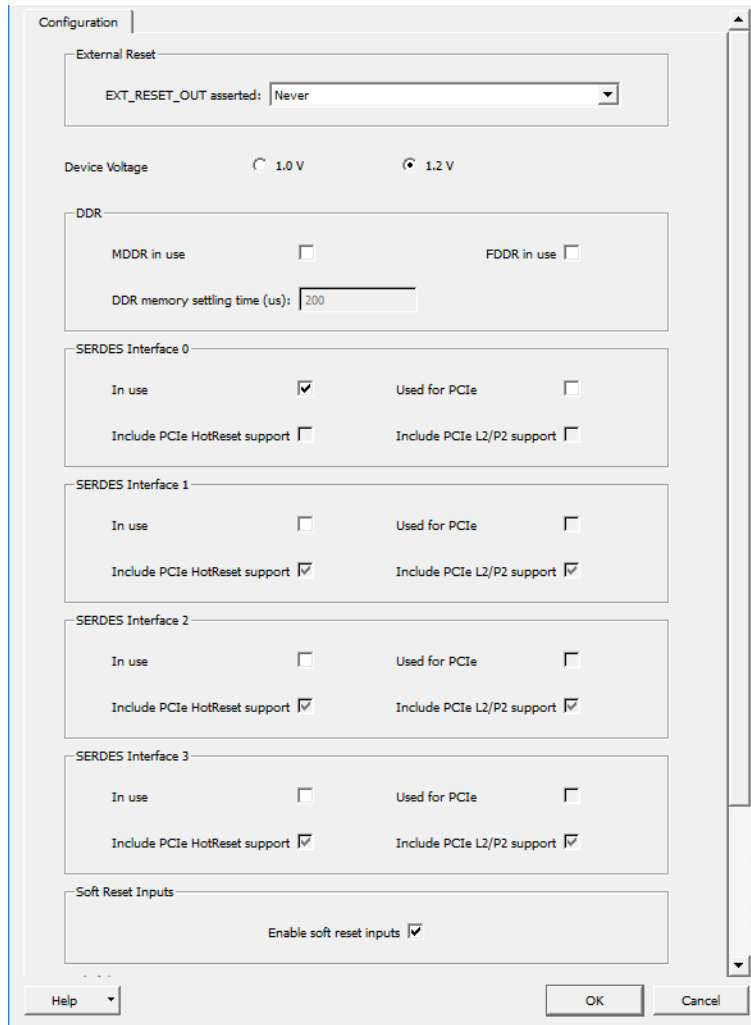
The diagram illustrates the reset signal architecture for the i.MX7D SoC. It features several main components:

- HOTRESET_0**: Receives inputs from `RBOOT_AS_SOFTZ`, `CLK_RSTZ`, `APB_RSTZ`, `SDF_RSTN_LOAD`, `PWRON_RESET_N`, and `PERST_N`. Its output `HOT_RESET_N_PULSE` connects to `CoreResetP_0`.
- CoreResetP_0**: A central reset controller receiving multiple signals including `RESET_N_MZF`, `FC2_APB_M_PRESET_N`, `FC2_APB_M_CLK`, `POWER_ON_RESET_N`, `FAB_RESET_N`, `WDOG_C_29_10MHz`, `SDIO_SALL_LOCK`, `CONFIG_DONE`, `SOFT_EXT_RESET_OUT`, `SOFT_RESET_P2M`, `SOFT_M0_RESET`, `SOFT_SDIO_PHY_RESET`, and `SOFT_SDIO_CORE_RESET`. It outputs `MSS_HRPS_READY`, `SDF_READY`, `RESET_N_P2M`, `M0_RESET_N`, `SDIO_CORE_RESET_N`, `SDIO_PHY_RESET_N`, `SDF_RELEASED_INT_DONE`, and `INT_DONE`.
- COREABC_0**: Receives `NSV_SRESET` and `POLK_O_INQ` as inputs. Its output `PRESETN_O_OUTQ` connects to `APB_MUX_0`.
- APB_MUX_0**: Receives `APB_M` and `PRESETN_O_OUTQ` as inputs.
- CoreConfigP_0**: Receives `FC2_APB_M_PRESET_N`, `FC2_APB_M_CLK`, `APB_S_PCLK`, `APB_S_PRESET_N`, `CONFGL_DONE`, `SOFT_EXT_RESET_OUT`, `SOFT_RESET_P2M`, `SOFT_M0_RESET`, `SOFT_SDIO_PHY_RESET`, `SOFT_SDIO_CORE_RESET`, `R_SFIO_PSEL`, `R_SFIO_WRITE`, and `R_SFIO_MAXCAPU` as inputs. It outputs `SDF_RELEASED_INT_DONE`, `INT_DONE`, `YAC_S_DONE`, `YAC_S_RESET_N`, and `SWP_AFFINITY`.

Additional signals shown include `SOF_READY`, `SDF_CORE_RESET_N`, `SDF_PHY_RESET_N`, `APB_M`, `PRESETN_O_OUTQ`, `YAC_S_DONE`, `YAC_S_RESET_N`, and `SWP_AFFINITY`.

The following figure shows the CoreResetP configured for SERDES_IF_0 block.

Figure 4 • CoreResetP Configuration



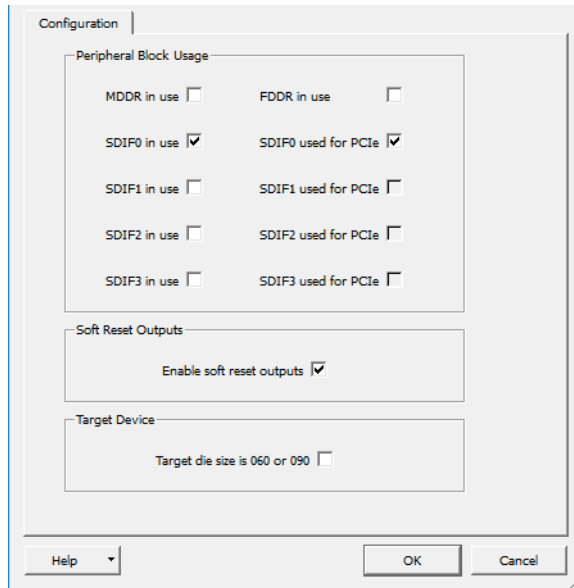
The image shows a 'Configuration' dialog box for the CoreResetP block. It contains several sections for configuring reset behavior:

- External Reset:** A dropdown menu for 'EXT_RESET_OUT asserted:' is set to 'Never'.
- Device Voltage:** Radio buttons for '1.0 V' and '1.2 V'. '1.2 V' is selected.
- DDR:** Checkboxes for 'MDDR in use' and 'FDDR in use' are both unchecked. A text field for 'DDR memory settling time (us):' is set to '200'.
- SERDES Interface 0:** 'In use' is checked. 'Used for PCIe' is unchecked. 'Include PCIe HotReset support' and 'Include PCIe L2/P2 support' are both unchecked.
- SERDES Interface 1:** 'In use' is unchecked. 'Used for PCIe' is unchecked. 'Include PCIe HotReset support' and 'Include PCIe L2/P2 support' are both checked.
- SERDES Interface 2:** 'In use' is unchecked. 'Used for PCIe' is unchecked. 'Include PCIe HotReset support' and 'Include PCIe L2/P2 support' are both checked.
- SERDES Interface 3:** 'In use' is unchecked. 'Used for PCIe' is unchecked. 'Include PCIe HotReset support' and 'Include PCIe L2/P2 support' are both checked.
- Soft Reset Inputs:** 'Enable soft reset inputs' is checked.

At the bottom, there are 'Help', 'OK', and 'Cancel' buttons.

The following figure shows the CoreConfigP configured for SERDES_IF_0 block.

Figure 5 • CoreConfigP Configuration



The image shows a 'Configuration' dialog box for CoreConfigP. It contains three main sections: 'Peripheral Block Usage', 'Soft Reset Outputs', and 'Target Device'. The 'Peripheral Block Usage' section has a table of checkboxes for MDDR, FDDR, and four SDIF blocks, with 'SDIF0 in use' and 'SDIF0 used for PCIe' checked. The 'Soft Reset Outputs' section has 'Enable soft reset outputs' checked. The 'Target Device' section has 'Target die size is 060 or 090' unchecked. At the bottom are 'Help', 'OK', and 'Cancel' buttons.

Peripheral Block Usage	
MDDR in use ☐	FDDR in use ☐
SDIF0 in use ☒	SDIF0 used for PCIe ☒
SDIF1 in use ☐	SDIF1 used for PCIe ☐
SDIF2 in use ☐	SDIF2 used for PCIe ☐
SDIF3 in use ☐	SDIF3 used for PCIe ☐

Soft Reset Outputs

Enable soft reset outputs ☒

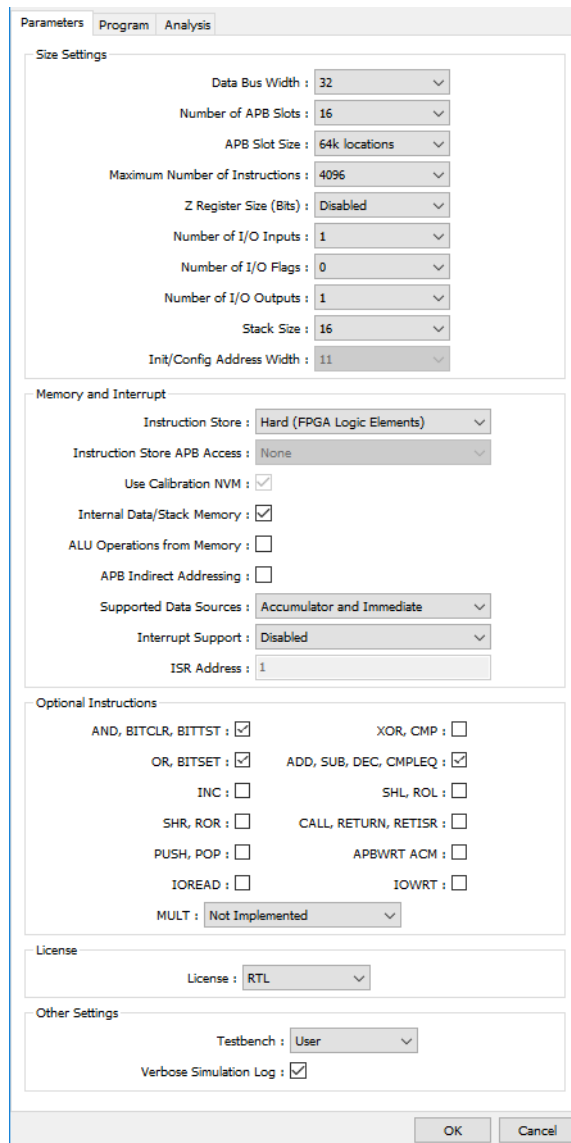
Target Device

Target die size is 060 or 090 ☐

Help OK Cancel

CoreABC is configured, as shown in the following figure.

Figure 6 • CoreABC Configuration



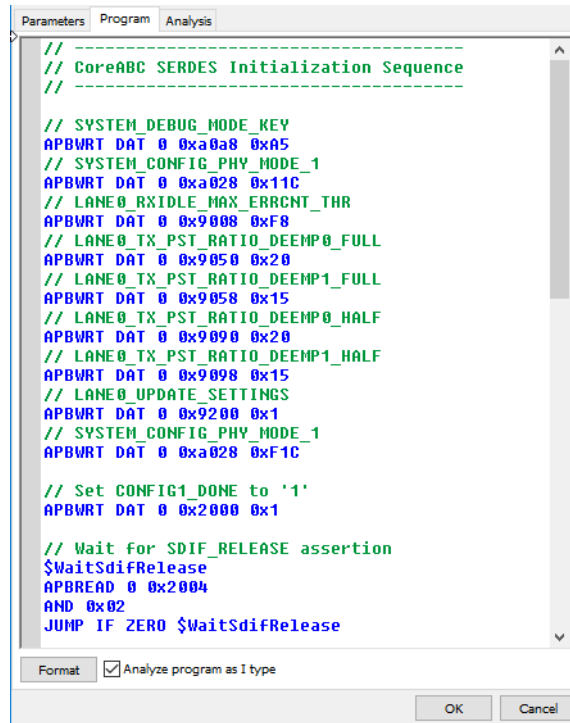
The figure shows the CoreABC Configuration dialog box with the following settings:

- Parameters** tab is selected.
- Size Settings:**
 - Data Bus Width: 32
 - Number of APB Slots: 16
 - APB Slot Size: 64k locations
 - Maximum Number of Instructions: 4096
 - Z Register Size (Bits): Disabled
 - Number of I/O Inputs: 1
 - Number of I/O Flags: 0
 - Number of I/O Outputs: 1
 - Stack Size: 16
 - Init/Config Address Width: 11
- Memory and Interrupt:**
 - Instruction Store: Hard (FPGA Logic Elements)
 - Instruction Store APB Access: None
 - Use Calibration NVM: ☒
 - Internal Data/Stack Memory: ☒
 - ALU Operations from Memory: ☐
 - APB Indirect Addressing: ☐
 - Supported Data Sources: Accumulator and Immediate
 - Interrupt Support: Disabled
 - ISR Address: 1
- Optional Instructions:**
 - AND, BITCLR, BITTST: ☒
 - XOR, CMP: ☐
 - OR, BITSET: ☒
 - ADD, SUB, DEC, CMPLAQ: ☒
 - INC: ☐
 - SHL, ROL: ☐
 - SHR, ROR: ☐
 - CALL, RETURN, RETISR: ☐
 - PUSH, POP: ☐
 - APBWRT ACM: ☐
 - IOREAD: ☐
 - IOWRT: ☐
 - MULT: Not Implemented
- License:**
 - License: RTL
- Other Settings:**
 - Testbench: User
 - Verbose Simulation Log: ☒

Buttons: OK, Cancel

The Libero generates the `SERDESIF_O_init_abc.txt` file at `<prt_location>\Libero_Project\component\work\top\SERDES_IF_0` with CoreABC code to initialize the SERDES (PCIe). This code is used for CoreABC program as shown in the following figure.

Figure 7 • CoreABC Code



```
// -----
// CoreABC SERDES Initialization Sequence
// -----

// SYSTEM_DEBUG_MODE_KEY
APBVRT DAT 0 0xa0a8 0xa5
// SYSTEM_CONFIG_PHY_MODE_1
APBVRT DAT 0 0xa028 0x11c
// LANE0_RXIDLE_MAX_ERRCNT_THR
APBVRT DAT 0 0x9008 0xf8
// LANE0_TX_PST_RATIO_DEEMP0_FULL
APBVRT DAT 0 0x9050 0x20
// LANE0_TX_PST_RATIO_DEEMP1_FULL
APBVRT DAT 0 0x9058 0x15
// LANE0_TX_PST_RATIO_DEEMP0_HALF
APBVRT DAT 0 0x9090 0x20
// LANE0_TX_PST_RATIO_DEEMP1_HALF
APBVRT DAT 0 0x9098 0x15
// LANE0_UPDATE_SETTINGS
APBVRT DAT 0 0x9200 0x1
// SYSTEM_CONFIG_PHY_MODE_1
APBVRT DAT 0 0xa028 0xf1c

// Set CONFIG1_DONE to '1'
APBVRT DAT 0 0x2000 0x1

// Wait for SDIF_RELEASE assertion
$WaitSdifRelease
APBREAD 0 0x2004
AND 0x02
JUMP IF ZERO $WaitSdifRelease
```

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

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

2.7.2.1 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]`.

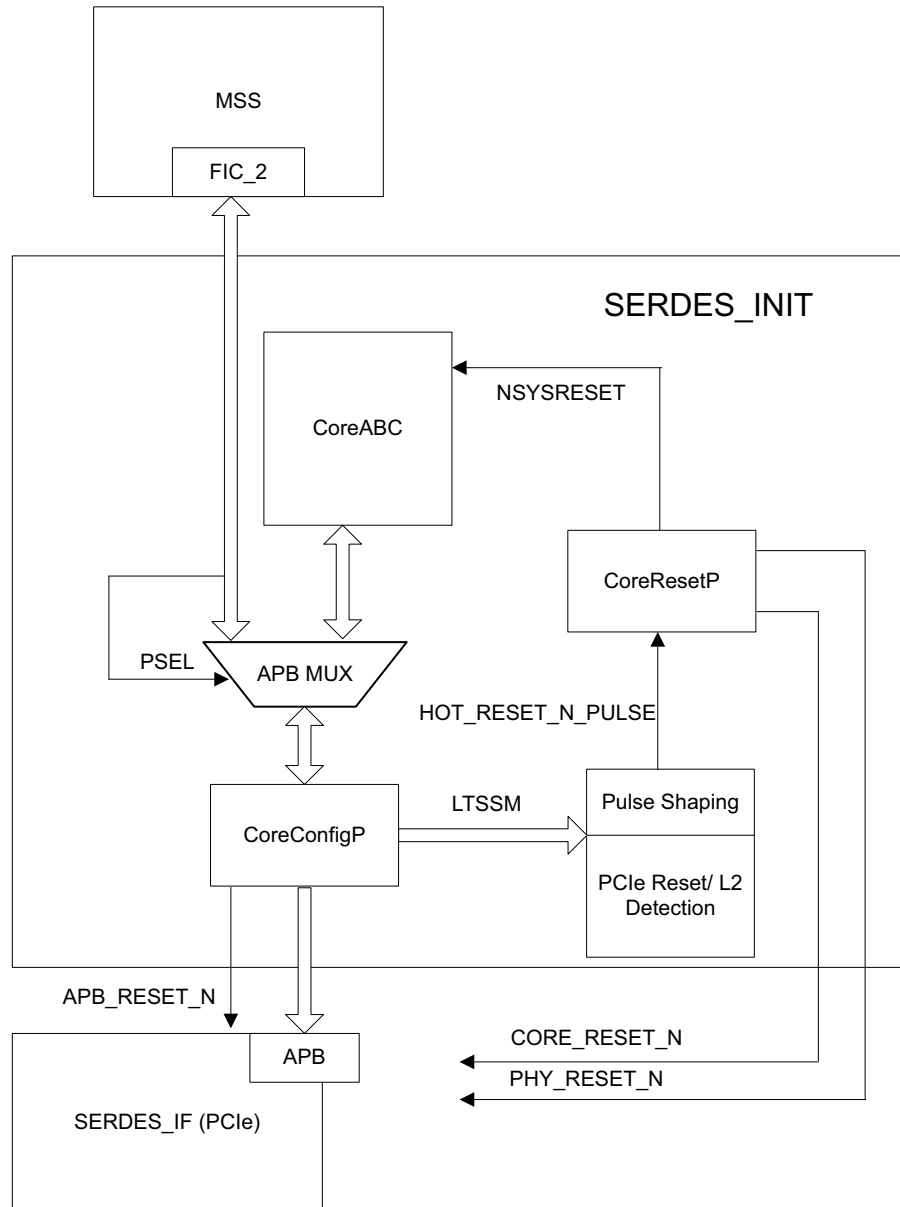
Generate `hot_reset_n` as the following: `hot_reset_n = l2_detected_n & hot_reset_n_ltssm`.

2.7.2.2 SERDES (PCIe) Reset Generation

Use pulse shaping logic to generate the reset pulse from the `hot_reset_n` and connect it to the CoreResetP asynchronous reset input to 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. 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.

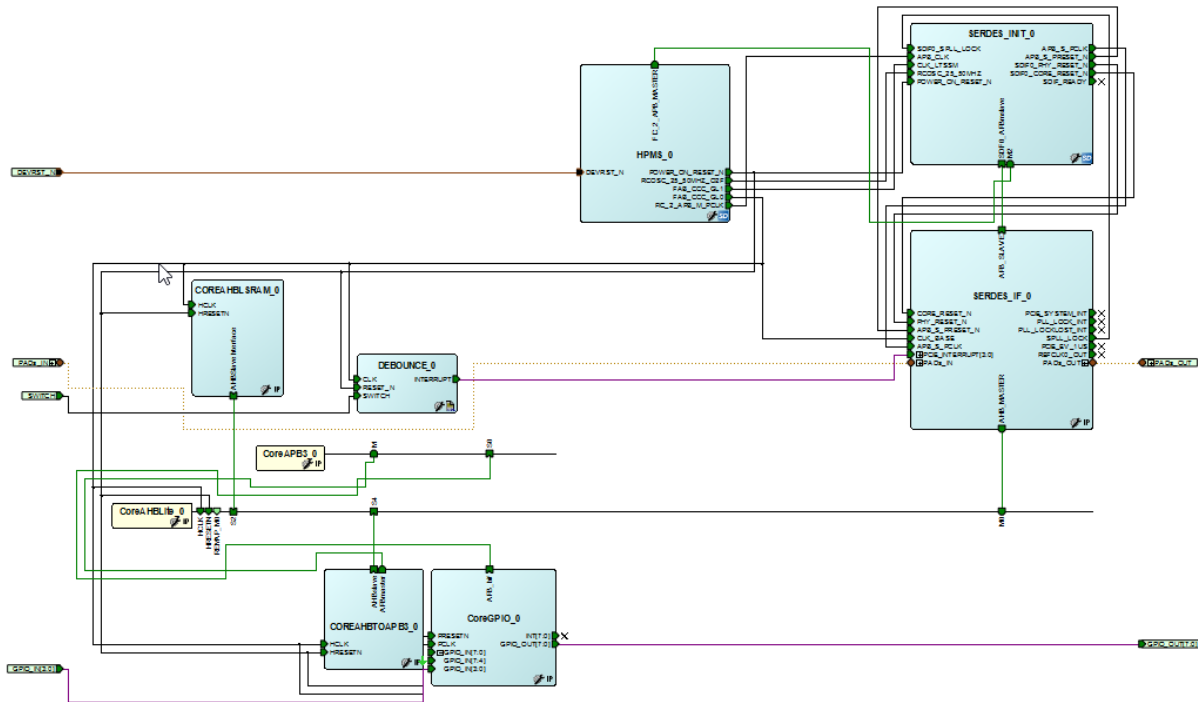
**Figure 8 • M2S010 / M2S025 / M2S050 / M2S150 / M2GL010 / M2GL025 / M2GL050 / M2GL150
Devices PCIe Reset Generation without PERSTn**



2.7.2.3 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 the following figure. The SERDES_INIT is implemented to detect the PCIe reset without monitoring the PERSTn and to generate the SERDES (PCIe) resets.

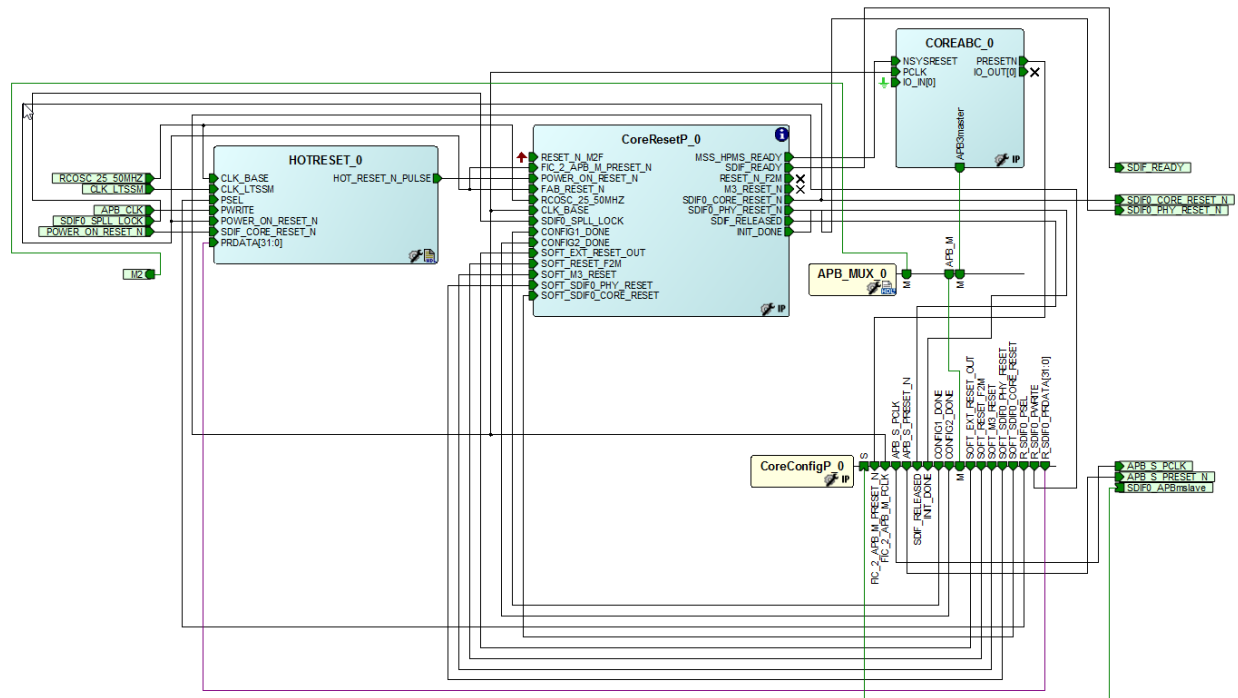
Figure 9 • Top-Level Design



The SERDES_INIT has CoreABC, CoreResetP, CoreConfigP, and HOTRESET blocks, as shown in the following figure. 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](#), page 5.

The following figure shows the SERDES INIT SmartDesign.

Figure 10 • SERDES INIT SmartDesign



2.7.3 M2S060 / M2S090 / M2GL060 / M2GL090 Devices with PERSTn

The following figure shows the implementation of the 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. Also, 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 Cortex-M3 or utilizing the SmartDebug SERDES utility, which is available for both SmartFusion2 and IGLOO2 devices.

The CoreABC takes few APB cycles to reset the SERDES Core and AXI IF. During these cycles, the SERDES AXI master initiates buffered AXI/AHB transactions and the fabric AXI/AHB slave logic should indicate the busy state to AXI master. In this application note, an AXI_AHB reset is generated to AHB_to_AHB.v logic (which asserts the HREADY to '0' during reset) during the reset operation of SERDES Core/AXI IF.

2.7.3.1 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.

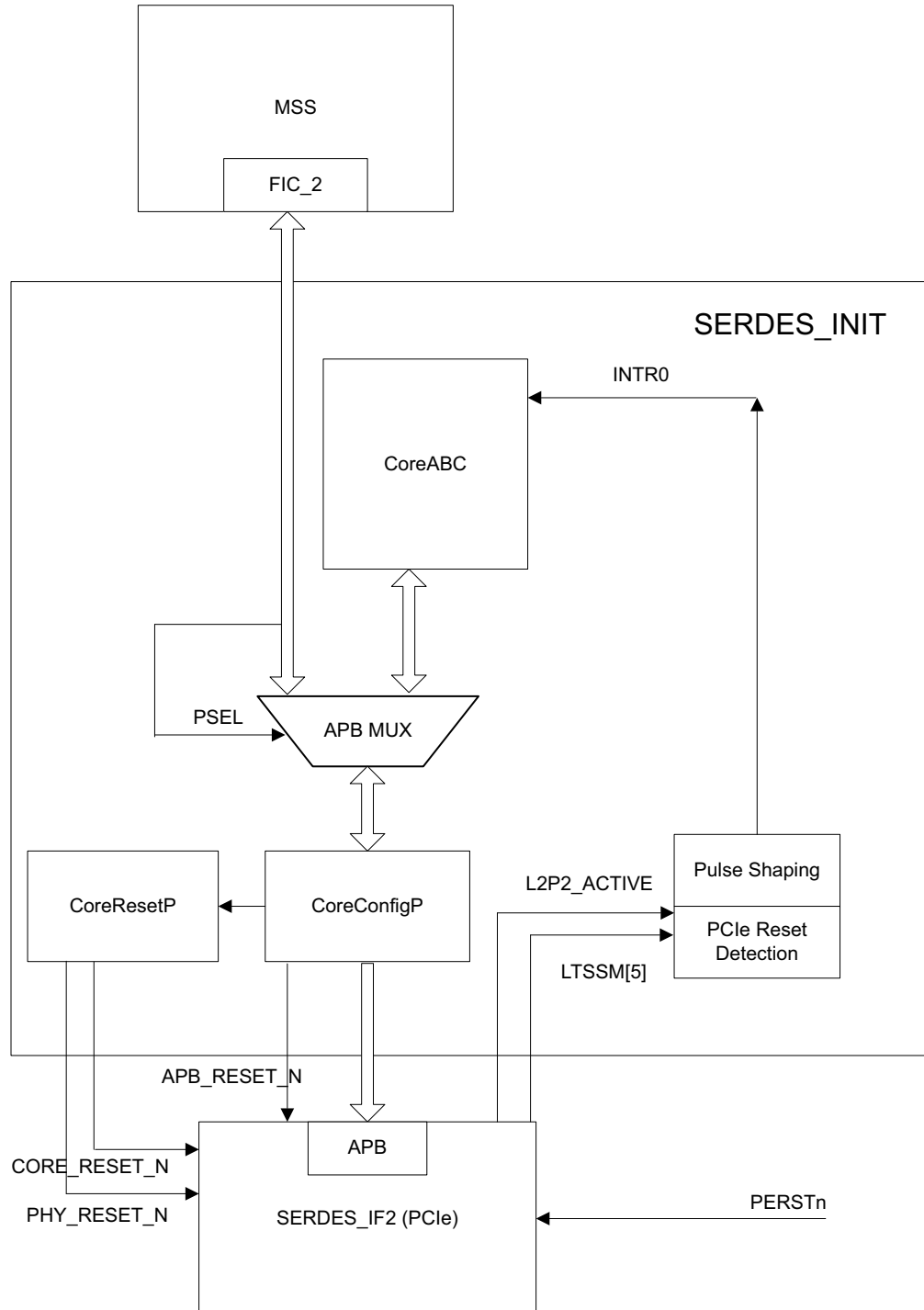
Generate the hot_reset_n as the following: $\text{hot_reset_n} = (! \text{L2P2_ACTIVE}) \& (! \text{LTSSM}[5])$.

Connect the SERDES PCIe PERST_N directly to PERSTn on board.

2.7.3.2 SERDES (PCIe) Reset Generation

Use `hot_reset_n` to generate interrupt INTR0 to CoreABC. The pulse shaping logic generates INTR0 synchronized to the CoreABC clock domain. The CoreABC interrupt routine resets the SERDES PCIE controller reset, and AXI reset using the soft reset register.

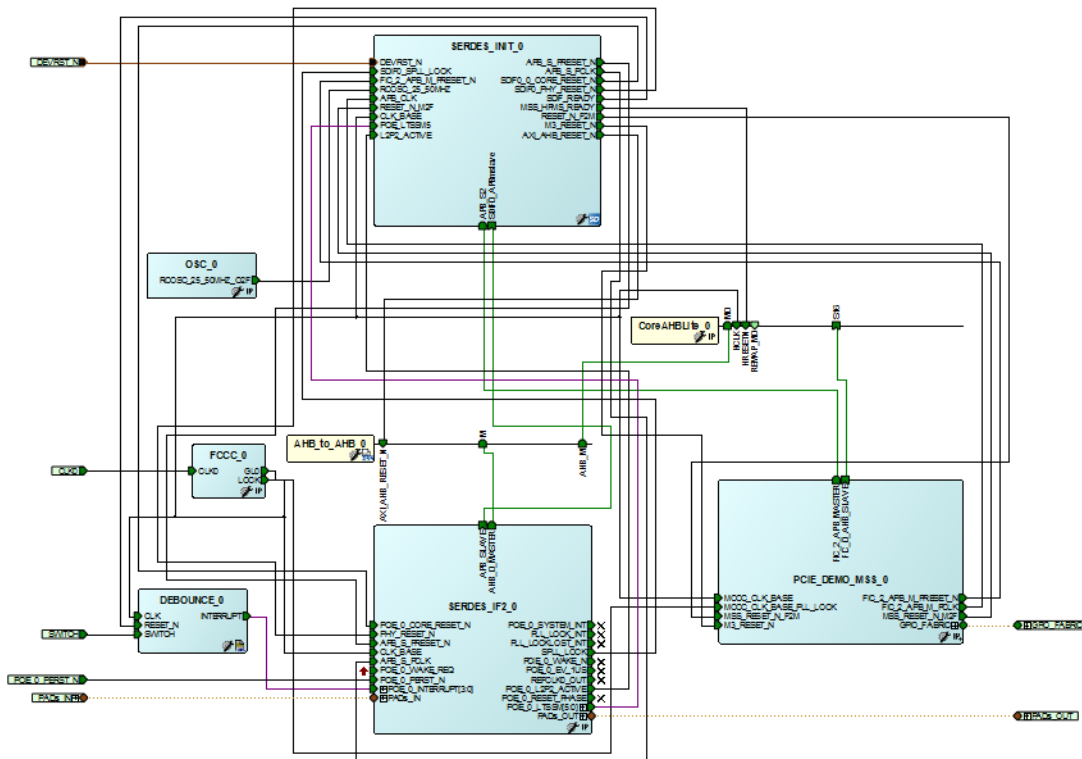
Figure 11 • M2S060 / M2S090 / M2GL060 / M2GL090 Devices PCIe Reset Generation with PERSTn



2.7.3.3 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 are shown in the following figure. 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.

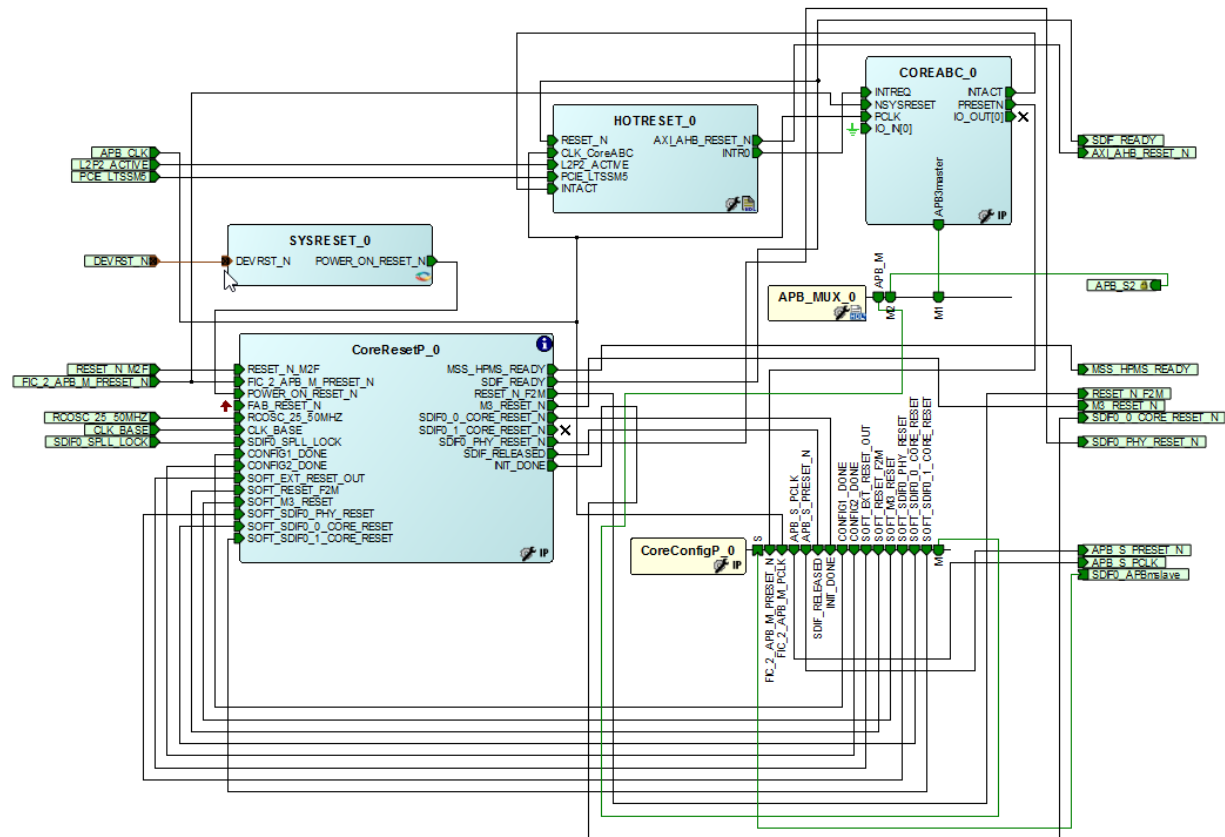
Figure 12 • Top Level Design



The SERDES_INIT SmartDesign has CoreABC, CoreResetP, CoreConfigP, APB_MUX, and HOTRESET blocks, as shown in the following figure. 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.

Note: The HOTRESET logic uses INTACT signal from CoreABC to de-assert the INTR0 interrupt, which is synchronous to CoreABC/HOTRESET logic clock. The INTACT - interrupt acknowledge signals needs to be synchronized to HOTRESET logic if any other processor is used for implementing PCIe reset sequence.

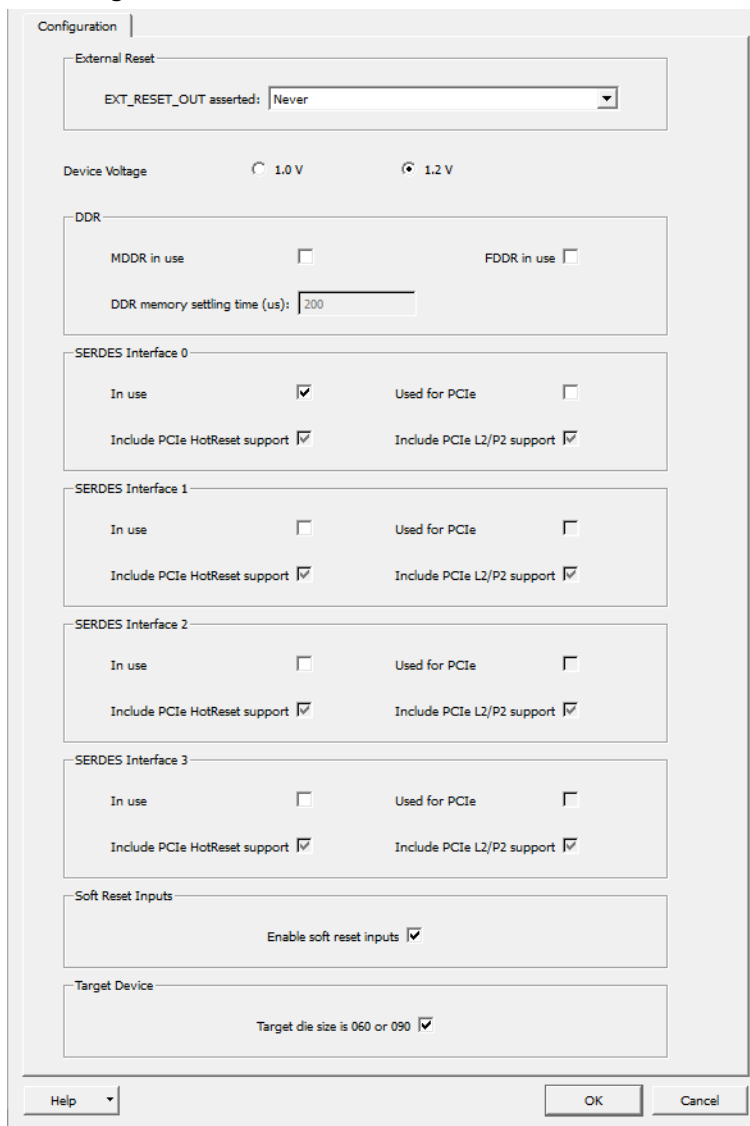
Figure 13 • SERDES INIT SmartDesign



CoreResetP is configured to generate resets for SERDES_IF2_0, as shown in the following figure.

Select only the SERDES interface check box. Do not select the other PCIe check boxes as those functionalities are already implemented in the HOTRESET block.

Figure 14 • CoreResetP Configuration



The figure shows the CoreResetP Configuration dialog box. The configuration is as follows:

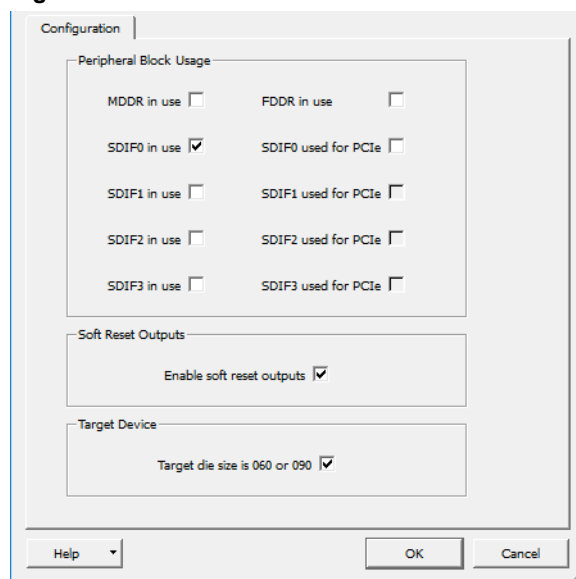
- External Reset:** EXT_RESET_OUT asserted: Never
- Device Voltage:** 1.0 V (radio button), 1.2 V (radio button, selected)
- DDR:**
 - MDDR in use: ☐
 - FDDR in use: ☐
 - DDR memory settling time (us): 200
- SERDES Interface 0:**
 - In use: ☒
 - Used for PCIe: ☐
 - Include PCIe HotReset support: ☒
 - Include PCIe L2/P2 support: ☒
- SERDES Interface 1:**
 - In use: ☐
 - Used for PCIe: ☐
 - Include PCIe HotReset support: ☒
 - Include PCIe L2/P2 support: ☒
- SERDES Interface 2:**
 - In use: ☐
 - Used for PCIe: ☐
 - Include PCIe HotReset support: ☒
 - Include PCIe L2/P2 support: ☒
- SERDES Interface 3:**
 - In use: ☐
 - Used for PCIe: ☐
 - Include PCIe HotReset support: ☒
 - Include PCIe L2/P2 support: ☒
- Soft Reset Inputs:**
 - Enable soft reset inputs: ☒
- Target Device:**
 - Target die size is 060 or 090: ☒

Buttons: Help, OK, Cancel

CoreConfigP is configured for the SERDES_IF_0 block, as shown in the following figure.

Select only the SERDES interface checkbox. Do not select the other PCIe checkboxes as those functionalities are already implemented in the HOTRESET block.

Figure 15 • CoreConfigP Configuration



The image shows a 'Configuration' dialog box for CoreConfigP. It has three main sections: 'Peripheral Block Usage', 'Soft Reset Outputs', and 'Target Device'. The 'Peripheral Block Usage' section contains two columns of checkboxes. The first column has 'MDDR in use' (unchecked), 'SD1F0 in use' (checked), 'SD1F1 in use' (unchecked), 'SD1F2 in use' (unchecked), and 'SD1F3 in use' (unchecked). The second column has 'FDDR in use' (unchecked), 'SD1F0 used for PCIe' (unchecked), 'SD1F1 used for PCIe' (unchecked), 'SD1F2 used for PCIe' (unchecked), and 'SD1F3 used for PCIe' (unchecked). The 'Soft Reset Outputs' section has a single checkbox 'Enable soft reset outputs' which is checked. The 'Target Device' section has a single checkbox 'Target die size is 060 or 090' which is checked. At the bottom, there is a 'Help' button with a dropdown arrow, and 'OK' and 'Cancel' buttons.

Peripheral Block Usage	
MDDR in use ☐	FDDR in use ☐
SD1F0 in use ☒	SD1F0 used for PCIe ☐
SD1F1 in use ☐	SD1F1 used for PCIe ☐
SD1F2 in use ☐	SD1F2 used for PCIe ☐
SD1F3 in use ☐	SD1F3 used for PCIe ☐

Soft Reset Outputs

Enable soft reset outputs ☒

Target Device

Target die size is 060 or 090 ☒

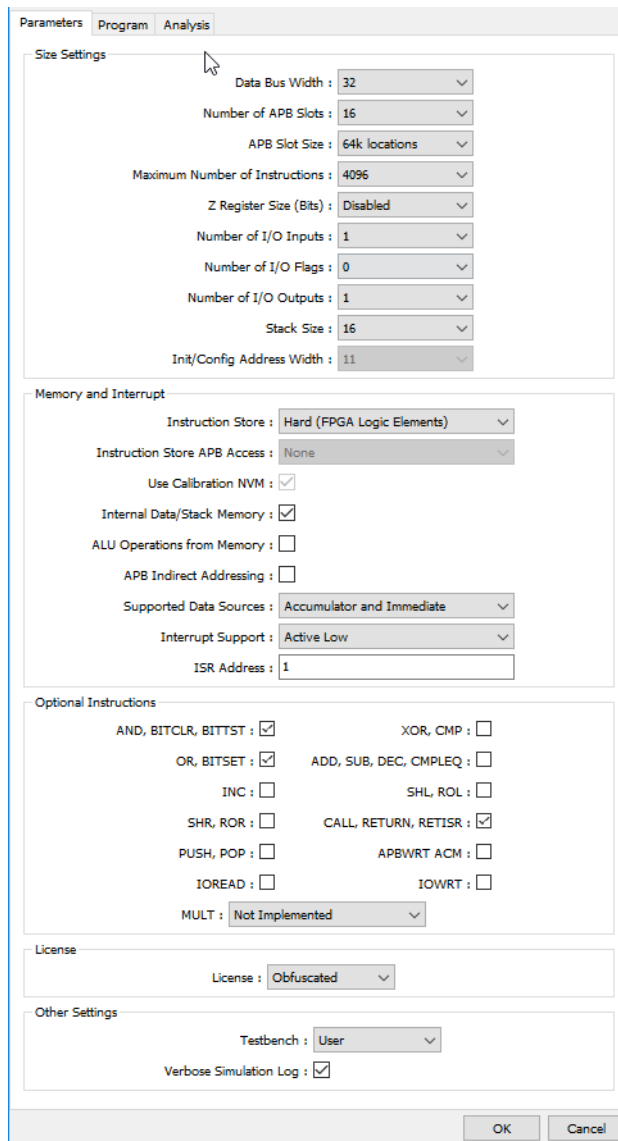
Help OK Cancel

The CoreABC configuration, is as shown in the following figure. Libero generates SERDESIF_0_init_abc.txt file at

<proj_location>\Libero_Project\component\work\top\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.

The following figure shows the CoreABC Configuration.

Figure 16 • CoreABC Configuration



The figure shows the CoreABC Configuration dialog box with the following settings:

- Parameters** tab is selected.
- Size Settings:**
 - Data Bus Width: 32
 - Number of APB Slots: 16
 - APB Slot Size: 64k locations
 - Maximum Number of Instructions: 4096
 - Z Register Size (Bits): Disabled
 - Number of I/O Inputs: 1
 - Number of I/O Flags: 0
 - Number of I/O Outputs: 1
 - Stack Size: 16
 - Init/Config Address Width: 11
- Memory and Interrupt:**
 - Instruction Store: Hard (FPGA Logic Elements)
 - Instruction Store APB Access: None
 - Use Calibration NVM: ☒
 - Internal Data/Stack Memory: ☒
 - ALU Operations from Memory: ☐
 - APB Indirect Addressing: ☐
 - Supported Data Sources: Accumulator and Immediate
 - Interrupt Support: Active Low
 - ISR Address: 1
- Optional Instructions:**
 - AND, BITCLR, BITTST: ☒
 - XOR, CMP: ☐
 - OR, BITSET: ☒
 - ADD, SUB, DEC, CMPEQ: ☐
 - INC: ☐
 - SHL, ROL: ☐
 - SHR, ROR: ☐
 - CALL, RETURN, RETISR: ☒
 - PUSH, POP: ☐
 - APBWRT ACM: ☐
 - IOREAD: ☐
 - IOWRT: ☐
 - MULT: Not Implemented
- License:**
 - License: Obfuscated
- Other Settings:**
 - Testbench: User
 - Verbose Simulation Log: ☒

Buttons: OK, Cancel

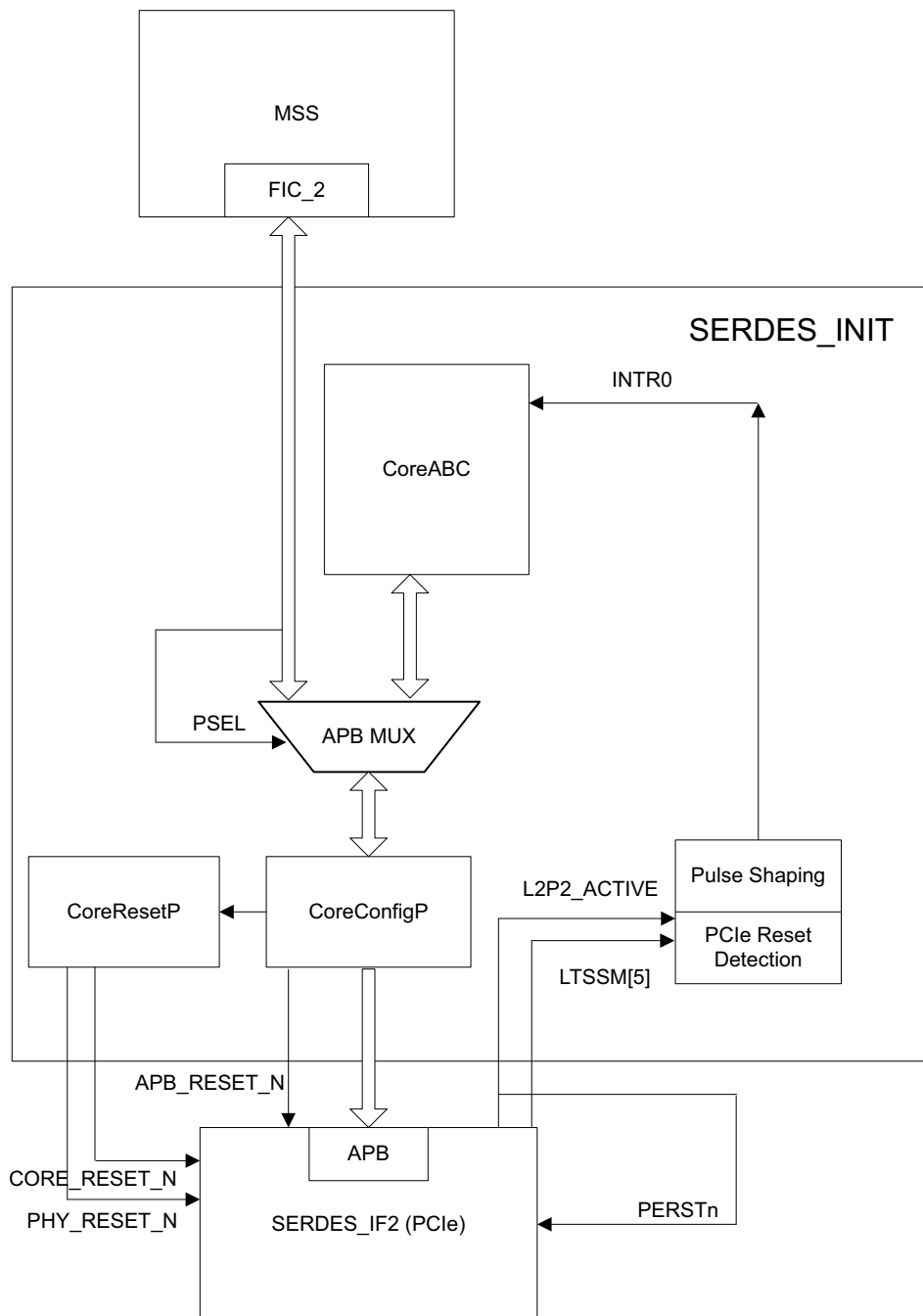
2.7.4 M2S060 / M2S090 / M2GL060 / M2GL090 Devices without PERSTn

The following figure shows the implementation of the PCIe reset sequence for M2S060 / M2S090 / M2GL060 / M2GL090 devices without PERSTn.

Note: This implementation is similar to the PCIe reset sequence implementation for M2S060 / M2S090 / M2GL060 / M2GL090 Devices with PERSTn, page 15 except that the SERDES_IF_2 PERST_N signal is directly connected to the SERDES_IF_2 L2P2_ACTIVE signal instead of PERSTn.

The SERDES_IF_2_L2P2_ACTIVE signal gets asserted to '1' when host/root port initiates L2 state. This assertion causes the EndPoint reset through PERST_N to exit from L2 state.

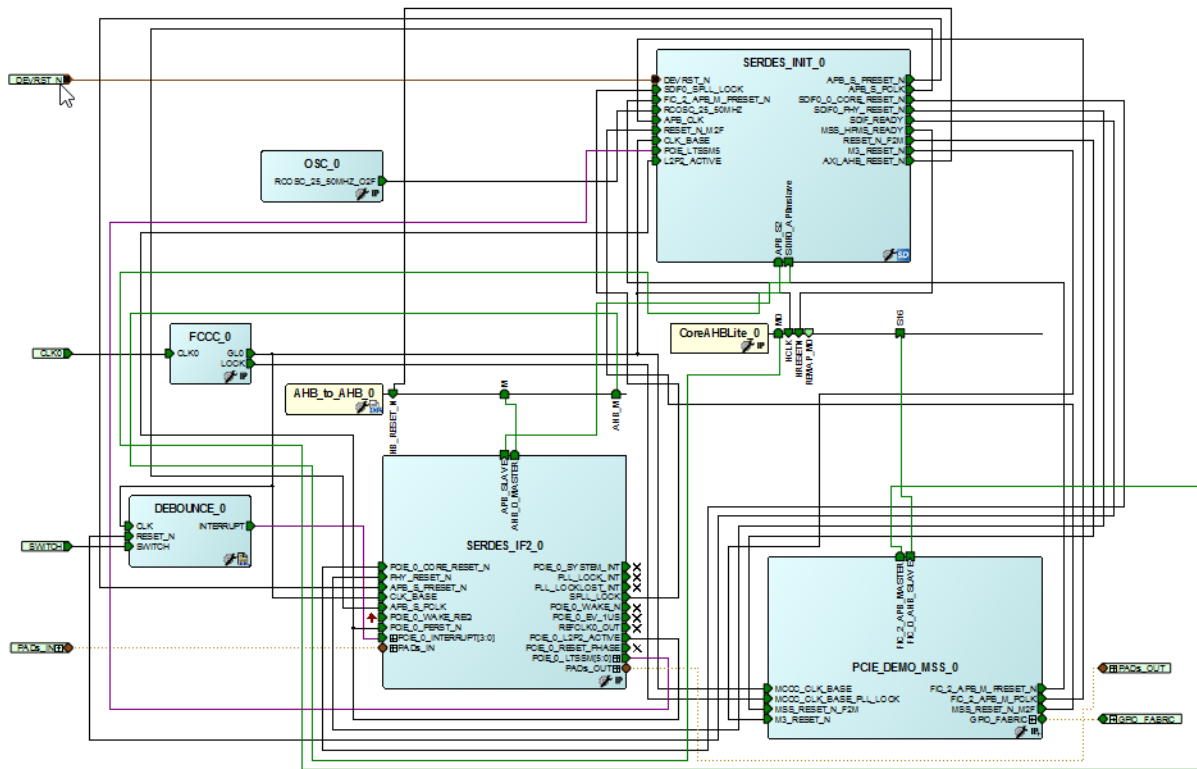
Figure 17 • M2S060 / M2S090 / M2GL060 / M2GL090 Devices PCIe Reset Generation without PERSTn



2.7.4.1 Implementation using M2S090 Security Evaluation Kit

The PCIe control plane demo design for the 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 are shown in the following figure.

Figure 18 • Top-Level Design



2.7.5 M2S/M2GL 060/090 Device Dual PCIe with PERSTn

The following figure shows the implementation of the PCIe reset sequence for M2S/M2GL 060/090 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 a soft reset. Also, 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 Cortex-M3 or utilizing the SmartDebug SERDES utility available for both SmartFusion2 and IGLOO2 devices.

The CoreABC takes few APB cycles to reset the SERDES core and AXI IF. During these cycles, the SERDES AXI master initiates buffered AXI/AHB transactions and the fabric AXI/AHB slave logic should indicate the busy state to AXI master. In this application note, an AXI_AHB reset is generated to AHB_to_AHB.v logic (which asserts the HREADY to '0' during reset) during the reset operation of SERDES Core/AXI IF.

2.7.5.1 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.

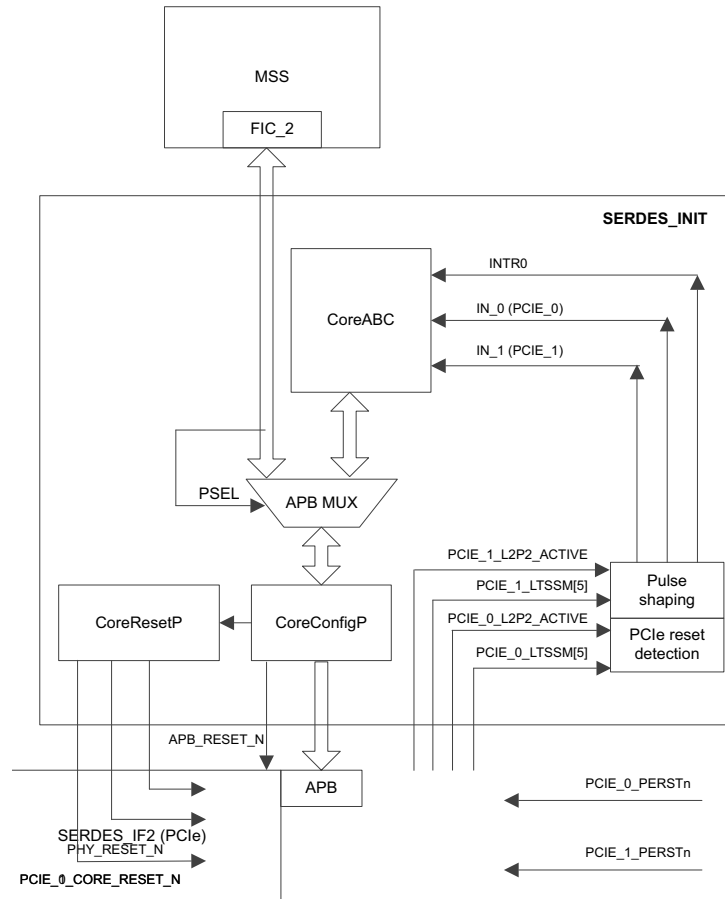
Generate hot_reset_n as: $\text{hot_reset_n} = (!\text{PCIE_x_L2P2_ACTIVE}) \& (!\text{PCIE_x_LTSSM}[5])$.

Connect the SERDES PCIe PERST_N to PERSTn on the board.

2.7.5.2 SERDES (PCIe) Reset Generation

Use `hot_reset_n` to generate interrupt `INTR0` to CoreABC. The pulse shaping logic generates `INTR0` synchronized to the CoreABC clock domain. The CoreABC interrupt routine resets the SERDES PCIe controller0/controller1 reset, and AXI interface reset using the soft reset register.

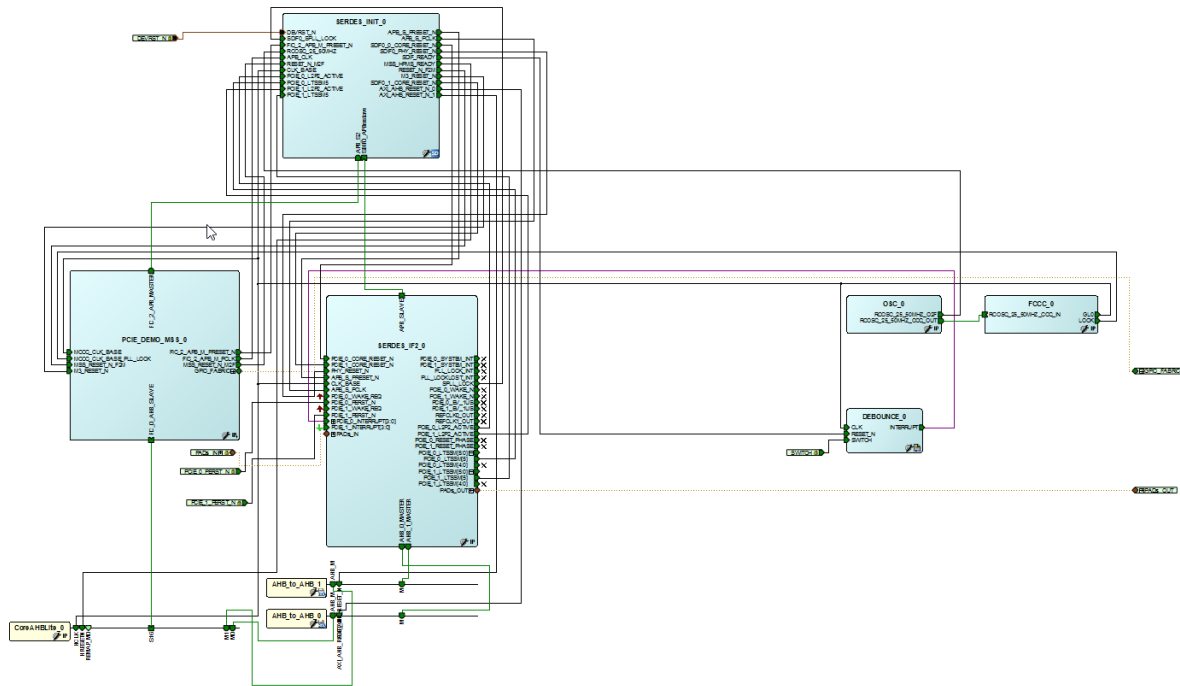
Figure 19 • M2S/M2GL 060/090 Device PCIe Reset Generation with PERSTn



2.7.5.3 Implementation Using M2S090 FG676 Device

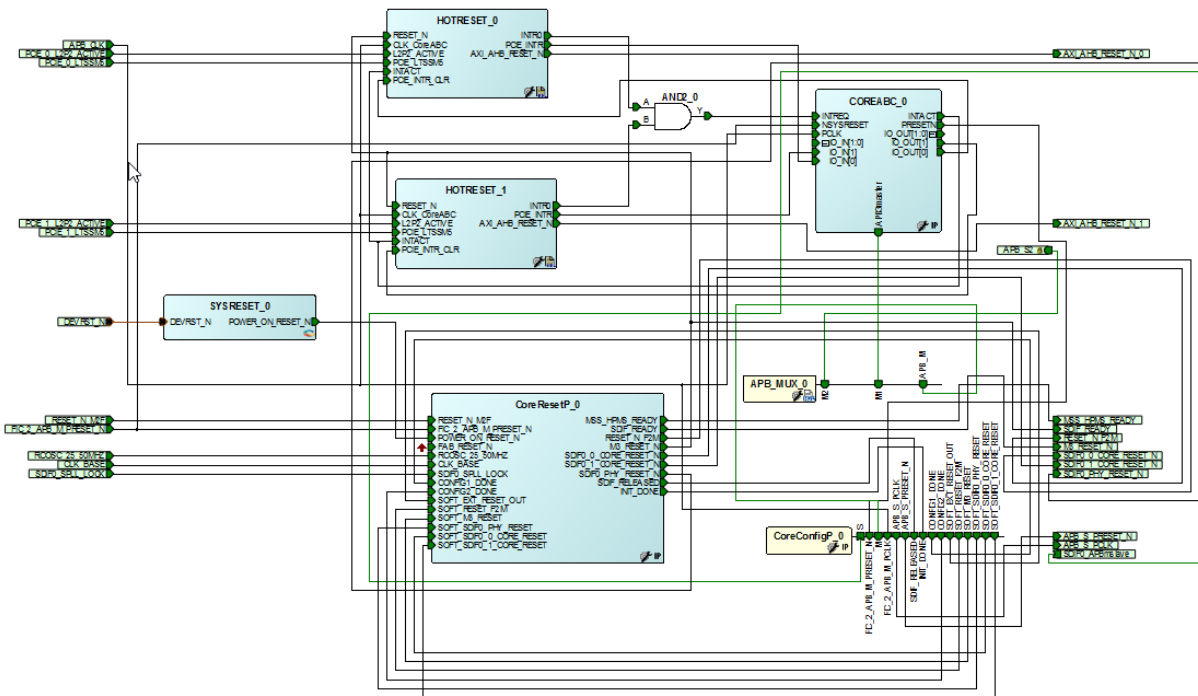
The PCIe control plane demo design for M2S090 is created using CoreABC standalone peripheral initialization method, and HOTRESET logic is implemented to detect the PCIe reset. The top-level SmartDesign with the SERDES_INIT and SERDES_IF blocks are shown in the following figure. SERDES_INIT is a SmartDesign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



The SERDES_INIT SmartDesign has CoreABC, CoreResetP, CoreConfigP, APB_MUX, and HOTRESET blocks as shown in the following figure. 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.

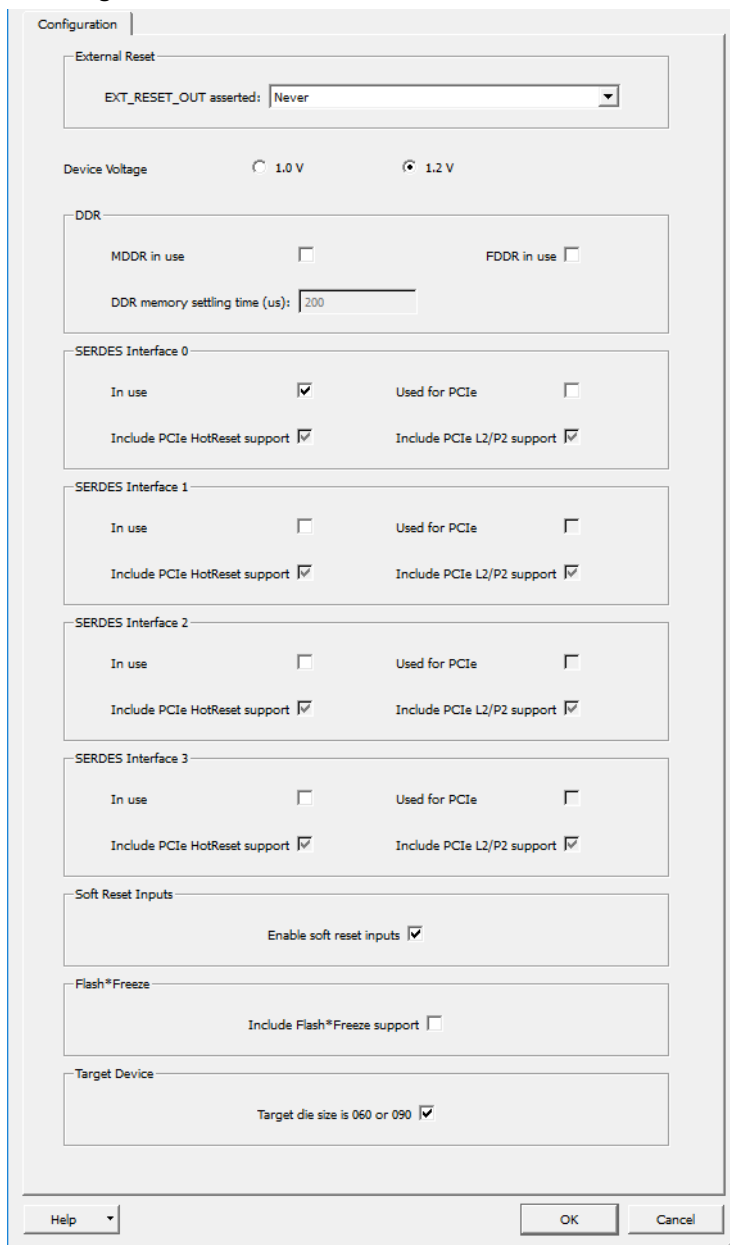
Figure 21 • SERDES_INIT SmartDesign



CoreResetP is configured to generate resets for SERDES_IF2_0, as shown in the following figure.

Select only the SERDES interface check box. Do not select the other PCIe check boxes as those functionalities are already implemented in the HOTRESET block.

Figure 22 • CoreResetP Configuration

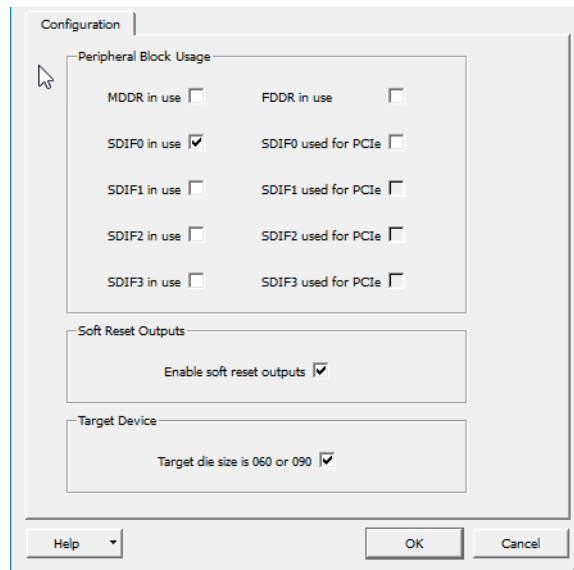


The figure shows the CoreResetP Configuration dialog box. The 'Configuration' tab is selected. The 'External Reset' section has a dropdown menu for 'EXT_RESET_OUT asserted:' set to 'Never'. The 'Device Voltage' section has two radio buttons: '1.0 V' and '1.2 V', with '1.2 V' selected. The 'DDR' section has two checkboxes: 'MDDR in use' and 'FDDR in use', both unchecked. Below them is a text field for 'DDR memory settling time (us):' with the value '200'. The 'SERDES Interface 0' section has four checkboxes: 'In use' (checked), 'Used for PCIe' (unchecked), 'Include PCIe HotReset support' (checked), and 'Include PCIe L2/P2 support' (checked). The 'SERDES Interface 1' section has the same four checkboxes, with 'In use' and 'Used for PCIe' unchecked, and the other two checked. The 'SERDES Interface 2' section has the same four checkboxes, with 'In use' and 'Used for PCIe' unchecked, and the other two checked. The 'SERDES Interface 3' section has the same four checkboxes, with 'In use' and 'Used for PCIe' unchecked, and the other two checked. The 'Soft Reset Inputs' section has a checkbox 'Enable soft reset inputs' which is checked. The 'Flash*Freeze' section has a checkbox 'Include Flash*Freeze support' which is unchecked. The 'Target Device' section has a checkbox 'Target die size is 060 or 090' which is checked. At the bottom, there are 'Help', 'OK', and 'Cancel' buttons.

CoreConfigP is configured for SERDES_IF_0 block, as shown in the following figure.

Select only the SERDES interface check box. Do not select the other PCIe check boxes as those functionalities are already implemented in the HOTRESET block.

Figure 23 • CoreConfigP Configuration



The image shows a 'Configuration' dialog box for CoreConfigP. It has a 'Peripheral Block Usage' section with a table of checkboxes. The 'SDIF0 in use' checkbox is checked. Below this is a 'Soft Reset Outputs' section with 'Enable soft reset outputs' checked. At the bottom is a 'Target Device' section with 'Target die size is 060 or 090' checked. The dialog has 'Help', 'OK', and 'Cancel' buttons at the bottom.

Peripheral Block Usage	
MDDR in use ☐	FDDR in use ☐
SDIF0 in use ☒	SDIF0 used for PCIe ☐
SDIF1 in use ☐	SDIF1 used for PCIe ☐
SDIF2 in use ☐	SDIF2 used for PCIe ☐
SDIF3 in use ☐	SDIF3 used for PCIe ☐

Soft Reset Outputs

Enable soft reset outputs ☒

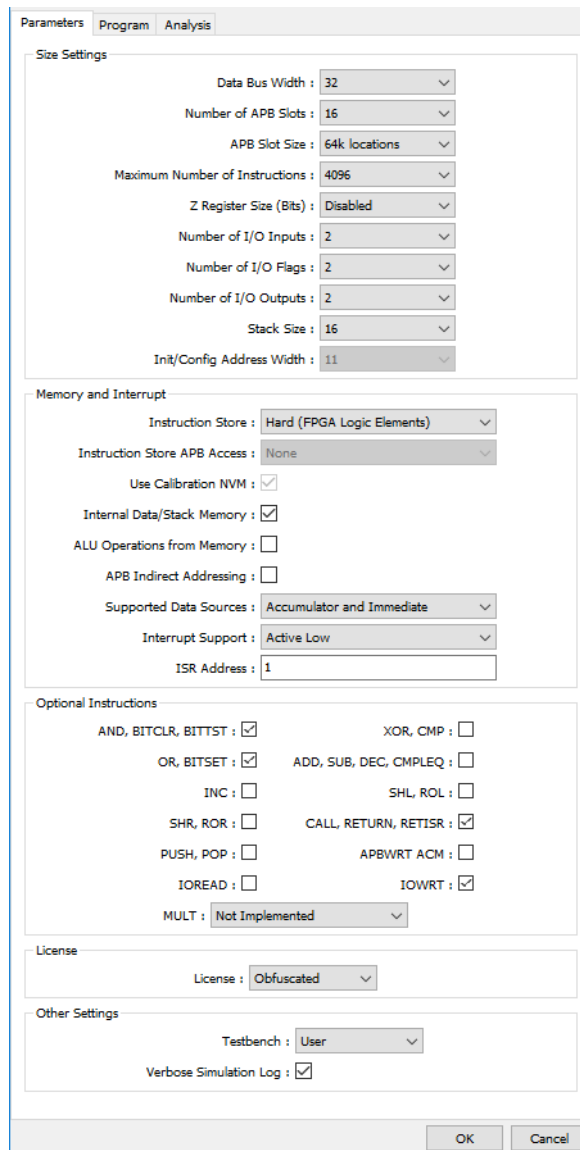
Target Device

Target die size is 060 or 090 ☒

Help OK Cancel

CoreABC configuration is shown in the following figure. Libero generates the `SERDESIF_0_init_abc.txt` file at `<proj_location>\Libero_Project\component\work\top\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



The figure shows the CoreABC Configuration dialog box with the following settings:

- Parameters** tab is selected.
- Size Settings:**
 - Data Bus Width: 32
 - Number of APB Slots: 16
 - APB Slot Size: 64k locations
 - Maximum Number of Instructions: 4096
 - Z Register Size (Bits): Disabled
 - Number of I/O Inputs: 2
 - Number of I/O Flags: 2
 - Number of I/O Outputs: 2
 - Stack Size: 16
 - Init/Config Address Width: 11
- Memory and Interrupt:**
 - Instruction Store: Hard (FPGA Logic Elements)
 - Instruction Store APB Access: None
 - Use Calibration NVM: ☒
 - Internal Data/Stack Memory: ☒
 - ALU Operations from Memory: ☐
 - APB Indirect Addressing: ☐
 - Supported Data Sources: Accumulator and Immediate
 - Interrupt Support: Active Low
 - ISR Address: 1
- Optional Instructions:**
 - AND, BITCLR, BITTST: ☒
 - XOR, CMP: ☐
 - OR, BITSET: ☒
 - ADD, SUB, DEC, CMPEQ: ☐
 - INC: ☐
 - SHL, ROL: ☐
 - SHR, ROR: ☐
 - CALL, RETURN, RETISR: ☒
 - PUSH, POP: ☐
 - APBWRT ACM: ☐
 - IOREAD: ☐
 - IOWRT: ☒
 - MULT: Not Implemented
- License:**
 - License: Obfuscated
- Other Settings:**
 - Testbench: User
 - Verbose Simulation Log: ☒

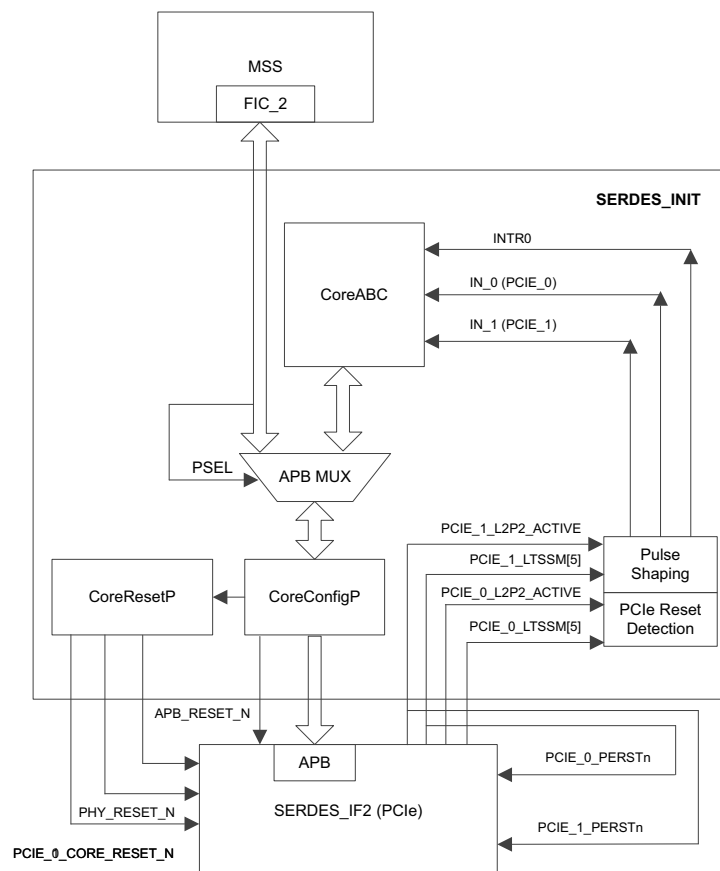
Buttons: OK, Cancel

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

The following figure shows the implementation of the PCIe reset sequence for the M2S/M2GL 060/090 device dual PCIe without PERSTn. This implementation is similar to the PCIe reset sequence implementation for [M2S/M2GL 060/090 Device Dual PCIe with PERSTn](#), page 23 except that the SERDES_IF_2 PERST_N signal is connected to the SERDES_IF_2 L2P2_ACTIVE instead of the PERSTn signal.

The SERDES_IF_2_L2P2_ACTIVE signal gets asserted to '1' when the host/root port initiates L2 state. This assertion causes the EndPoint reset through PERST_N to exit from L2 state.

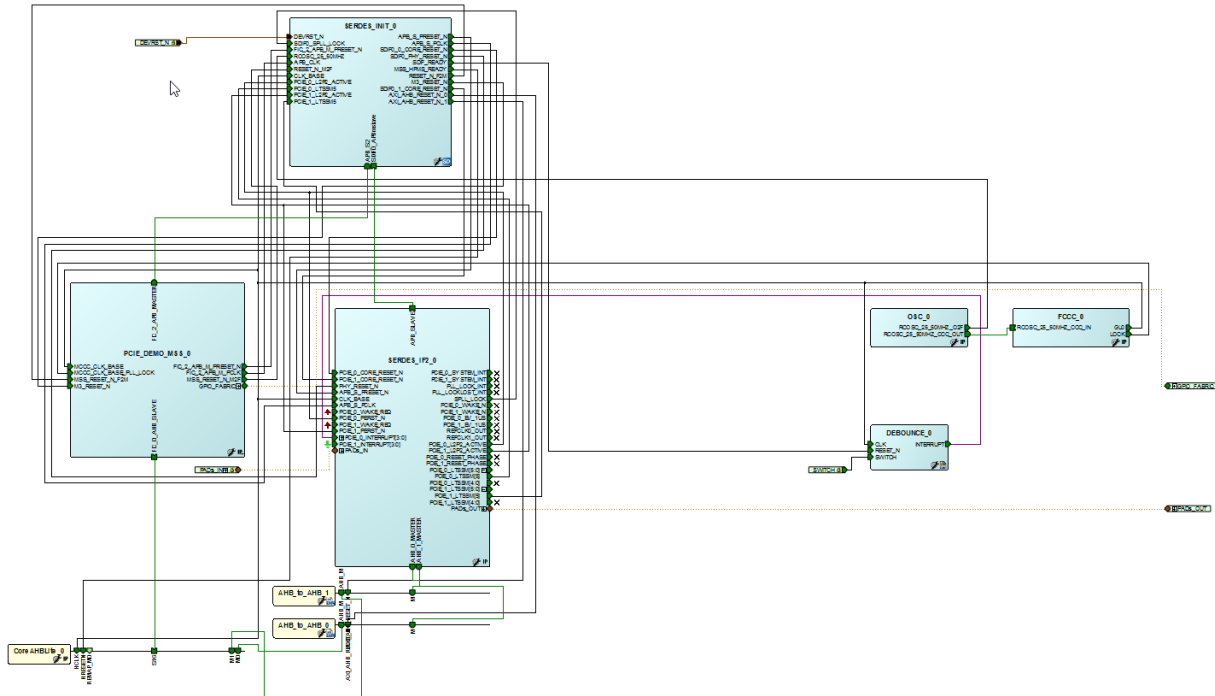
Figure 25 • M2S/M2GL 060/090 Device Dual PCIe without PERSTn



2.7.6.1 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 are shown in the following figure.

Figure 26 • Top-Level Design



M2S060/M2S090/M2GL060/M2GL090 devices also support configuring the SERDES in triple mode (two PCIe endpoint controllers and one EPCS controller).

The possible configurations are:

1. 1. PCIeX1 on L0 + PCIeX1 on L1 + EPCSX2 on L2 and L3
2. 2. PCIeX1 on L0 + PCIeX1 on L1 + EPCSX1 on L2
3. 3. PCIeX1 on L0 + PCIeX1 on L1 + EPCSX1 on L3

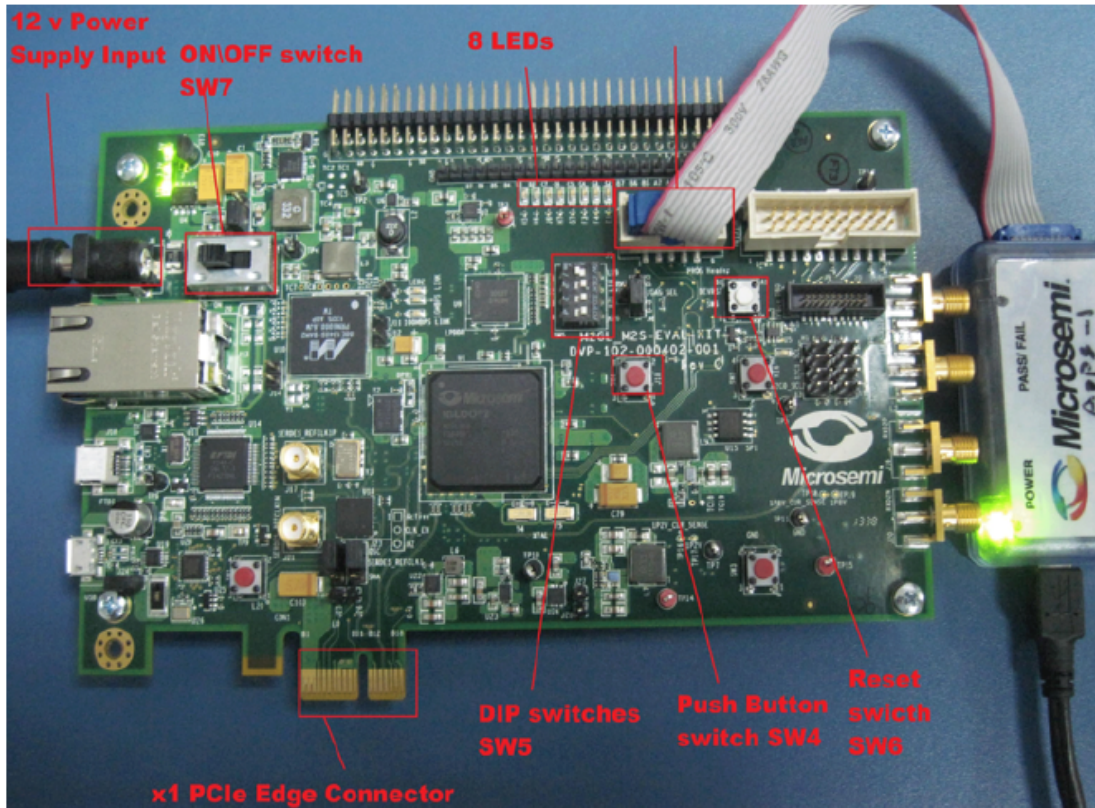
The PCIe reset sequence for the third configuration is similar to the PCIe reset sequence implementation for M2S/M2GL 060/090 device dual PCIe with/without PERSTn. It is not possible to implement the PCIe reset sequence for the first and second configurations as the `pcie_1_l2p2_active` signal is overlaid by the `epcs_rxclk_2` signal.

2.8 Running the Design

Program the SmartFusion2 Security Evaluation kit board or IGLOO2 Evaluation kit board with the job file provided as part of the design files using FlashPro Express software, refer to [Appendix: Programming the Device Using FlashPro Express](#), page 34.

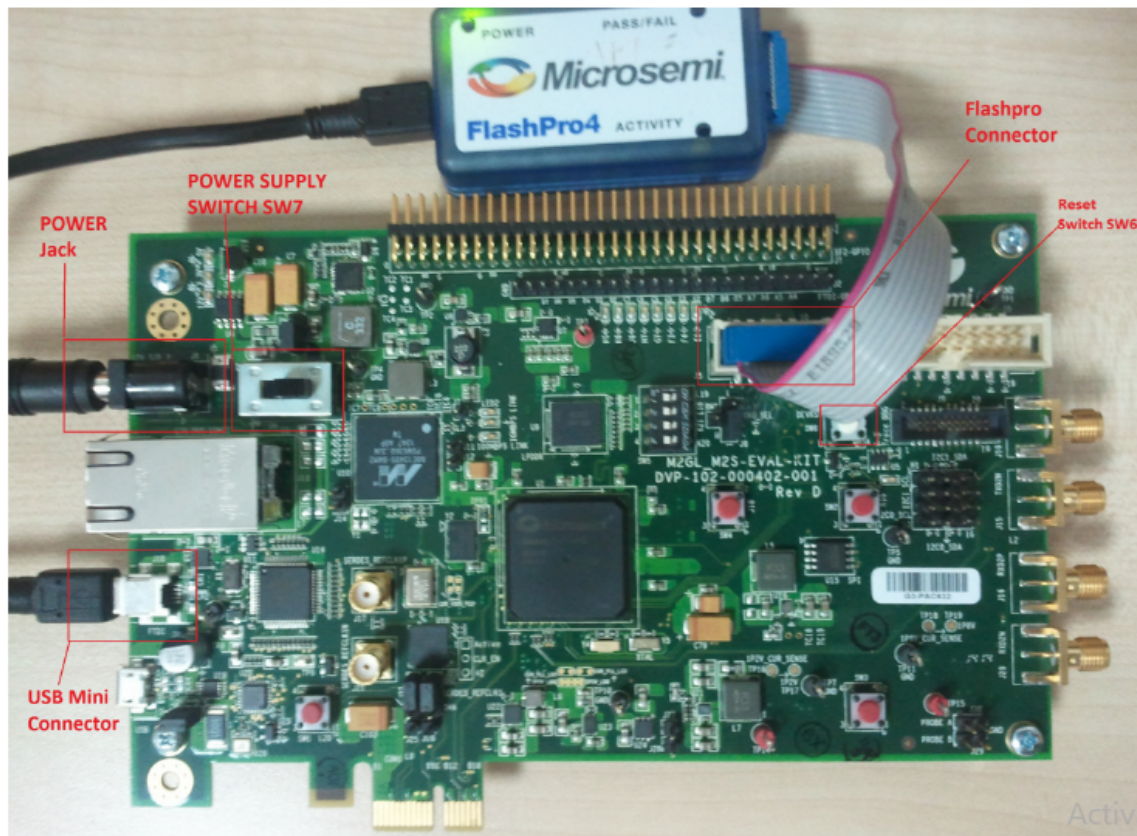
The IGLOO2 Evaluation kit board is shown in the following figure.

Figure 27 • IGLOO2 Evaluation Kit Board



The SmartFusion2 Security Evaluation kit board is shown in the following figure.

Figure 28 • SmartFusion2 Security Evaluation Kit Board



2.8.1 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.

Note: Laptop PCIe adapter cards do not have PERSTn signal.

Note: 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 a specific motherboard configuration. If this occurs, the device needs to be reset to be detected.

2.9 Conclusion

This application note describes the recommended implementation of the 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.

3 Appendix: Programming the Device Using FlashPro Express

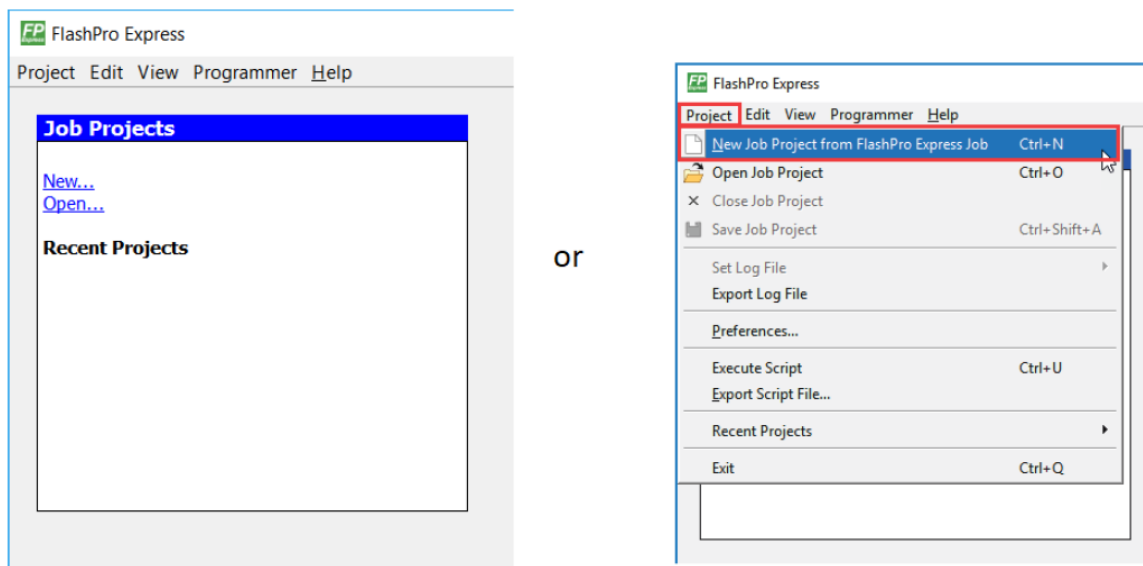
This section describes how to program the SmartFusion2 and IGLOO2 devices with the programming job file using FlashPro Express.

The following steps describe how to program the FlashPro Express:

1. Switch **ON** the power supply switch **SW7**.
2. Launch the FlashPro Express software.
3. Click **New Project**.
4. In the **New Project** window, type the project name as:
 - IGLOO2: IGL2_MDDR_Demo
 - SmartFusion2: SF2_MDDR_Demo
5. Click **Browse** and navigate to the location where you want to save the project.
6. Select **Single device** as the **Programming mode**.
7. Click **OK**.

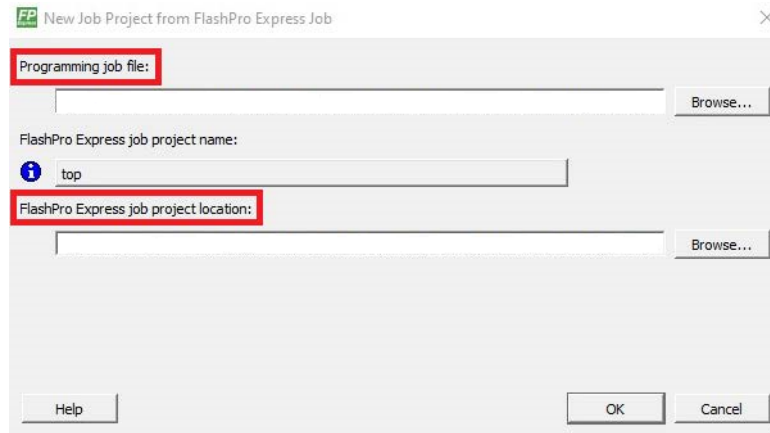
The FlashPro new project is shown in the following figure.

Figure 29 • FlashPro Express Job Project



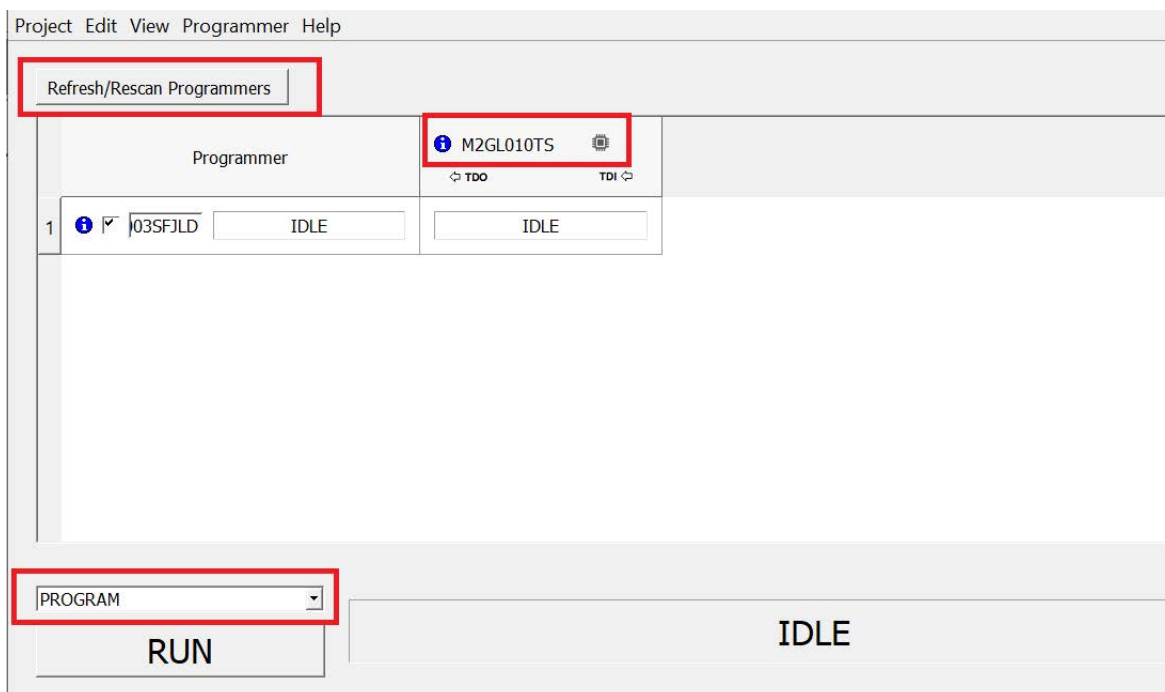
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:
 - **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:
`<download_folder>\m2s_m2gl_ac437_df\Programming_Job`
 - **FlashPro Express job project name:** Click **Browse** and navigate to the location where you want to save the project.

Figure 30 • New Job Project from FlashPro Express Job

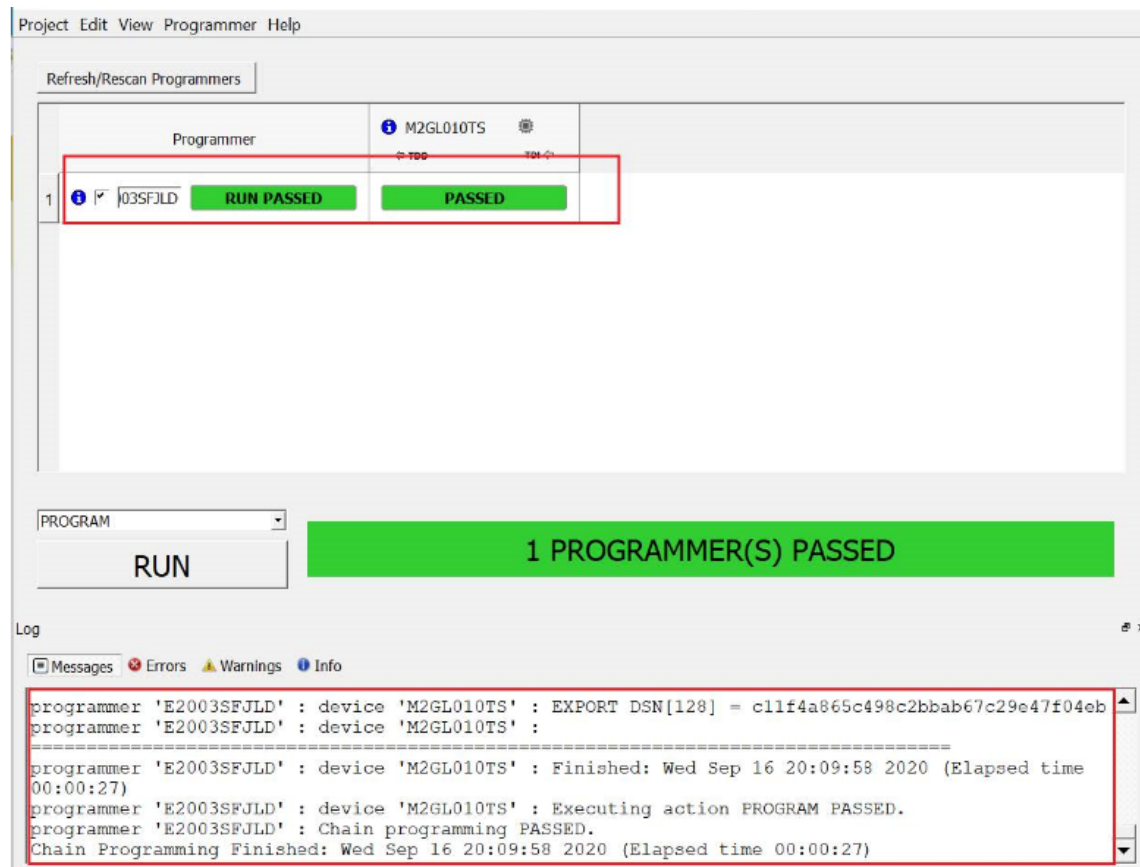


9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan** Programmers.

Figure 31 • Programming the Device



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 32 • FlashPro Express—RUN PASSED

12. Close **FlashPro Express** or in the Project tab, click **Exit**.