



Synaptic Labs'

xSPI-Multi-Bus Memory Controller (xSPI-xSPI-MBMC)

Tutorial

T001A: A Qsys based Nios II Reference design with a simple self test of the Flash Memory using SLL xSPI-MBMC IP

This tutorial describes a simple reference design for SLL xSPI-MBMC IP targeted specifically to Trenz MAX10 / Cruvi Reference board (CR00100-01). Most xSPI-MBMC customers using any of these boards board will want to start with this tutorial. This tutorial describes key aspects of a pre-configured .qsys reference project and then walks through the process of generating and compiling that .Qsys project. This tutorial then describes how to compile the example Nios II source code, integrate the firmware into the FPGA bitstream and then run the reference design on the development board.

Table of Contents

Synaptic Labs' xSPI-Multi-Bus Memory Controller (xSPI-xSPI-MBMC) Tutorial.....	1
Synaptic Labs 2020 info@synaptic-labs.com 8 th March 2021 V1.1 page 38.....	4
1.0 Contents of the reference project.....	5
2.0 Open the reference Quartus Project.....	6
2.1 Check the correct FPGA device is selected	6
3.0 Open the reference Qsys project	8
4.0 Explore and configuring the reference Qsys project	9
4.1 Components employed in the reference project	9
4.2 Nios II/f processor configuration.....	10
4.3 Configuring SLL xSPI-Multi-Bus Memory Controller	11
4.5 Configuration of Altera's SDRAM.....	20
5.0 Generating the Qsys Design	21
6.0 Demo test Application	22
6.1 Flash test Application File Structure.....	23
7.0 Preparing the firmware.....	24
7.1 Open the NIOS II Software Built Tools for Eclipse	24
7.2 Create a simple application and BSP.....	25
7.3 Configure the Board Support Package (BSP).....	28
7.4 Generate the BSP and clean the project.....	32
7.5 Copy the memory testing source code.....	33
7.6 Build the Nios II Application.....	34
8.0 Synthesize and assemble the Design	35
9.0 Program the FPGA Bitstream into the FPGA device	36
10.0 Run the application from within Nios II SBT	37

Set-Up Requirements:

Step 1: Obtain core materials

- Download and install Quartus Prime Standard/Lite 18.1 on your PC, please ensure that your PC meets the required minimum specification.
- Create a folder/directory for your work. We suggest: [C:\SLL_lab](#)
- Ask Synaptic-Labs for the reference design Project (supplied with EMxxLX user guide)
- Extract to: C:\SLL_lab\

It is suggested to shorten folders that are too long. This is due to the path length limitation when the Quartus is running in Windows. Rename any long folders .

Step 2: License Setup

- Next you need to apply for Synaptic Labs' xSPI-Multi-Bus Memory Controller license.
- You can skip this step if you already installed the license at some earlier stage.
- Contact S/Labs for a license (info@synaptic-labs.com).

Step 3 : Install xSPI-MBMC Qsys Component into the project IP Folder

- In this tutorial we assume that SLL xSPI-Multi-Bus Memory Controller (xSPI-MBMC) will be located in the Project directory.
- Other Qsys component installation methods are described in the above mentioned installation Guides.
- Contact Synaptic Labs' for the latest version of Synaptic Labs' xSPI-MBMC IP
- Extract to the project/ip directory :
 - C:\SLL_lab\CR00100-01_project_18V_J1_xxx\ip

Step 4: MAX 10 LP Development Board Cruvi Channels



The Trenz CR00100-01 board contains a power controller that sets the voltage level on the CRUVI channel. Ensure that the EP53A7HQI power controller signals in the top level verilog project file are set to the voltage level required by the memory devices mounted on the Cruvi channels. Check the schematic for more info.

EP53A7HQI	pwr_ctr_vid	pwr_ctr_vid	pwr_ctr_vid	Cruvi J1 Voltage
	2	1	0	
	GND	GND	GND	3.3V
	VCC	GND	GND	2.5V
	VCC	VCC	VCC	3.3V

```
//-----  
//Power Control  
//# EP53A7HQI power and VID control  
//# VID=000 3.3V  
//# VID=100 2.5V  
//# VID=111 1.8V  
//-----  
assign pwr_ctr_enable = 1'b1;  
assign pwr_ctr_vid0   = 1'b1;  
assign pwr_ctr_vid1   = 1'b1;  
assign pwr_ctr_vid2   = 1'b1;
```

1.0 Contents of the reference project

Synaptic Labs' xSPI-Multi-Bus Memory Controller (xSPI-MBMC) Reference design projects includes the following files and directories:

- CR00100-01_project_18V_J1_xxx folder contains the Quartus Prime and Qsys project files for the first reference project.
- The CR00100-01_project_18V_J1_xxx→ ip folder will contain SLL xSPI-MBMC encrypted ip (refer to Setup requirements: step3 for more info)
- The CR00100-01_project_18V_J1_xxx → software folder is the workspace folder for Eclipse
- The CR00100-01_project_18V_J1_xxx →source folder contains the source code for the FlashTest program as used in this xSPI-MBMC Tutorial 001.

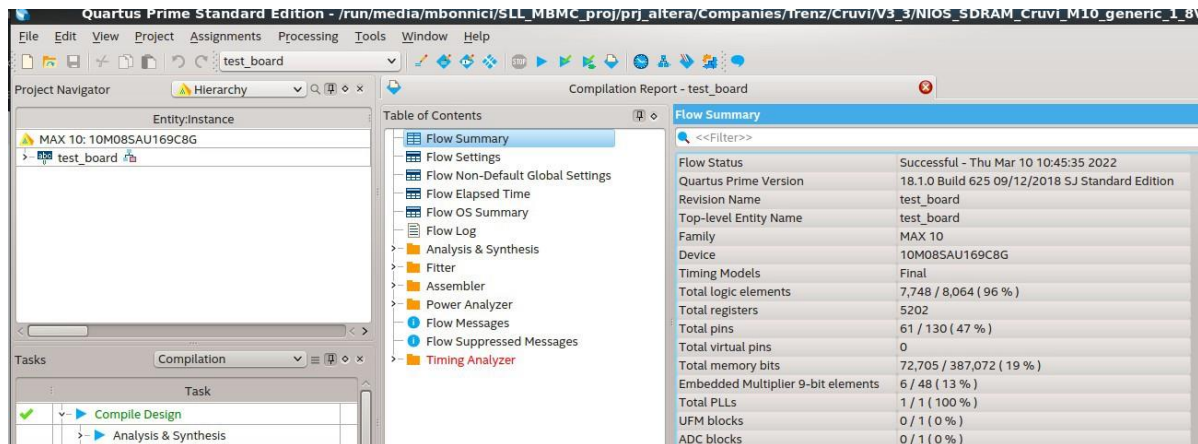
Note: Synaptic Labs' xSPI-Multi-Bus Memory Controller (xSPI-MBMC) IP can ONLY be simulated with Altera's Modelsim Simulator. Please contact Synaptic Labs for a simulation model if required.

2.0 Open the reference Quartus Project

In the menu bar of Quartus Prime, select File → Open Project...

- Select the file test_board.qpf in the project directory
- Click the [Open] button.

2.1 Check the correct FPGA device is selected



Every Quartus project is targeted to a specific FPGA device.

The Trenz CR00100-01 board employs the [10M08SAU169C8G](#)

device. If you need to change the FPGA device for your specific

board:

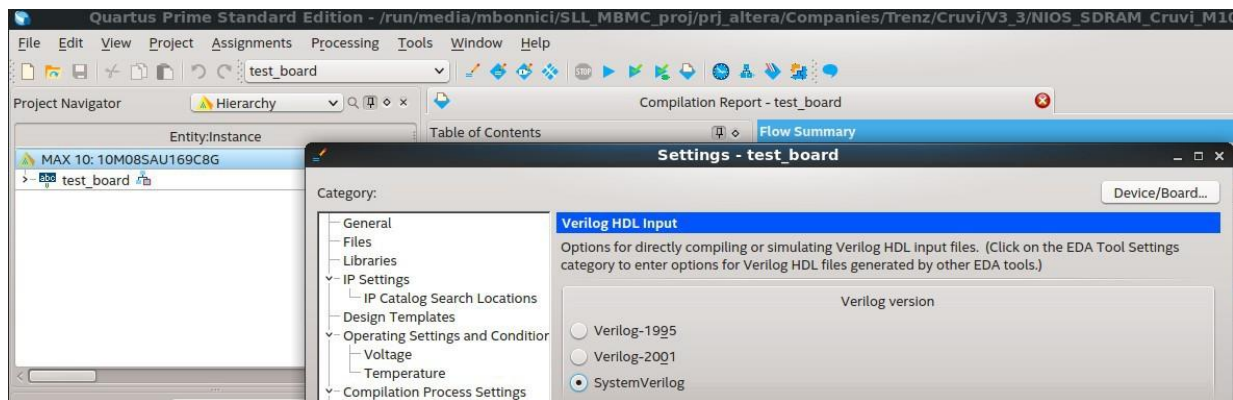
- Ensure that there are no instances of the Qsys application running.
- Right click on the device name.
- Select “Devices...” in the pop up window.
- A new window will open. Select the “Device” tab.
- Copy the required device name into the “Name filter:” field.
- A popup window will ask: “Do you want to remove all location assignments?” Click on the [No] button.
- Select the requested device in the “available devices:” field so that it is highlighted in blue.
- Then click on the [Okay] button.

The Quartus project and the Qsys project are now configured for the FPGA device you selected.

Enable System Verilog in Quartus

You will need to enable System Verilog in the Compiler Settings. This can be done by

- Right click on the device name and select Settings.
- Locate Compiler Settings / Verilog HDL Input
- Select System Verilog



3.0 Open the reference Qsys project

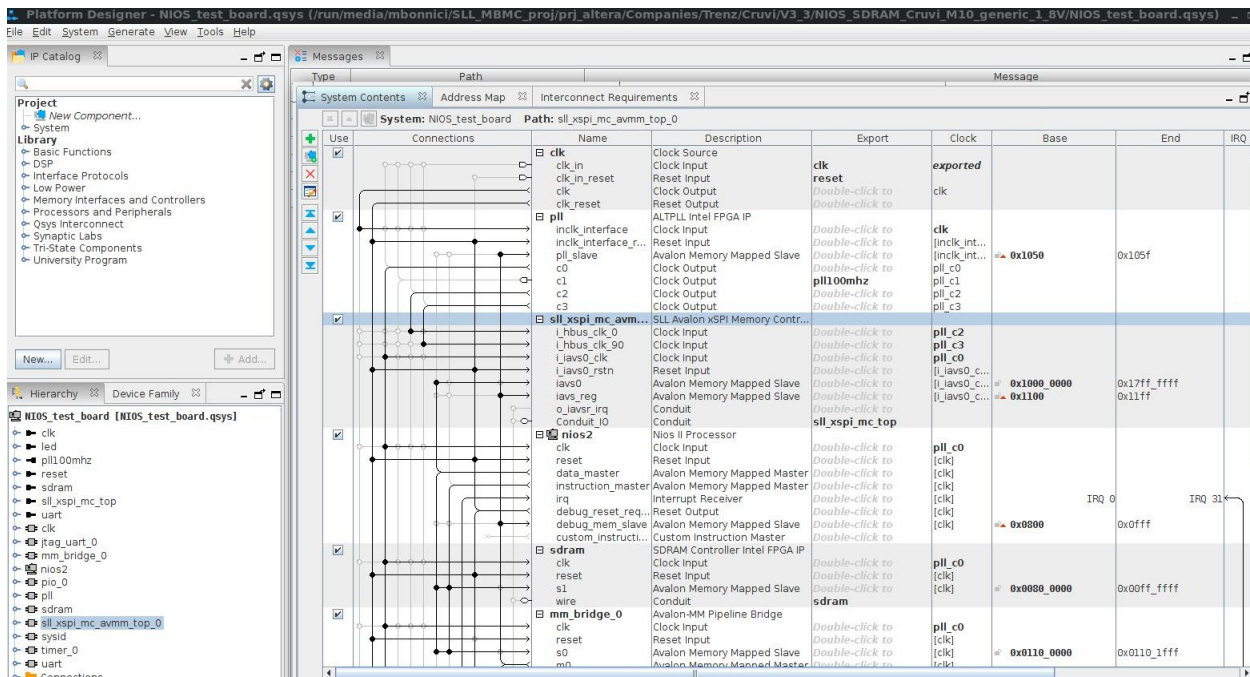
In the menu bar of Quartus Prime, select Tools→Qsys

- Select the file Nios_test_board.qsys in the project directory
- Click the [Open] button.

4.0 Explore and configuring the reference Qsys project

4.1 Components employed in the reference project

The reference Qsys project in this tutorial employs a NiosII/f processor, Synaptic Labs' Multi-Bus Memory Controller (xSPI-MBMC) IP, Altera's SDRAM controller to store code and data, Altera's DMA controllers and various peripherals such as Altera's JTAG UART and timer modules as illustrated below. All these Qsys components are connected together.



4.2 Nios II/f processor configuration

In this example, the Nios II/f Reset and Exception vectors are mapped to SDRAM memory as illustrated below. This means that the Nios II/f processor will look for the boot code and exception handling / interrupt code in the SDRAM memory module.

The screenshot shows the 'Vectors' tab in the Nios II/f processor configuration window. It contains three sections: 'Reset Vector', 'Exception Vector', and 'Fast TLB Miss Exception Vector'. Each section has three fields: 'Reset vector memory:', 'Reset vector offset:', and 'Reset vector:'. The 'Reset vector memory' is set to 'sdram.s1' for all three. The 'Reset vector offset' is '0x00000000' for the Reset Vector and '0x00000020' for the Exception Vector. The 'Reset vector' is '0x00800000' for the Reset Vector and '0x00800020' for the Exception Vector. The 'Fast TLB Miss Exception Vector' section has 'Fast TLB Miss Exception vector memory' set to 'None', and both 'Fast TLB Miss Exception vector offset' and 'Fast TLB Miss Exception vector' set to '0x00000000'.

The screenshot shows the 'Caches and Memory Interfaces' tab in the Nios II/f processor configuration window. It contains four sections: 'Instruction Cache', 'Flash Accelerator', 'Data Cache', and 'Tightly-coupled Memories'. The 'Instruction Cache' section has 'Size' set to '2 Kbytes' and 'Add burstcount signal to instruction_master' set to 'Enable'. The 'Flash Accelerator' section has 'Line Size' set to 'None' and 'Number of Cache Lines' set to '2'. The 'Data Cache' section has 'Size' set to '2 Kbytes', 'Victim buffer implementation' set to 'RAM', 'Add burstcount signal to data_master' set to 'Enable', and a checked box for 'Use most-significant address bit in processor to bypass data cache'. The 'Tightly-coupled Memories' section has 'Number of tightly coupled instruction master ports' and 'Number of tightly coupled data master ports' both set to 'None'. The 'Peripheral Region' section has 'Size' set to 'None' and 'Base Address' set to '0x00000000'.

As illustrated below, the instruction and data caches of the Nios II/f core have both been set to 2KB or 4Kbytes in size to accelerate software performance. The instruction and data caches have both been configured with their burstcount signal enabled so that both caches issue burst memory transfer requests. This is done because: (a) the HyperBus protocol employs burst memory transfer requests with closed page mode of operation; and (b) SLL's xSPI-MBMC employs an Avalon interface with burst mode of operation.

4.3 Configuring SLL xSPI-Multi-Bus Memory Controller

Synaptic Labs' xSPI-Multi-Bus Memory Controller has been pre-configured in this reference project. The user does not need to change any settings in the configuration. In this section we will describe various configuration options used in the design.

SLL Avalon xSPI Memory Controller IP (xSPI-MC)
sll_xspi_mc_avmm_top

Block Diagram
Show signals

sll_xspi_mc_avmm_top_0

- i hbus_clk_0 clock
- i hbus_clk_90 clock
- i iavs0_clk clock
- i iavs0_rstn reset
- iavs0 avalon
- Conduit IO conduit

sll_xspi_mc_avmm_top

Info Master Configuration Clock and PLL Configuration IAVS0 Configuration Device 0 Info Device 1 Info

Synaptic Laboratories Limited (SLL)

License And Confidentiality Agreement

By Using SLL xSPI Memory Controller (xSPI-MC) IP, You acknowledge that You have read the accompanying License and Confidentiality Agreement, understand it, accept it and agree to be bound by all its Terms.

SLL xSPI Memory Controller (xSPI-MC)

Synaptic Laboratories Ltd (SLL) xSPI Memory Controller (xSPI-MC) provisions a single Memory memory channel with support for up to 2 Memory memory devices.

You can instantiate multiple instances of SLL xSPI-MC IP in your design.

This version of SLL xSPI-MC IP supports both HyperRAM and HyperFlash devices.

Visit www.synaptic-labs.com to access free documentation, free reference designs and to download the latest version of this IP.

Installation of License Key

The accompanying License Credential with embedded License Key provides a description of the License Type, Target Application, Supported Device/s, and what capabilities have been enabled/disabled under Your License. The full terms and conditions contained in the License Credential with embedded License Key are included by reference into the License and Confidentiality Agreement.

Your License Credential with embedded License Key is typically found:

- (a) in the xSPI-MC IP folder; or
- (b) as an attachment sent to you by e-mail.

That License Key must be installed (copied and pasted) in Quartus Prime to synthesise designs.

There are two options to configure the memory device selection mode in SLL xSPI-MBMC IP .

1) Fixed/static Mode implementation

In this mode, the user configures the controller for the exact memory device present on the Cruvi adapter.

In the **Master Configuration** Window, the **Memory Channel 0 /1 Configuration** contain a list of memory devices. These lists are grouped according to memory protocols.

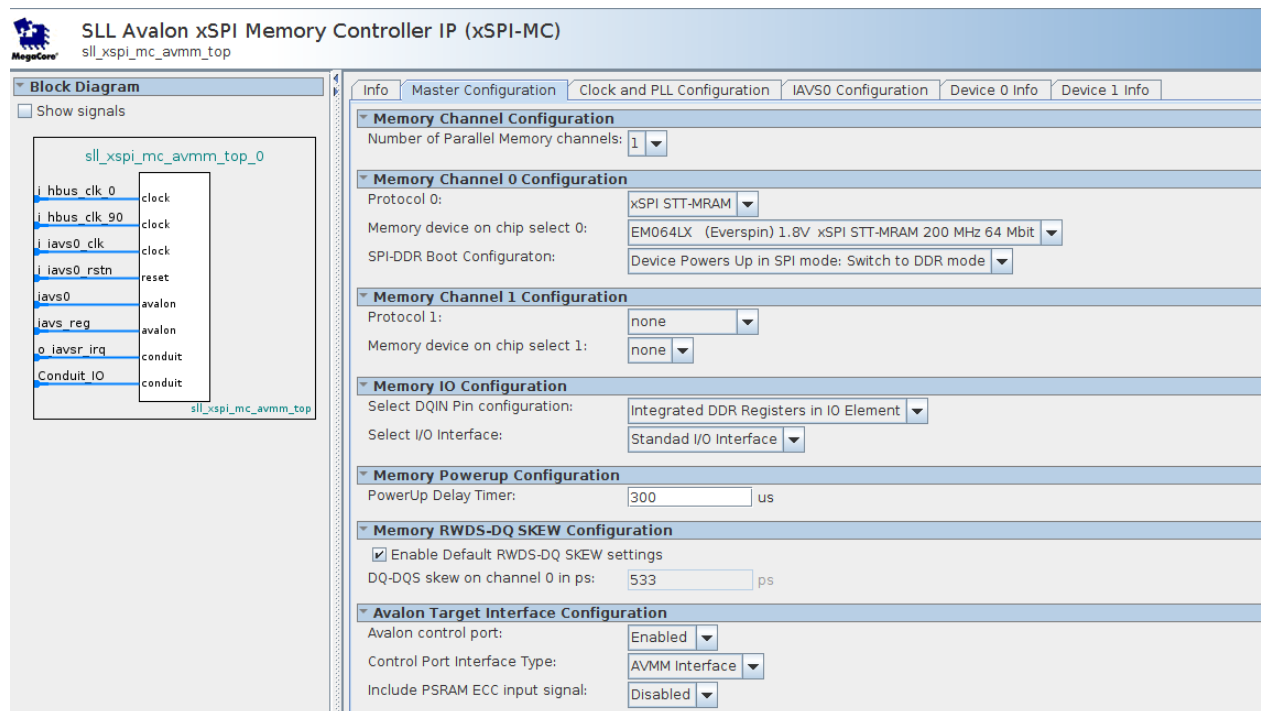
For example, to connect an Everspin EM064LX Flash memory (Everspin EMLX family) on chip select 0, the user needs to set the follow:

The Memory Channel 0 Configuration field is set to :

- **Protocol1** as [SPI STT-MRAM](#)
- **Memory device on chip select 0** as [EM06LX](#).
- **SPI-DDR Boot Configuration** as [Device Powers up in SPI : Switch to DDR](#)

The Memory Channel 1 Configuration field is set to :

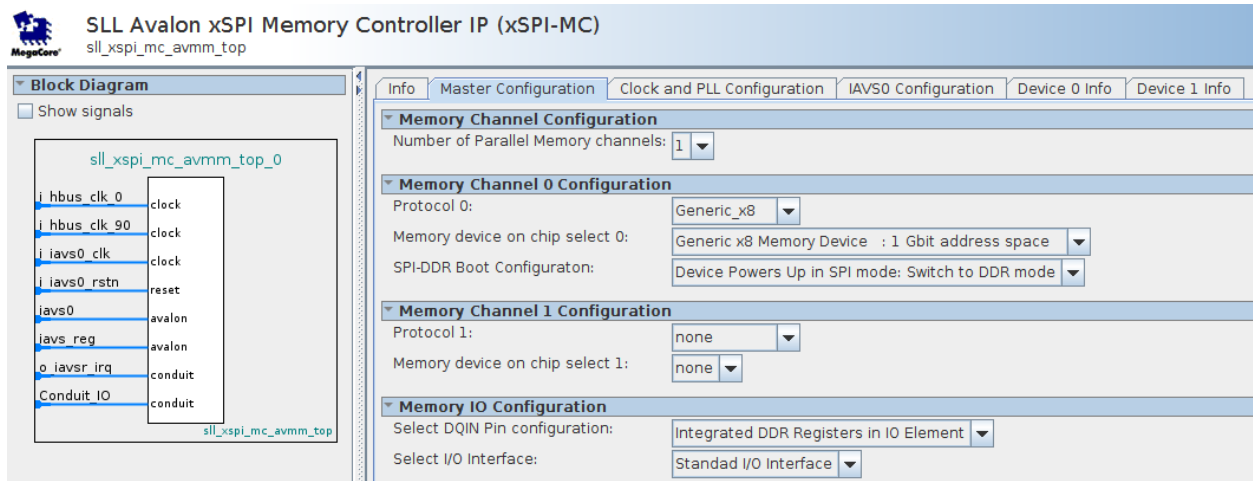
- **Protocol1** as [None](#)



After a power up delay, SLL xSPI-MBMC will automatically program the Flash or PSRAM configuration registers. The user does NOT need to program the memory controller in software. This implementation results in a much lower resource utilization in FPGA.

2) Generic Mode implementation

In this mode, the user configures the controller in Generic mode. This means that the user can connect any memory device supported by the controller. The memory devices will later on be configured through software.



Check xSPI-MBMC user manual for more information.

In this project, the memory controller is configured as a static memory device controller. This means that only the selected memory device can be connected on the CRUVI channel.

The user will **NOT** need to configure the controller in software. Check the user manual for more information or study the software application example provided with this tutorial.

In static mode, the user cannot test different devices without the need to re-compile the Quartus design.

In this example, we will use this port to test Everspin EMLX Flash memory device.

In the "Master Configuration" tab

.....for this project CR00100-01 MAX10 Kit

The Memory Channel 0 Configuration field is set to :

- **Protocol1** as **SPI STT-MRAM**
- **Memory device on chip select 0** as **EM06LX**.
- **SPI-DDR Boot Configuration** as **Device Powers up in SPI : Switch to DDR**

The Memory Channel 1 Configuration field is set to :

- **Protocol1** as **None**

SLL Avalon xSPI Memory Controller IP (xSPI-MC)
sll_xspi_mc_avmm_top

Block Diagram
Show signals

Info Master Configuration Clock and PLL Configuration IAVS0 Configuration Device 0 Info Device 1 Info

Memory Channel Configuration
Number of Parallel Memory channels: 1

Memory Channel 0 Configuration
Protocol 0: xSPI STT-MRAM
Memory device on chip select 0: EM064LX (Everspin) 1.8V xSPI STT-MRAM 200 MHz 64 Mbit
SPI-DDR Boot Configuration: Device Powers Up in SPI mode: Switch to DDR mode

Memory Channel 1 Configuration
Protocol 1: none
Memory device on chip select 1: none

Memory IO Configuration
Select DQIN Pin configuration: Integrated DDR Registers in IO Element
Select I/O Interface: Standad I/O Interface

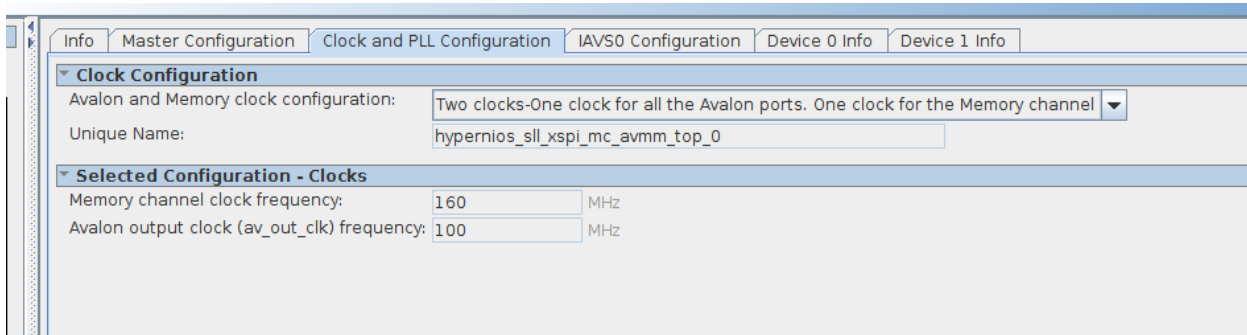
Memory Powerup Configuration
PowerUp Delay Timer: 300 us

Memory RWDS-DQ SKEW Configuration
☒ Enable Default RWDS-DQ SKEW settings
DQ-DQS skew on channel 0 in ps: 533 ps

Avalon Target Interface Configuration
Avalon control port: Enabled
Control Port Interface Type: AVMM Interface
Include PSRAM ECC input signal: Disabled

In the Clock and PLL Configuration Tab

- The Avalon and Memory clock configuration field is set to Two clocks
- The Memory channel clock frequency field will automatically be set to the incoming frequency on the Hyperbus clock
- The Avalon output clock frequency field will automatically be set to the incoming frequency on the Avalon slave clock



The screenshot shows a software configuration window with several tabs: Info, Master Configuration, Clock and PLL Configuration (selected), IAVSO Configuration, Device 0 Info, and Device 1 Info. The 'Clock Configuration' section is expanded, showing 'Avalon and Memory clock configuration' set to 'Two clocks-One clock for all the Avalon ports. One clock for the Memory channel' and a 'Unique Name' of 'hypernios_sll_xspi_mc_avmm_top_0'. Below this, the 'Selected Configuration - Clocks' section shows 'Memory channel clock frequency' at 160 MHz and 'Avalon output clock (av_out_clk) frequency' at 100 MHz.

Clock Configuration	
Avalon and Memory clock configuration:	Two clocks-One clock for all the Avalon ports. One clock for the Memory channel
Unique Name:	hypernios_sll_xspi_mc_avmm_top_0
Selected Configuration - Clocks	
Memory channel clock frequency:	160 MHz
Avalon output clock (av_out_clk) frequency:	100 MHz

In the Ingress Avalon Slave 0 (IAVS0) configuration tab

The IAVS0 port is used to access all memory devices connected to the xSPI Multi-Bus Memory Controller IP. The most common configuration of the IAVS0 port is as follows:

IAVS0: Ingress Avalon port stage

- The **Enable Avalon write** capability field is [checked](#)
- The **Enable Avalon byte-enable** capability field is [checked](#)
- The **Register Avalon Write** data field is left [unchecked](#)

IAVS0: Burst Converter and address decoder stage

- The **max BurstSize** (in Words) field is set to [64 words](#)
- The **BurstonBurstBoundaries** is set to [True](#)

IAVS0: Ingress Avalon return stage

- The **Register Avalon Read data** path field is left [unchecked](#)
- The **Use Avalon Transaction Response** field is left [unchecked](#)

The screenshot displays the 'IAVS0 Configuration' tab within a software interface. The tab is divided into four expandable sections:

- IAVS0: Ingress Avalon port stage**: Contains checkboxes for 'Enable Avalon write capability' (checked), 'Enable Avalon byte-enable capability' (checked), and 'Register Avalon write data path' (unchecked). A text field for 'Access capabilities' is set to 'Read/Write'.
- IAVS0: Ingress Avalon address/data**: Contains fields for 'Address width' (21 bits), 'Address units' (Words), and 'Word width' (32 bits).
- IAVS0: Burst converter and address decoder stage**: Contains dropdown menus for 'maxBurstSize (in words)' (64) and 'burstOnBurstBoundariesOnly' (true).
- IAVS0: Ingress Avalon return stage**: Contains checkboxes for 'Register Avalon read data path' (unchecked) and 'Use Avalon transaction responses' (unchecked).

The GUI interface includes significant inbuilt documentation. Moving your mouse over a configuration field pops up the on-screen help for that field.

In the Device 0 Info tab

The “Device 0 Info” tab provides information about the Memory device connected to chip select device 0.

If in the "Master Configuration" tab, the Memory Channel 0 Configuration/Protocol field is set to None, these fields remain empty.

If in the "Master Configuration" tab, the Memory Channel 0 Configuration/Protocol field is set to any other device , the table will show the parameters, configuration and timing for the selected device memory.

The screenshot shows a configuration window with several tabs: Info, Master Configuration, Clock and PLL Configuration, IAVS0 Configuration, Device 0 Info (selected), and Device 1 Info. The Device 0 Info tab is active, displaying two sections: Device 0 Parameters and Device 0 Timings.

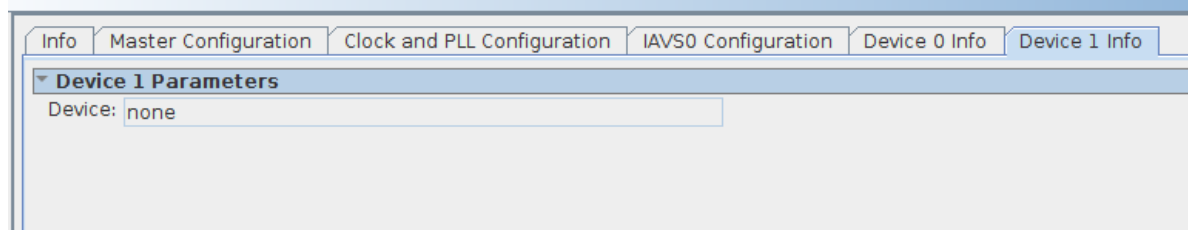
Device 0 Parameters		
Device:	em064lx	
Memory Type:	Xccelaflash	
Device storage capacity:	8	MBytes
Device Speed Grade:	200	MHz
Device DQ data width:	8	
Device Wrap Support:	0	
<input checked="" type="checkbox"/> Non-default configuration settings are programmed after Power-Up		

Device 0 Timings		
Tacc:	16	cycles
Tcshi - Trwr:	15	cycles
Tcss:	2	cycles
Tcsh:	1	cycles

The Device1 Info Tab

The table below will show the parameters, configuration and timing for the memory device.

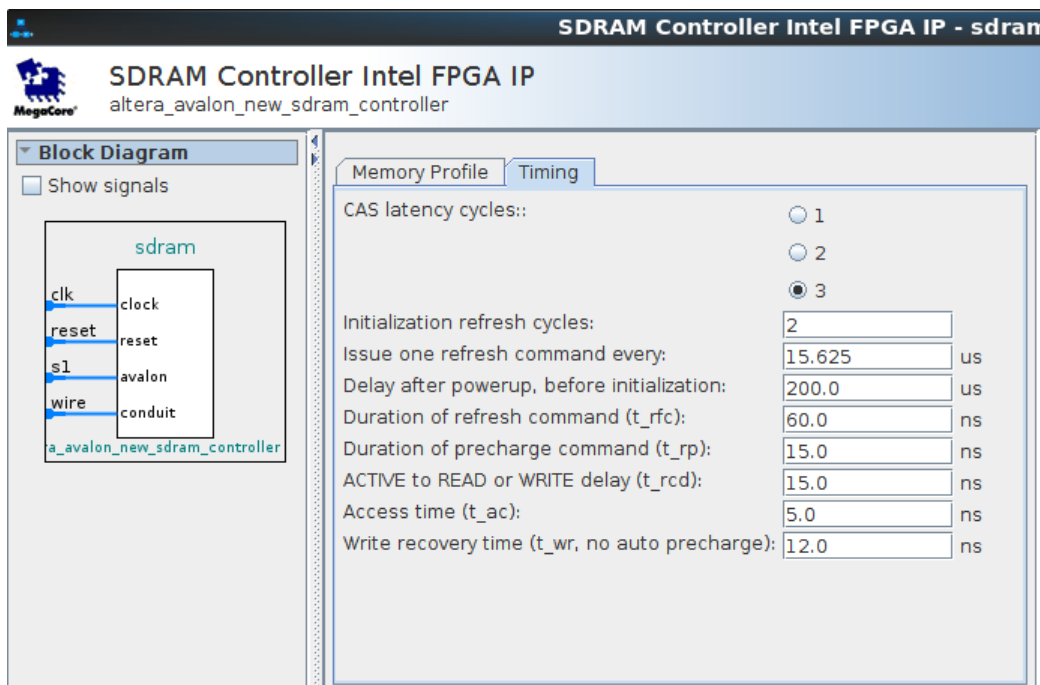
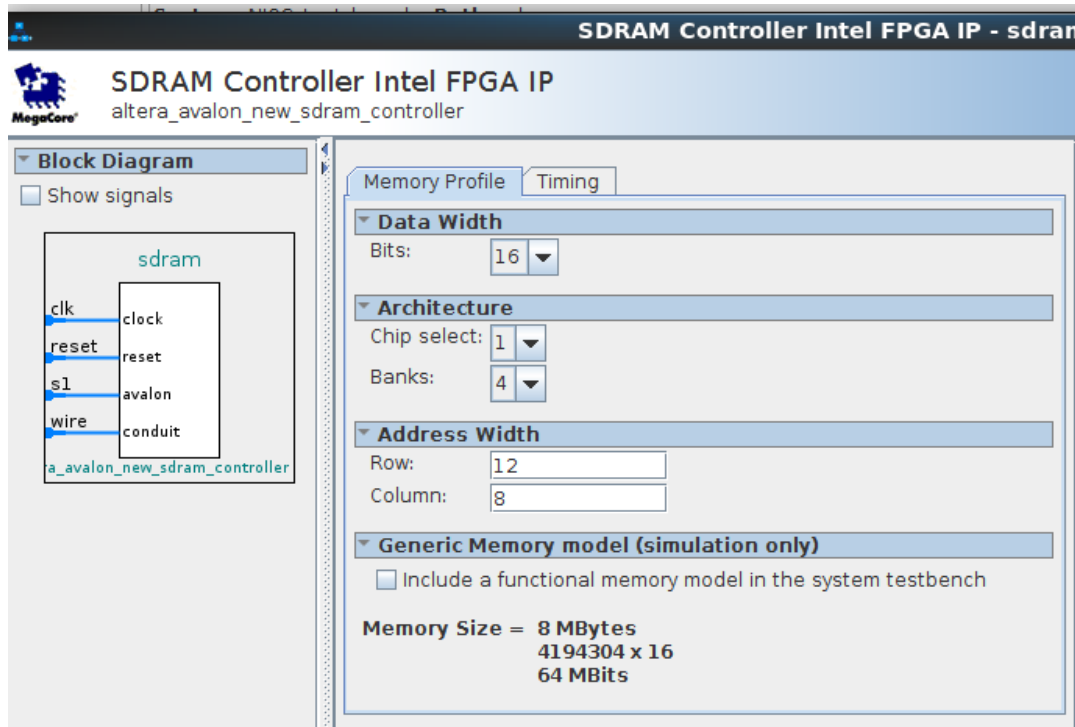
If in the "Master Configuration" tab, the Memory Channel 1 Configuration/Protocol field is set to None, these fields remain empty.



The screenshot shows a software interface with several tabs: "Info", "Master Configuration", "Clock and PLL Configuration", "IAVS0 Configuration", "Device 0 Info", and "Device 1 Info". The "Device 1 Info" tab is selected. Under this tab, there is a section titled "Device 1 Parameters". Below this section, there is a label "Device:" followed by a text input field containing the word "none".

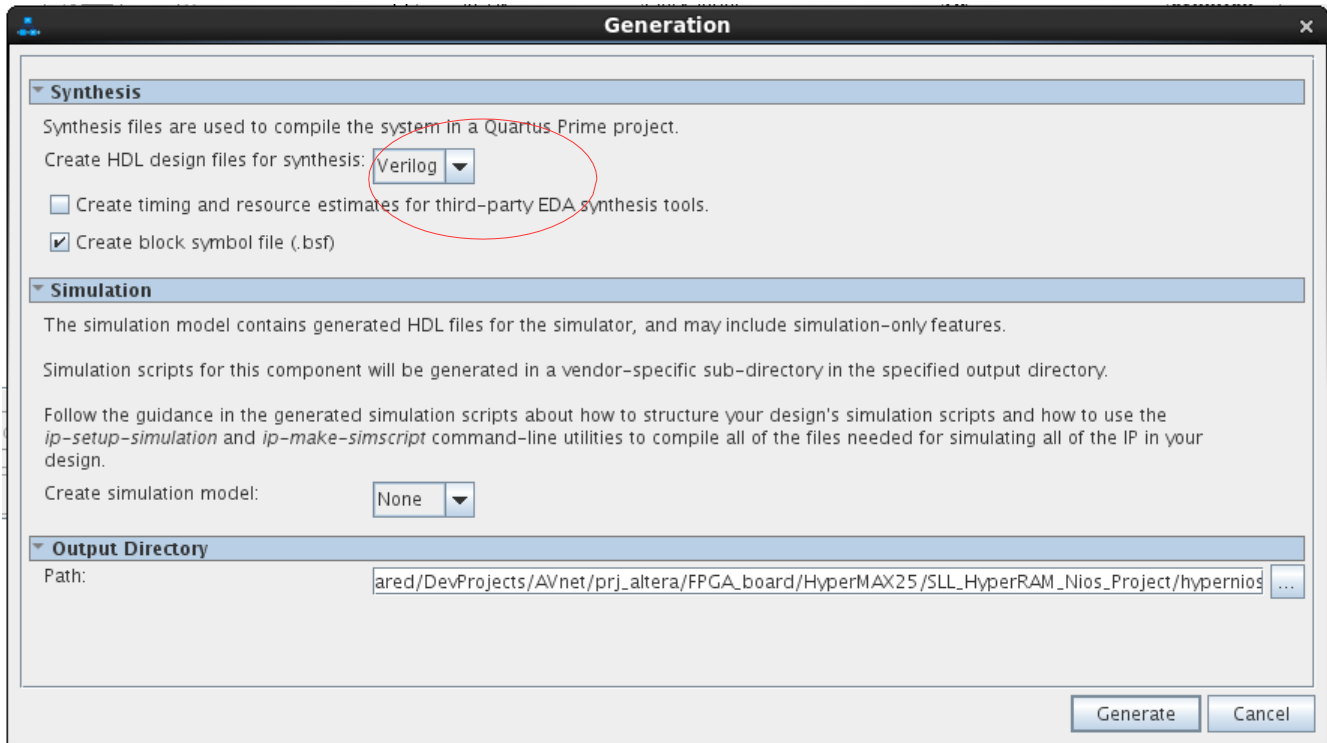
4.5 Configuration of Altera's SDRAM

In this reference project, Altera's SDRAM controller is configured as an 8 Mbyte SDRAM device.



5.0 Generating the Qsys Design

Once the Qsys project has been correctly configured, press the [Generate HDL...] button on the bottom right hand side of the Qsys window.



- In the Synthesis section, set the Create HDL design files for synthesis field to Verilog.
- In the Simulation section, set the Create simulation model field to None.
- Then click on the [Generate] button.
- You may see a Save System window.
- Click the [Close] button to close the save window.
- Generating the .qsys project updates the .SOPC file which will be used by the Nios II Software Build Tools (SBT) environment.
- Click the [Close] button to close the generate window.

You may want to close the Qsys window.

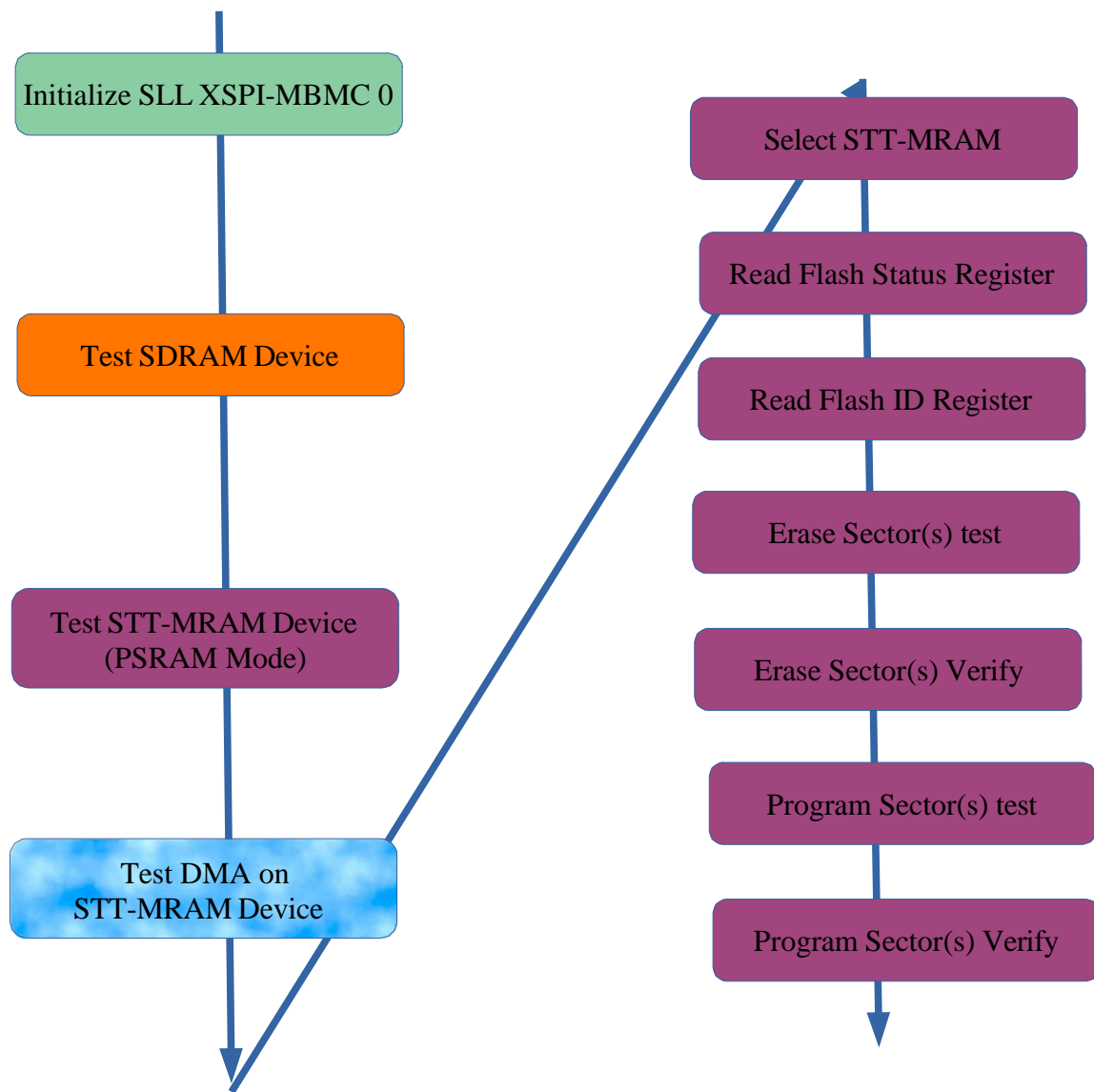
6.0 Demo test Application

The pre-compiled project has the following setup

- NiosII running at 100 Mhz
- SDRAM (8 Mbyte) running at 100 Mhz
- Everspin EM064LX (on J1) memory running at 150 Mhz

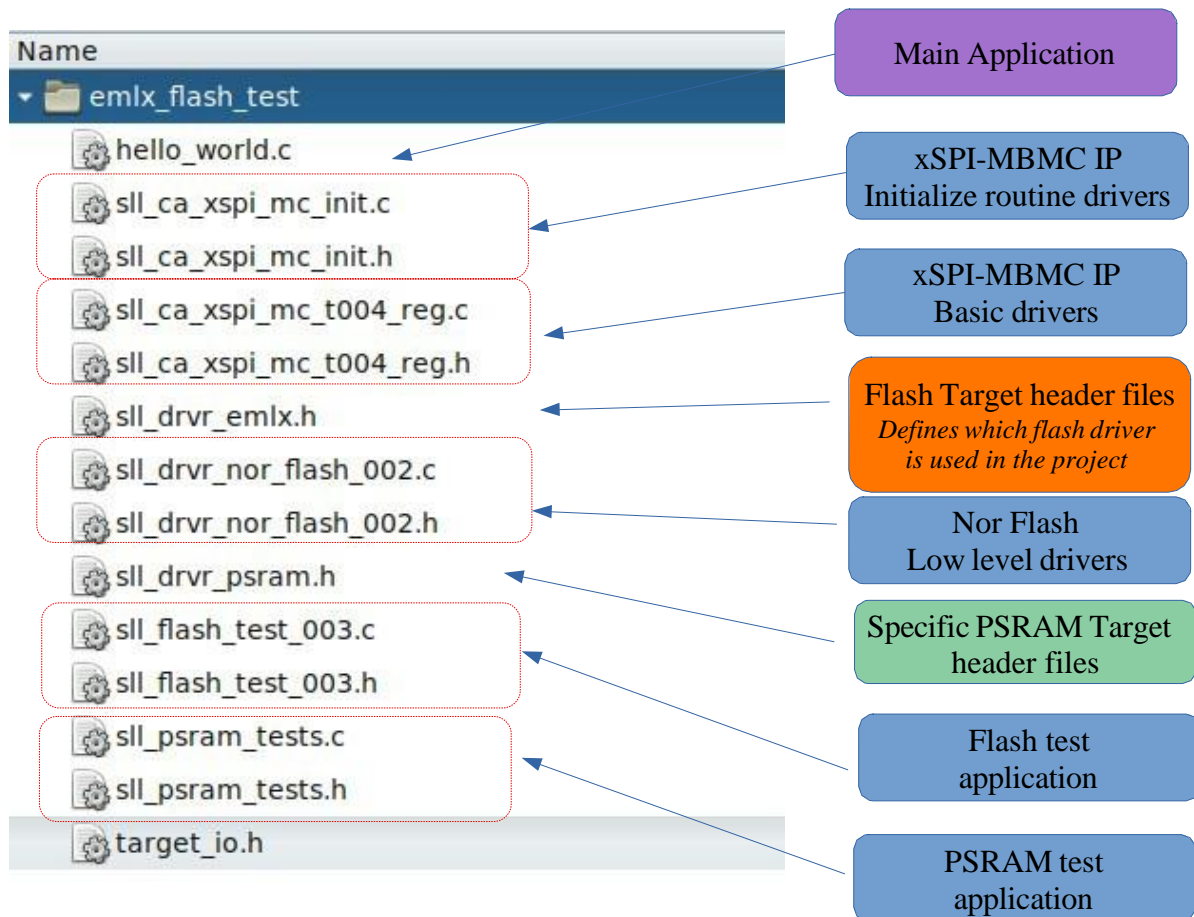
The Flash/SDRAM test application provided with this bundle performs a number of functions to test the SDRAM memory and Flash memory devices.

If errors are encountered, the subsequent tests are bypassed.



6.1 Flash test Application File Structure

Software flash low level drivers and a sample test application is provided with this tutorial. The low level drivers can be found under the source_files directory. To date, low level drivers are provided for Everspin EMLX family, Cypress Semper Flash with octospi (28HS/28FL series), Micron xSPI Flash (MT35XU series) and GigaDevice (GD55LX series). In this demo application, the Flash Drivers for Everspin Flash (EMLX/28FL series) are provided. Kindly contact Synaptic Labs' if you want additional flash drivers or functionality / API for a specific memory device.

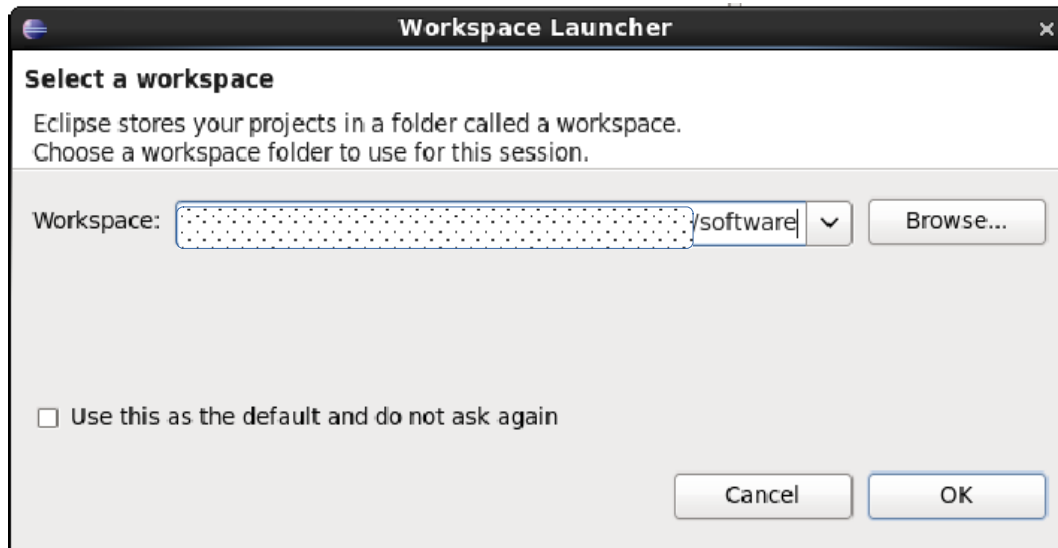


7.0 Preparing the firmware

7.1 Open the NIOS II Software Built Tools for Eclipse

In Quartus Prime, go to the menu bar and select

- Tools → NIOS II Software Built Tools for Eclipse.



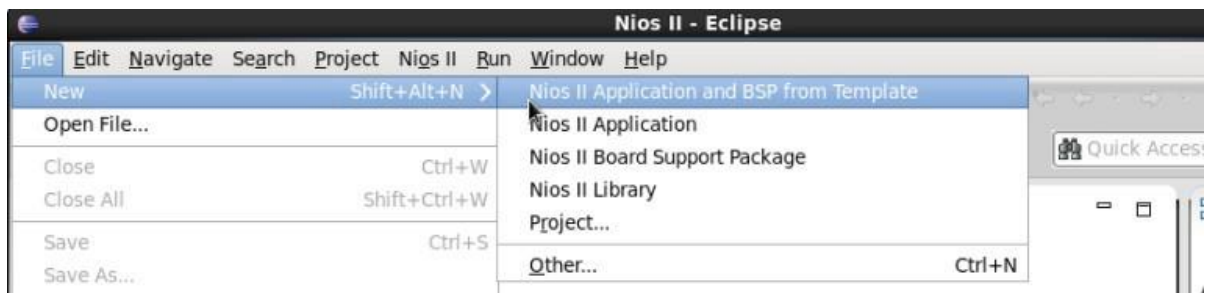
- Click the [Browse...] button. A new file selector window will open. In this tutorial we are going to select the software folder located inside the project folder as the workspace and then click the [OK] button.
- Be sure to leave the [] Use this as the default field unticked.
- Click the [OK] button.

7.2 Create a simple application and BSP

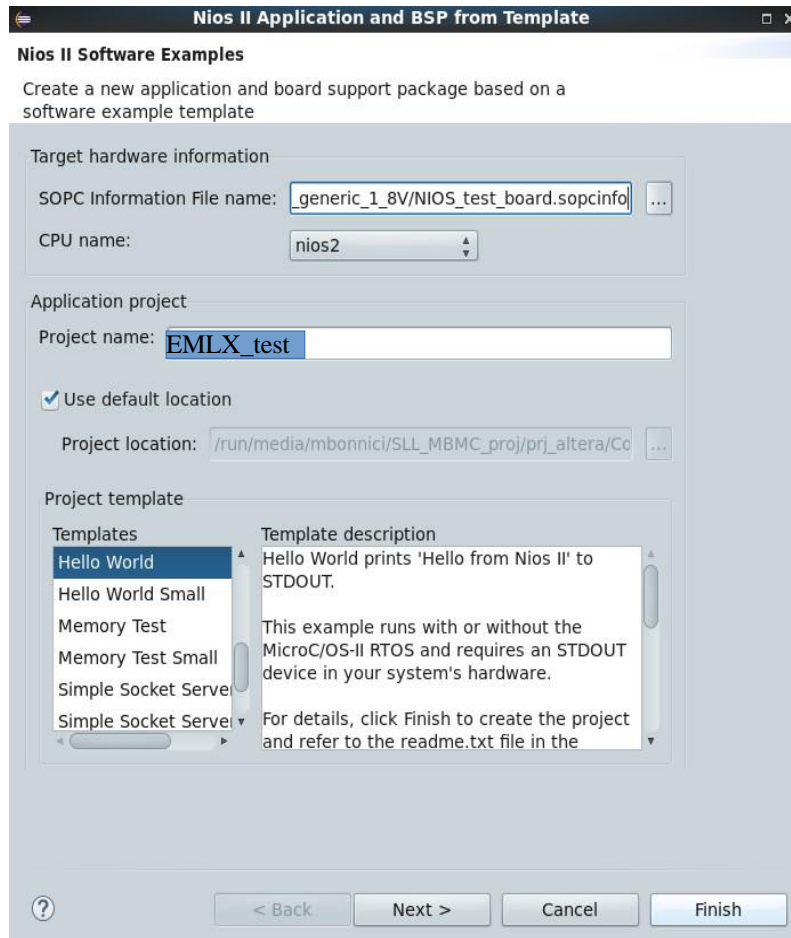
The software folder in the reference project is empty. This is because problems can be experienced when moving the Eclipse Workshop folder between Windows and Linux Systems. We need to create a Nios II application, and Nios II board support package for that Nios II application:

In the Eclipse window, go the menu bar and select:

- File → New → NIOS II Application and BSP from Template



A new window will pop up: (most of the fields below will initially be empty)



- In the Target hardware information, click on the [...] button
- A file browser window will open. Locate and select the **Nios_test_board.sopcinfo** file generated by Qsys and stored in the project directory. Click [Open].
- It may take around 30 seconds for the Eclipse application to parse the .sopcinfo file.
- Select a Project name. In this example, we are using **EMLX_test** as the project name.
- Ensure that: [x] Use default location is ticked.
- We now need to select a template from the Project Template list. In this example, select the Hello World template.
- Press the [Finish] button to complete the current step.

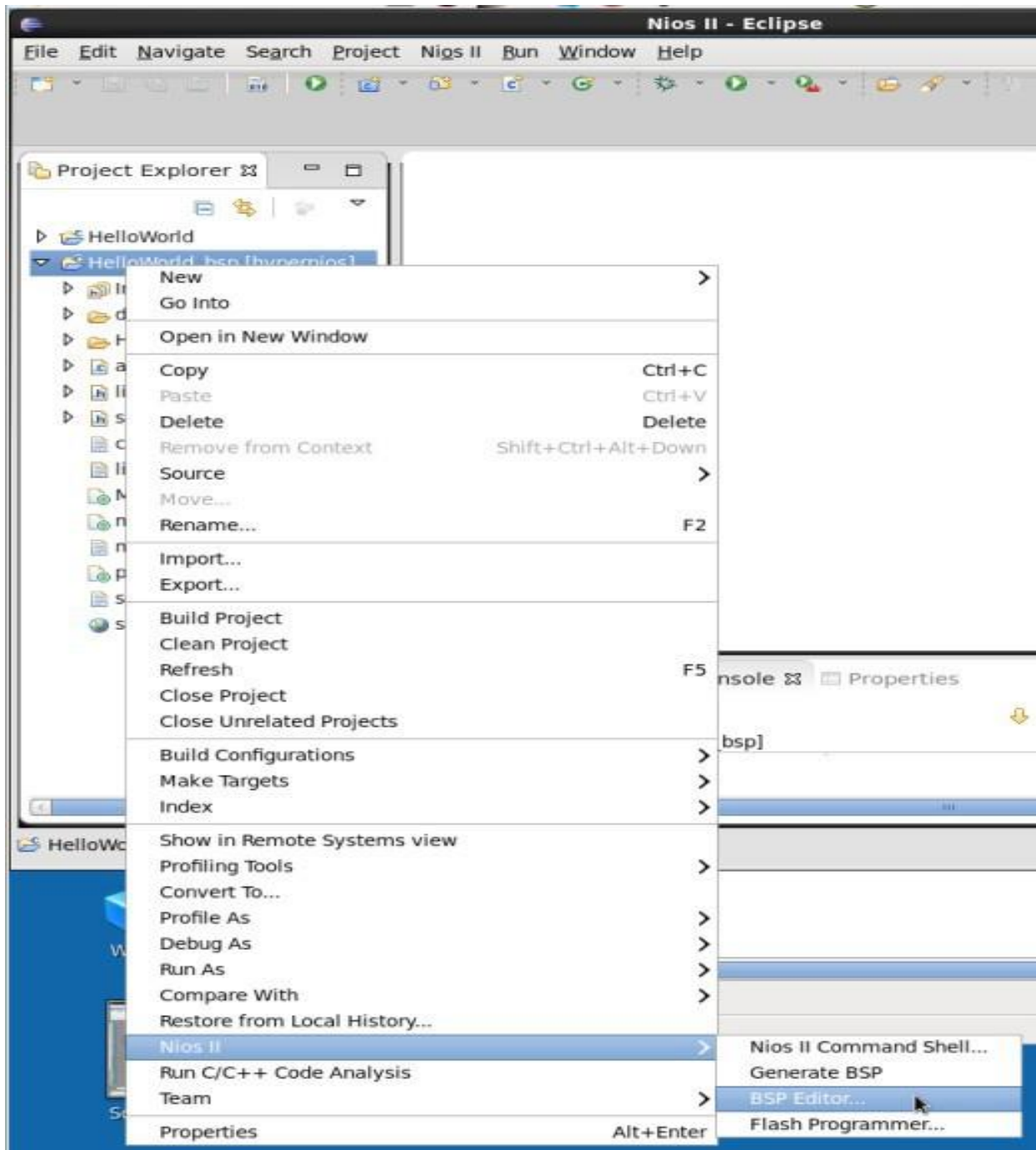
The Nios II SBT will now generate:

- a **EMLX_test** application folder that contains the hello_world.c file. We will replace that hello_world.c file with a custom program that tests the Flash device later in this tutorial.
- a **EMLX_test_bsp** folder that contains the Nios II Board Support Package (BSP) hardware abstraction layer (HAL).

7.3 Configure the Board Support Package (BSP)

The Nios II BSP must be configured before we can compile the source code.

- In the Project Explorer tab, right click on:
EMLX_test_bsp → Nios II -> BSP Editor...



In the Main Tab of the BSP editor, in the panel on the left hand side, select:

- **Settings** → **Common**
- Set the **sys_clk_timer** field to `timer_0`. This is used to generate a recurring system clock interrupt for the hardware abstraction layer.
- Set the **timestamp_timer** field to `interval_timer`. This field is used to enable the hardware abstraction layer to perform fine precision timing.

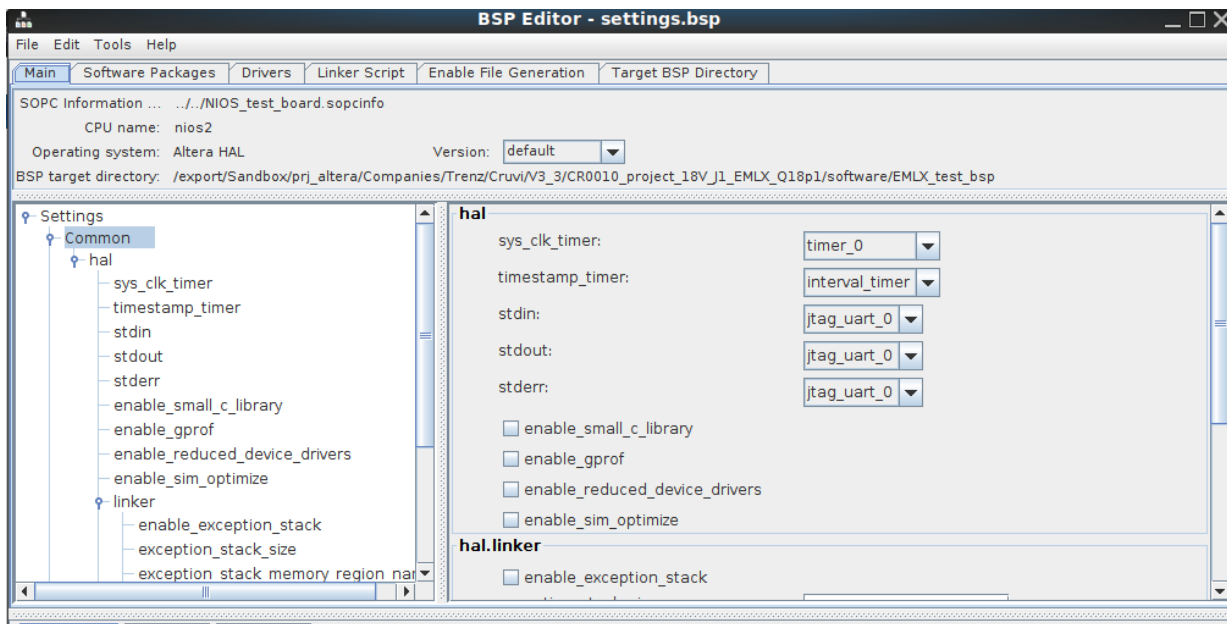
The Newlib ANSI C standard library can be configured as small or normal

Generally, when mapping code and data to on-chip memory:

Tick the **[x] Enable small C library** field to reduce the size of the executable code generated by the hardware abstraction layer (HAL). Ticking this option also reduces the functionality and performance of the HAL. Please note that the inbuilt `memset()` and `memcpy()` routines will be very slow.

Generally, when mapping code and data to HyperRAM and/or SDRAM:

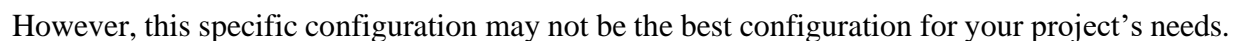
Untick the **[] Enable small C library** field to increase the functionality and performance of the executable code generated by the hardware abstraction layer (HAL). The inbuilt `memset()` and `memcpy()` routines will achieve good performance. However, the executable code will be considerably larger.



- Settings → Advanced → hal.linker

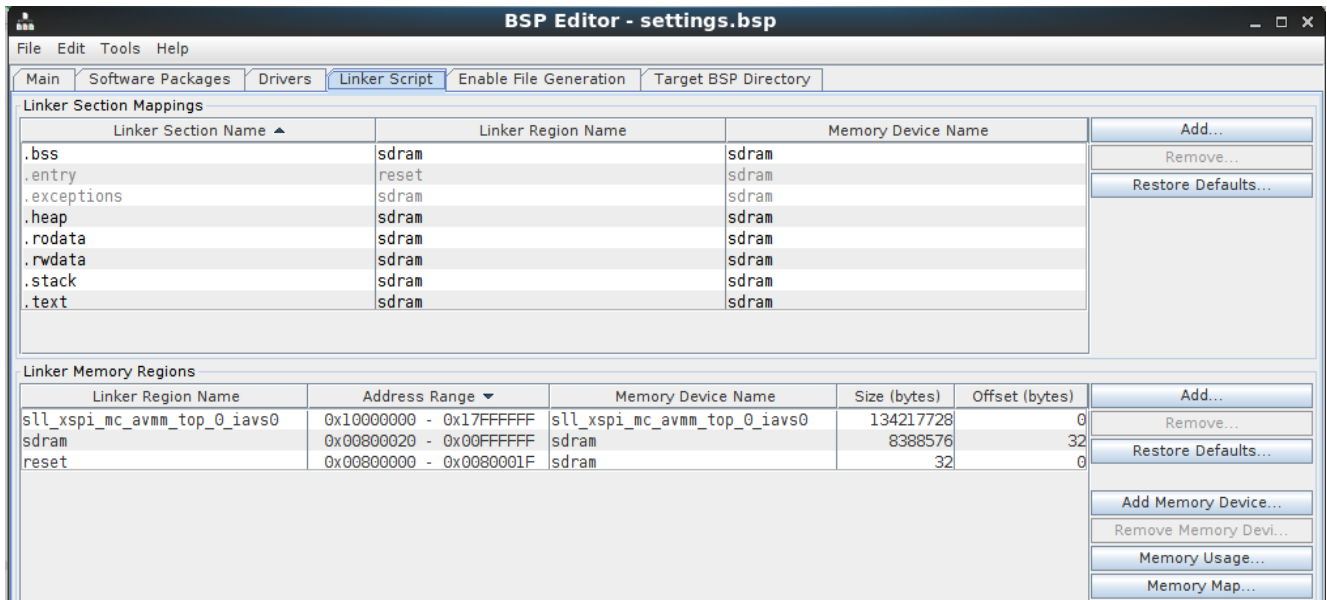
For the purpose of this tutorial, the following configuration will generally work:

- Tick [x] allow_code_at_reset
- Tick [x] enable_alt_load
- Tick [x] enable_alt_load_copy_rodata
- Tick [x] enable_alt_load_copy_rwdata
- Tick [x] enable_alt_load_copy_exception



Generic Nios II Booting Methods User Guide, UG-20001, 2016.05.24
https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/niosii_generic_booting_methods.pdf

Select the Linker Script Tab of the BSP editor.



For this tutorial example, we are going to:

- Map the reset vector (.reset) to the sdram memory (sdram) .
This is generated by Qsys and depends on the location of the Nios II reset vector.
- Map the exception vector (.exceptions) to the sdram memory (sdram) .
This is generated by Qsys and depends on the location of the Nios II exception vector.
- Map the instruction code (.text) in the sdram memory (sdram) .
- Map all other data regions (.bss, .heap, .rodata, .rwdata, .stack) to the sdram memory (sdram) .

This will map all memory regions generated by the GCC tools to the SDRAM memory.

Now:

- Click on the [Exit] button on the bottom right hand corner of the BSP Editor window.
- Then click on the [Yes, Save] button on the Save Changes window to save the BSP settings.

7.4 Generate the BSP and clean the project

The software developer must re-generate the BSP every time the Qsys project is regenerated. This ensures that the device drivers and addresses of peripherals are reflected correctly in the hardware abstract library. To (re)generate the BSP:

- Go to the Nios II eclipse window.
- Right click on **EMLX_test_bsp** project then select Nios II then select Generate BSP.
- Right click on the **EMLX_test_bsp** project then select Clean Project to delete any intermediate files generated by the gcc compiler for this application library.
- Right click on the **EMLX_test** project then select Clean Project to delete any intermediate files generated by the gcc compiler for this application folder.

7.5 Copy the memory testing source code

- We now want to replace the original HelloWorld.c source code with software that checks the Flash Memory. Copy and replace all files located in:
 - CR00100-01_project_18V_J1_xxx/source_files/EMLX_testto:
 - CR00100-01_project_18V_J1_xxx/software/EMLX_test

In the project explorer window of Eclipse, right click on the EMLX_test folder.

Then select Refresh. The new source code files should now be visible within Eclipse.

7.6 Build the Nios II Application

We now want to run the compiler and linker:

- Go to the Nios II eclipse window.
- Go to the menu bar and select:
Project ->Build All

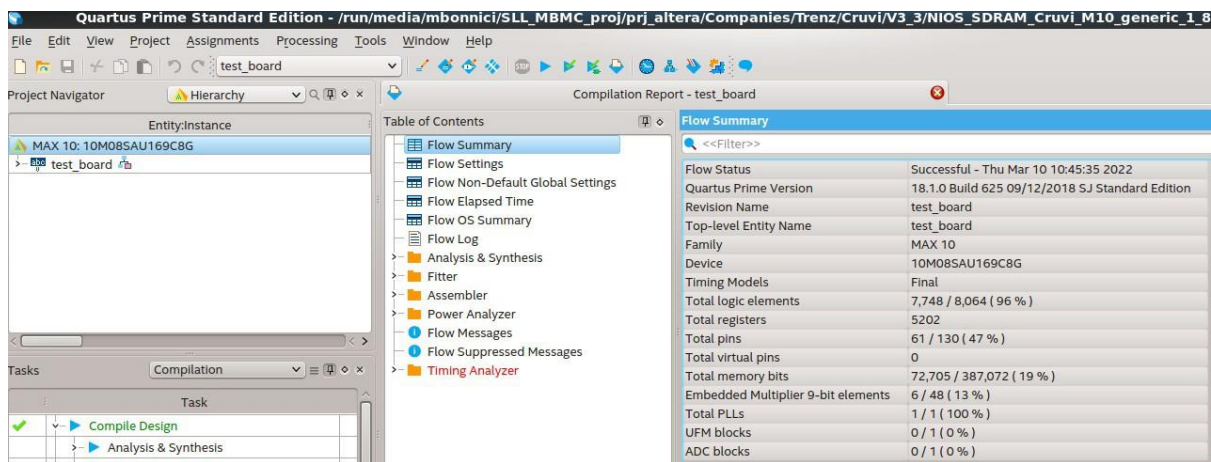
If the project produces warning / error messages, you may need to build the project twice.

The **EMLX_test** executable firmware (.ELF) is now generated. The .ELF can be downloaded directly into the off-chip SDRAM using the Nios II Debugger.

8.0 Synthesize and assemble the Design

- Go to the Quartus Prime window.
- In the menu bar, select: Processing → Start Compilation

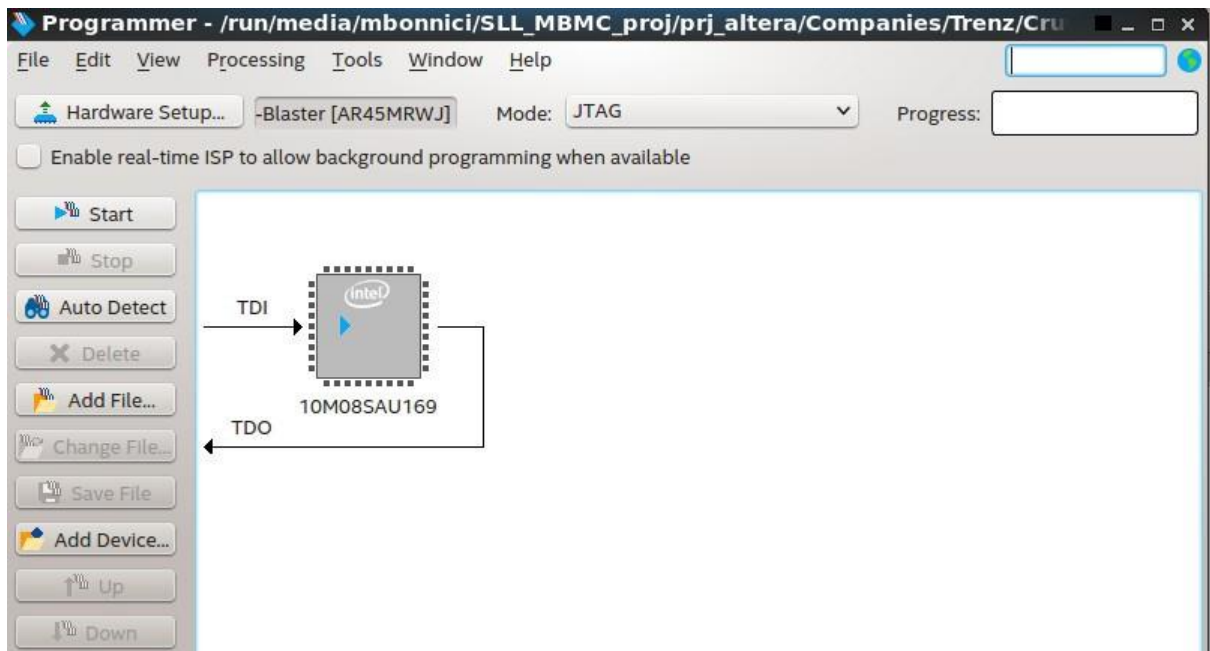
Windows users, please note: If the compilation fails to start you may need to reduce the path length of your project folder. This is because some versions of Windows have a maximum path length of 260 characters which can be exceeded when compiling projects in Quartus Prime.



The assembler step will create the SRAM FPGA Bitstream file (.SOF) with the memory initialisation file (.hex) for the SRAM embedded in that FPGA Bitstream file.

9.0 Program the FPGA Bitstream into the FPGA device

- Connect the Trenz CR00100-01 Evaluation kit to the USB port of your computer
- Open the Quartus Prime window
- In the menubar, click on Tools then Programmer to start the Altera Programmer

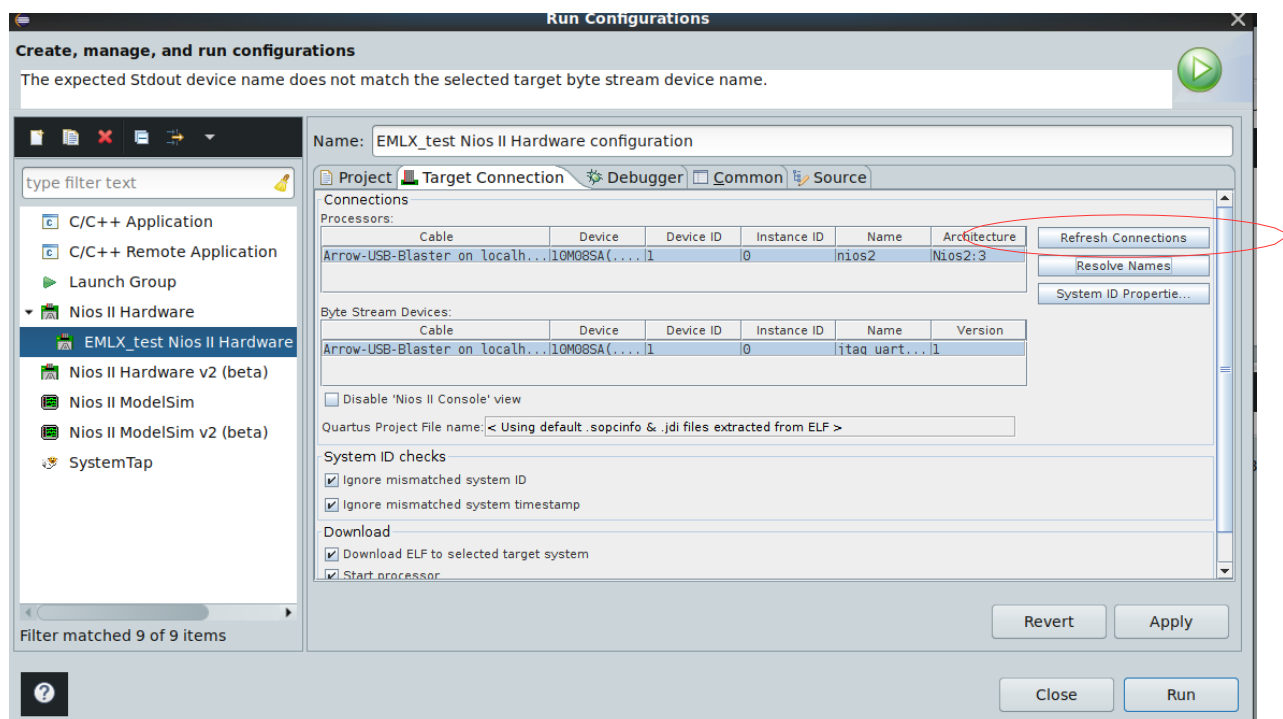


- Click on “Hardware Setup...”. A new window will open.
- Double Click on the (Arrow-USB-Blaster [AR45MRWJ]) device, then click the [Close] button.
- If the test_board.sof (or testboard_time_limited.sof) is not already selected:
- Click “Add File...” in the programmer window.
- Go to the output_files folder
- Double click on test_board.sof (or testboard_time_limited.sof)
- Click the [Start] button and the FPGA bitstream will be programmed into the SRAM configuration memory of the FPGA device.

For the trial version a window “OpenCore Plus Status” will opens Do not close this window.

10.0 Run the application from within Nios II SBT

- Select the Nios II Software Built Tools for Eclipse window.
- Right click on EMLX_test → Run As → Run Configurations...
- A new window will open
- Make sure the Project name: field says EMLX_test.
- Select the Target Configuration tab.
- Press the [Refresh Connection] button to detect the Nios II processor.



- Tick the [Ignore mismatched System ID] field.
- Tick the [Ignore mismatched System timestamp] field.
- Press the [Run] button to download the firmware from the desktop and copy it directly into SDRAM and then run the firmware from SDRAM.

If the download does not work, please check that the static timing constraints of your project are passing.

Messages similar to the one below are displayed in the Nios II Console Window.

```
EMLX_test Nios II Hardware configuration - cable: Arrow-USB-Blaster on localhost [AR45MRWJ] device ID: 1 instance ID: 0 name: jtag_uart_0.jtag
SDRAM memory buffer allocated at address 0x900000
Testing SDRAM

.... Nios II Read/write tests
PSRAM Test : Running pass 1 of 1 passes
Test: Sliding single data bit test within 32-bit word at Address: 0x00900000
- PASS

Test: PSRAM 8/16 bit Byte Enable Test at address: 0x00900000
= 4x08-bit Write:
- 32-bit Read [0]: PASS
- 16-bit Read [0]: PASS
- 16-bit Read [1]: PASS
- 08-bit Read [0]: PASS
- 08-bit Read [1]: PASS
- 08-bit Read [2]: PASS
- 08-bit Read [3]: PASS
= 2x16-bit Write:
- 32-bit Read [0]: PASS
- 16-bit Read [0]: PASS
- 16-bit Read [1]: PASS
```