IGLOO2 Standalone Peripheral Initialization Methodology User Guide

Introduction (Ask a Question)

When creating a design using an IGLOO2 device, if you use any of the two DDR controllers (FDDR or MDDR) or Serial High speed controller (SERDESIF) blocks, you must initialize the configuration registers of these blocks at run-time before they can be used. For example, for the DDR controller, you must set the DDR mode (DDR3/DDR2/LPDDR), PHY width, burst mode and ECC. Similarly, for the SERDESIF block used as a PCIe endpoint, you must set the PCIE BAR to AXI (or AHB) window.

In this document, we describe all the steps necessary to create a Libero design that automatically initializes the DDR controller and SERDESIF blocks at power up, with the Standalone Initialization mode ON.

First we provide a detailed description of the theory of operation. We introduce the major components of the Peripheral Initialization Solution and outline how they interact.

Unlike the normal flow (Standalone Initialization OFF) where the initialization solution is created by the System Builder, in the case of Standalone Initialization mode ON, the initialization solution has to be put together in SmartDesign using different soft IP cores (mentioned in the latter sections), whether you choose to use System Builder or not. System Builder will not create any initialization logic for any of the peripherals. You have to build the initialization logic that sits outside the System Builder block, should you choose to use System Builder at all.

Note that as the name suggests, the standalone initialization logic has to be built separately for each of the peripherals (DDR/SERDES) used.

Next, we describe how to build designs with the Standalone Initialization mode ON in cases where you choose to use System Builder and in cases where you choose not to.

In this section we address:

- The creation of the configuration data for DDR controller and SERDESIF configuration registers
- The creation of the FPGA logic required to transfer the configuration data to the different ASIC configuration registers

For complete details about the DDR controller and SERDESIF configuration registers please refer to the UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide and UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide.

Table of Contents

1. Theory of Operation (Ask a Question)

The Standalone Peripheral Initialization solution for each peripheral uses the following major components:

- The CoreABC soft IP core, which has to be loaded with a program to initialize the peripheral's configuration registers, so that it orchestrates the initialization process. The program contains the registers specific to a peripheral that's being initialized.
- The CoreConfigP soft IP core, whose function is to initialize the peripherals' configuration registers.
- The CoreResetP soft IP core, whose function is to manage the reset sequence of the HPMS, DDR controllers, and SERDESIF blocks.

One set of these 3 soft IP cores is dedicated to initialize a single peripheral, and similar logic involving these cores should be built separately for each peripheral used in the design.

The peripheral initialization process works as follows:

1. Upon reset, the CoreABC runs the program it is loaded with.
2. The program starts writing to the registers of the peripheral being initialized. If the peripheral is MDDR/FDDR, then the program writes configuration data to the DDR controllers, and if the peripheral is SERDES, then the program writes the SERDESIF configuration registers, via the CoreABC master BIF. This interface is connected to the soft CoreConfigP core instantiated in the FPGA fabric.
3. After all the registers are configured, the CoreABC program writes to the CoreConfigP control registers to indicate the completion of the register configuration phase; the CoreConfigP output signal CONFIG1_DONE and CONIG2_DONE are then asserted.
   There are two phases of register configuration (CONFIG1 and CONFIG2) depending upon the peripherals used in the design.
4. If the peripheral being initialized is DDR (FDDR/MDDR), then both the signals CONFIG1_DONE and CONFIG2_DONE are asserted at the same time.
5. If the peripheral being initialized is SERDESIF, then there are 2 phases of register configuration depending upon whether SERDES is configured in PCIE mode or not.
   - CONFIG1_DONE is asserted after the first phase of register configuration is complete. SERDESIF system and lane registers are configured in this phase. If SERDES is configured in a non-PCIE mode, then CONFIG2_DONE signal is also asserted immediately.
   - The second phase of register configuration then follows (if SERDESIF is configured in PCIE mode). The following are the different events that happen in this second phase:
     - Once CoreResetP de-asserts PHY_RESET_N and CORE_RESET_N signals of the SERDESIF blocks, it also asserts an output signal SDIF_RELEASED.
     - Once the SDIF_RELEASED signal is asserted, the CoreABC program starts polling for the assertion of PMA_READY on the appropriate SERDESIF lane. Once the PMA_READY is asserted, the second set of SERDESIF registers (PCIE registers) are configured/written by the CoreABC program.
     - After all the PCIE registers are configured, the CoreABC program writes to the CoreConfigP control registers to indicate the completion of the second phase of register configuration; the CoreConfigP output signal CONIG2_DONE is then asserted.
6. Apart from the above signal assertions/de-assertions, CoreResetP also manages the initialization of the peripheral being initialized by performing the following functions (depending upon the peripheral being initialized):
   - De-asserting the MDDR/FDDR core reset

- De-asserting the SERDESIF blocks PHY and CORE resets
- Monitoring of the FDDR PLL (FPLL) lock signal. The FPLL must be locked to guarantee that the FDDR AXI/AHBLite data interface and the FPGA fabric can communicate correctly.
- Monitoring of the SERDESIF block PLL (SPLL) lock signals. The SPLL must have locked to guarantee that the SERDESIF blocks AXI/AHBLite interface (PCIe mode) or XAUI interface can communicate properly with the FPGA fabric.
- Waiting for the external DDR memories to settle and be ready to be accessed by the DDR controllers.

7. When the peripheral is initialized and is ready to communicate, CoreResetP asserts the INIT_DONE signal; the CoreConfigP internal register INIT_DONE is then asserted.
   - If the peripheral is MDDR/FDDR, and the DDR initialization time is reached, CoreResetP output signal DDR_READY is asserted. Assertion of this signal DDR_READY can be monitored as an indication that the DDR (MDDR/FDDR) is ready for communication.
   - If the peripheral is SERDESIF, and the second phase of register configuration is successfully completed, CoreResetP output signal SDIF_READY is asserted. Assertion of this signal SDIF_READY can be monitored as an indication that this SERDESIF block is ready for communication.
8. The CoreABC program which has been waiting for INIT_DONE to be asserted completes its execution now.

Note: In case of an IGLOO2 design with fabric logic (say fabric master) waiting to communicate with the peripheral, it should wait for the assertion of INIT_DONE (OR DDR_READY/SDIF_READY based on which peripheral is being used) signal of the CoreResetP instance (that belongs to the initialization logic of the peripheral) before it attempts to communicate with the peripheral.

The methodology described in this document relies on the CoreABC executing the initialization process as part of its program (microcode). All the initialization logic is taken care by the CoreABC program and the soft IP cores CoreConfigP and CoreResetP.

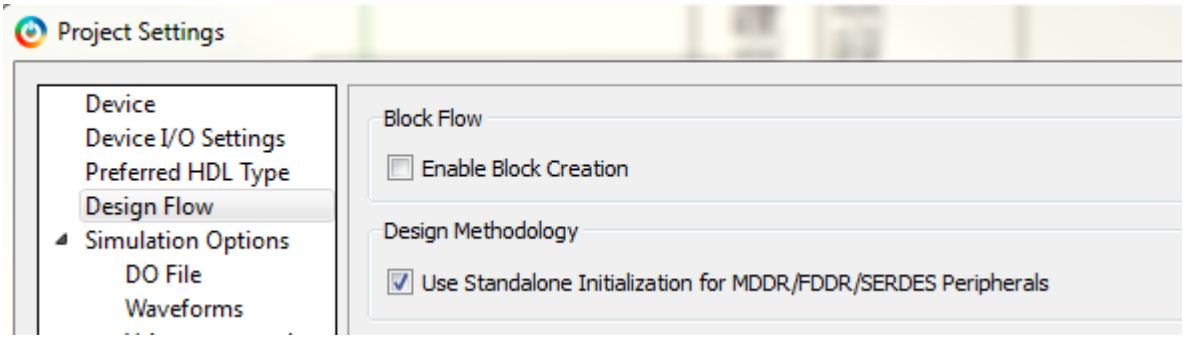
2. Switching the Standalone Initialization Mode ON (Ask a Question)

You can turn the Standalone Initialization mode ON when you first create a project for IGLOO2 in the Design Methodology section in the New Project dialog as shown in figure below.

Figure 2-1. Design Methodology – Use Standalone Initialization for MDDR/FDDR/SERDES

If you already have your project open you can turn the Standalone Initialization mode ON from the Project Settings Design Flow window as shown in figure below.

Figure 2-2. Standalone Initialization from Project Settings window



3. Using System Builder to Create a Design Using DDR blocks (Ask a Question)

The IGLOO2 System Builder is a powerful design tool that helps you capture your system-level requirements and produces a design implementing those requirements. With the Standalone Initialization mode ON, if you are building a design using FDDR, you can choose to use System Builder which automatically instantiates and configures the FDDR block. Alternatively, without using System Builder also, you can just instantiate and configure the FDDR block manually to build your design. If you want to build a design using the MDDR block, then you must use System Builder. In any case, the peripheral initialization logic using CoreABC, CoreConfigP and CoreResetP has to be built manually for every peripheral you use. If you are building a design using SERDESIF and fabric logic only (that means if you don't want to use anything else in the HPMS), you don't have to use System Builder at all. Build everything using regular Smart Design. In Using SmartDesign To Create A Design Using SERDESIF And FDDR Blocks, we describe in detail how to create such a solution without the System Builder.

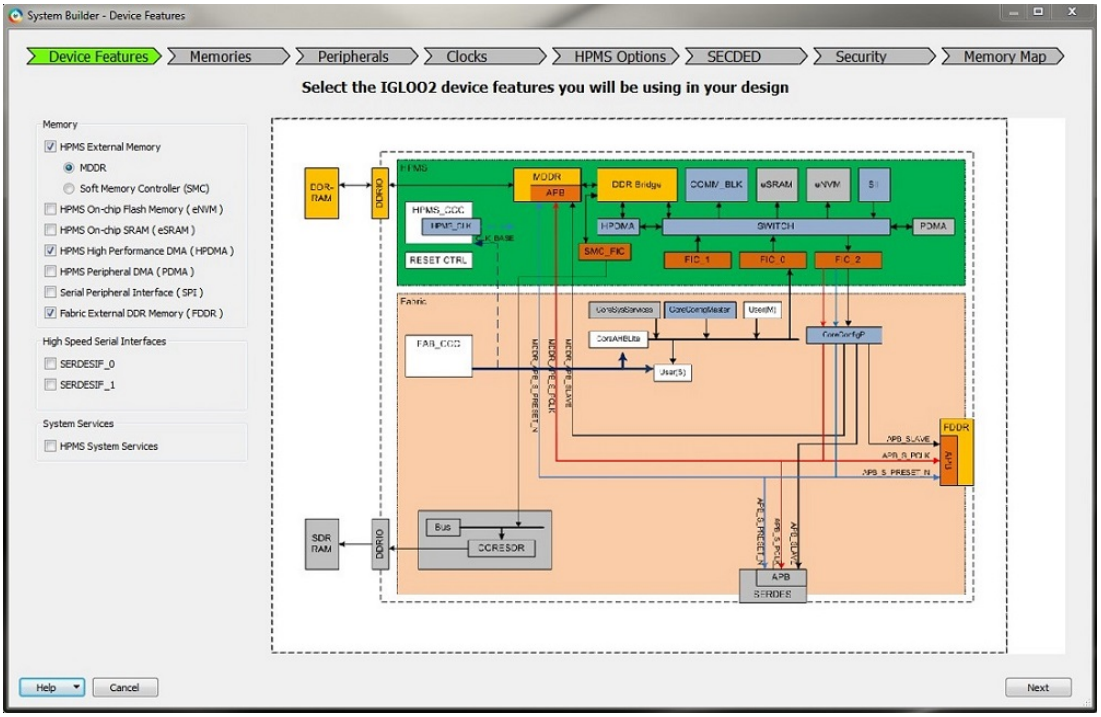
If you are using System Builder, you must perform the following tasks to create a design that will instantiate and configure your DDR blocks (MDDR/FDDR), and then create and interface the initialization logic required to initialize the DDR blocks (MDDR/FDDR).

1. In the **Device Features** page, specify which DDR controllers are used in your design.
2. In the **Memory** page, specify the type of DDR (DDR2/DDR3/LPDDR) and the configuration data for your external DDR memories. See the Memory Page section for details.
3. In the **Peripherals** page, add fabric masters configured as AHBLite/AXI to the Fabric DDR Subsystem and/or HPMS DDR FIC Subsystem (optional).
4. In the **Clock Settings** page, specify the clock frequencies for the DDR sub-systems, and configure the Chip Oscillator and Fabric CCC resources required to drive the fabric logic outside the System Builder block.
5. Complete your design specification and click Finish. System Builder will then build the design instantiating and configuring the HPMS(MDDR)/FDDR blocks.
6. Build CoreABC based standalone initialization logic required to initialize the DDR blocks (MDDR/FDDR).
7. Interface the initialization logic with the System Builder block (which has MDDR/FDDR), and continue with the design flow.

3.1. System Builder Device Features Page (Ask a Question)

In the Device Features page, specify which DDR controllers (MDDR and/or FDDR) are used in your design. You can also choose to use HPDMA for memory transfers between MDDR and eNVM / eSRAM / fabric logic via FIC0/1, in this page.

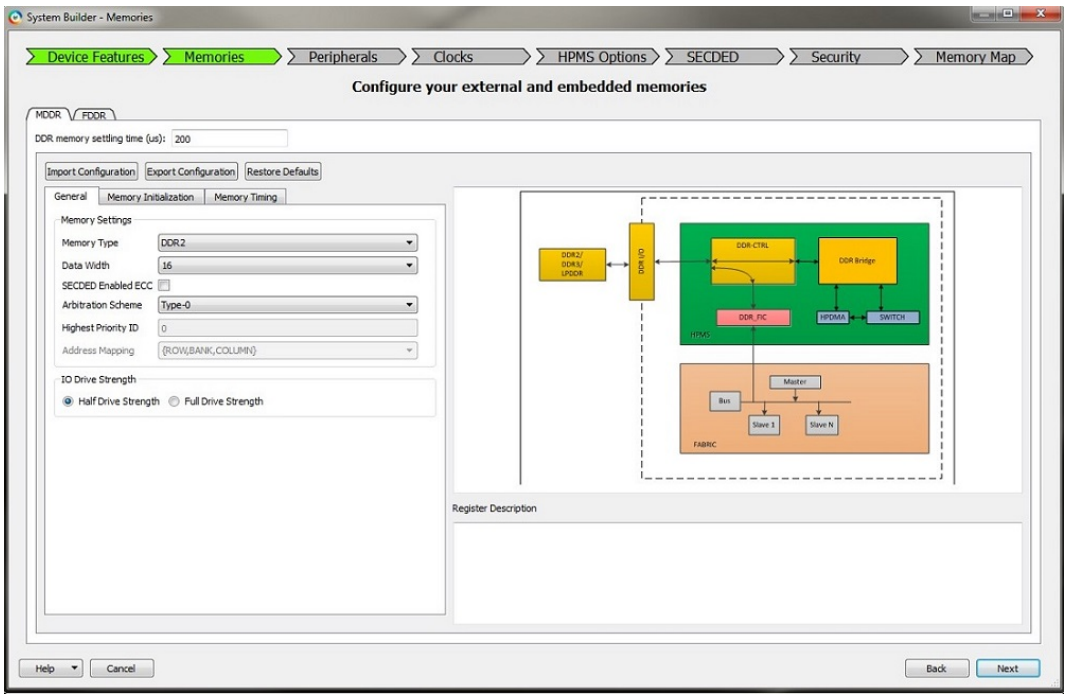
Figure 3-1. System Builder Device Features Page



3.2. System Builder Memory Page (Ask a Question)

To use the HPMS DDR (MDDR) or Fabric DDR (FDDR), select the Memory Type from the dropdown list.

Figure 3-2. HPMS External Memory



You must:

1. Select the Memory type (**DDR2**, **DDR3** or **LPDDR**).
2. Define the **DDR memory settling time**. Consult your external DDR Memory Specifications to set the correct memory setting time. The DDR memory may fail to initialize correctly if the memory settling time is not correctly set.
3. Either import the DDR register configuration data or set your DDR Memory Parameters. For details, consult the UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide.

This data is used to generate the CoreABC program files corresponding to the DDR registers being configured. For complete details on DDR controller configuration registers please refer to the UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide.

An example of the configuration file syntax is shown in figure below. The register names used in this file are the same as the ones described in the UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide.

Figure 3-3. Configuration File Syntax Example

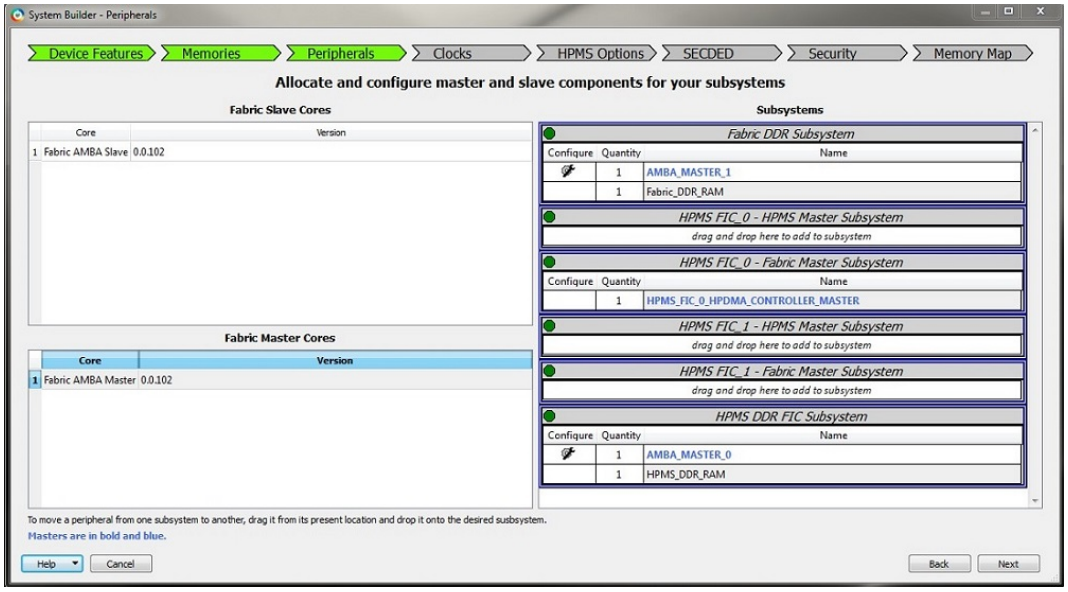
```
## PHY_16_DDR2_NO_ECC_BL8_INTER

ddrc_dyn_soft_reset_CR          0x00 ;
ddrc_dyn_refresh_1_CR           0x27DE ;
ddrc_dyn_refresh_2_CR           0x030F ;
ddrc_dyn_powerdown_CR           0x02 ;
ddrc_dyn_debug_CR               0x00 ;
ddrc_ecc_data_mask_CR           0x0000 ;
ddrc_addr_map_col_1_CR          0x3333 ;
ddrc_addr_map_col_3_CR          0x3300 ;
ddrc_init_1_CR                  0x0001 ;
ddrc_cke_rstn_cycles_CR1        0x0100 ;
ddrc_cke_rstn_cycles_CR2        0x0008 ;
ddrc_init_emr2_CR               0x0000 ;
ddrc_init_emr3_CR               0x0000 ;
ddrc_dram_bank_act_timing_CR    0x1947;
ddrc_odt_param_1_CR             0x0010 ;
ddrc_odt_param_2_CR             0x0000 ;
ddrc_debug_CR                   0x3300 ;
ddrc_mode_reg_rd_wr_CR          0x0000 ;
ddrc_mode_reg_data_CR           0x0000 ;
ddrc_pwr_save_2_CR              0x0000 ;
ddrc_hpr_queue_param_CR1        0x80F8 ;
ddrc_hpr_queue_param_CR2        0x0007 ;
ddrc_lpr_queue_param_CR1        0x80F8 ;
ddrc_lpr_queue_param_CR2        0x0007 ;
ddrc_wr_queue_param_CR          0x0200 ;
ddrc_dfi_min_ctrlupd_timing_CR  0x0003 ;
ddrc_dfi_max_ctrlupd_timing_CR  0x0040 ;
ddrc_dfi_wr_lvl_control_CR1     0x0000 ;
ddrc_dfi_wr_lvl_control_CR2     0x0000 ;
ddrc_dfi_rd_lvl_control_CR1     0x0000 ;
ddrc_dfi_rd_lvl_control_CR2     0x0000 ;
ddrc_dfi_ctrlupd_time_interval_CR  0x0309 ;
ddrc_perf_param_3_CR            0x0000 ;
ddrc_ecc_int_clr_reg            0x0000 ;
```

3.3. System Builder Peripherals Page (Ask a Question)

In the Peripherals page, for each DDR controller a separate subsystem is created (Fabric DDR Subsystem for FDDR and HPMS DDR FIC Subsystem for MDDR). You can add a Fabric AMBA Master (configured as AXI/AHBLite) core to each of these subsystems to enable fabric master access to the DDR controllers. Upon generation, System Builder automatically instantiates bus cores (depending

on the type of AMBA Master added) and exposes the master BIF of the bus core and the clock and reset pins of the corresponding subsystems (FDDR/MDDR) under appropriate pin groups, to the top. All you have to do is connect the BIFs to the appropriate Fabric Master cores that you would instantiate in the design. In the case of MDDR, it is optional to add a Fabric AMBA Master core to the HPMS DDR FIC Subsystem. Instead you could choose to have HPDMA transfer data between MDDR and eNVM / eSRAM / fabric logic via FIC0/1; for that you must check the HPDMA checkbox in the Device Features page.

Figure 3-4. System Builder Peripherals Page



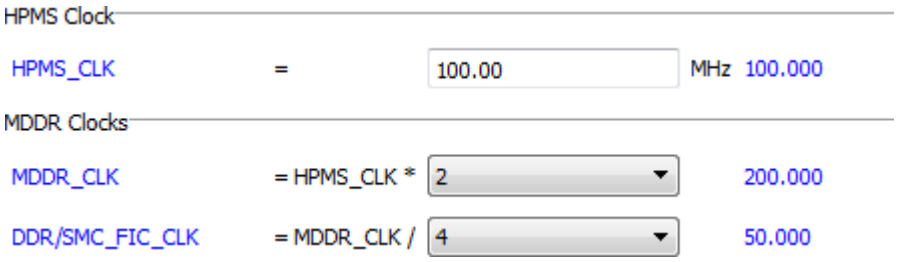
3.4. System Builder Clock Settings Page (Ask a Question)

In the Clock Settings page, for each DDR controller you must specify the clock frequencies related to each DDR (MDDR and/or FDDR) sub-system.

For MDDR, you must specify:

- **MDDR_CLK** - This clock determines the operating frequency of the DDR Controller and should match the clock frequency you wish your external DDR memory to run at. Note that this clock is defined as a multiple of the HPMS_CLK as shown in figure below. The MDDR_CLK must be less than 333 MHz.
- **DDR_FIC_CLK** - If you have chosen to also access the MDDR from the FPGA fabric, you need to specify the DDR_FIC_CLK. This clock frequency is defined as a ratio of the MDDR_CLK and it should match the frequency at which the FPGA fabric sub-system that accesses the MDDR is running.

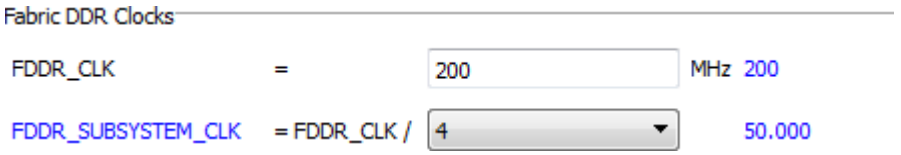
Figure 3-5. HPMS Main Clock And MDDR Clocks



For FDDR you must specify:

- **FDDR_CLK** - Determines the operating frequency of the DDR Controller and should match the clock frequency at which you wish your external DDR memory to run. The FDDR_CLK must be within 20 MHz and 333 MHz.
- **FDDR_SUBSYSTEM_CLK** - This clock frequency is defined as a ratio of the FDDR_CLK and should match the frequency at which the FPGA fabric sub-system that accesses the FDDR is running.

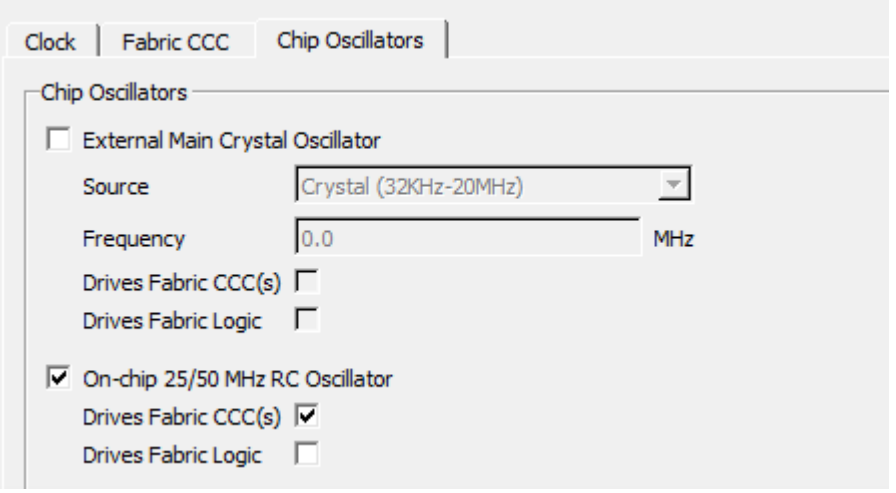
Figure 3-6. Fabric DDR Clocks



3.4.1. Chip Oscillators Tab – Clocks Page (Ask a Question)

In the Chip Oscillators tab of the System Builder Clocks page, check the 'On-chip 25/50 MHz RC Oscillator' and the Drives Fabric CCC(s) checkboxes as shown in figure below. This exposes an output pin RCOSC_25_50MHZ_CCC_OUT on the System Builder block which can be used to drive a fabric CCC. This helps in reusing the oscillator block that's already instantiated inside the System Builder block to drive the CoreResetP cores being used for the peripherals, that sit outside the System Builder block. This is necessary if you are using System Builder because there's only 1 RCOSC per device.

Figure 3-7. Chip Oscillators Tab Of System Builder



3.5. Generating your System Builder design (Ask a Question)

Once you are done configuring all the System Builder pages with your desired settings, click 'Finish' in the last page. The System Builder component is generated to a SmartDesign, with all required top level pins and BIF ports exposed on the System Builder block under appropriate pin groups. Next you need to build the initialization logic for the DDR blocks (MDDR/FDDR) used in your design, interface it to the System Builder block to initialize the DDR blocks, generate and then continue with the design flow.

Upon generating the System Builder component, separate text files containing the CoreABC program corresponding to MDDR and FDDR register configuration are created to the disk under the <project_location>/.../*_HPMS/ directory and the <project_location>/.../FABDDR_0/ directories respectively with the names MDDR_init_abc.txt and FDDR_init_abc.txt. This CoreABC program generated for MDDR/FDDR has to be loaded/copied to the CoreABC instance used for the initialization of the peripheral (MDDR/FDDR). This will be discussed again in the following sections.

3.6. Building Standalone Initialization Logic for MDDR (Ask a Question)

In order to initialize the MDDR, you must create the initialization subsystem in the FPGA fabric. The FPGA fabric initialization subsystem moves data from the CoreABC program to the DDR configuration registers, manages the reset sequences required for the MDDR block to be operational and signals when the MDDR block is ready to communicate with the rest of your design. To create the initialization subsystem you must:

- Instantiate and configure CoreABC soft IP core.
- Load CoreABC with the initialization program generated to the MDDR_init_abc.txt file.
- Instantiate and configure the CoreConfigP and the CoreResetP cores.
- Connect these components to the peripheral's (MDDR) configuration interfaces, clocks, resets and PLL lock ports.

3.6.1. CoreABC configuration (Ask a Question)

1. Create a new SmartDesign component (MDDR_INIT).
2. Instantiate CoreABC into your SmartDesign. This core can be found in the Libero Catalog (under Processors).
3. Double-click the core to open the configurator.
4. Configure the core as shown in the figure below.
   a. Configure the data bus width to be 16.
   b. Configure the maximum number of instructions to at least 256.
   c. Configure to use AND and OR operations as optional instructions. Configure Instruction Store to Hard (FPGA Tiles).
5. Copy the CoreABC program generated for MDDR from the MDDR_init_abc.txt file created under the <project_location>/.../*_HPMS/ directory, to the CoreABC Program tab. See the figure below.

Figure 3-8. CoreABC Configuration

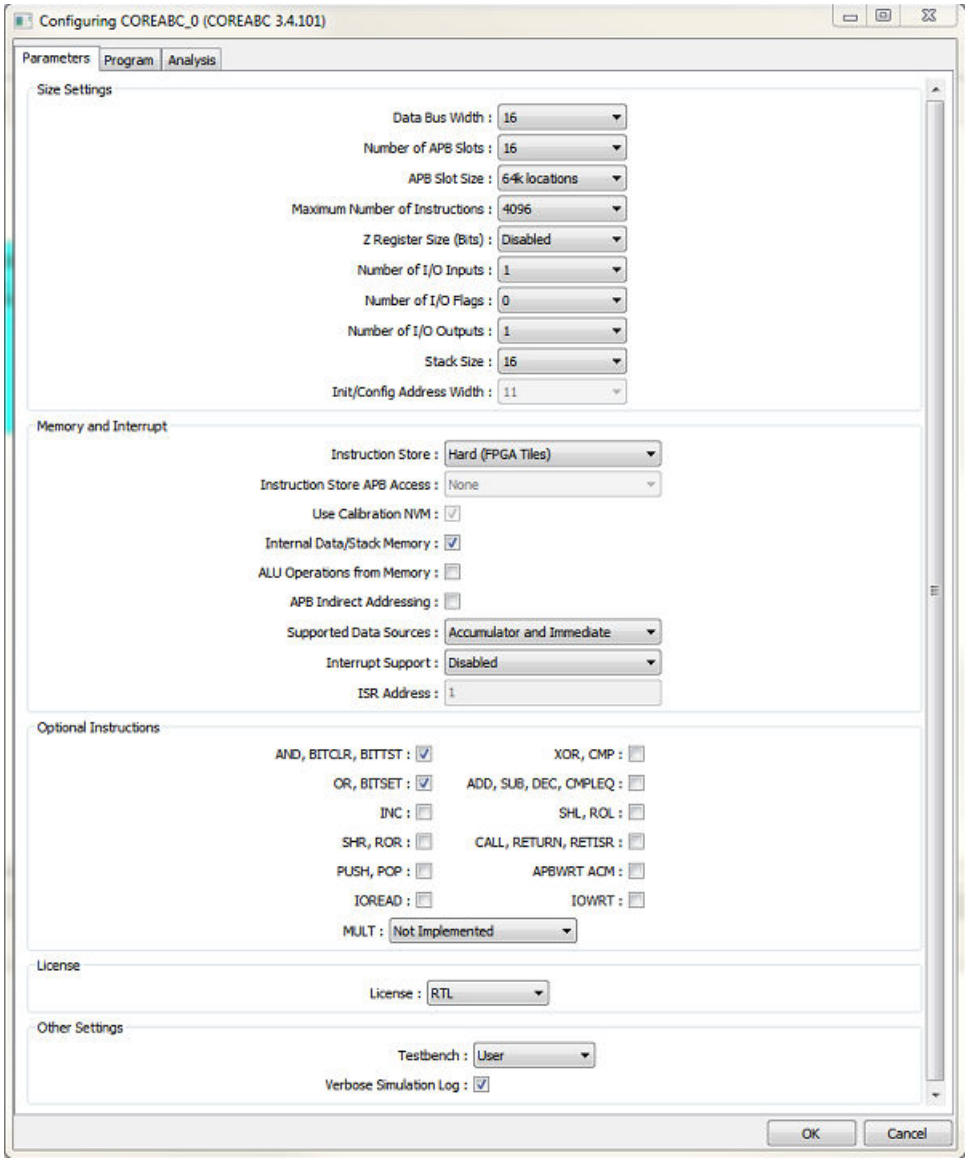
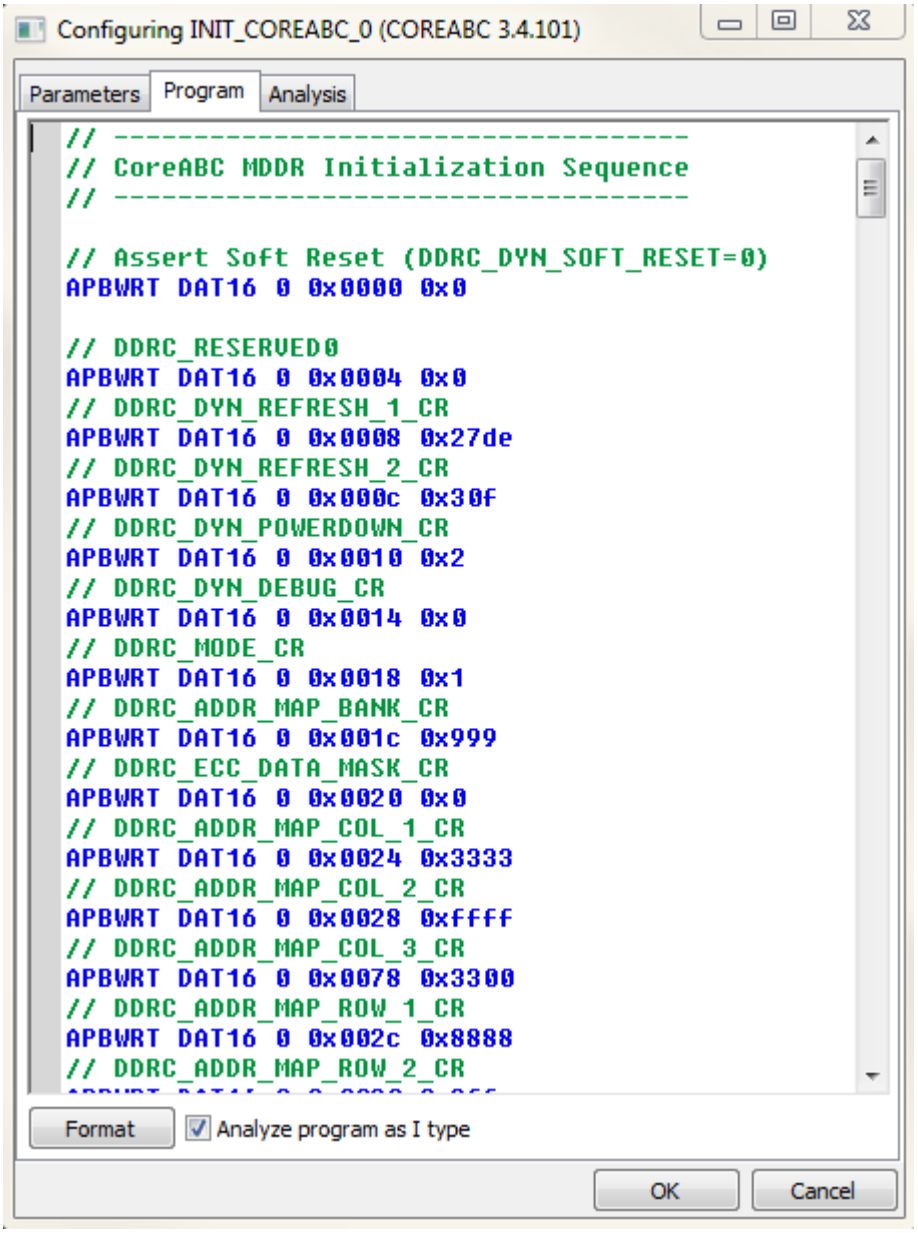


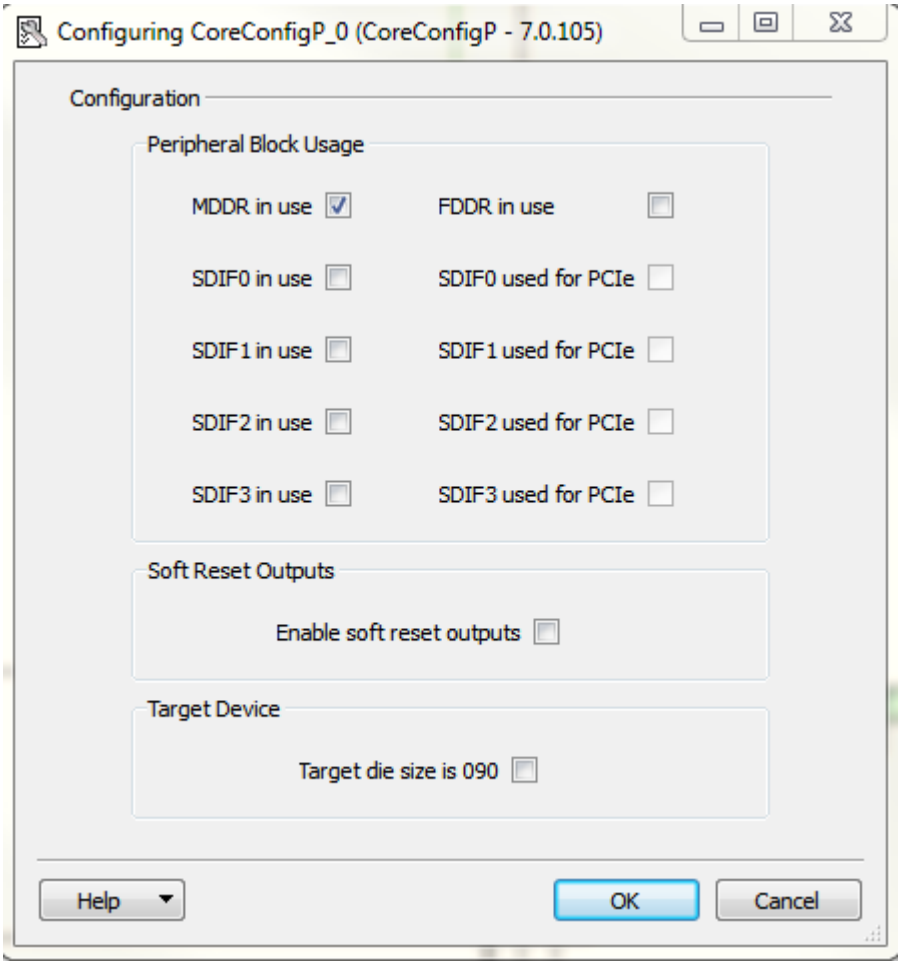
Figure 3-9. CoreABC Program For MDDR



3.6.2. CoreConfigP (Ask a Question)

1. Instantiate CoreConfigP into the same SmartDesign. This core can be found in the Libero Catalog (under Peripherals).
2. Double-click the core to open the configurator.
3. Configure the core to specify which peripherals need to be initialized as shown in figure below.

Figure 3-10. CoreConfigP Dialog Box



3.6.3. CoreResetP (Ask a Question)

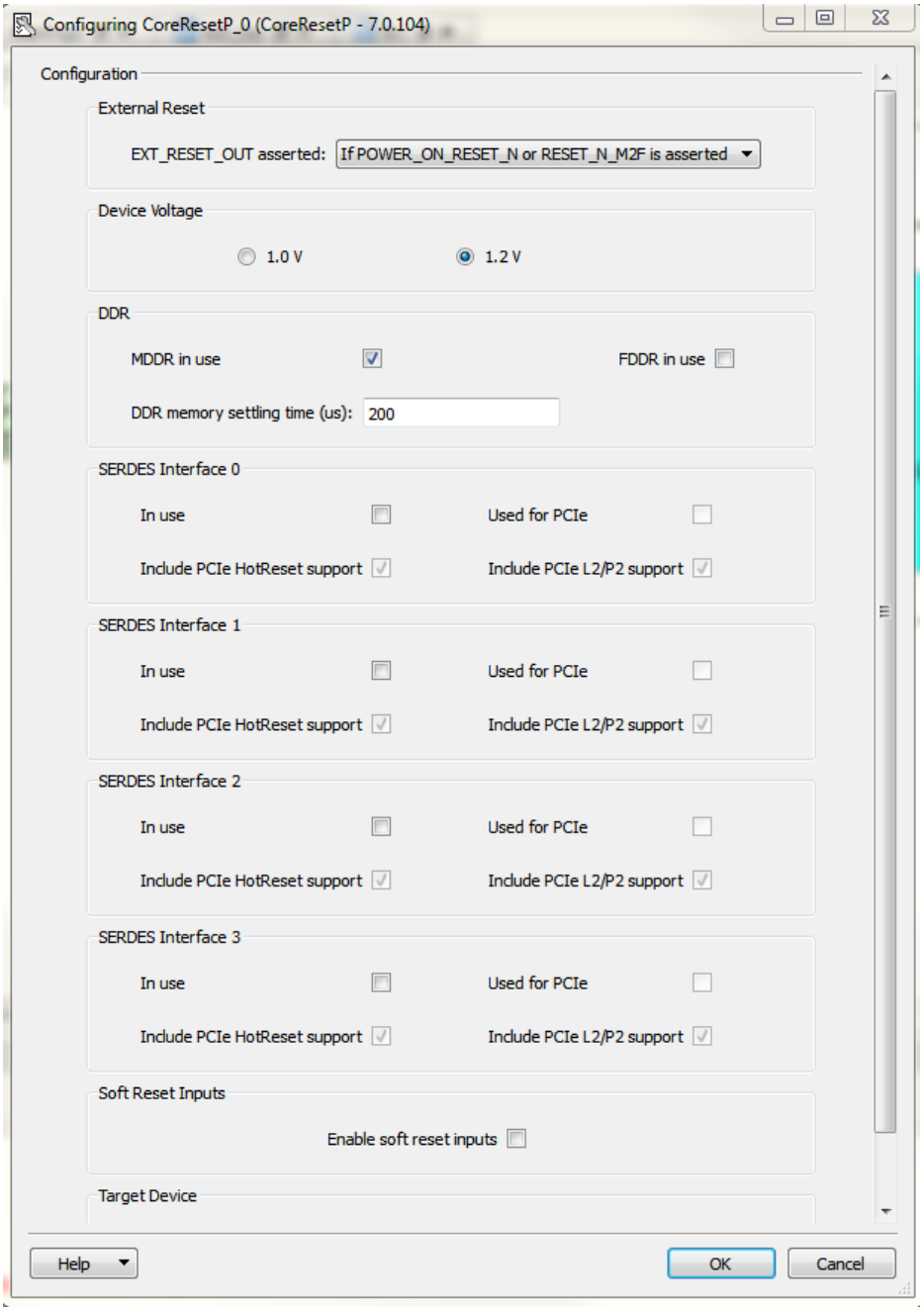
1. Instantiate CoreResetP into the same SmartDesign. This core can be found in the Libero Catalog, under Peripherals.
2. Double-click the core inside the SmartDesign Canvas to open the Configurator.
3. Configure the core to:
   - Specify the external reset behavior (EXT_RESET_OUT asserted). Choose one of four options:
     - EXT_RESET_OUT is never asserted.
     - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) is asserted.
     - EXT_RESET_OUT is asserted if FAB_RESET_N is asserted.
     - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) or FAB_RESET_N is asserted.

- Specify the Device Voltage. The selected value should match the voltage you selected in the Libero Project Settings dialog.
- Check the appropriate checkboxes to indicate which peripherals you are using in your design..
- Specify the external DDR memory setting time. Refer to the external DDR memory vendor datasheet to configure this parameter. 200us is a good default value for DDR2 and DDR3 memories running at 200MHz. This is a very important parameter to guarantee a working

simulation and a working system on silicon. Incorrect value for the settling time may result in simulation errors.

Refer to the CoreResetP handbook for details on the options available to you in this configurator.

Figure 3-11. CoreResetP Configurator



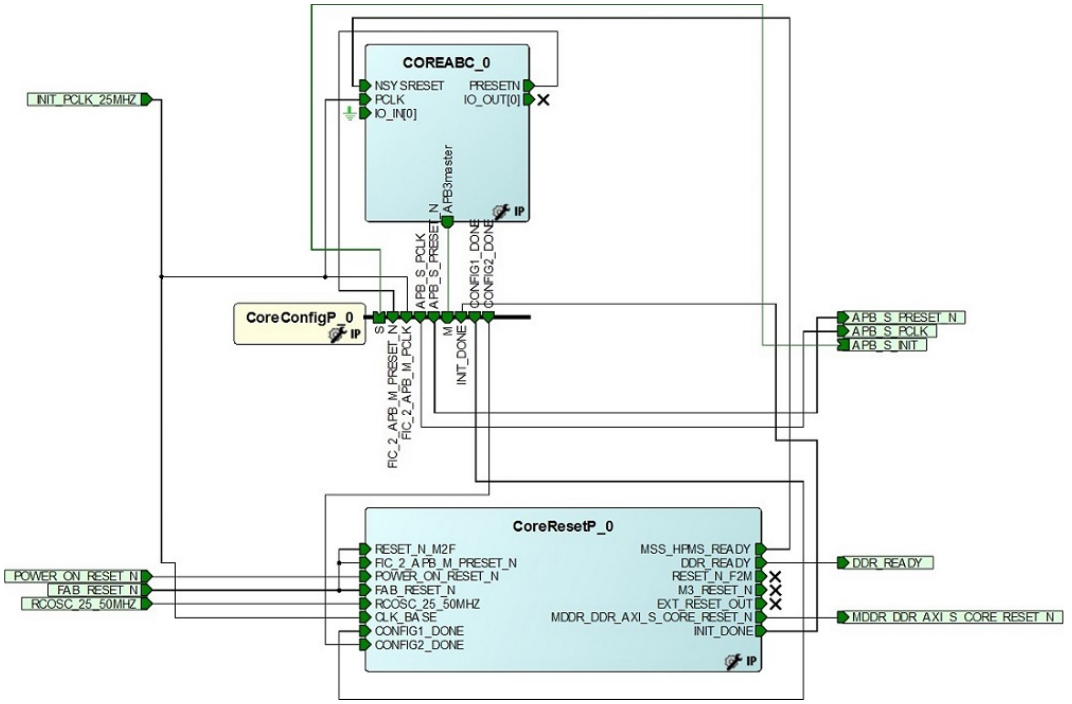
3.6.4. Overall Connectivity of the Initialization Logic (MDDR_INIT) (Ask a Question)

After you have instantiated and configured the 3 cores CoreABC, CoreConfigP and CoreResetP, appropriate connections have to be made to make the initialization logic operational. See the figure below to understand how the connections are made.

The following is a list of signals that need to be promoted to the top which will be needed when interfacing this initialization logic with the actual peripheral (MDDR).

- CoreConfigP:
  - APB_S_PRESET_N
  - APB_S_PCLK
  - APB_S_INIT (APB BIF MDDR_APBmslave)
- CoreResetP:
  - RCOSC_25_50MHZ
  - FAB_RESET_N
  - POWER_ON_RESET_N
  - DDR_READY
  - MDDR_DDR_AXI_S_CORE_RESET_N
- INIT_PCLK_25MHz connecting together the PCLK of CoreABC, the FIC_2_APB_M_PCLK of CoreConfigP and the CLK_BASE of CoreResetP).

Figure 3-12. MDDR_INIT (MDDR Initialization Logic)



3.7. Interfacing MDDR With The Initialization Logic Built For It (Ask a Question)

In the same SmartDesign the System Builder block is present, instantiate the Smart Design containing the MDDR initialization logic (MDDR_INIT), and do necessary interconnections to interface the System Builder block (containing the MDDR) to the initialization logic. See the figure below to understand how the connections are made.

The following is a list of signal interconnections that need to be made to properly interface the System Builder block (MDDR) to the initialization logic.

From Port or Bus Interface (BIF)/ Component	To Port/Bus Interface (BIF)/Component
MDDR_APB_S_PCLK/ System Builder Block	APB_S_PCLK/ initialization logic.
MDDR_APB_S_PRESET_N/ System Builder Block	APB_S_PRESET_N/ initialization logic.
MDDR_APB_SLAVE BIF/ System Builder Block	APB_S_INIT/ initialization logic
POWER_ON_RESET_N/ System Builder Block	POWER_ON_RESET_N/ initialization logic

Interfacing MDDR With The Initialization Logic Built For It (continued)	
From Port or Bus Interface (BIF)/ Component	**To Port/Bus Interface (BIF)/Component**
FAB_RESET_N / System Builder Block	FAB_RESET_N / initialization logic
MDDR_CORE_RESET_N / System Builder Block	MDDR_DDR_AXI_S_CORE_RESET_N / initialization logic

Apart from the above connections, do the following also:

- Promote the FAB_RESET_N pin from the initialization logic (MDDR_INIT_0 instance) to the top level (this is the warm reset). It is recommended to synchronize FAB_RESET_N to CLK_BASE.
- Promote the HPMS_DDR_FIC_SUBSYSTEM_PINS to the top to drive the fabric logic that belongs to the HPMS_DDR_FIC_SUBSYSTEM.
- Promote the DDR_READY of the initialization logic to the top to monitor the status of the MDDR initialization.
- Drive the INIT_PCLK_25MHz input pin of the initialization logic with 25MHz clock. You can use the unused GLx in the System Builder block from the 'Fabric CCC' tab of the 'Clocks' page to drive any clock in the fabric logic.
- Instantiate a FABCCC block configured as shown in figures below and make the following connections:
  - Connect system builder output RCOSC_25_50MHZ_CCC_OUTPUT to FCCC input RCOSC_25_50MHZ_CCC_IN.
  - Connect FCCC output GL0 to MDDR_INIT input RCOSC_25_50MHZ_CCC

Figure 3-13. FAB CCC Configurator

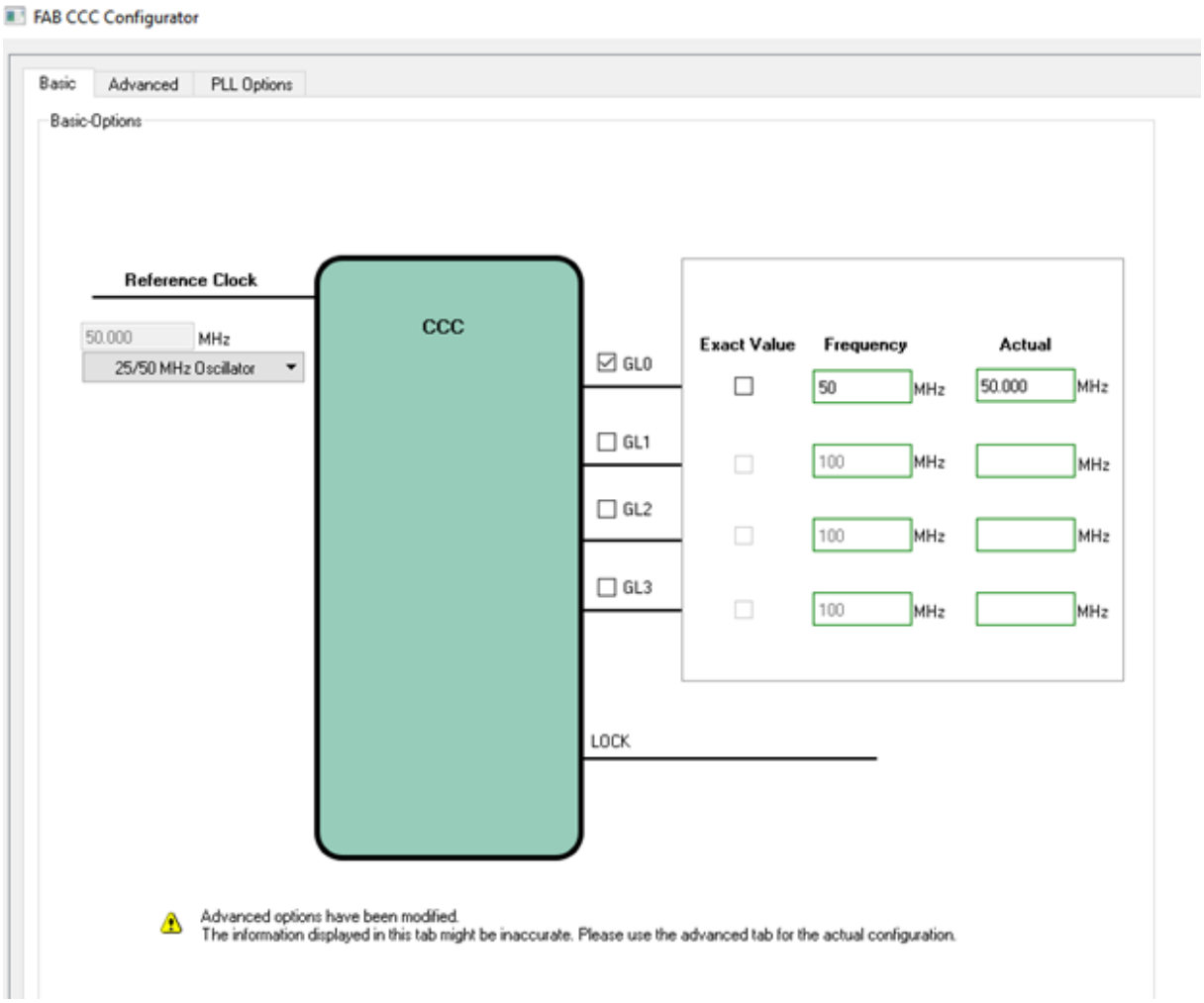


Figure 3-14. Advanced Tab of FAB CCC Configurator

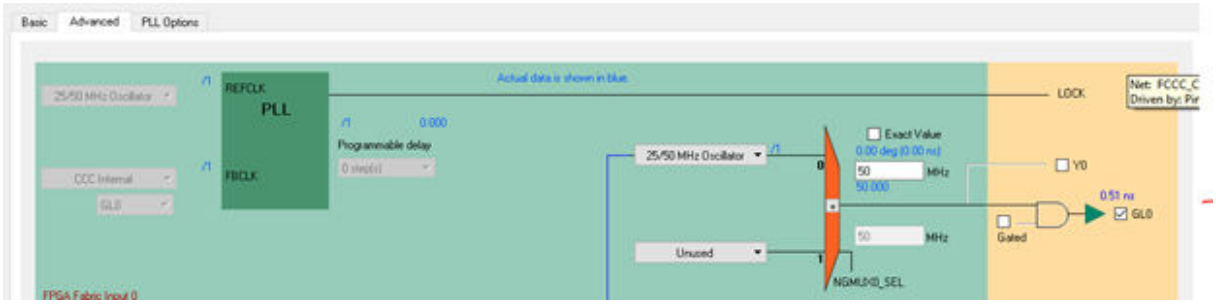
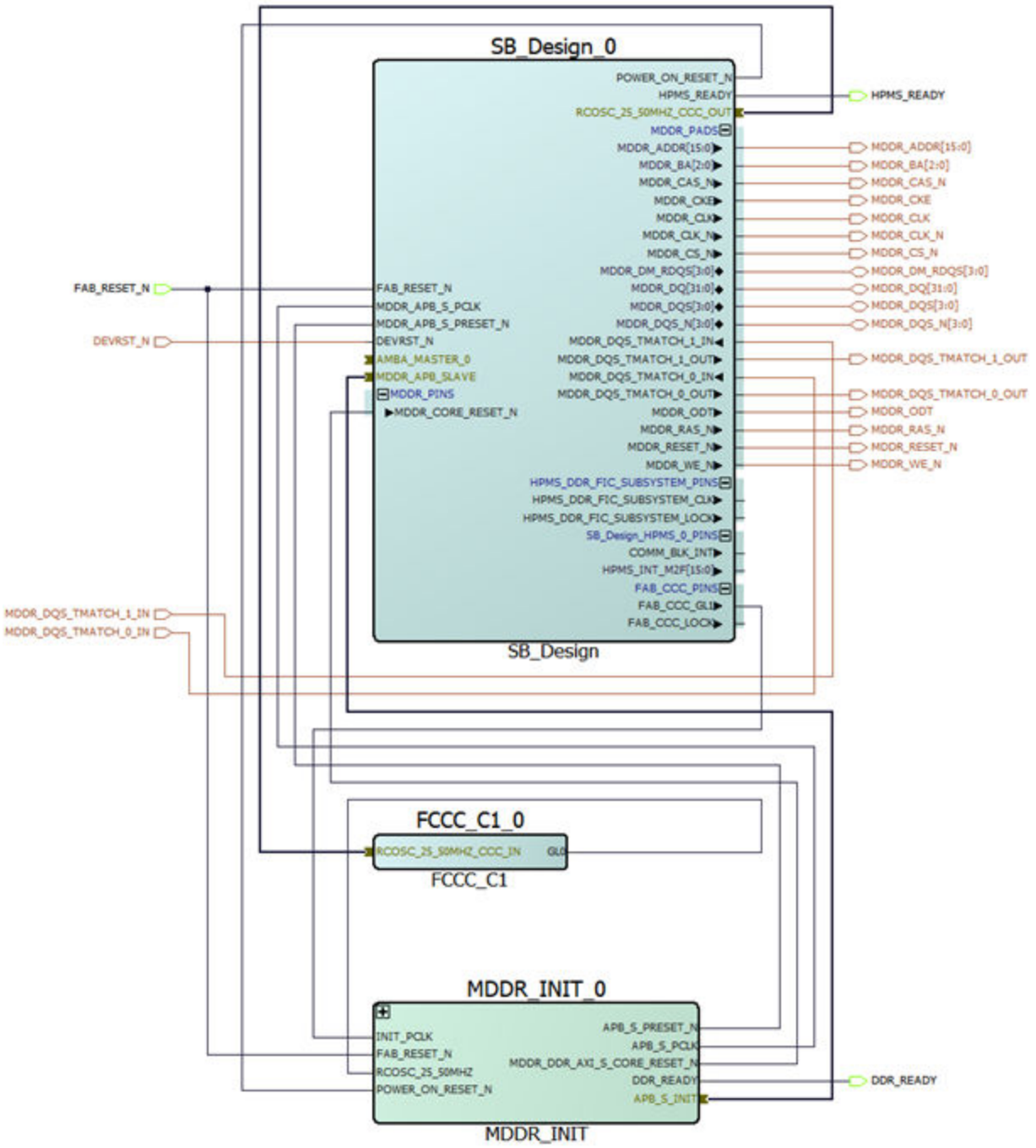


Figure 3-15. Interfacing System Builder (MDDR) With The Initialization Logic



3.8. Building Standalone Initialization Logic for FDDR (Ask a Question)

In order to initialize the FDDR, you must create the initialization subsystem in the FPGA fabric. The FPGA fabric initialization subsystem moves data from the CoreABC program to the DDR configuration registers, manages the reset sequences required for the FDDR block to be operational and signals when the FDDR block is ready to communicate with the rest of your design. To create the initialization subsystem you must:

- Instantiate and configure CoreABC soft IP core.
- Load CoreABC with the initialization program generated to the FDDR_init_abc.txt file.

- Instantiate and configure the CoreConfigP and CoreResetP cores.
- Connect these components to the peripheral's (FDDR) configuration interfaces, clocks, resets and PLL lock ports.

3.8.1. CoreABC configuration (Ask a Question)

1. Create a new SmartDesign component (FDDR_INIT).
2. Instantiate CoreABC into your SmartDesign. This core can be found in the Libero Catalog (under Processors).
3. Double-click the core to open the configurator.
4. Configure the core as shown in the figure below.
   - Configure the data bus width to be 16.
   - Configure the maximum number of instructions to at least 256.
   - Configure to use AND and OR operations as optional instructions.
   - Configure Instruction Store to Hard (FPGA Tiles).
5. Copy the CoreABC program generated for FDDR from the FDDR_init_abc.txt file created under the <project_location>/.../FABDDR_0/ directory, to the CoreABC Program tab. See the figure below.

Figure 3-16. CoreABC Configuration

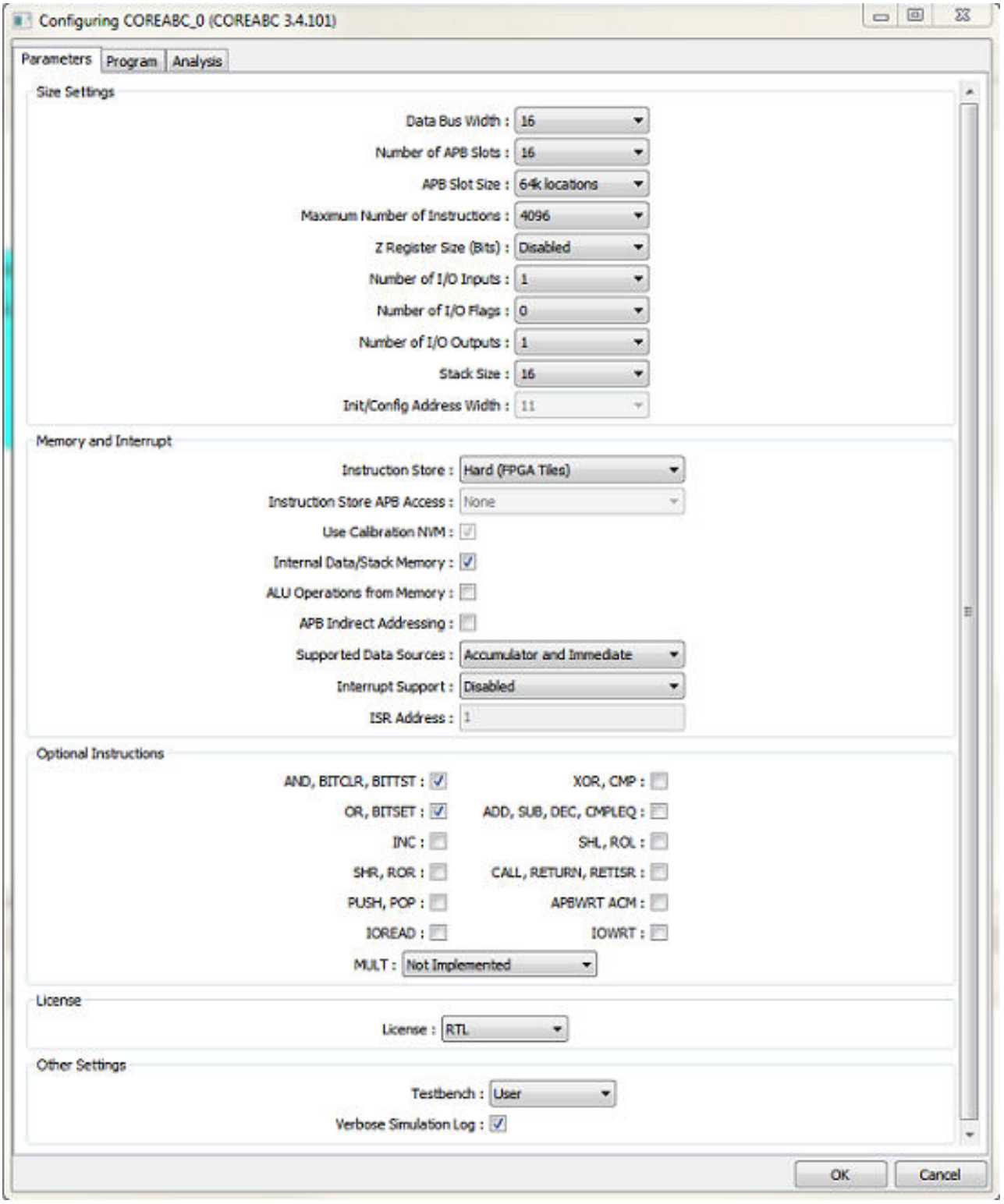
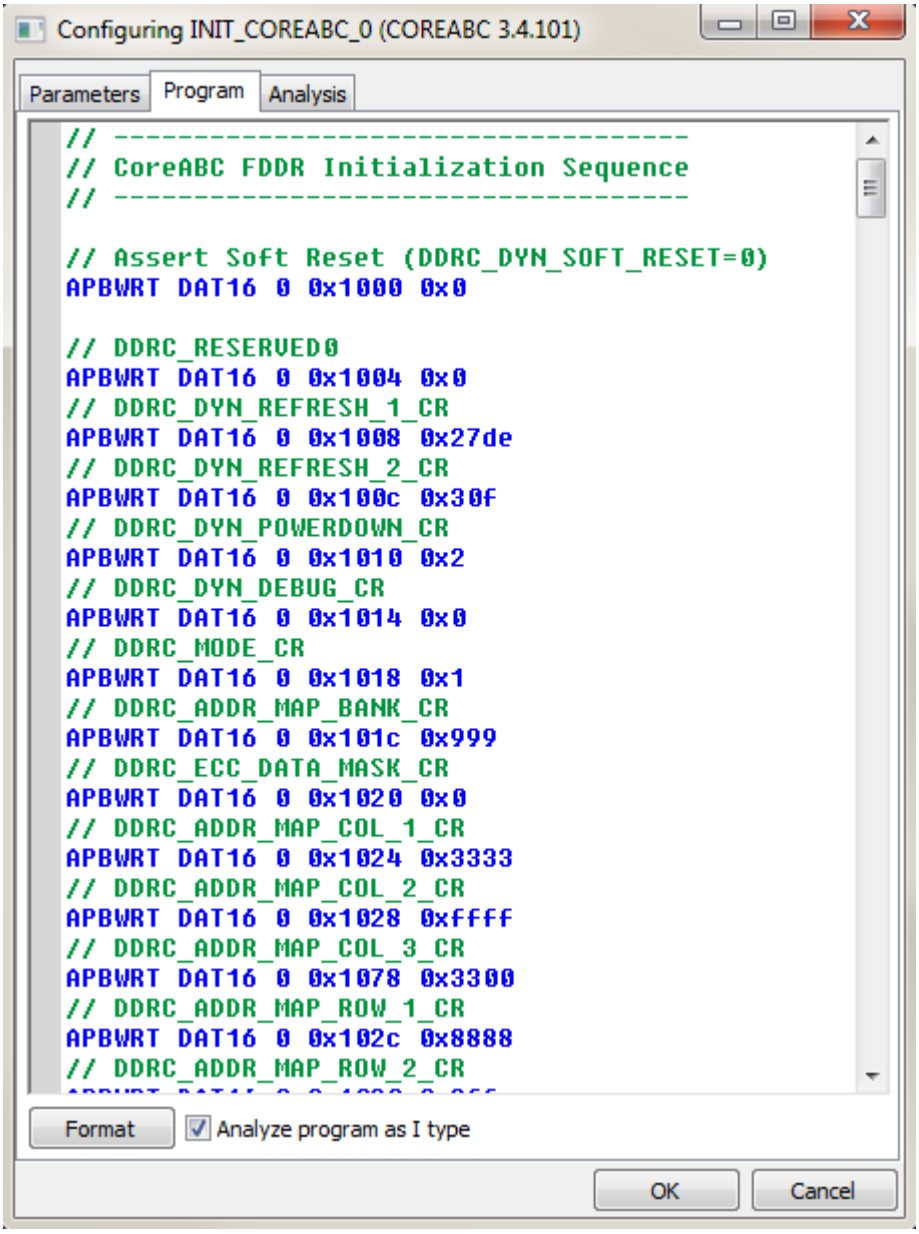


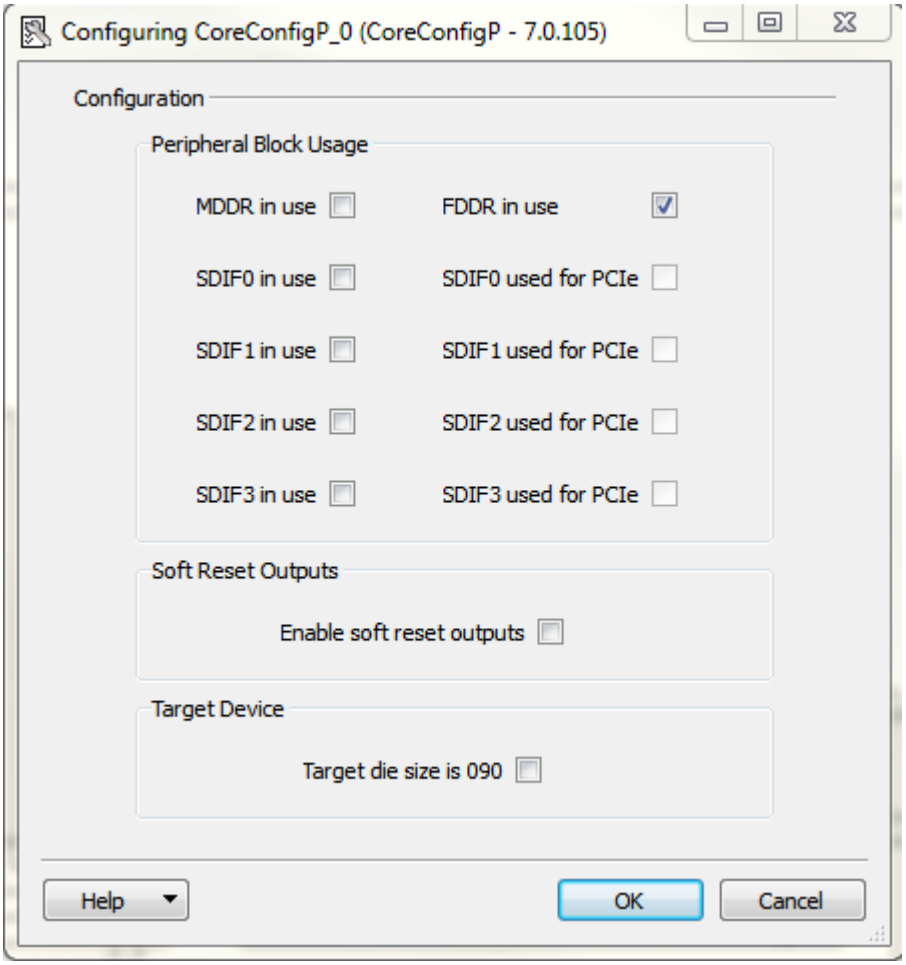
Figure 3-17. CoreABC Program For FDDR



3.8.2. CoreConfigP (Ask a Question)

1. Instantiate CoreConfigP into the same SmartDesign. This core can be found in the Libero Catalog (under Peripherals).
2. Double-click the core to open the configurator.
3. Configure the core to specify which peripherals need to be initialized.

Figure 3-18. CoreConfigP Dialog Box



3.8.3. CoreResetP (Ask a Question)

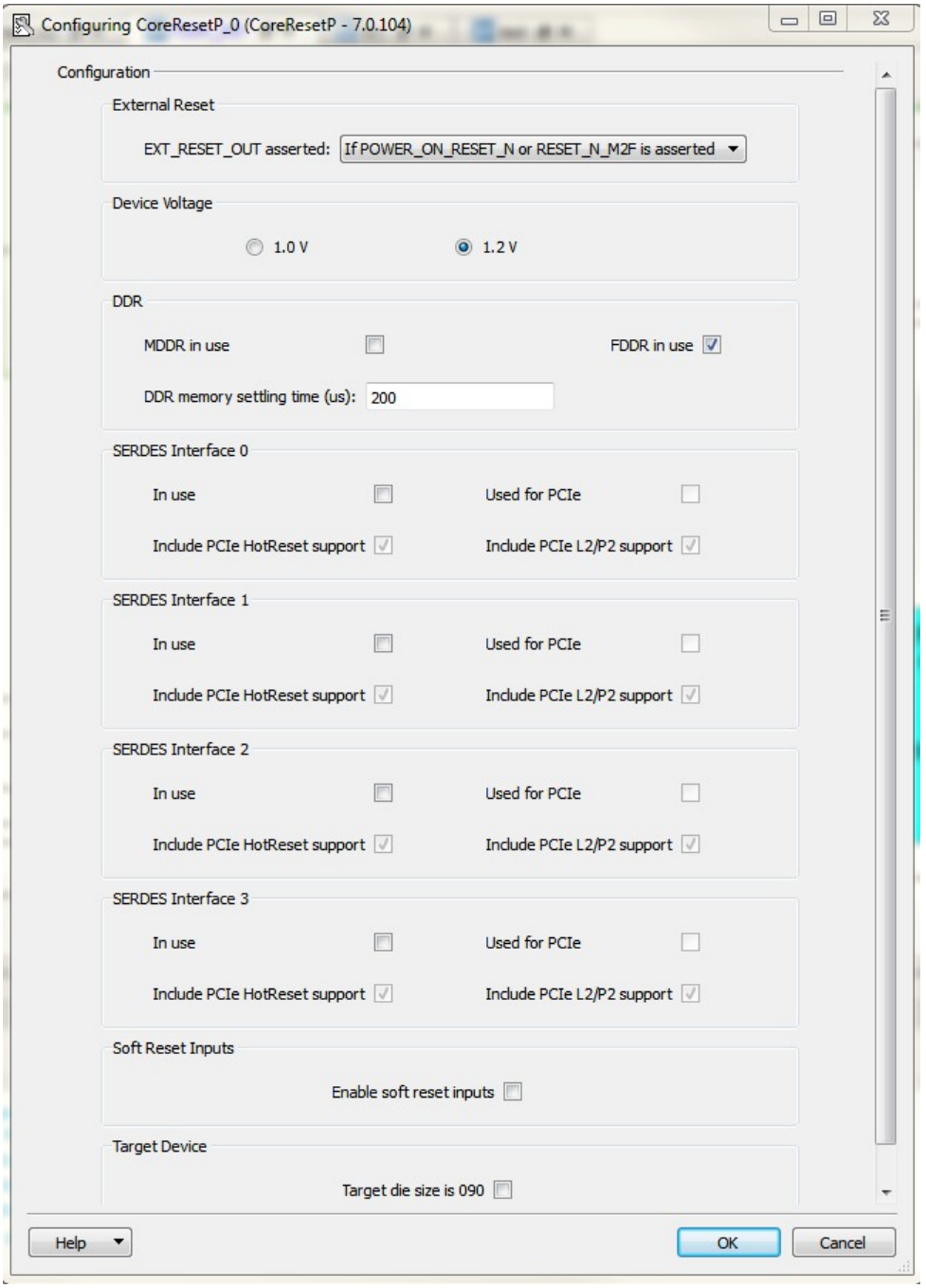
1. Instantiate CoreResetP into the same SmartDesign. This core can be found in the Libero Catalog, under Peripherals.
2. Double-click the core inside the SmartDesign Canvas to open the Configurator.
3. Configure the core to:

- Specify the external reset behavior (EXT_RESET_OUT asserted). Choose one of four options:
  - EXT_RESET_OUT is never asserted
  - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) is asserted
  - EXT_RESET_OUT is asserted if FAB_RESET_N is asserted
  - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) or FAB_RESET_N is asserted
- Specify the Device Voltage. The selected value should match the voltage you selected in the Libero Project Settings dialog.
- Check the appropriate checkboxes to indicate which peripherals you are using in your design.
- Specify the external DDR memory setting time. Refer to the external DDR memory vendor datasheet to configure this parameter. 200us is a good default value for DDR2 and DDR3 memories running at 200MHz. This is a very important parameter to guarantee a working

simulation and a working system on silicon. Incorrect value for the settling time may result in simulation errors.

Refer to the CoreResetP handbook for details on the options available to you in this configurator.

Figure 3-19. CoreResetP Configurator



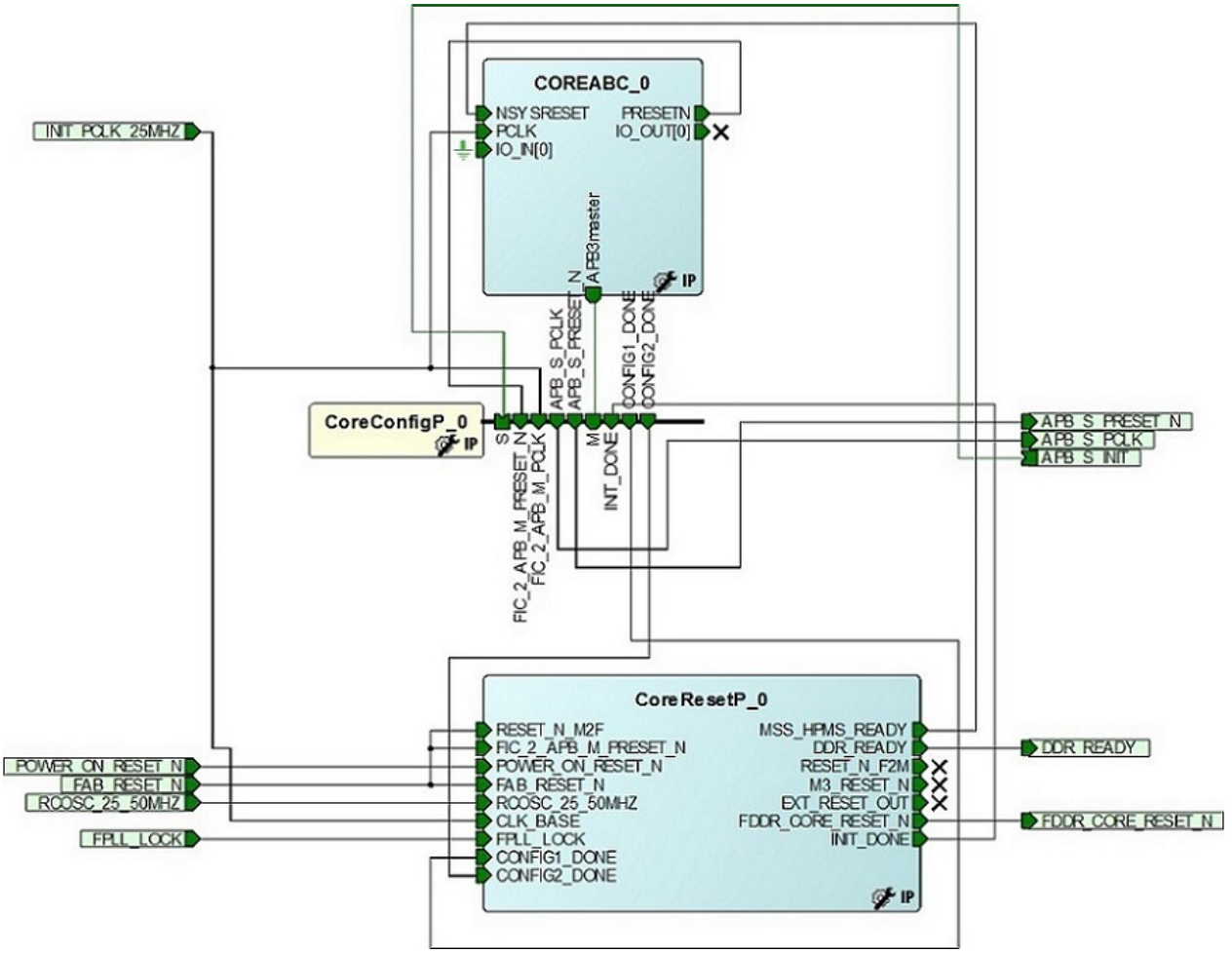
3.8.4. Overall Connectivity Of The Initialization Logic (FDDR_INIT) (Ask a Question)

After you have instantiated and configured the 3 cores CoreABC, CoreConfigP and CoreResetP, appropriate connections have to be made to make the initialization logic operational. See the figure below to understand how the connections are made.

The following is a list of signals that need to be promoted to the top which will be needed when interfacing this initialization logic with the actual peripheral (FDDR).

- CoreConfigP:
  - APB_S_PRESET_N
  - APB_S_PCLK
  - APB_S_INIT (APB BIF FDDR_APBmslave)
- CoreResetP:
  - RCOSC_25_50MHZ
  - FAB_RESET_N
  - POWER_ON_RESET_N
  - DDR_READY
  - FDDR_CORE_RESET_N
  - FPLL_LOCK
- INIT_PCLK_25MHz (connecting together the PCLK of CoreABC, the FIC_2_APB_M_PCLK of CoreConfigP and the CLK_BASE of CoreResetP).

Figure 3-20. FDDR_INIT (FDDR Initialization Logic)



3.8.5. Interfacing FDDR With The Initialization Logic Built For It (Ask a Question)

In the same SmartDesign the System Builder block is present, instantiate the Smart Design containing the FDDR initialization logic (FDDR_INIT), and do necessary interconnections to interface the System Builder block containing the FDDR, to the initialization logic. See the figure below to understand how the connections are made.

The following is a list of signal interconnections that need to be made to properly interface the FDDR to the initialization logic.

From Port or Bus Interface (BIF)/ Component	To Port/Bus Interface (BIF)/ Component
FDDR_APB_S_PCLK/ System Builder Block	APB_S_PCLK/ initialization logic
FDDR_APB_S_PRESET_N/ System Builder Block	APB_S_PRESET_N/ initialization logic
FDDR_APB_SLAVE BIF/ System Builder Block	APB_S_INIT/ initialization logic
POWER_ON_RESET_N/ System Builder Block	POWER_ON_RESET_N/ initialization logic
FAB_RESET_N / System Builder Block	FAB_RESET_N / initialization logic
FDDR_CORE_RESET_N / System Builder Block	FDDR_CORE_RESET_N / initialization logic
FDDR_FPLL_LOCK/ System Builder Block	FPLL_LOCK/ initialization logic

Apart from the above connections, do the following also:

- Promote the FAB_RESET_N pin from the initialization logic (FDDR_INIT_0 instance) to the top level (this is the warm reset). It is recommended to synchronize FAB_RESET_N to CLK_BASE.
- Promote the DDR_READY of the initialization logic to the top to monitor the status of the FDDR initialization.
- Promote the FDDR_SUSBSYSTEM_RESET_N pin to the top or drive it appropriately from the fabric logic.
- Instantiate a FABCCC block and do the following connections:
  - Drive the INIT_PCLK_25MHZ input pin of the initialization logic with the GLx of FABCCC block configured to 25MHz frequency.
  - Drive the FDDR_SUBSYSTEM_CLK input pin under the FDDR_SUBSYSTEM_PINS group of the System Builder block with the GLx of FABCCC block (configured to appropriate frequency).
  - Drive the FDDR_SUBSYSTEM_LOCK input pin under the FDDR_SUBSYSTEM_PINS group of the System Builder block with the LOCK pin of the FABCCC block.
- Instantiate a FABCCC block configured as shown in the figures below and make the following connections:
  - Connect system builder output RCOSC_25_50MHZ_CCC_OUTPUT to FCCC input RCOSC_25_50MHZ_CCC_IN.
  - Connect FCCC output GL0 to FDDR_INIT input RCOSC_25_50MHZ_CCC

Figure 3-21. FAB CCC Configurator

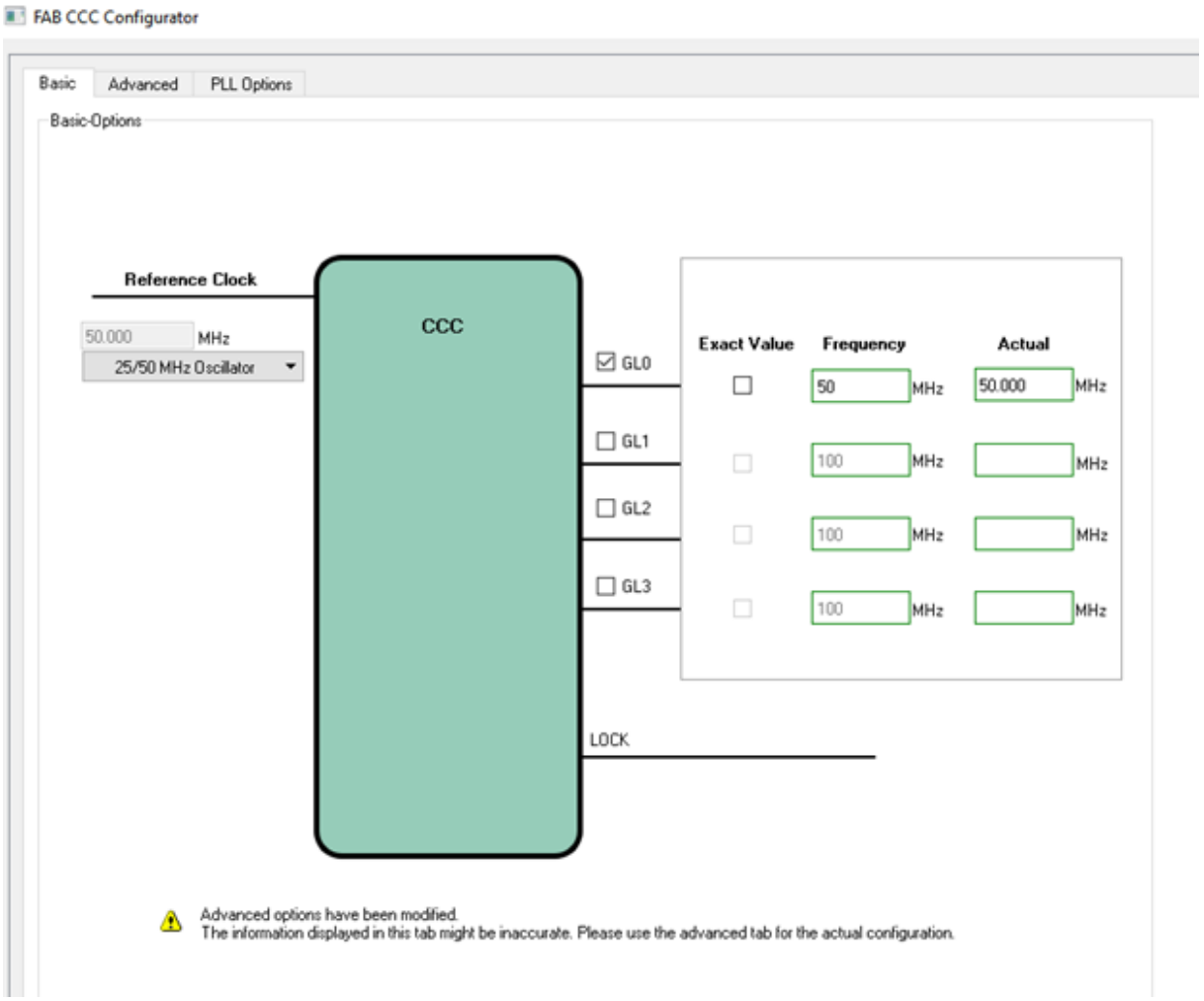


Figure 3-22. Advanced Tab in FAB CCC Configurator

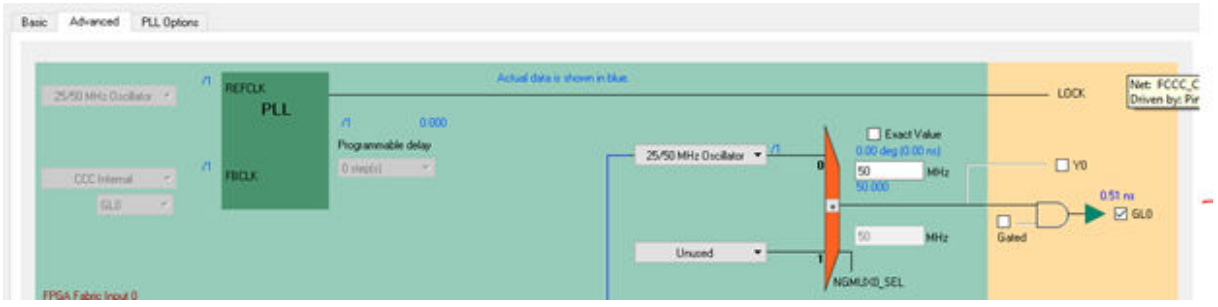
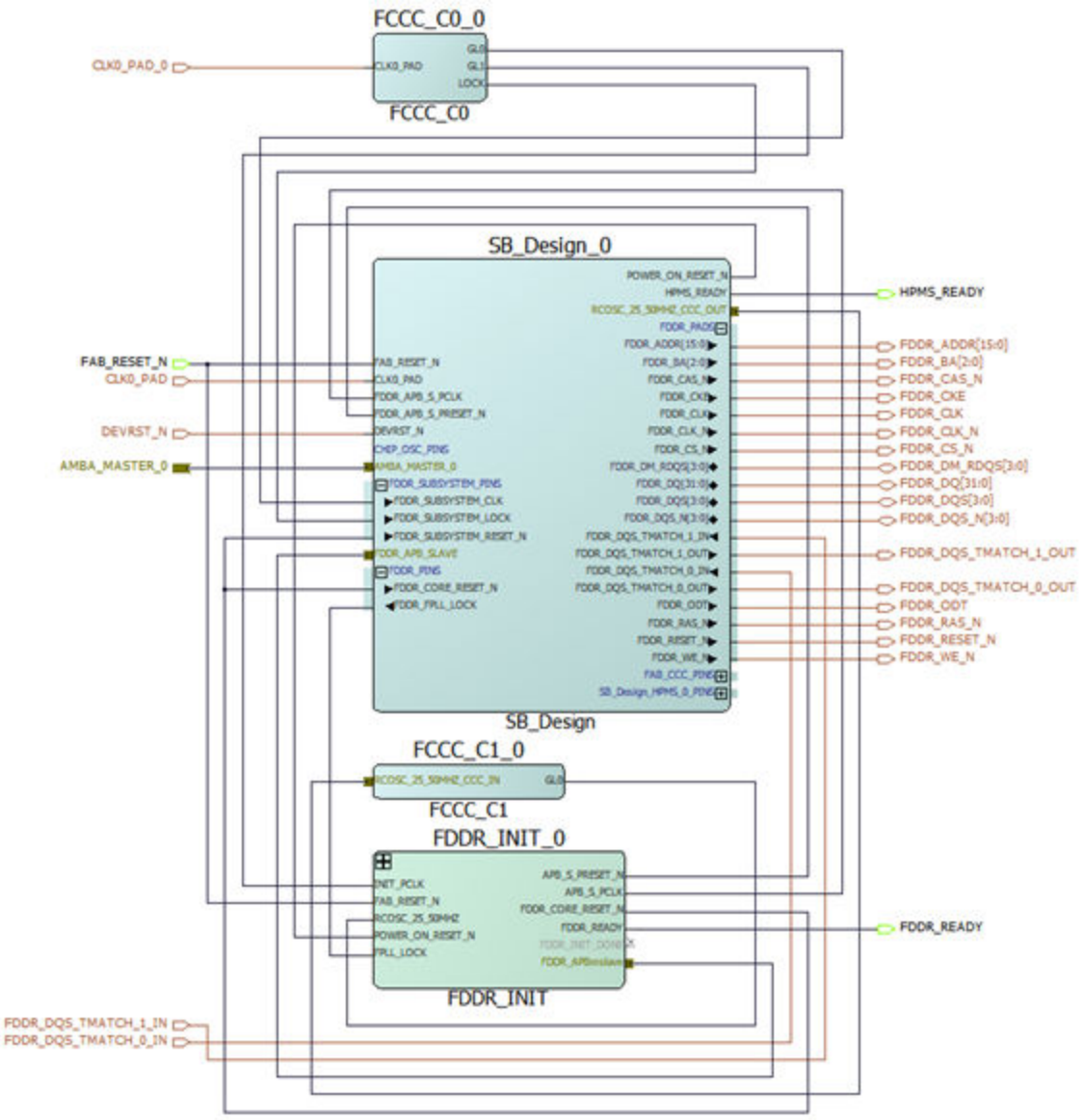


Figure 3-23. Interfacing System Builder (FDDR) With The Initialization Logic



3.8.6. Continuing with the Design Flow (Ask a Question)

Next step is to integrate any user logic that you might have with the System Builder block and the initialization logic. Once you have done that, you can generate your top level SmartDesign. This will generate all files that are necessary to implement and simulate your design. You can then proceed with the rest of the Design Flow.

4. Using SmartDesign To Create A Design Using SERDESIF And FDDR Blocks (Ask a Question)

In this section we describe how to put a complete 'initialization' solution together without using the IGLOO2 System Builder. The goal is to help you understand what you must do if you do not wish to use the System Builder. In this section we describe how to:

- Input the configuration data for FDDR controller and SERDESIF configuration registers.
- Instantiate and connect the Fabric Cores required to transfer the configuration data to the FDDR controller and SERDESIF configuration registers.

Note: If you want to use the MDDR block in your design, then you must use System Builder. In IGLOO2, it is not possible to build a design using MDDR without using the System Builder.

4.1. Design using SERDESIF_n (n=0/1/2/3) (Ask a Question)

4.1.1. Building Standalone Initialization Logic For SERDESIF_n (Ask a Question)

In order to initialize the SERDESIF_n registers, you must create the initialization subsystem in the FPGA fabric. The FPGA fabric initialization subsystem moves data from the CoreABC program to the SERDESIF_n configuration registers, manages the reset sequences required for the SERDESIF_n block to be operational and signals when the SERDESIF_n block is ready to communicate with the rest of your design. To create the initialization subsystem you must:

- Instantiate and configure CoreABC soft IP core.
- Load CoreABC with the initialization program generated to the SERDESIF_n_init_abc.txt file.
- Instantiate and configure the CoreConfigP and CoreResetP cores.
- Instantiate and configure the on-chip 50MHz RC oscillator.
- Instantiate the System Reset (SYSRESET) macro.
- Connect these components to the peripheral's (SERDESIF_n) configuration interface, clocks, resets and PLL lock ports.

4.1.1.1. CoreABC Configuration (Ask a Question)

1. Create a new SmartDesign component (SERDESIF_n_INIT).
2. Instantiate CoreABC into your SmartDesign. This core can be found in the Libero Catalog (under Processors).
3. Double-click the core to open the configurator.
4. Configure the core as shown in the figure below.
   - Configure the data bus width to be 32 (as 32-bit data needs to be written to some of the SERDES registers).
   - Configure the maximum number of instructions to at least 256.
   - Configure to use AND and OR operations as optional instructions.
   - Configure Instruction Store to Hard (FPGA Tiles).
5. Copy the CoreABC program generated for SERDESIF_n from the SERDESIF_n_init_abc.txt file created under the <project_location>/../SERDES_IF_n/ directory, to the CoreABC Program tab. See the figure below.

Note: The SERDESIF_n_init_abc.txt file will be generated only after you generate the Smart Design containing the SERDESIF_n block. So after you've made all the connections and generated all the blocks (including SERDESIF_n), you will need to copy the contents of the SERDESIF_n_init_abc.txt file to the CoreABC Program tab and regenerate the initialization logic Smart Design component containing CoreABC. You may defer doing this until you completely configure your SERDESIF_n block and generate the SmartDesign component containing it.

Figure 4-1. CoreABC Configuration

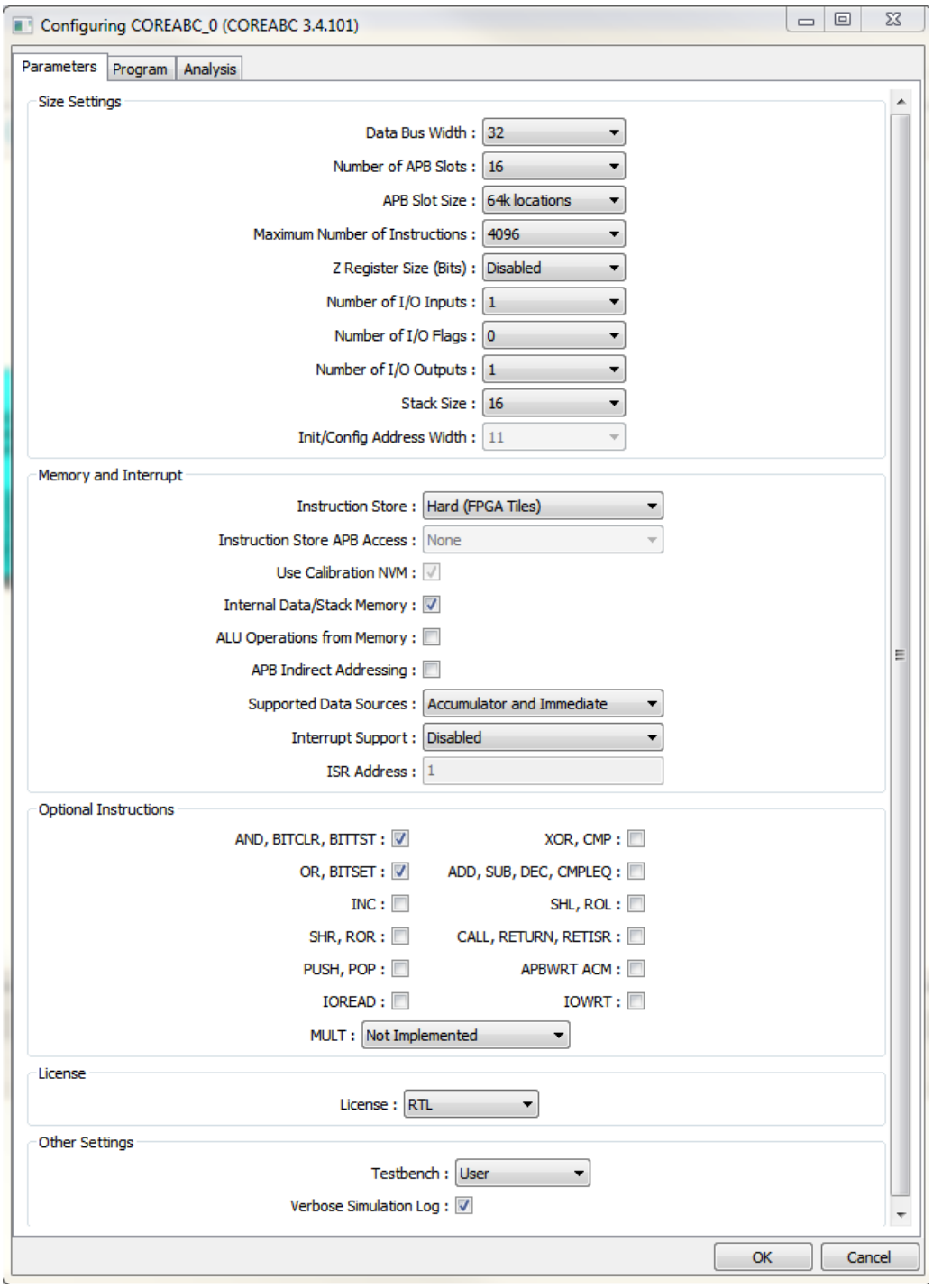
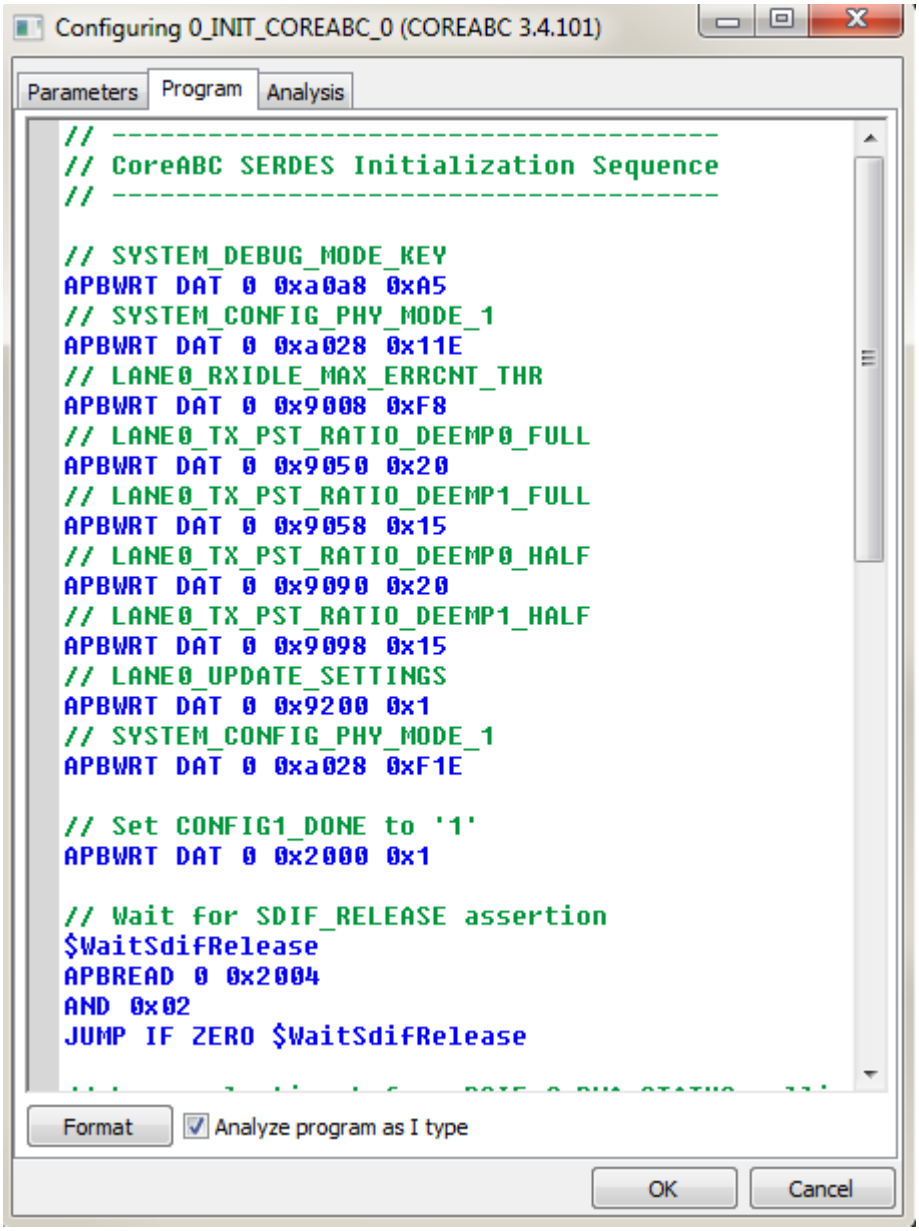


Figure 4-2. CoreABC Program For SERDESIF_n

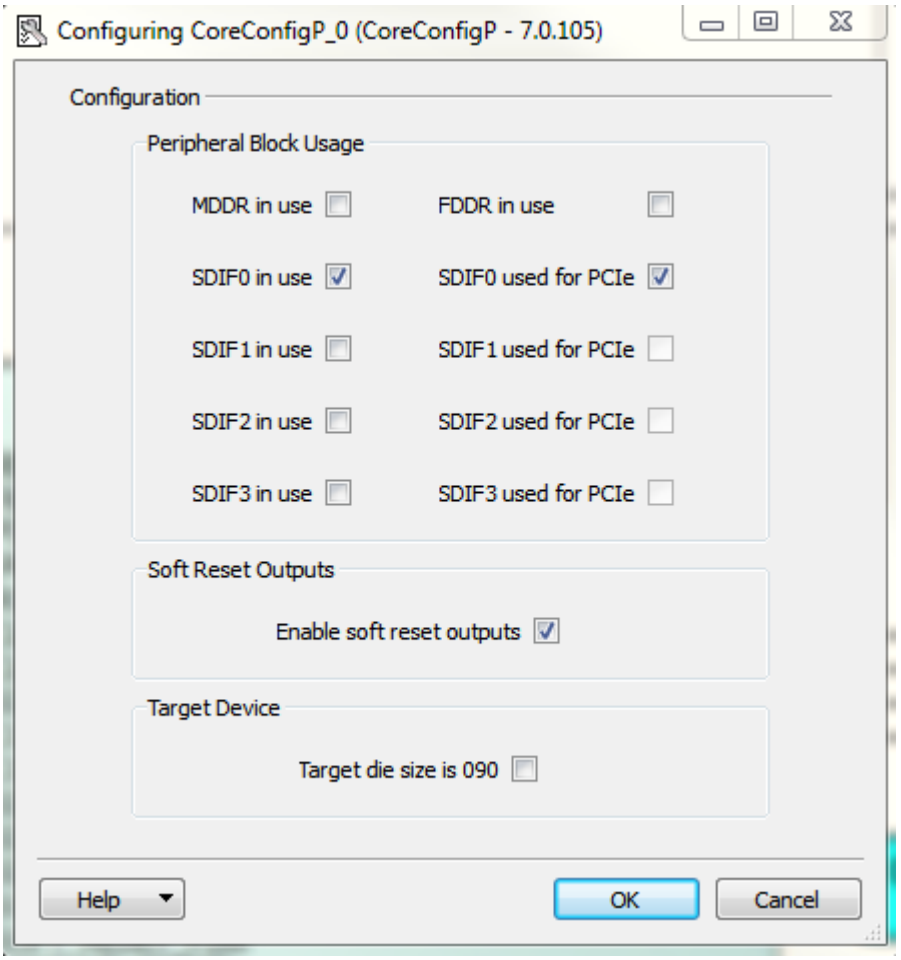


4.1.1.2. CoreConfigP (Ask a Question)

1. Instantiate CoreConfigP into the same SmartDesign. This core can be found in the Libero Catalog (under Peripherals).
2. Double-click the core to open the configurator.
3. Check the appropriate checkboxes as shown in the figure below.

Note: Irrespective of the SERDESIF_n location (n=0/1/2/3) you want to configure using this initialization logic, check only the “SDIF0 in use” and “SDIF0 used for PCIe” checkboxes. For example, even if you are building this initialization logic for say, the SERDESIF_1/2/3 location, you need to check the “SDIF0 in use” and “SDIF0 used for PCIe” checkboxes in the CoreConfigP instance. Do not check the checkboxes corresponding to the SERDESIF_1/2/3 locations in the CoreConfigP instance. This is a requirement for the Libero generated CoreABC code to work for all the SERDESIF_n locations (n = 0/1/2/3). And this rule is applicable only when you are configuring the CoreConfigP and CoreResetP instances as a part of building the initialization logic; you will have to specify the exact SERDESIF_n location in the SERDESIF configurator later when you instantiate and configure the SERDESIF block.

Figure 4-3. CoreConfigP Dialog Box



4.1.1.3. CoreResetP (Ask a Question)

1. Instantiate CoreResetP into the same SmartDesign. This core can be found in the Libero Catalog, under Peripherals.
2. Double-click the core inside the SmartDesign Canvas to open the Configurator.
3. Configure the core to:

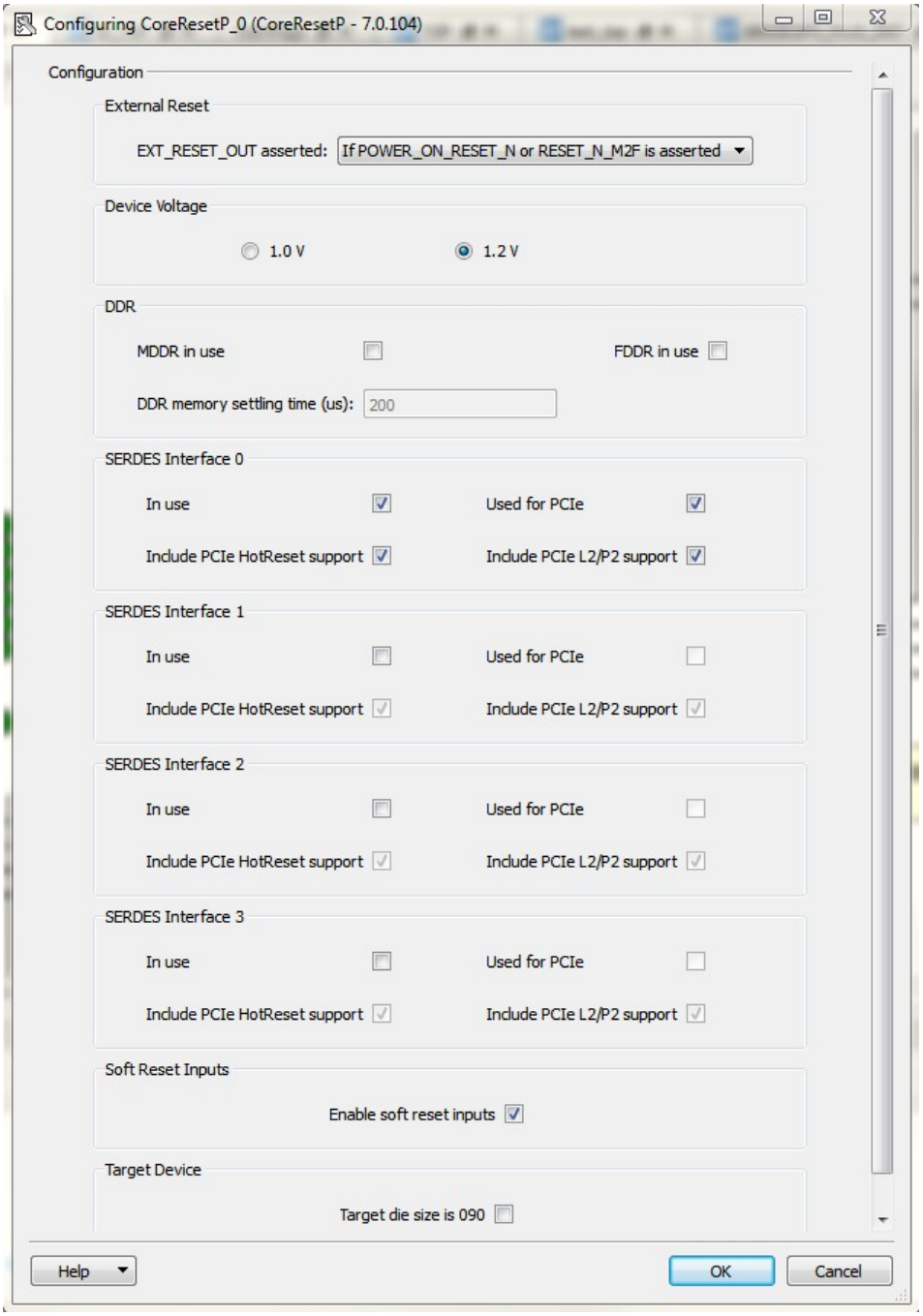
- Specify the external reset behavior (EXT_RESET_OUT asserted). Choose one of four options:
  - EXT_RESET_OUT is never asserted
  - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) is asserted
  - EXT_RESET_OUT is asserted if FAB_RESET_N is asserted

  - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) or FAB_RESET_N is asserted

- Specify the Device Voltage. The selected value should match the voltage you selected in the Libero Project Settings dialog.
- Check the appropriate checkboxes as shown in the figure below. Refer to the CoreResetP handbook for details on the options available to you in this configurator.

Note: Irrespective of the SERDESIF_n location (n=0/1/2/3) you want to configure using this initialization logic, check only the checkboxes under the “SERDES Interface 0”. For example, even if you are building this initialization logic for say, the SERDESIF_1/2/3 location, you need to check the checkboxes under the “SERDES Interface 0” in the CoreResetP instance. Do not check the checkboxes corresponding to the SERDESIF_1/2/3 locations. This is a requirement for the Libero generated CoreABC code to work for all the SERDESIF_n locations (n = 0/1/2/3). And this rule is applicable only when you are configuring the CoreConfigP and CoreResetP instances as a part of building the initialization logic; you will have to specify the exact SERDESIF_n location in the SERDESIF configurator later when you instantiate and configure the SERDESIF block.

Figure 4-4. CoreResetP Configurator



4.1.1.4. Overall Connectivity of the initialization logic (SERDESIF_n_INIT) (Ask a Question)

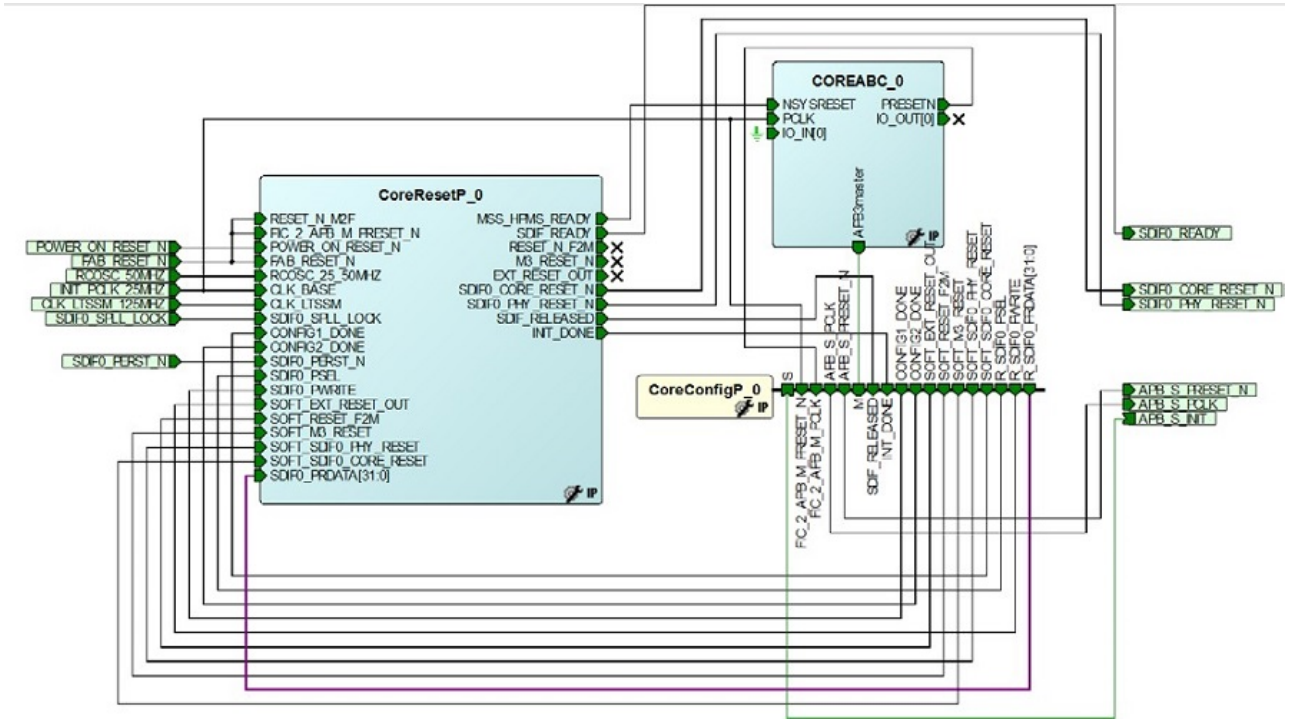
After you have instantiated and configured the 3 cores CoreABC, CoreConfigP and CoreResetP, appropriate connections have to be made to make the initialization logic operational. See the figure below to understand how the connections are made.

The following is a list of signals that need to be promoted to the top which will be needed when interfacing this initialization logic with the actual peripheral (SERDESIF_n).

- CoreConfigP:
  - APB_S_PRESET_N

  - APB_S_PCLK
  - APB_S_INIT (APB BIF SDIF0_APBmslave)
- CoreResetP:
  - RCOSC_25_50MHZ
  - CLK_LTSSM (promoted as CLK_LTSSM_125MHZ)
  - FAB_RESET_N
  - POWER_ON_RESET_N
  - SDIF_READY
  - SDIFn_PHY_RESET_N
  - SDIFn_CORE_RESET_N
  - SDIFn_SPLL_LOCK
  - SDIFn_PERST_N
- INIT_PCLK_25MHz (connecting together the PCLK of CoreABC, the FIC_2_APB_M_PCLK of CoreConfigP and the CLK_BASE of CoreResetP).

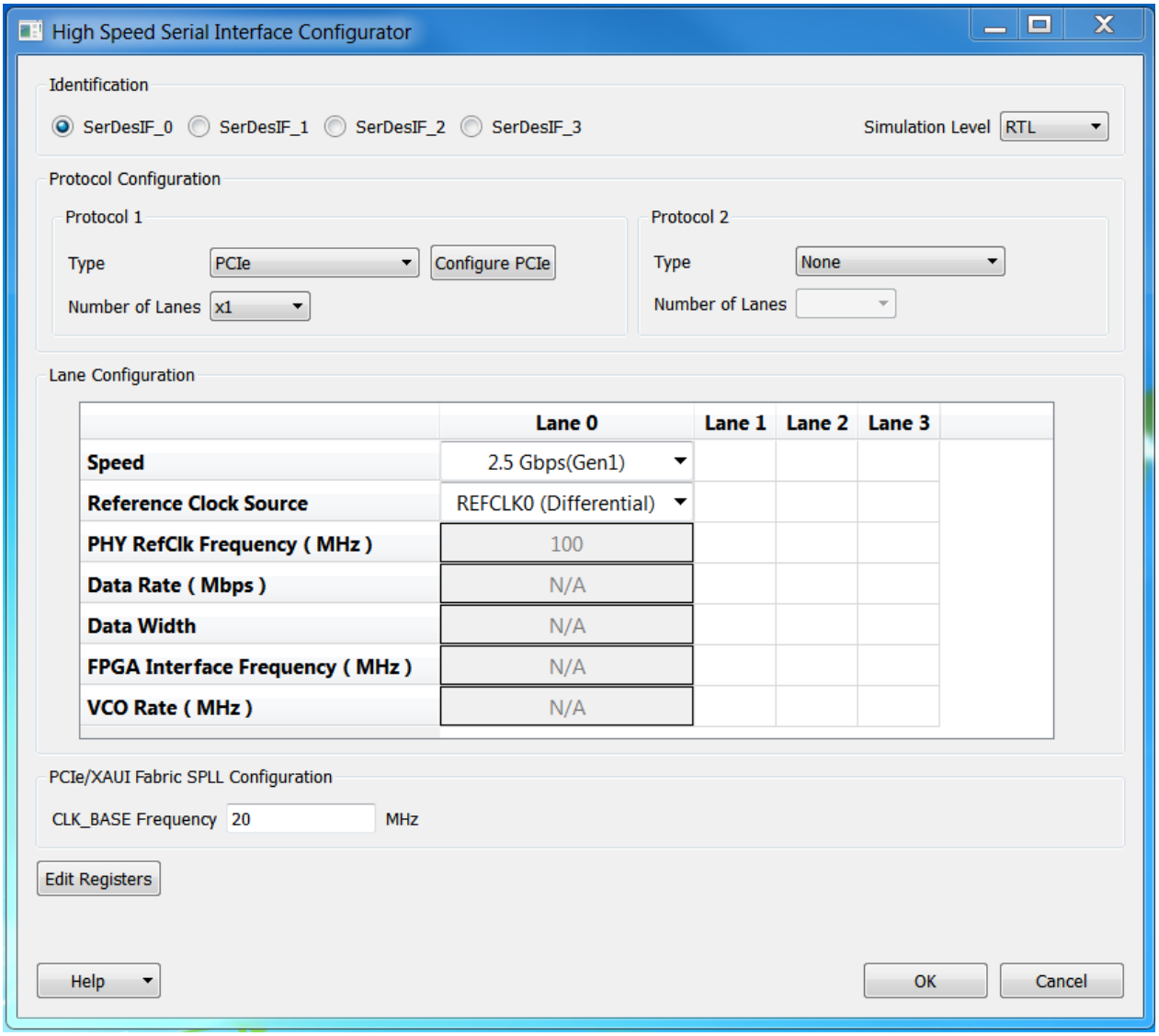
Figure 4-5. SERDESIF_n_INIT (SERDESIF_n Initialization Logic)



4.1.1.5. Instantiating And Configuring The SERDESIF_n Block (Ask a Question)

Create a Smart Design component and instantiate a SERDESIF block from the catalog window (from under the Peripherals). Double-click the SERDESIF block in the SmartDesign canvas to open the configurator to select a location (n= 0/1/2/3) and configure the SERDES. Similar to the DDR configuration registers, each SERDES block also has configuration registers that must be loaded at runtime. You can either import these register values or use the High Speed Serial Interface Configurator to enter your PCIe or EPCS parameters and the register values will automatically be computed for you. Consult the SmartFusion2 and IGLOO2 High-Speed Serial Interface Configuration User Guide for details.

Figure 4-6. High Speed Serial Interface Configurator



4.1.2. Interfacing SERDESIF_n With The Initialization Logic Built For It (Ask a Question)

In the same SmartDesign the SERDESIF_n block is present, instantiate the Smart Design containing the SERDESIF_n initialization logic (SERDESIF_n_INIT), and do necessary interconnections to interface the SERDESIF_n block to the initialization logic. See the figure below to understand how the connections are made.

The following is a list of signal interconnections that need to be made to properly interface the SERDESIF_n to the initialization logic.

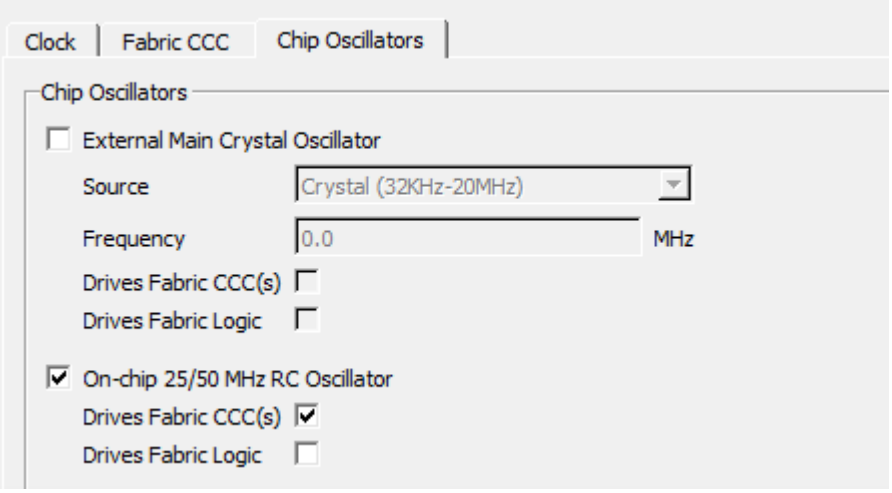
From Port or Bus Interface (BIF)/ Component	To Port/Bus Interface (BIF)/ Component
APB_S_PCLK/ SERDESIF_n	APB_S_PCLK/ initialization logic
APB_S_PRESET_N/ SERDESIF_n	APB_S_PRESET_N/ initialization logic
APB_SLAVE BIF/ SERDESIF_n	APB_S_INIT/ initialization logic
PHY_RESET_N/ SERDESIF_n	SDIFn_PHY_RESET_N/ initialization logic
CORE_RESET_N/ SERDESIF_n	SDIFn_CORE_RESET_N/ initialization logic
SPLL_LOCK/ SERDESIF_n	SDIFn_SPLL_LOCK/ initialization logic

4.1.2.1. 50MHz Oscillator Instantiation (Ask a Question)

CoreResetP needs to be clocked by the on-chip 50MHz RC oscillator. You must instantiate a 50MHz Oscillator and a FCCC for this purpose.

- Instantiate the Chip Oscillators core into the same SmartDesign the SERDESIF_n block is present. This core can be found in the Libero Catalog under Clock & Management.
- Configure this core such that the oscillator drives the Fabric CCC, as shown in figure below.

Figure 4-7. Chip Oscillators Configurator



- Click **OK**.
- Instantiate a FABCCC block configured as shown in the figures below.

Figure 4-8. FABCCC Configurator

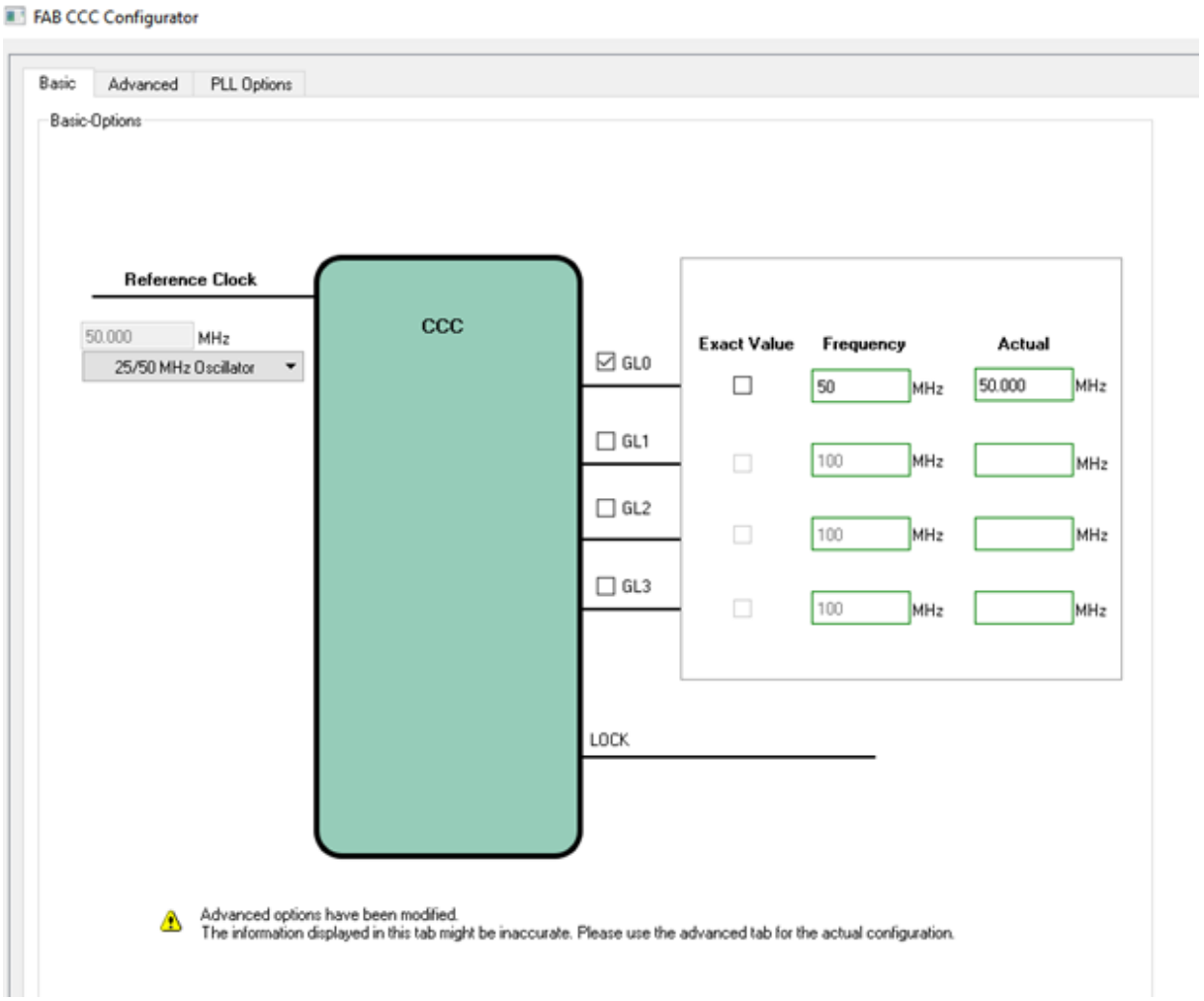
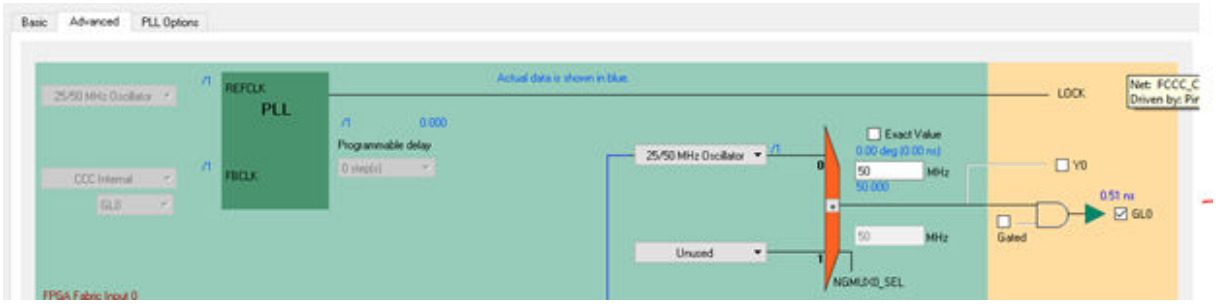


Figure 4-9. FABCCC Configurator - Advanced Tab



- Connect the RCOSC_25_50MHz_CCC_OUT output of the Oscillator to the RCOSC_25_50MHz_CCC_IN input of FCCC.
- Connect the GL0 output of the FCCC to the RCOSC_25_50MHz input of SERDESIF_n_INIT block (SERDESIF_n initialization logic).

4.1.2.2. System Reset (SYSRESET) Instantiation (Ask a Question)

The SYSRESET macro provides device level reset functionality to your design. The POWER_ON_RESET_N output signal is asserted/de-asserted whenever the chip is powered up or the external pin DEVRST_N is asserted/de-asserted. Instantiate the SYSRESET macro into the same

SmartDesign the SERDESIF_n block is present. This macro can be found in the Libero Catalog under Macro Library. No configuration of this macro is necessary. Drive the POWER_ON_RESET_N input of SERDESIF_n_INIT block (SERDESIF_n initialization logic with the POWER_ON_RESET_N output signal of this SYSRESET macro.

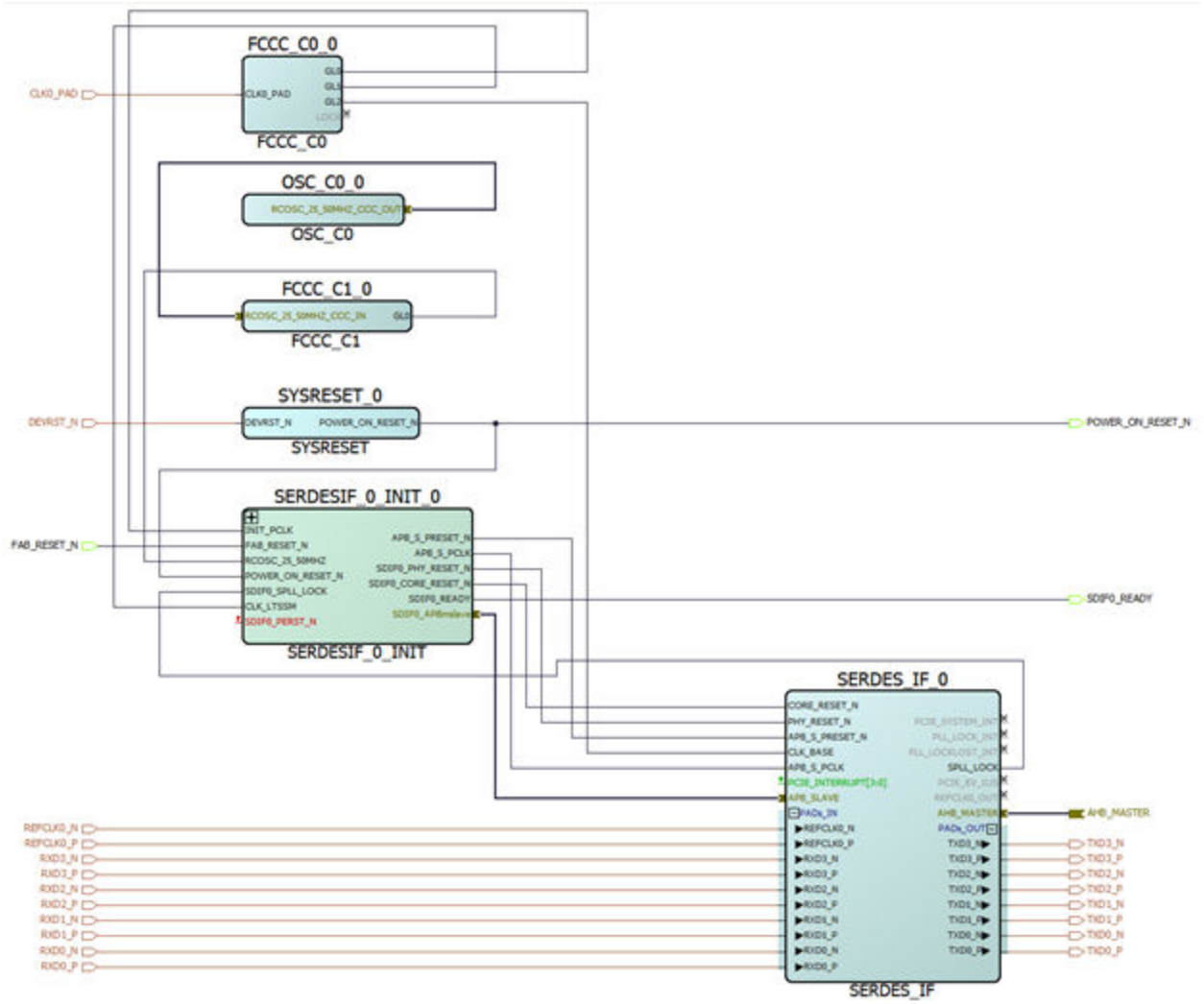
Figure 4-10. SYSRESET Macro

Apart from the above connections, do the following also:

- Promote the FAB_RESET_N pin from the initialization logic (SERDESIF_n_INIT_0 instance) to the top level (this is the warm reset). It is recommended to synchronize FAB_RESET_N to CLK_BASE.
- Promote the SDIFn_READY pin of the initialization logic to the top to monitor the status of the SERDESIF_n initialization.
- Promote the SDIFn_PERST_N pin of the initialization logic to the top or tie it high.
- Instantiate a second FABCCC block and do the following connections:
  - Drive the INIT_PCLK_25MHZ input pin of the initialization logic with the GLx of FABCCC block configured to 25MHz frequency.
  - Drive the CLK_BASE input pin of the SERDESIF_n block with the GLx of FABCCC block (configured to appropriate frequency).
  - Drive the CLK_LTSSM_125MHZ input pin of the initialization logic with the GLx of FABCCC block configured to 125MHz frequency.

Note: If more than 1 SERDESIF blocks are used in a design then you should put together separate initialization logic (using the CoreABC, CoreConfigP, and CoreResetP) for each one of them and do appropriate connections with the SERDESIF_n blocks.

Figure 4-11. Interfacing SERDESIF_n With The Initialization Logic



4.1.2.3. Continuing with the Design Flow (Ask a Question)

Next step is to integrate any user logic that you might have with the SERDESIF_n block and the initialization logic. Once you have done that, you can generate your top level SmartDesign. This will generate all files that are necessary to implement and simulate your design. You can then proceed with the rest of the Design Flow.

Note: After configuring all the desired SERDESIF_n registers in the SERDESIF configurator and upon generating the Smart Design component containing the SERDESIF_n block, the SERDESIF_n_init_abc.txt file will be generated to the disk. You will need to copy the contents of the SERDESIF_n_init_abc.txt file to the CoreABC Program tab and regenerate the initialization logic Smart Design component containing CoreABC.

4.2. Design using FDDR block (Ask a Question)

4.2.1. Building Standalone Initialization Logic For FDDR (Ask a Question)

In order to initialize the FDDR, you must create the initialization subsystem in the FPGA fabric. The FPGA fabric initialization subsystem moves data from the CoreABC program to the DDR configuration registers, manages the reset sequences required for the FDDR block to be operational and signals when the FDDR block is ready to communicate with the rest of your design. To create the initialization subsystem you must:

- Instantiate and configure CoreABC soft IP core.
- Load CoreABC with the initialization program generated to the FDDR_init_abc.txt file.
- Instantiate and configure the CoreConfigP and CoreResetP cores.
- Instantiate and configure the on-chip 50MHz RC oscillator.
- Instantiate the System Reset (SYSRESET) macro.
- Connect these components to the peripheral's (FDDR) configuration interface, clocks, resets and PLL lock ports.

4.2.1.1. CoreABC configuration (Ask a Question)

1. Create a new SmartDesign component (FDDR_INIT).
2. Instantiate CoreABC into your SmartDesign. This core can be found in the Libero Catalog (under Processors).
3. Double-click the core to open the configurator.
4. Configure the core as shown in the figure below.
   - Configure the data bus width to be 16.
   - Configure the maximum number of instructions to at least 256.
   - Configure to use AND and OR operations as optional instructions.
   - Configure Instruction Store to Hard (FPGA Tiles).
5. Copy the CoreABC program generated for FDDR from the FDDR_init_abc.txt file created under the <project_location>/../FABDDR_0/ directory, to the CoreABC Program tab. See the figure below.

Note: The FDDR_init_abc.txt file will be generated only when you generate the Smart Design containing the FDDR block. So after you've made all the connections and generated all the blocks (including FDDR), you will need to copy the contents of the FDDR_init_abc.txt file to the CoreABC Program tab and regenerate the initialization logic Smart Design component containing CoreABC. You may defer doing this until you completely configure your FDDR block and generate the SmartDesign component containing it.

Figure 4-12. CoreABC Configuration

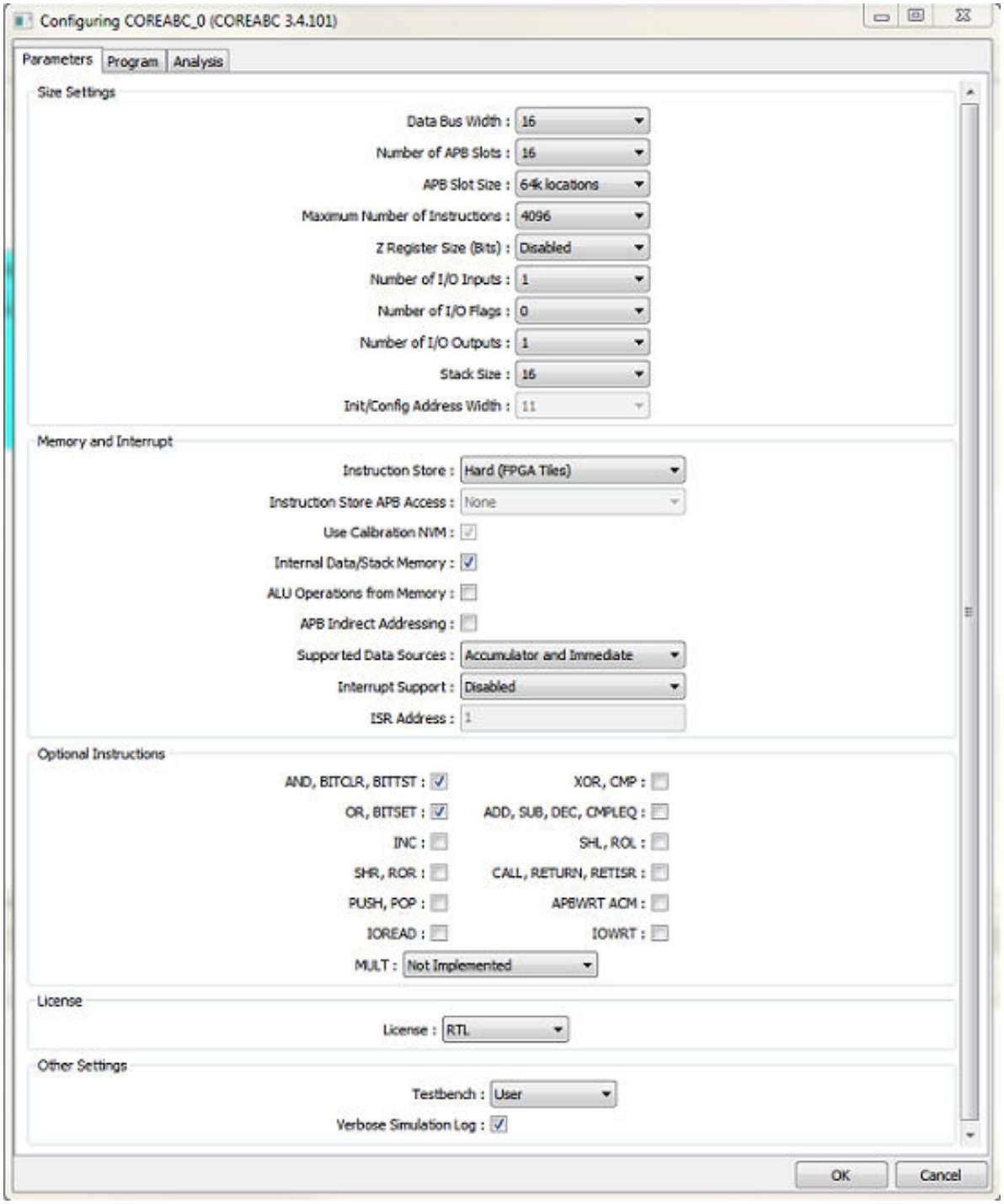
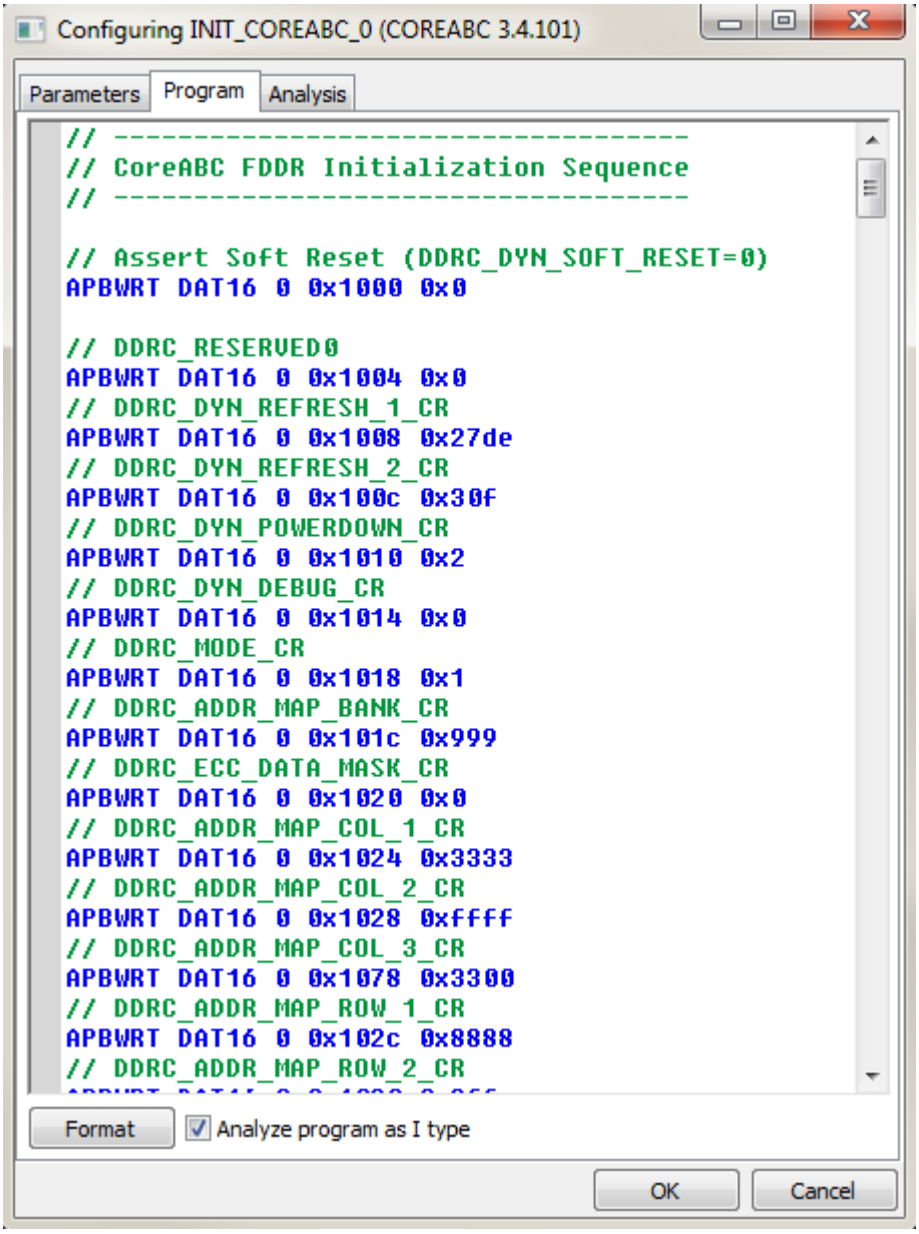


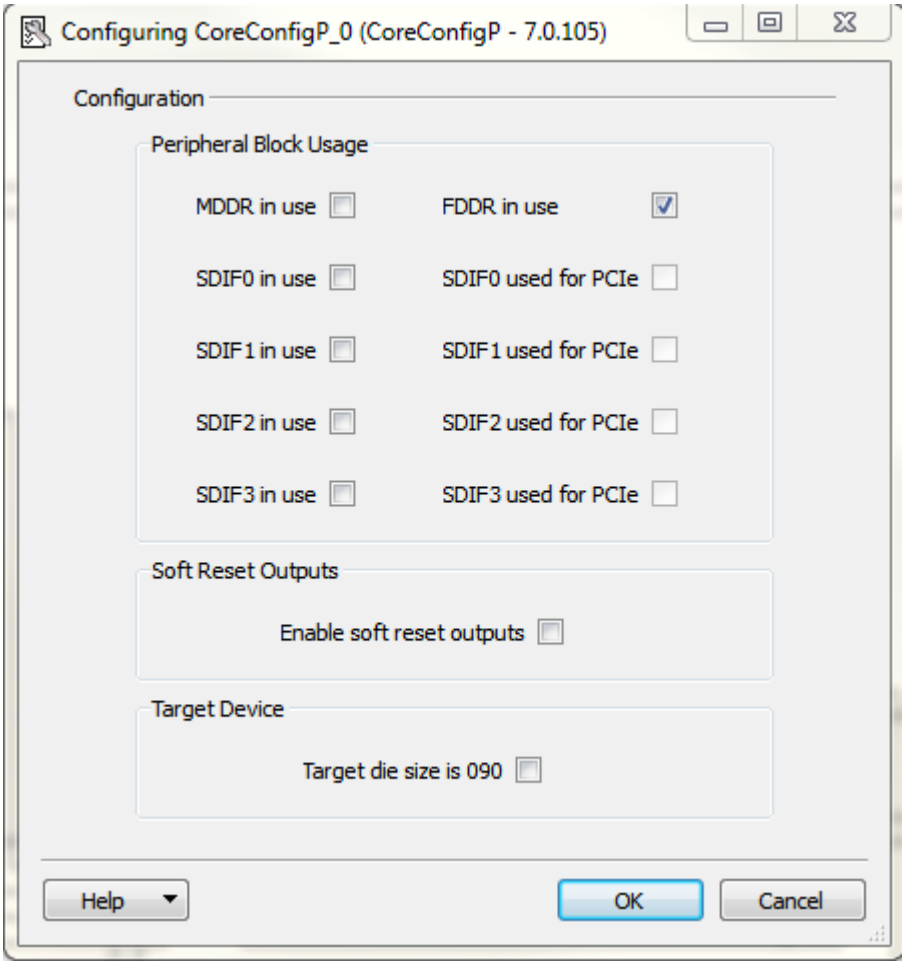
Figure 4-13. CoreABC Program For FDDR



4.2.1.2. CoreConfigP (Ask a Question)

1. Instantiate CoreConfigP into the same SmartDesign. This core can be found in the Libero Catalog (under Peripherals).
2. Double-click the core to open the configurator.
3. Configure the core to specify which peripherals need to be initialized.

Figure 4-14. CoreConfigP Dialog Box



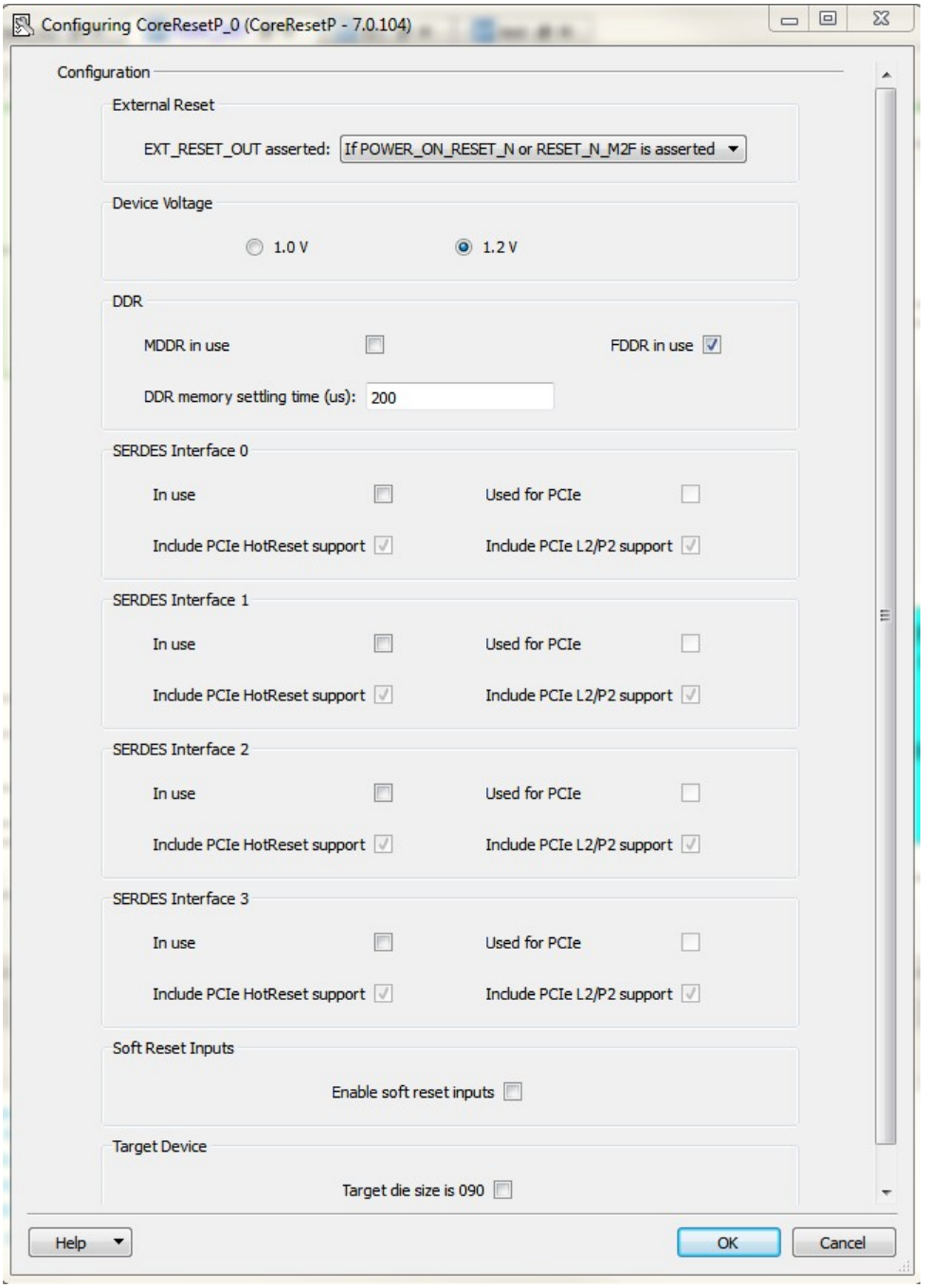
4.2.1.3. CoreResetP (Ask a Question)

1. Instantiate CoreResetP into the same SmartDesign. This core can be found in the Libero Catalog, under Peripherals.
2. Double-click the core inside the SmartDesign Canvas to open the Configurator.
3. Configure the core to:

- Specify the external reset behavior (EXT_RESET_OUT asserted). Choose one of four options:
  - EXT_RESET_OUT is never asserted
  - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) is asserted
  - EXT_RESET_OUT is asserted if FAB_RESET_N is asserted
  - EXT_RESET_OUT is asserted if power up reset (POWER_ON_RESET_N) or FAB_RESET_N is asserted
- Specify the Device Voltage. The selected value should match the voltage you selected in the Libero Project Settings dialog.
- Check the appropriate checkboxes to indicate which peripherals you are using in your design.
- Specify the external DDR memory setting time. Refer to the external DDR memory vendor datasheet to configure this parameter. 200us is a good default value for DDR2 and DDR3 memories running at 200MHz. This is a very important parameter to guarantee a working simulation and a working system on silicon. Incorrect value for the settling time may result in simulation errors.

Refer to the CoreResetP handbook for details on the options available to you in this configurator.

Figure 4-15. CoreResetP Configurator



4.2.2. Overall Connectivity Of The Initialization Logic (FDDR_INIT) (Ask a Question)

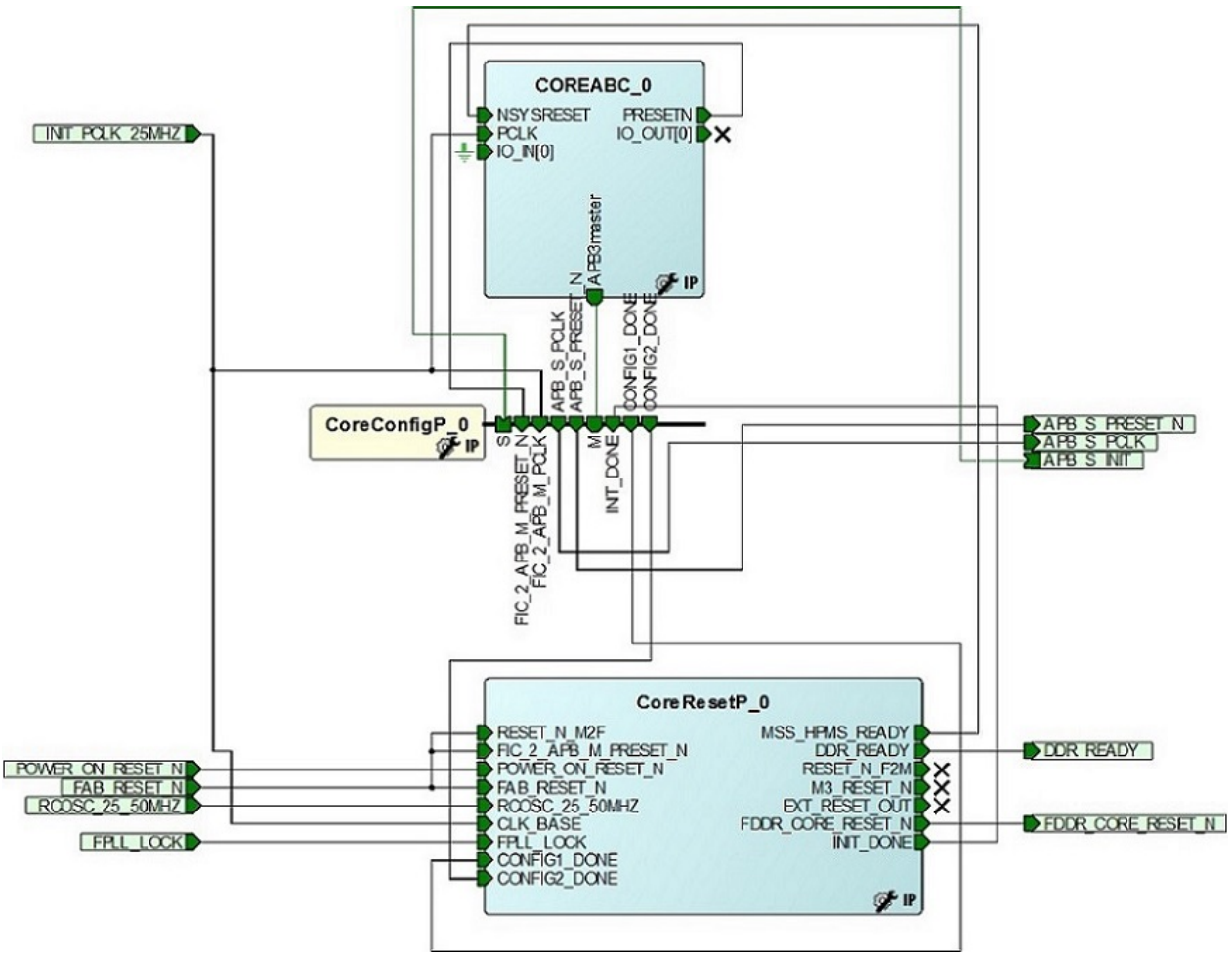
After you have instantiated and configured the 3 cores CoreABC, CoreConfigP and CoreResetP, appropriate connections have to be made to make the initialization logic operational. See the figure below to understand how the connections are made.

The following is a list of signals that need to be promoted to the top which will be needed when interfacing this initialization logic with the actual peripheral (FDDR).

- CoreConfigP:

  - APB_S_PRESET_N
  - APB_S_PCLK
  - APB_S_INIT (APB BIF FDDR_APBmslave)
- CoreResetP:
  - RCOSC_25_50MHZ
  - FAB_RESET_N
  - POWER_ON_RESET_N
  - DDR_READY
  - FDDR_CORE_RESET_N
  - FPLL_LOCK
- INIT_PCLK_25MHz (connecting together the PCLK of CoreABC, the FIC_2_APB_M_PCLK of CoreConfigP and the CLK_BASE of CoreResetP).

Figure 4-16. FDDR_INIT (FDDR Initialization Logic)



4.2.2.1. Instantiating And Configuring The FDDR Block (Ask a Question)

Create a Smart Design component and instantiate the FDDR block from the catalog window (from under the Memory and Controllers). Double-click the FDDR block in the SmartDesign canvas to open the configurator and configure the FDDR. The Fabric DDR (FDDR) controller must be configured dynamically (at runtime) to match the external DDR memory configuration requirements (DDR mode, PHY width, burst mode, ECC, etc.). Data entered in the FDDR configurator is written to

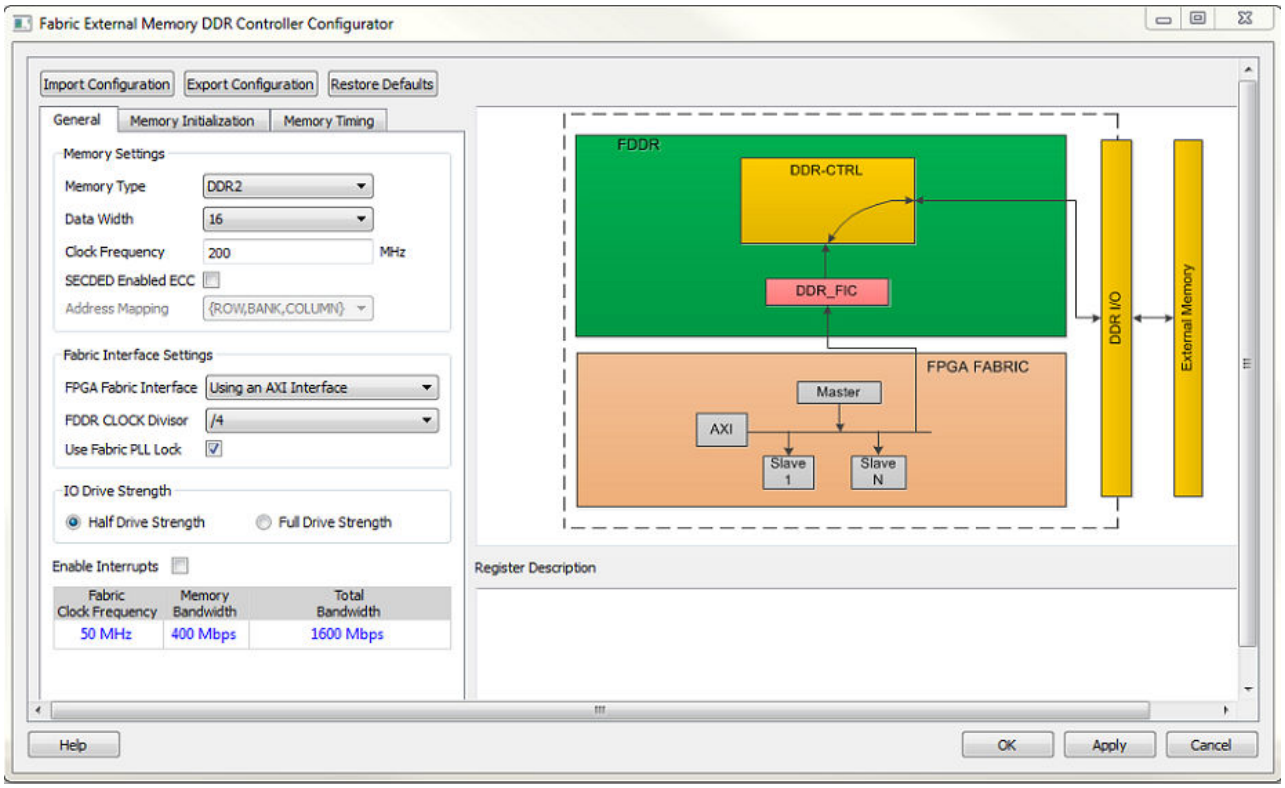
the DDR controller configuration registers by the CoreABC program. The Configurator has three different tabs for entering different types of configuration data:

- General data (DDR mode, Data Width, Clock Frequency, ECC, Fabric Interface, Drive Strength)
- Memory Initialization data (Burst Length, Burst Order, Timing Mode, Latency etc)
- Memory Timing data

Consult the specifications of your external DDR memory and configure the DDR Controller to match the requirements of your external DDR memory.

For details on DDR Configuration, refer to the IGLOO2 HPMS DDR Configuration User Guide.

Figure 4-17. Fabric DDR Configurator



4.2.3. Interfacing The FDDR With The Initialization Logic Built For It (Ask a Question)

In the same SmartDesign the FDDR block is present, instantiate the Smart Design containing the FDDR initialization logic (FDDR_INIT), and do necessary interconnections to interface the FDDR block to the initialization logic. See the figure below to understand how the connections are made.

The following is a list of signal interconnections that need to be made to properly interface the FDDR to the initialization logic.

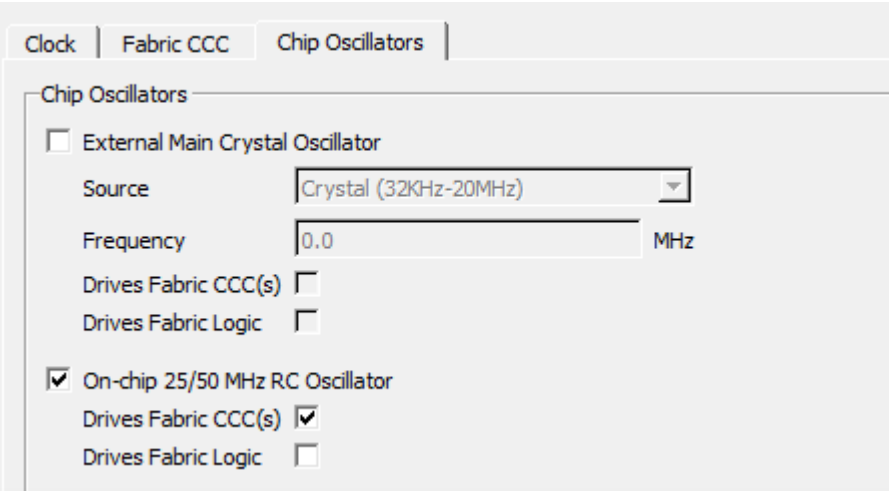
From Port or Bus Interface (BIF)/ Component	To Port/Bus Interface (BIF)/ Component
APB_S_PCLK/ FDDR	APB_S_PCLK/ initialization logic.
APB_S_PRESET_N/ FDDR	APB_S_PRESET_N/ initialization logic
APB_SLAVE BIF/ FDDR	APB_S_INIT/ initialization logic
FPLL_LOCK/ FDDR	FPLL_LOCK/ initialization logic
CORE_RESET_N / FDDR	FDDR_CORE_RESET_N / initialization logic

4.2.3.1. 50MHz Oscillator Instantiation (Ask a Question)

CoreResetP needs to be clocked by the on-chip 50MHz RC oscillator. You must instantiate a 50MHz Oscillator and a FCCC for this purpose.

- Instantiate the Chip Oscillators core into the same SmartDesign the FDDR block is present. This core can be found in the Libero Catalog under Clock & Management.
- Configure this core such that the oscillator drives the Fabric CCC, as shown in figure below.

Figure 4-18. Chip Oscillators Configurator



- Click **OK**.
- Instantiate an FABCCC block configured as shown in the figures below.

Figure 4-19. FAB CCC Configurator

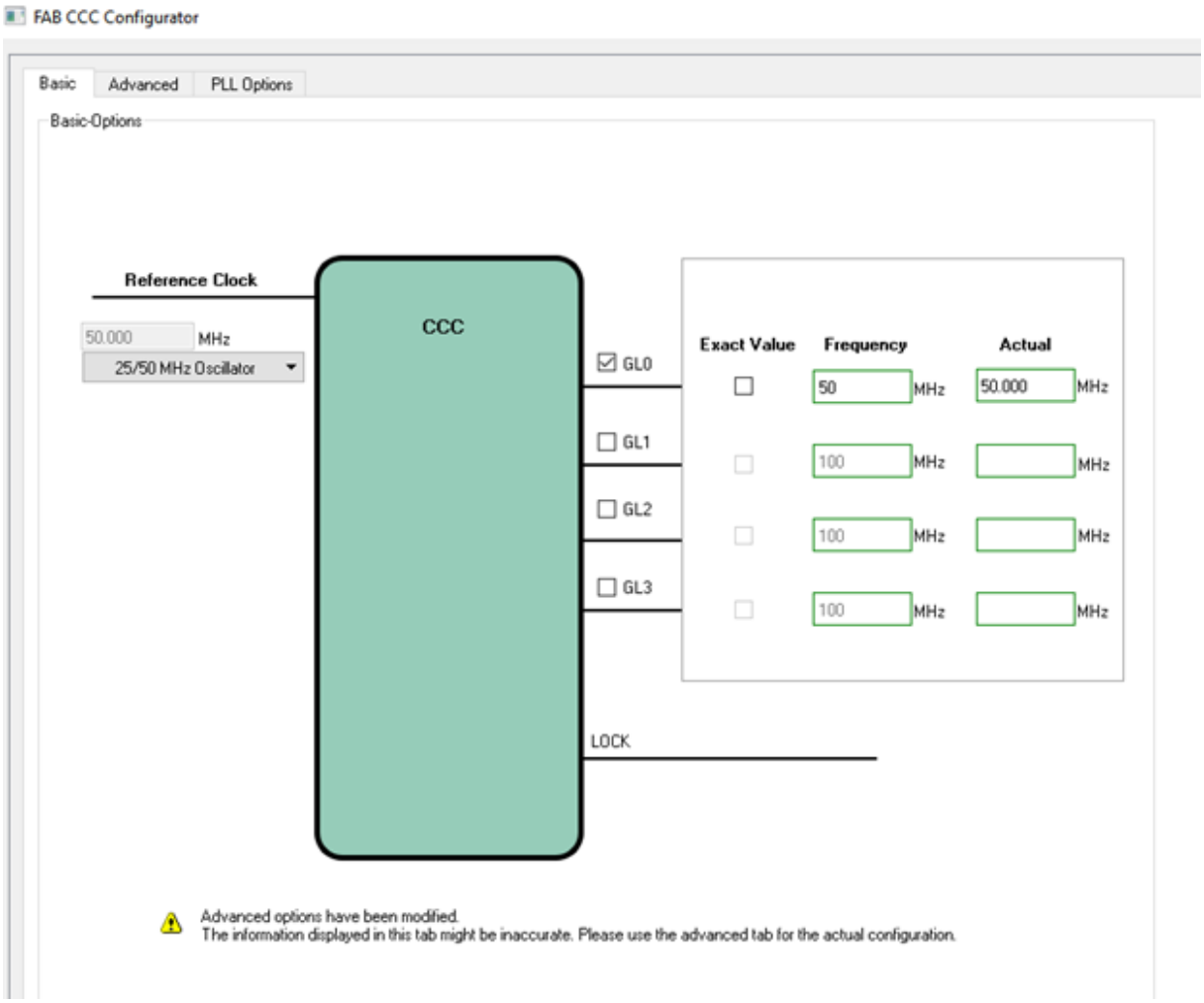
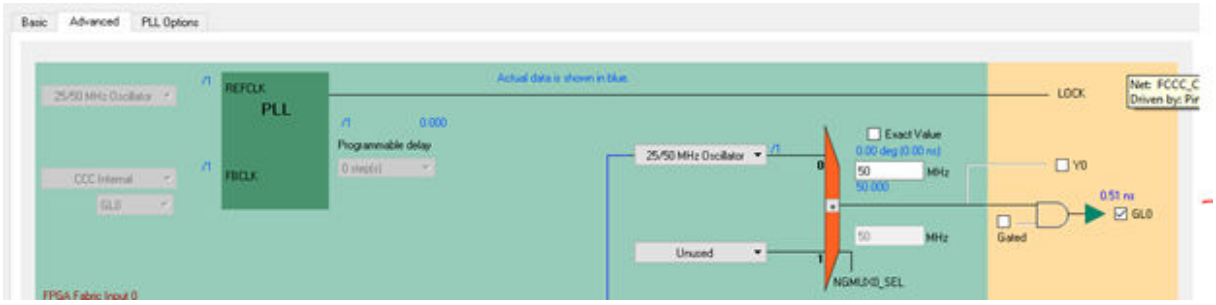


Figure 4-20. FAB CCC Configurator - Advanced Tab



- Connect the RCOSC_25_50MHz_CCC_OUT output of the Oscillator to the RCOSC_25_50MHz_CCC_IN input of FCCC.
- Connect the GL0 output of the FCCC to the RCOSC_25_50MHz input of FDDR_INIT block (FDDR_INIT_n initialization logic).

4.2.3.2. System Reset (SYSRESET) Instantiation (Ask a Question)

The SYSRESET macro provides device level reset functionality to your design. The POWER_ON_RESET_N output signal is asserted/de-asserted whenever the chip is powered up or the external pin DEVRST_N is asserted/de-asserted. Instantiate the SYSRESET macro into the same

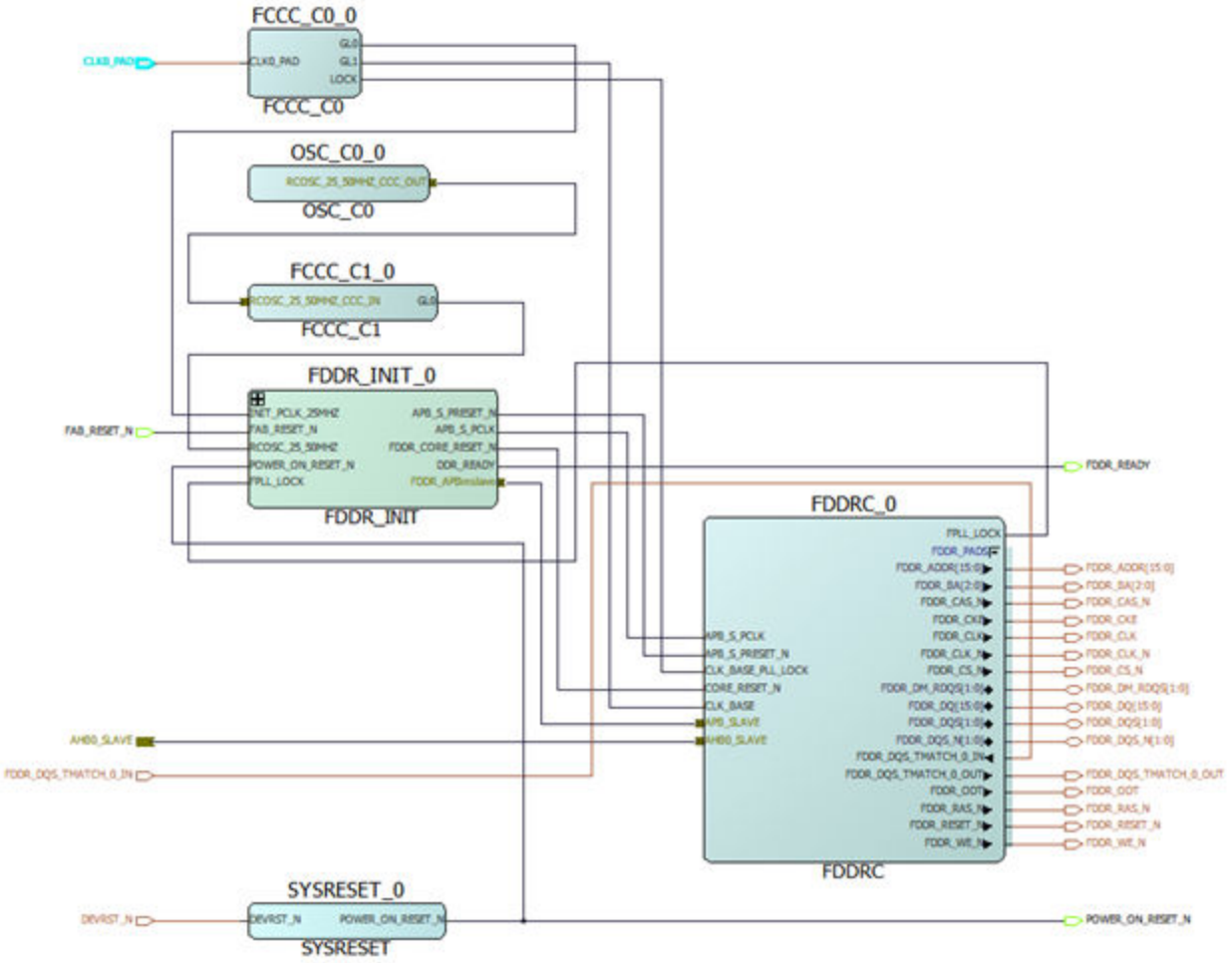
SmartDesign the FDDR block is present. This macro can be found in the Libero Catalog under Macro Library. No configuration of this macro is necessary. Drive the POWER_ON_RESET_N input of FDDR_INIT block (FDDR initialization logic with the POWER_ON_RESET_N output signal of this SYSRESET macro.

Figure 4-21. SYSRESET Macro

Apart from the above connections, do the following also:

- Promote the FAB_RESET_N pin from the initialization logic (FDDR_INIT_0 instance) to the top level (this is the warm reset). It is recommended to synchronize FAB_RESET_N to CLK_BASE.
- Promote the DDR_READY of the initialization logic to the top to monitor the status of the FDDR initialization.
- Instantiate a second FABCCC block and do the following connections:
  - Drive the INIT_PCLK_25MHZ input pin of the initialization logic with the GLx of FABCCC block configured to 25MHz frequency.
  - Drive the CLK_BASE input pin of the FDDR block with the GLx of FABCCC block (configured to appropriate frequency).
  - Drive the CLK_BASE_PLL_LOCK input pin of the FDDR block with the LOCK pin of the FABCCC block.

Figure 4-22. Interfacing FDDR With The Initialization Logic



4.2.3.3. Continuing with the Design Flow (Ask a Question)

Next step is to integrate any user logic that you might have with the FDDR block and the initialization logic. Once you have done that, you can generate your top level SmartDesign. This will generate all files that are necessary to implement and simulate your design. You can then proceed with the rest of the Design Flow.

Note: After configuring all the desired FDDR registers in the FDDR configurator and upon generating the Smart Design component containing the FDDR block, the FDDR_init_abc.txt file will be generated to the disk. You will need to copy the contents of the FDDR_init_abc.txt file to the CoreABC Program tab and regenerate the initialization logic Smart Design component containing CoreABC.

5. Simulating the Design (Ask a Question)

Unlike the normal flow (Standalone Initialization OFF) where the peripheral initialization involves various `*_init.reg` files to mimic the initialization in simulations, no such files are required in case of the Standalone Initialization mode ON.

Note: Unlike the normal flow, `ENVM_init.mem` file that's created upon invoking simulations doesn't have any peripheral's register configuration information in case of the Standalone Initialization mode ON. It only has the data corresponding to the ENVM clients specified in the design.

The CoreABC program is solely responsible for the peripheral initialization both in simulations and on board (device).

When you generate a Smart Design component containing SERDESIF_n (configured in PCIe mode), then the following files are generated in the <project dir>/simulation directory:

- `SERDESIF_n_user.bfm` - Contains the user commands. Edit this file to enter your BFM commands that would exercise the SERDESIF_n PCIe.

6. Revision History (Ask a Question)

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

Revision	Date	Description
A	12/2023	The document was migrated to the Microchip template.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3,

ISBN: 978-1-6683-3600-7

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office	**Australia - Sydney**	**India - Bangalore**	**Austria - Wels**
2355 West Chandler Blvd.	Tel: 61-2-9868-6733	Tel: 91-80-3090-4444	Tel: 43-7242-2244-39
Chandler, AZ 85224-6199	**China - Beijing**	**India - New Delhi**	Fax: 43-7242-2244-393
Tel: 480-792-7200	Tel: 86-10-8569-7000	Tel: 91-11-4160-8631	**Denmark - Copenhagen**
Fax: 480-792-7277	**China - Chengdu**	**India - Pune**	Tel: 45-4485-5910
Technical Support:	Tel: 86-28-8665-5511	Tel: 91-20-4121-0141	Fax: 45-4485-2829
www.microchip.com/support	**China - Chongqing**	**Japan - Osaka**	**Finland - Espoo**
Web Address:	Tel: 86-23-8980-9588	Tel: 81-6-6152-7160	Tel: 358-9-4520-820
www.microchip.com	**China - Dongguan**	**Japan - Tokyo**	**France - Paris**
Atlanta	Tel: 86-769-8702-9880	Tel: 81-3-6880- 3770	Tel: 33-1-69-53-63-20
Duluth, GA	**China - Guangzhou**	**Korea - Daegu**	Fax: 33-1-69-30-90-79
Tel: 678-957-9614	Tel: 86-20-8755-8029	Tel: 82-53-744-4301	**Germany - Garching**
Fax: 678-957-1455	**China - Hangzhou**	**Korea - Seoul**	Tel: 49-8931-9700
Austin, TX	Tel: 86-571-8792-8115	Tel: 82-2-554-7200	**Germany - Haan**
Tel: 512-257-3370	**China - Hong Kong SAR**	**Malaysia - Kuala Lumpur**	Tel: 49-2129-3766400
Boston	Tel: 852-2943-5100	Tel: 60-3-7651-7906	**Germany - Heilbronn**
Westborough, MA	**China - Nanjing**	**Malaysia - Penang**	Tel: 49-7131-72400
Tel: 774-760-0087	Tel: 86-25-8473-2460	Tel: 60-4-227-8870	**Germany - Karlsruhe**
Fax: 774-760-0088	**China - Qingdao**	**Philippines - Manila**	Tel: 49-721-625370
Chicago	Tel: 86-532-8502-7355	Tel: 63-2-634-9065	**Germany - Munich**
Itasca, IL	**China - Shanghai**	**Singapore**	Tel: 49-89-627-144-0
Tel: 630-285-0071	Tel: 86-21-3326-8000	Tel: 65-6334-8870	Fax: 49-89-627-144-44
Fax: 630-285-0075	**China - Shenyang**	**Taiwan - Hsin Chu**	**Germany - Rosenheim**
Dallas	Tel: 86-24-2334-2829	Tel: 886-3-577-8366	Tel: 49-8031-354-560
Addison, TX	**China - Shenzhen**	**Taiwan - Kaohsiung**	**Israel - Ra'anana**
Tel: 972-818-7423	Tel: 86-755-8864-2200	Tel: 886-7-213-7830	Tel: 972-9-744-7705
Fax: 972-818-2924	**China - Suzhou**	**Taiwan - Taipei**	**Italy - Milan**
Detroit	Tel: 86-186-6233-1526	Tel: 886-2-2508-8600	Tel: 39-0331-742611
Novi, MI	**China - Wuhan**	**Thailand - Bangkok**	Fax: 39-0331-466781
Tel: 248-848-4000	Tel: 86-27-5980-5300	Tel: 66-2-694-1351	**Italy - Padova**
Houston, TX	**China - Xian**	**Vietnam - Ho Chi Minh**	Tel: 39-049-7625286
Tel: 281-894-5983	Tel: 86-29-8833-7252	Tel: 84-28-5448-2100	**Netherlands - Drunen**
Indianapolis	**China - Xiamen**		Tel: 31-416-690399
Noblesville, IN	Tel: 86-592-2388138		Fax: 31-416-690340
Tel: 317-773-8323	**China - Zhuhai**		**Norway - Trondheim**
Fax: 317-773-5453	Tel: 86-756-3210040		Tel: 47-72884388
Tel: 317-536-2380			**Poland - Warsaw**
Los Angeles			Tel: 48-22-3325737
Mission Viejo, CA			**Romania - Bucharest**
Tel: 949-462-9523			Tel: 40-21-407-87-50
Fax: 949-462-9608			**Spain - Madrid**
Tel: 951-273-7800			Tel: 34-91-708-08-90
Raleigh, NC			Fax: 34-91-708-08-91
Tel: 919-844-7510			**Sweden - Gothenberg**
New York, NY			Tel: 46-31-704-60-40
Tel: 631-435-6000			**Sweden - Stockholm**
San Jose, CA			Tel: 46-8-5090-4654
Tel: 408-735-9110			**UK - Wokingham**
Tel: 408-436-4270			Tel: 44-118-921-5800
Canada - Toronto			Fax: 44-118-921-5820
Tel: 905-695-1980			
Fax: 905-695-2078			