

# Synaptic Labs'

## Intel xSPI Memory Controller (xSPI-MC – Version 3.3x)

### Design Guidelines (for Intel FPGA ) **draft**

#### Table of Contents

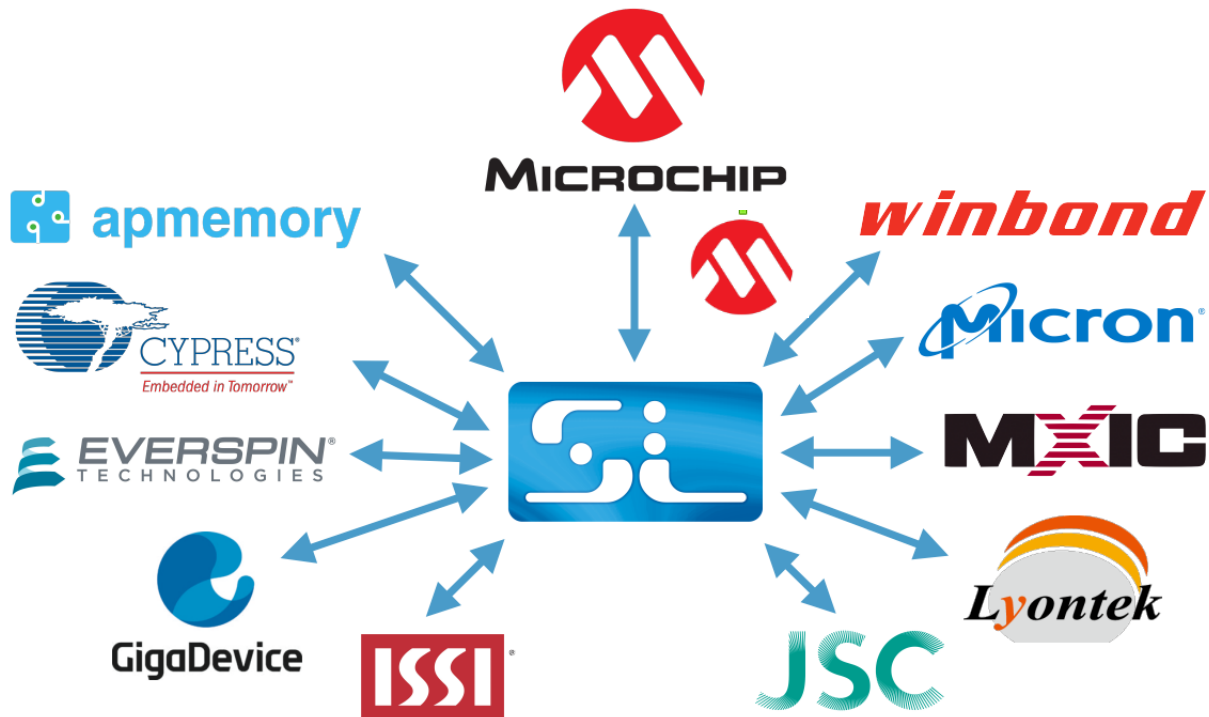
Introduction.....	3
1.0 Set-Up Requirements.....	8
2.1 Instantiating SLL xSPI-Multiple Bus Memory Controller IP.....	10
2.2 Configuring SLL xSPI-MBMC IP in the Master User Interface Tab.....	11
2.3 Configuring SLL xSPI-MBMC IP in the Clock and PLL Configuration Tab.....	17
2.4 Configuring SLL MBMC IP in the IAVS0 Configuration Tab.....	19
2.4.1 Configuring Ingress Avalon port slave parameters.....	19
2.4.2 Configuring the burst count on the Ingress Avalon port slave.....	20
2.4.3 Configuring the Ingress Avalon port slave return path.....	21
2.5 Configuring SLL xSPI-MBMC IP in the Device 0/1 Tab.....	22
3.0 Connecting SLL xSPI-Multi Bus Memory Controller to the NIOS II/f embedded processor.....	23
3.1 SLL xSPI-MBMC with external PLL Configuration.....	25
3.1.1 Option A – Same Clock for the PSRAM/Flash memory channel an Avalon-MM channel.....	26
3.1.2 Clocking (PLL) Wizard Configuration.....	27
3.1.3 SLL xSPI-MBMC Configuration.....	27
3.1.4 SLL xSPI-MBMC wiring.....	28
3.2 Option B – Different Clocks for the PSRAM/Flash memory channel and Avalon-MM channel .....	29
3.2.1 Clocking Wizard Configuration.....	29
3.2.2 SLL xSPI-MBMC Configuration.....	29
3.2.3 SLL xSPI-MBMC wiring.....	30
4.0 Signal Description.....	32
4.1 Memory Bus Signal Interface.....	32
4.2 Memory Clock Signal Interface.....	33
4.3 Avalon Clock/reset interface.....	33
4.4 Avalon Data Slave Interface.....	33
4.5 Avalon Control Slave Interface (optional).....	34
5.0 Avalon-MM Register Control Port.....	35
6.0 Avalon-MM Register Control Port.....	36
6.1 Register MAP.....	37
6.2 Register Definitions.....	40
6.3 Accessing PsuedoRAM Registers.....	55
6.4 Accessing HyperFlash Memory using the Control Port.....	56
6.5 Accessing xSPI Flash Memory using the Control Port.....	57
6.6 xSPI Profile 1.0 Transaction : Format 1.A.....	58
6.8 xSPI Profile 1.0 Transaction : Format 1.B.....	59
6.9 xSPI Profile 1.0 Transaction : Format 1.C.....	60

6.10 xSPI Profile 1.0 Transaction : Format 1.D.....	61
7.0 : AXI4 Memory Address Mapping *** .....	62
8.0 xSPI Controller Core Operation.....	64
9.0 Core Reset.....	65
9.1 Global Asynchronous Reset.....	65
9.2 Software Core Reset ( <i>to be defined</i> ).....	65
9.3 Software Device Reset ( <i>to be defined</i> ).....	65
9.3 Device Sleep/Power Down ( <i>to be defined</i> ).....	65
10.0 Powering Up ( <i>generic mode only</i> ).....	66
11.0 CycloneV Support.....	74
11.1 SLL xSPI-MBMC Configuration for Integrated DDIO Support.....	75
12.0 Static Timing Consideration.....	76
12.1 Data Input Timing Constraint.....	76
12.2 Timing Constraints.....	76
12.3 Pin Clustering.....	77
12.4 Using 3V PSRAM and Flash devices.....	78
12.4.1 Selecting the correct operating frequency in Qsys.....	78
12.4.2 Selecting the correct voltage in Quartus.....	78
12.4.3 Connecting the PSRAM signals to the FPGA I/O.....	79

## Introduction

SLL xSPI Memory Controller (xSPI-MC) supports the following Octal DDR PseudoRAMs and Flash from the following memory vendors :

- Cypress/ISSI (HyperRAM , HyperFlash, SemperFlash, xSPI Flash, xSPI PRAM)
- Winbond (HyperRAM)
- AP Memory (OctaRAM and Xccela RAM)
- JSC (OctaRAM)
- LyonTek (Xccela RAM)
- Micron (xSPI Flash)
- GigaDevice (xSPI Flash)
- Macronix (MX25 Flash series)



The pseudoRAM / Flash memories operate in a Double Data Rate (DDR) mode. They have an 8-bit Serial Interface and are used in all applications such as automotive semiconductor market, Industrial Ethernet, AI, Wearable device market and IoT market. They support a maximum operating frequency of 200 MHz Speed and a maximum bandwidth of 400 Mbits/s.

Furthermore, when compared to traditional DDR memories, the PseudoRAM memories are considered as low power alternatives.

Synaptic Labs' xSPI Memory Controller IP has an Avalon-MM slave interface. It supports burst access (up to 128 words). It can be configured in single access or burst mode access. Furthermore, burst mode access can be configured with either Avalon burst wrap support or Avalon burst-on-burst-boundaries support.

All external I/O pads to the memory signals are generated from within SLL xSPI-MC IP. The user does not need to manually instantiate the I/O pads in the design.

An external PLL is used to generate all the necessary clocks.

Well documented example design projects and tutorials are provided upon request.

Note: Synaptic Labs' xSPI Memory Controller IP does NOT support DCARS functionality (Hyper-RAM PSC mode).

- DCARS is a very specific capability requested by a very specific customer / chipset partner and is only supported by Cypress devices.
- ISSI devices do not have DCARS support.
- Cypress do not recommend the implementation of DCARS functionality.
- DCARS has a maximum frequency of 133 MHz.

SLL is targeting the following 23 memory device families in xSPI-MC IP :

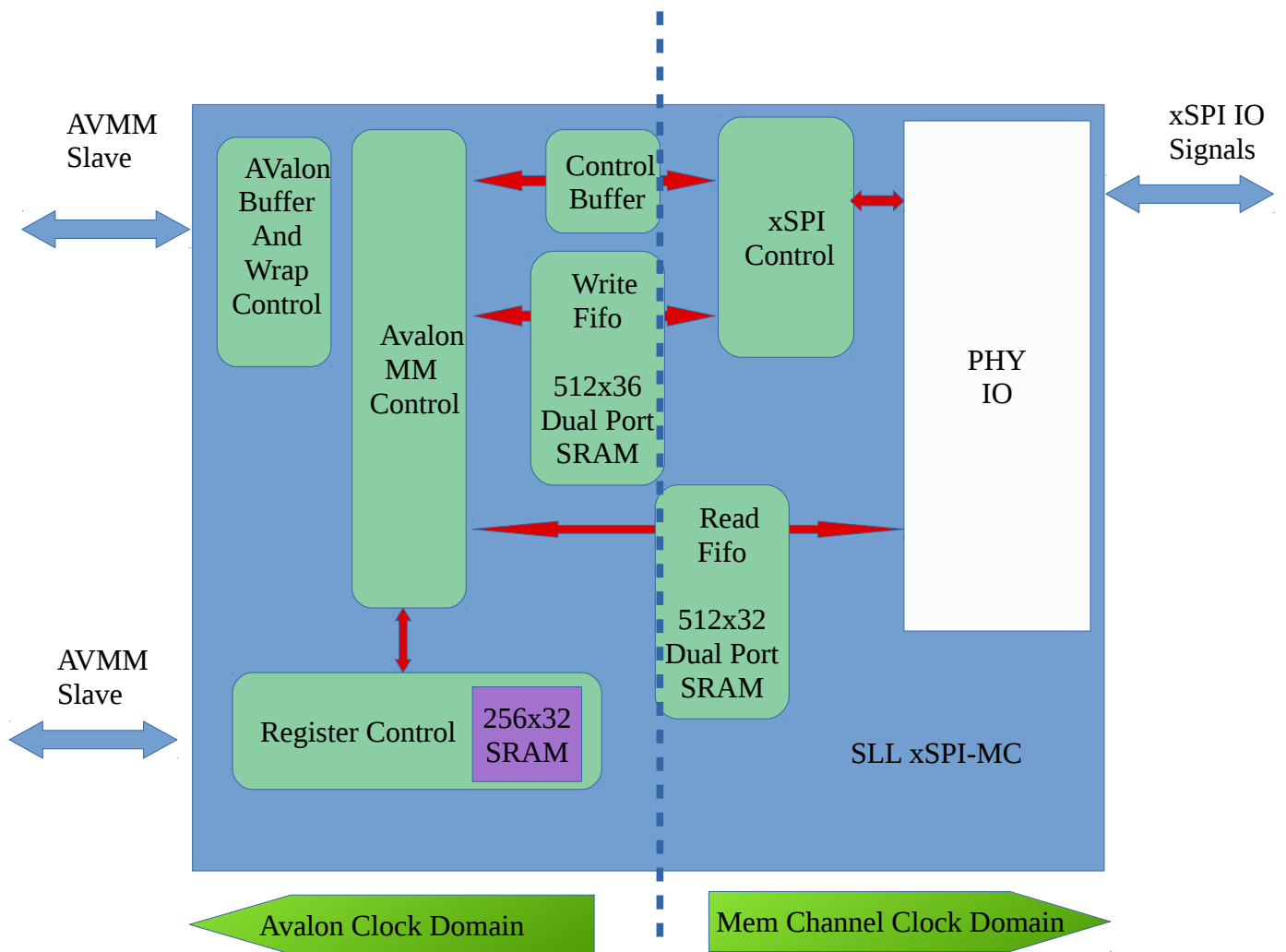
	Memory Vendor	1.8V Device Family	Device Type(s)	DQ (bits)	Device Speed (MHz)	Data Rate
1	AP Memory	Xccela PSRAM	PSRAM	8	200	DDR
2	AP Memory	OctaRAM	PSRAM	8	200	DDR
3	Cypress	HyperRAM 1.0	PSRAM	8	166	DDR
4	Cypress	HyperFlash 1.0	NOR Flash	8	166	DDR
5	Cypress	HyperMCP 1.0	PSRAM + NOR Flash	8	166	DDR
6	Cypress	HyperRAM 2.0	PSRAM	8	200	DDR
7	Cypress	Semper Flash with HyperBus Interface	NOR Flash	8	200	DDR
8	Cypress	HyperMCP 2.0	PSRAM + NOR Flash	8	200	DDR
9	Cypress	Semper Flash with Octal Interface (xSPI Profile 1.0)	NOR Flash	8	200	DDR
10	Everspin	STT-MRAM (ET-35x)	MRAM	8	200	DDR
11	ISSI	HyperFlash 1.0	NOR Flash	8	166	DDR
12	ISSI	Octal Flash (xSPI Profile 1.0)	NOR Flash	8	200	DDR
13	ISSI	HyperRAM 1.0	PSRAM	8	166	DDR
14	ISSI	HyperRAM 1.0 with ECC	PSRAM	8	166	DDR
15	ISSI	OctaRAM	PSRAM	8	200	DDR
16	ISSI	OctaRAM with ECC	PSRAM	8	200	DDR
17	GigaDevice	GD25LX256E	NOR Flash	8	200	DDR
18	JSC	OctaRAM	PSRAM	8	200	DDR
19	Lyontek	Xccela PSRAM	PSRAM	8	200	DDR
20	Macronix	OctaFlash	NOR Flash	8	200	DDR
21	Micron	Xccela Flash	NOR Flash	8	200	DDR
22	Winbond	HyperRAM 1.0	PSRAM	8	166	DDR
23	Winbond	HyperRAM 2.0	PSRAM	8	200	DDR

PSRAM memory support is implemented predominantly in hardware. Certain PSRAM operations, such as entering and exiting low power mode, can be controlled using software via the control port of the xSPI-MC IP.

NOR Flash memory support is implemented as a combination of hardware and software device drivers:

- The data port of the xSPI-MC IP core in hardware is designed to permit:
- Fast read/write operations to PSRAM
- Fast read operations to NOR Flash
- The control port of the xSPI-MC IP core in hardware is designed to permit:
- All xSPI Profile 1.0 command types (1.A, 1.B, 1.C, 1.D), and
- The vast majority of the xSPI Profile 1.0 command type variations
- The control port of the xSPI-MC IP core in hardware will support:
  - All the command type variations employed in the above 23 memory device families.
  - Software drivers for NOR Flash memory devices can therefore be expanded over time (by SLL, Microchip, or the customer) to access all memory device specific operations as required.

## SLL xSPI-MC Block Diagram



## Memory Bandwidth

The memory bandwidth of SLL xSPI-MC controller is close to the wire speed bandwidth of the memory device. The bandwidth depends on the memory channel frequency and the burst length of the memory transaction.

For a x8 memory channel running at 200 MHz DDR, and considering an initial access latency of 8, the effective memory bandwidth increase as the burst length increases:

Burst Length in Bytes	MHz	Initial Access Latency	Effective memory bandwidth in Megabytes/s	Wirespeed bandwidth in Megabytes/s	Effective memory bandwidth efficiency %
..	...	...	...	...	...
32	200	8	173.0	400.0	43.2
64	200	8	241.5	400.0	60.4
128	200	8	301.2	400.0	75.3
256	200	8	343.6	400.0	85.9
512	200	8	369.7	400.0	92.4
1024	200	8	384.2	400.0	96.1

For large burst lengths, the effective memory bandwidth is around 95% of the wire speed bandwidth and can be calculated as follows:

- Memory bandwidth in Mbytes/s =  $0.95 * 2 * \text{memory channel frequency}$

# 1.0 Set-Up Requirements

## Step 1: Obtain core materials

1. Download and install Intel's Quartus (2018.01 or later) on your PC, please ensure that your PC meets the required minimum specification.

## Step 2: License Setup

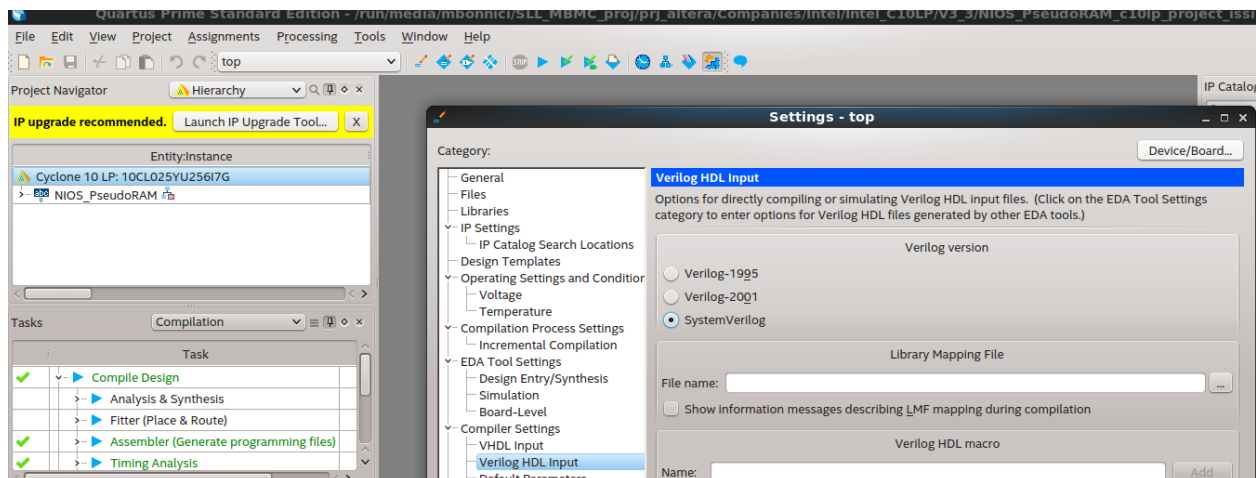
1. Next you need to register for Synaptic Labs' xSPI-MC Memory Controller IP ([info@synaptic-labs.com](mailto:info@synaptic-labs.com)).

## Step 3: Install xSPI-MC Intel Component into the project IP Folder

1. In this tutorial we assume that SLL xSPI Memory Controller(xSPI-MC) will be located in the Project directory.

## Step 4: Enable System Verilog in Quartus

1. You will need to enable System Verilog in the Compiler Settings. This can be done by
  - Right Click on the FPGA device and select Settings.
  - Locate Compiler Settings / Verilog HDL Input
  - Select System Verilog





## 2.0 Finding SLL Multiple Bus Memory Controller IP in the Qsys IP Catalog

Within Qsys, SLL xSPI-Multi-Bus Memory Controller IP can be found in the Qsys IP Catalog panel under:

- + Synaptic Labs
- + Memory
  - + xSPI
    - + SLL Avalon xSPI Memory Controller

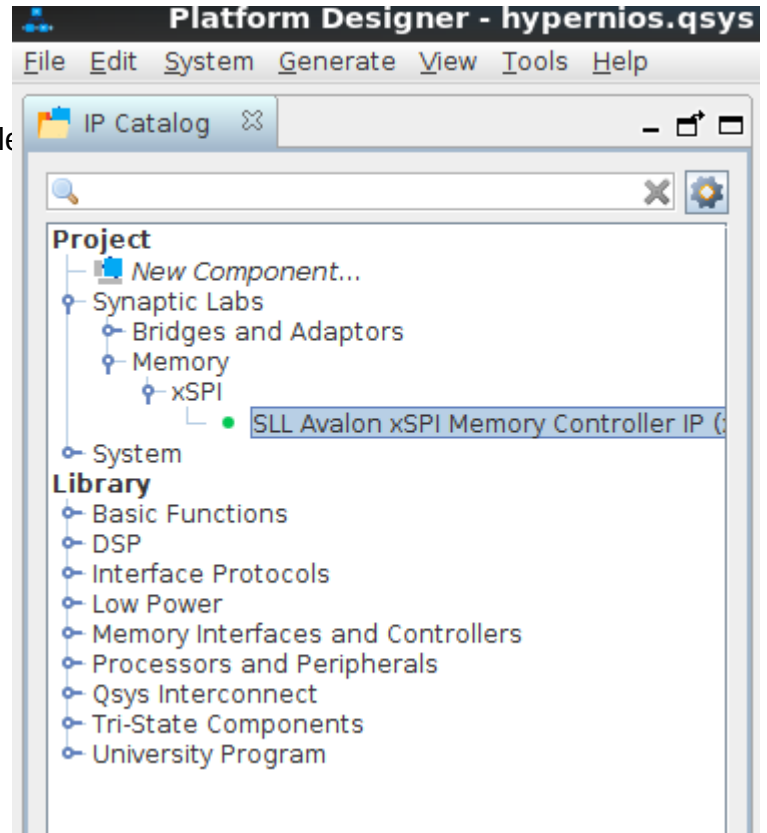
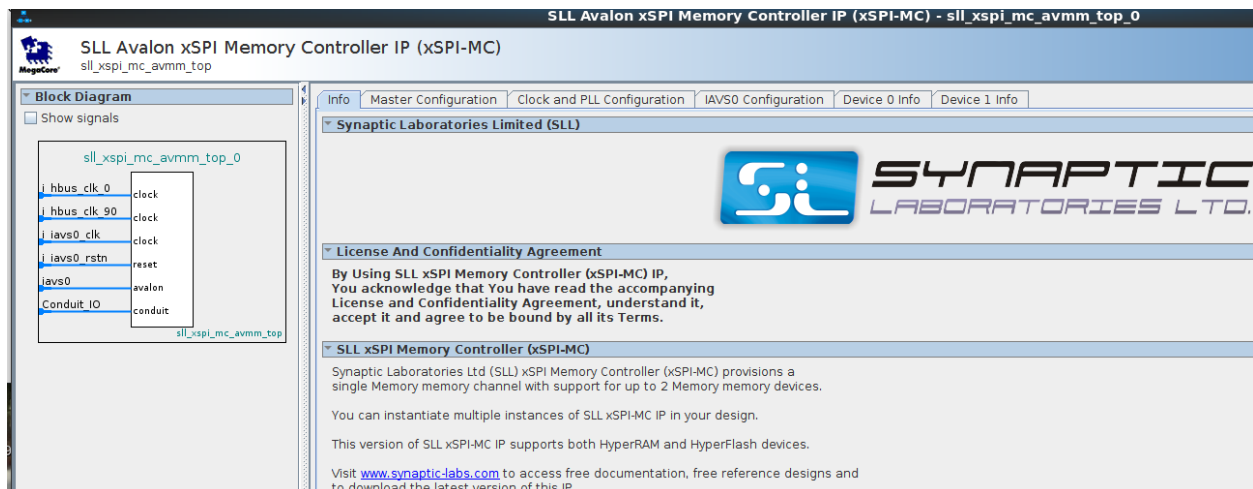


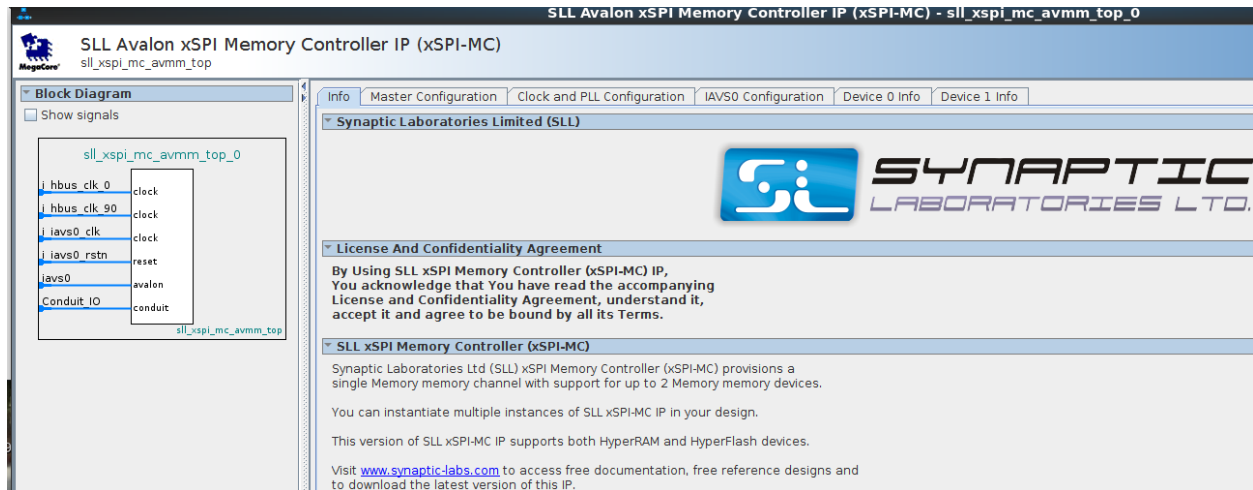
Figure 1: Finding SLL xSPI-Multi Bus Memory Controller IP in the Qsys IP Catalog



## 2.1 Instantiating SLL xSPI-Multiple Bus Memory Controller IP

Double click on Synaptic-Labs xSPI-Multi Bus Memory Controller IP to instantiate an instance of this IP in your Qsys project.

The following IP configuration options will appear on your screen.



SLL xSPI-MC component has to be configured according to the memory device(s) present on the FPGA board

## 2.2 Configuring SLL xSPI-MBMC IP in the Master User Interface Tab

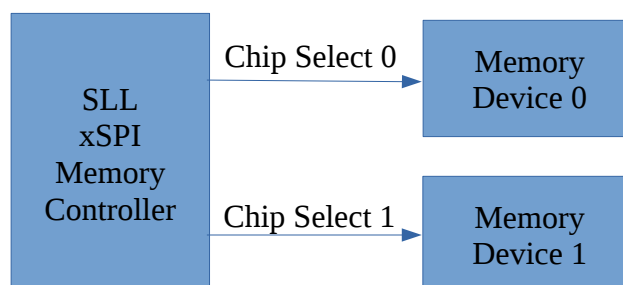
The screenshot shows the 'Master Configuration' tab in the Master User Interface. It contains several configuration sections:

- Memory Channel Configuration:** Number of Parallel Memory channels: 1
- Memory Channel 0 Configuration:** Protocol 0: none, Memory device on chip select 0: none
- Memory Channel 1 Configuration:** Protocol 1: HyperRAM, Memory device on chip select 1: IS66WVH16M8 ALL (ISSI) 1.8V HyperRAM 166 MHz 128 Mbit
- 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: 150 us
- Memory RWDS-DQ SKEW Configuration:** ☒ Enable Default RWDS-DQ SKEW settings, DQ-DQS skew on channel 0 in ps: 500 ps
- Avalon Target Interface Configuration:** Avalon control port: Disabled, Include PSRAM ECC input signal: Disabled

The following is a short description of SLL xSPI-MC IP parameters.

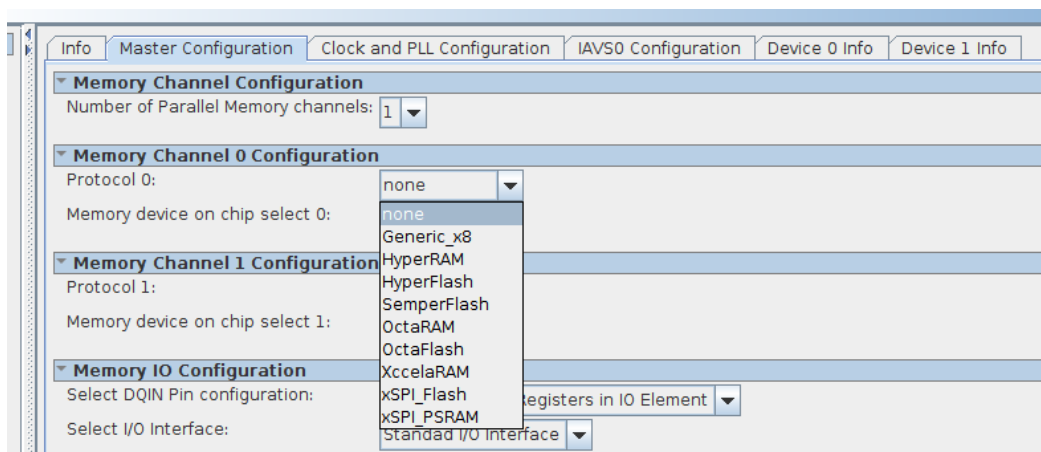
**Memory Device 0 Configuration** contains a subset of configurable fields for the memory device connected on Chip Select 0. If no device is present select **None** in the **Memory Device 0 Name**.

**Memory Device 1 Configuration** contains a subset of configurable fields for the memory device connected on Chip Select 1. If no device is present select **None** in the **Memory Device 1 Name**.

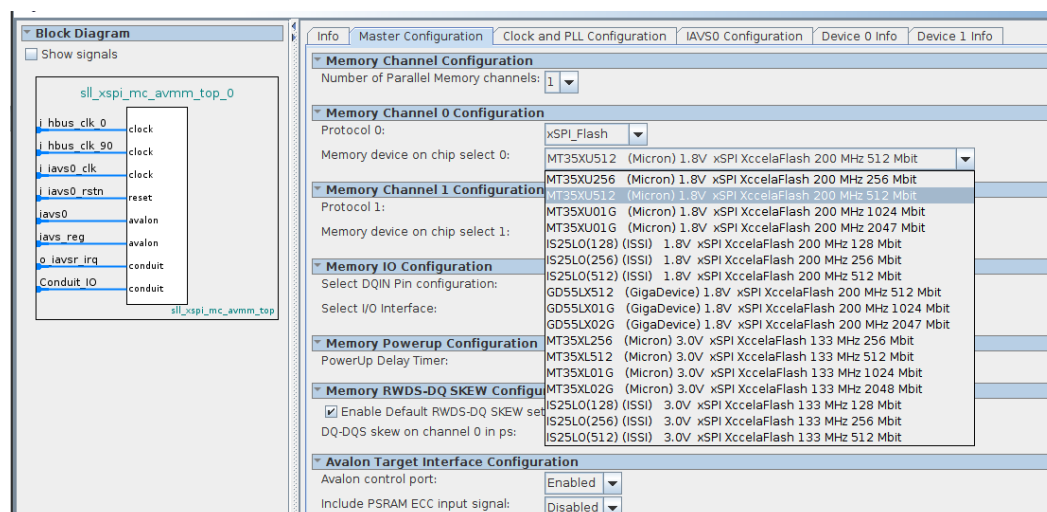


## 2.2.1 Memory Channel 0/1 Configuration

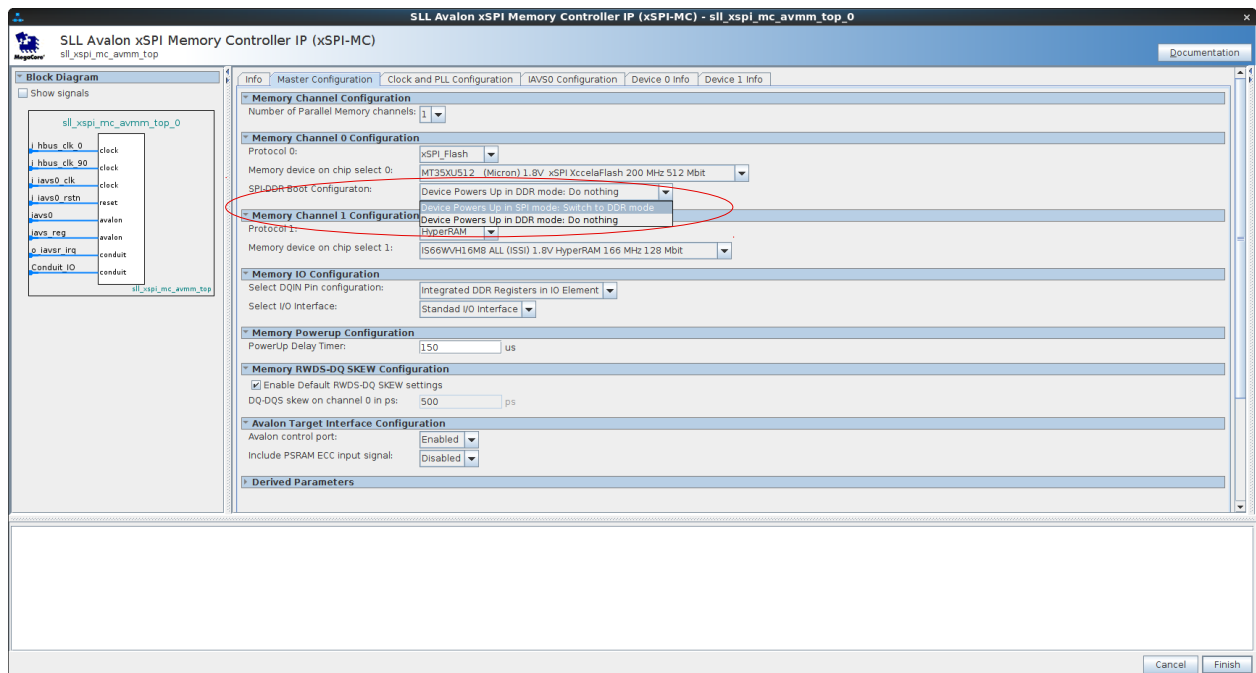
- Memory Device 0/1 Name** : Selects the Flash/PSRAM memory device present on the FPGA board. Currently, SLL xSPI-MC supports the following protocols : Hyperram, HyperFlash 1.0, SemperFlash, OctaRAM, OctaFlash, XccelaRAM and xSPI Profile 1.0 Flash and xSPI-Profile 2.0 PSRAM. If no device is present select **None**. The example below shows a section of the list of devices supported. Kindly contact Synaptic Labs' ([info@synaptic-labs.com](mailto:info@synaptic-labs.com)) in case the Flash device present on your FPGA board is not available in the list.



- Memory Device on chip select 0/1:** This is a list of supported devices based on the protocol selected in the **Memory Device Protocol** field. Select the device present on the FPGA board. The example below shows a list of devices supported under the xSPI Flash Protocol. Kindly contact Synaptic Labs' ([info@synaptic-labs.com](mailto:info@synaptic-labs.com)) in case the PSRAM/Flash device present on your FPGA board is not available in the list.



## Device 0/1 SPI-DDR Boot Config



- **SPI- DDR Boot Config** (where applicable) : Most memory devices supported by SLL xSPI-MC IP powers up in x8 DDR mode. However, certain memory devices powers up in SPI (x1) mode after reset. In this cases, the memory device needs to be configured to switch to x8 DDR mode after powerup. This can be achieved by selecting the
  - **Device Powers Up in SPI mode : Switch to DDR** field. SLL xSPI-MC IP will automatically configure the memory device to x8 DDR after reset.

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.

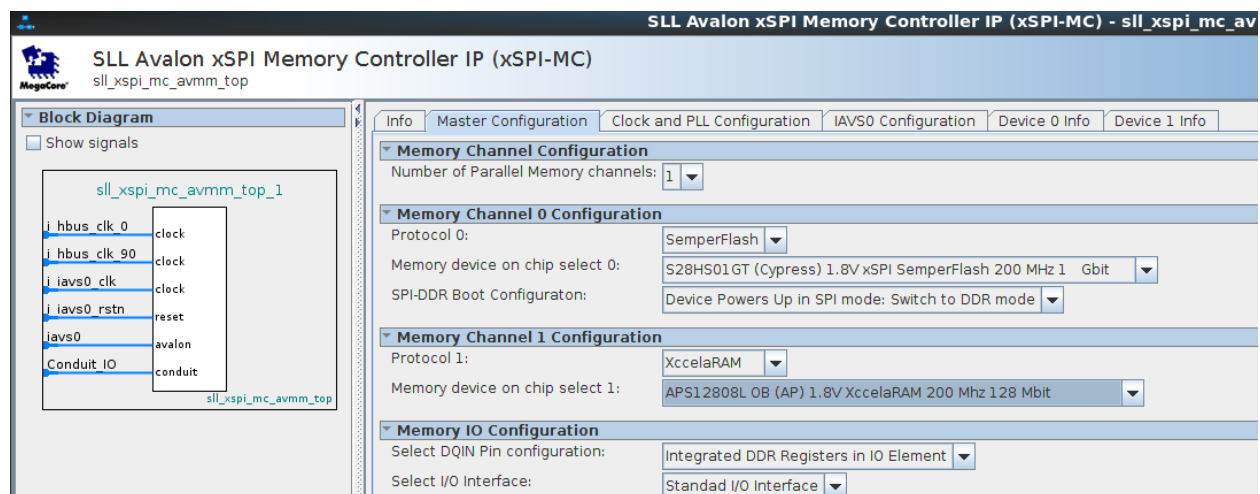
For example, to connect an Infineon S28HS01GT Flash memory (Infineon S28HS family) on chip select 0, and AP Memory APS12808 (XccelaRAM family) on chip select 1, the user needs to set the follow:

The Memory Channel 0 Configuration field is set to :

- **Protocol1** as [SemperFlash](#)
- **Memory device on chip select 0** as [S28HS01GT](#).
- **SPI-DDR Boot Configuration** as [Device Powers up in SPI : Switch to DDR](#)

The Memory Channel 1 Configuration field is set to :

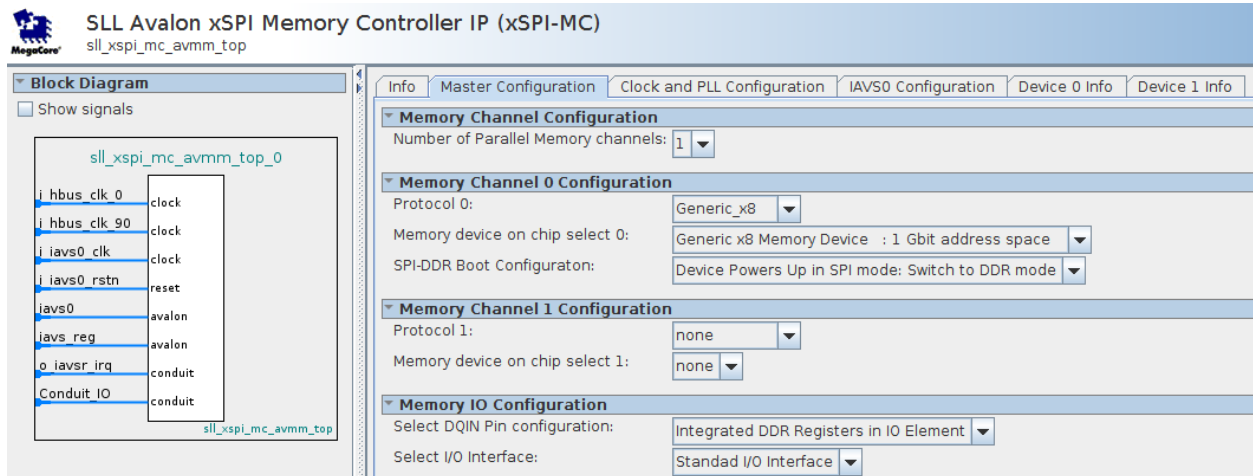
- **Protocol1** as [XccelaRAM](#)
- **Memory device on chip select 0** as [APS12808](#).



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.

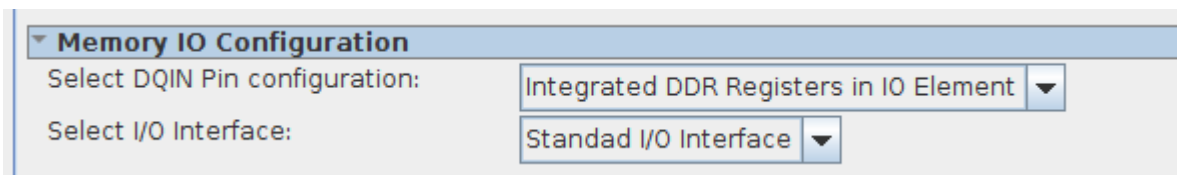


## 2.2.2 Memory IO Configuration

The user can either select

- Direct Connection or
- Integrated DDR Register in the IO Element (preferred option)

The Integrated DDR feature can be used for all device families (not just Cyclone V). The Integrated DDR feature instantiates the PHY in a way that enables the DDR FF inside the I/O Elements. The Integrated DDR Register in the IO Element option allows for potential higher clock speeds to be achieved.

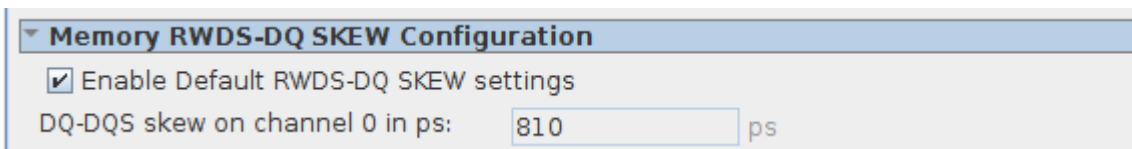


▼ **Memory IO Configuration**

Select DQIN Pin configuration: Integrated DDR Registers in IO Element ▼

Select I/O Interface: Standard I/O Interface ▼

## 2.2.3 RWDS-DQ Skew Configuration



▼ **Memory RWDS-DQ SKEW Configuration**

☒ Enable Default RWDS-DQ SKEW settings

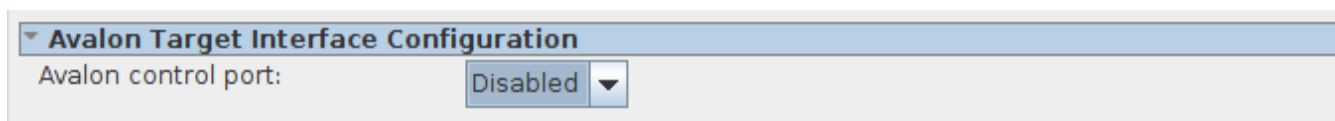
DQ-DQS skew on channel 0 in ps: 810 ps

When the **[x] Enable Default RWDS-SKEW** is checked, the RWDS-DQ skew will be automatically calculated.

When the **[ ] Enable Default RWDS-SKEW** option is left unchecked, the user can enter the RWDS-DQ skew manually.

The value will be used to set the correct timing parameters in the timing constraint script automatically generated by SLL MBMC IP.

## 2.2.4 Avalon Target interface Configuration



▼ **Avalon Target Interface Configuration**

Avalon control port: Disabled ▼

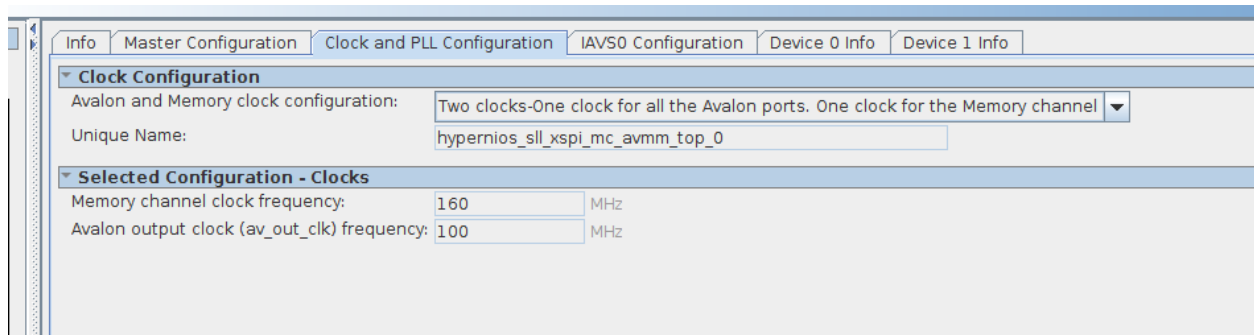
When enabled, a secondary Avalon-MM port is created.

This can be used for programming HyperFlash or accessing pseudoRAM registers.



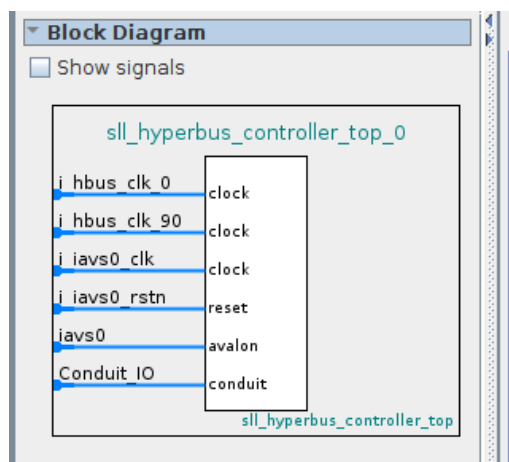
## 2.3 Configuring SLL xSPI-MBMC IP in the Clock and PLL Configuration Tab

SLL xSPI-MBMC IP is configured in external PLL mode. Internal PLL mode is no longer supported. The option is no longer visible in the configuration settings.



### 2.3.1 External PLL mode

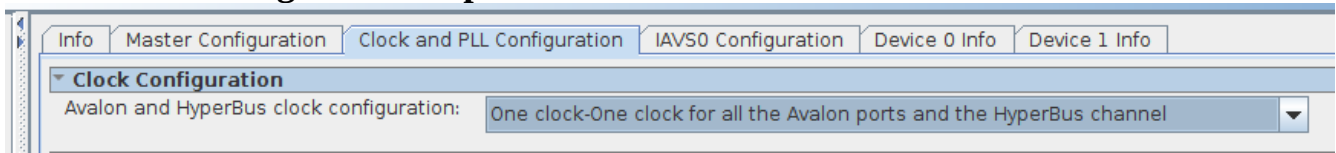
The different Avalon-MM master/slave clocks and PSRAM/Flash Clocks in the design are generated by an external PLL.



In the external PLL mode,  
SLL-xSPI MBMC IO uses 3 clock signals

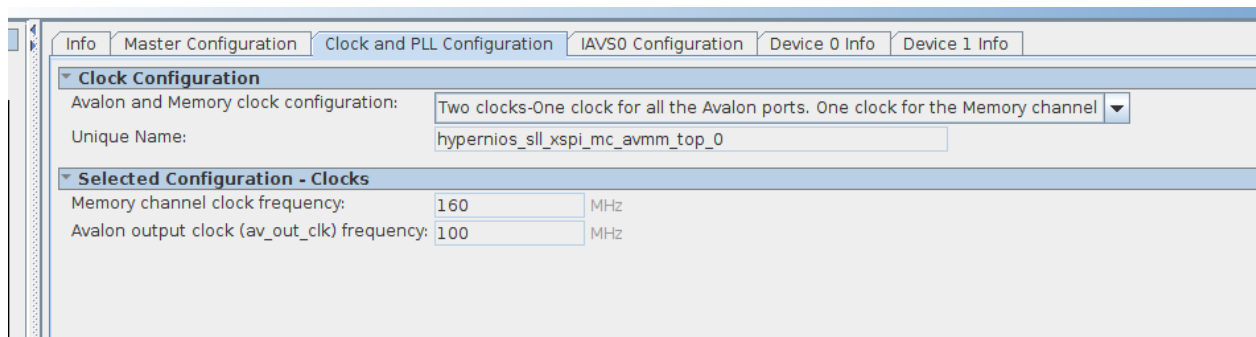
**i\_hbus\_clk\_0** : input clock driving the Hyperbus controller  
**i\_hbus\_clk\_90** : input clock driving the Hyperbus controller  
Operates at same frequency as i\_hbus\_clk\_0  
Phase shifted 90 degrees  
**i\_iavs\_clk** : input clock driving the Avalon-MM interface

### 2.3.1.1 For Single Clock Operation



Select this option when the PSRAM/Flash memory channel is operating at the same frequency as the Avalon Memory slave interface.

### 2.3.1.2 For Dual Clock Operation



Select this option when the PSRAM/Flash memory channel is operating at a different frequency than the Avalon Memory slave interface.

## 2.4 Configuring SLL MBMC IP in the IAVS0 Configuration Tab

The Avalon port slave has its own port configuration tab. This offers great flexibility and enables the interface to be configured under different parameters.

The screenshot shows the 'IAVS0 Configuration' tab in a software interface. The tab is divided into several sections:

- IAVS0: Ingress Avalon port stage**
  - ☒ Enable Avalon write capability
  - ☒ Enable Avalon byte-enable capability
  - Access capabilities: Read/Write
  - ☐ Register Avalon write data path (generally recommended for high clock speed designs)
- IAVS0: Ingress Avalon address/data**
  - Address width: 22 bits
  - Address units: Words
  - Word width: 32 bits
- IAVS0: Burst converter and address decoder stage**
  - maxBurstSize (in words): 8
  - linewrapBursts: true
  - burstOnBurstBoundariesOnly: false
- IAVS0: Ingress Avalon return stage**
  - ☐ Register Avalon read data path (sometimes used to increase top clock speeds)
  - ☐ Use Avalon transaction responses

### 2.4.1 Configuring Ingress Avalon port slave parameters

The "Enable Avalon write capability" feature is enabled by default and adds the Avalon write signal conduits on this slave interface. If the Avalon slave interface is a read only connection, such as the Nios II instruction master, the write signals can be disabled by un-ticking this box.

The "Enable Avalon byte-enable capability" feature is enabled by default and adds the Avalon byeeenable signal conduit on this slave interface. For bus masters that do NOT employ the byte-enable signals, this feature can be turned off.

The "Register Avalon Write data" feature allows the registration of the write data signal on the Avalon slave to potentially increase clock speeds and ease place-and-route.

## 2.4.2 Configuring the burst count on the Ingress Avalon port slave

The "**maxBurstSize (in words)**" GUI label select the maximum burst length of read and/or write burst memory transfer requests on this port in words. The value of **maxBurstSize** must be set to the value of the largest **maxBurstSize** of all Avalon bus masters interfaces that are connected to this slave interface.

When a burst size greater than 1 is selected, additional parameters become available to configure the burst behaviour.

If one or more of the Altera Avalon bus master interfaces connected to this slave port employs **linewrapBursts=true**, then set the **linewrapBursts** of this slave port to true.

An additional parameter **burstOnBurstBoundariesOnly** becomes available.

- Set the value of **burstOnBurstBoundariesOnly** to true for the slave port when all bus master interfaces connected to this port have their **burstOnBurstBoundariesOnly** parameter set to true.
- Otherwise, set the value of **burstOnBurstBoundariesOnly** to false for the slave port.

Example : if the Nios II processor is configured with

- Instruction cache enabled and the "Add burstcount signal to instruction master " enabled. (This will configure the Nios II instruction master with a burst size of 8 and **linewrapBursts=true**).
- Data cache enabled and the "Add burstcount signal to data master " enabled. (This will configure the Nios II data master with a burst size of 8, **linewrapBursts=false** and **burstOnBurstBoundariesOnly=true**).
- and we are connecting both the Instruction Master and the Data master to SLL Multiple Bus Memory Controller Avalon-MM slave

then we set **maxBurstSize (in words)** to 8, and **linewrapBursts=true**.

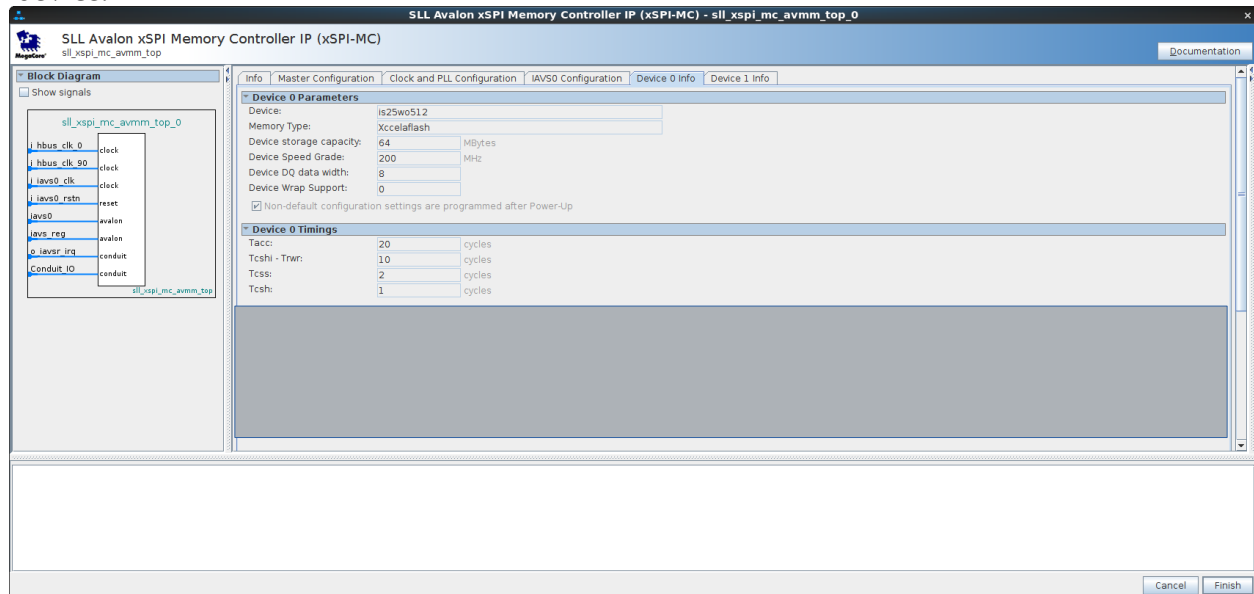
### 2.4.3 Configuring the Ingress Avalon port slave return path

The “**Register Avalon Read data**” feature allows the registration of the read data signal on the Avalon slave to potentially increase clock speeds and eases place-and-route.

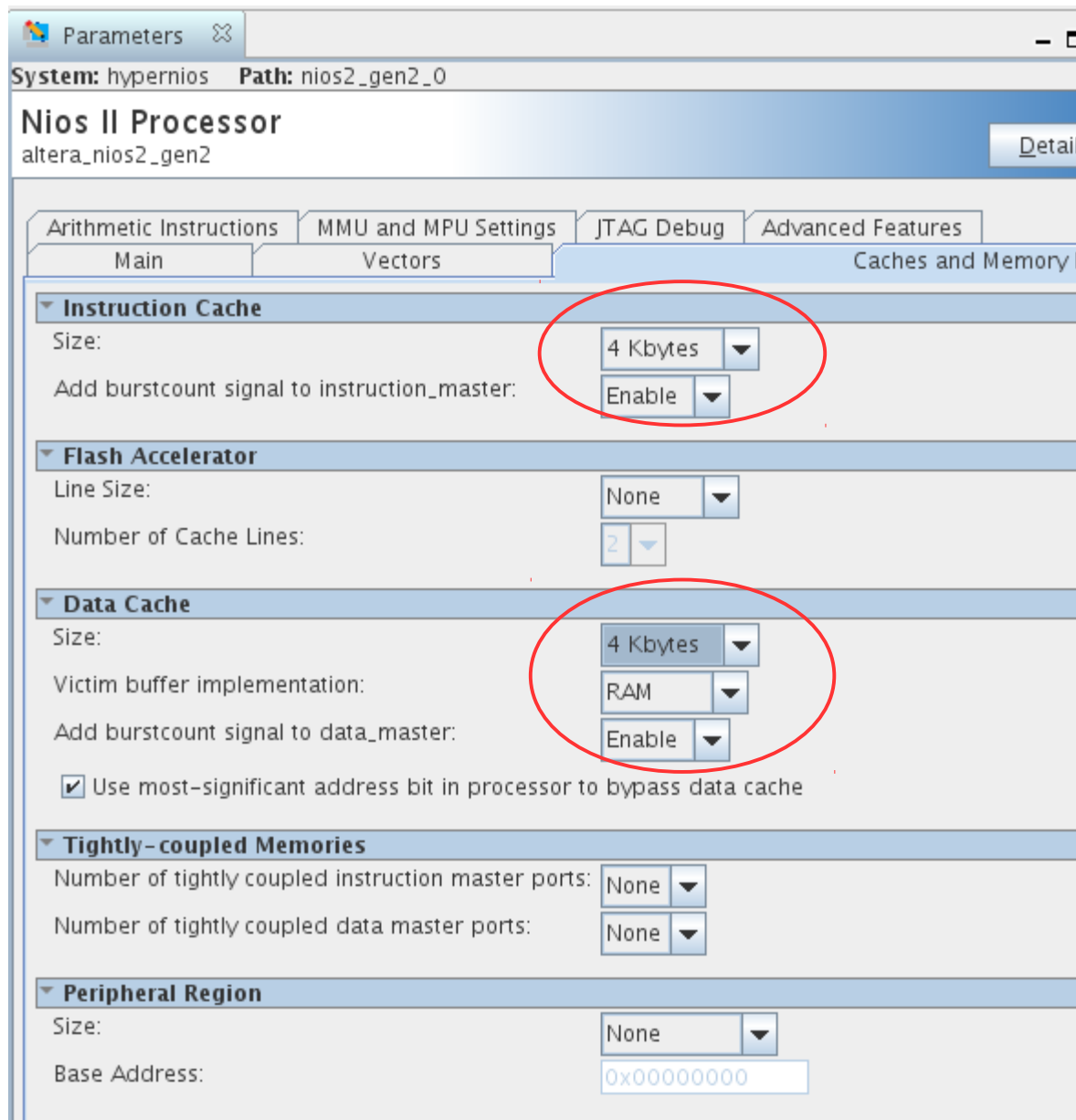
The “**Use Avalon Transaction Response**” feature adds the Avalon transaction response conduit.

## 2.5 Configuring SLL xSPI-MBMC IP in the Device 0/1 Tab

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



### 3.0 Connecting SLL xSPI-Multi Bus Memory Controller to the NIOS II/f embedded processor



Configure the Nios II/f processor with Instruction/Data caches enabled and **burstcount** signals enabled.

Configure the SLL xSPI-Multi Bus Memory Controller Avalon slave (IAVS0 configuration tab) with a **burstsize** of 8 and **linewrap burst** enabled.

The screenshot shows the 'IAVS0 Configuration' tab of a software interface. The tab is selected, and the configuration is organized into four expandable sections:

- IAVS0: Ingress Avalon port stage**
  - ☒ Enable Avalon write capability
  - ☒ Enable Avalon byte-enable capability
  - Access capabilities: Read/Write
  - ☐ Register Avalon write data path (generally recommended for high clock speed designs)
- IAVS0: Ingress Avalon address/data**
  - Address width: 22 bits
  - Address units: Words
  - Word width: 32 bits
- IAVS0: Burst converter and address decoder stage**
  - maxBurstSize (in words): 8
  - linewrapBursts: true
  - burstOnBurstBoundariesOnly: false
- IAVS0: Ingress Avalon return stage**
  - ☐ Register Avalon read data path (sometimes used to increase top clock speeds)
  - ☐ Use Avalon transaction responses



### 3.1 SLL xSPI-MBMC with external PLL Configuration

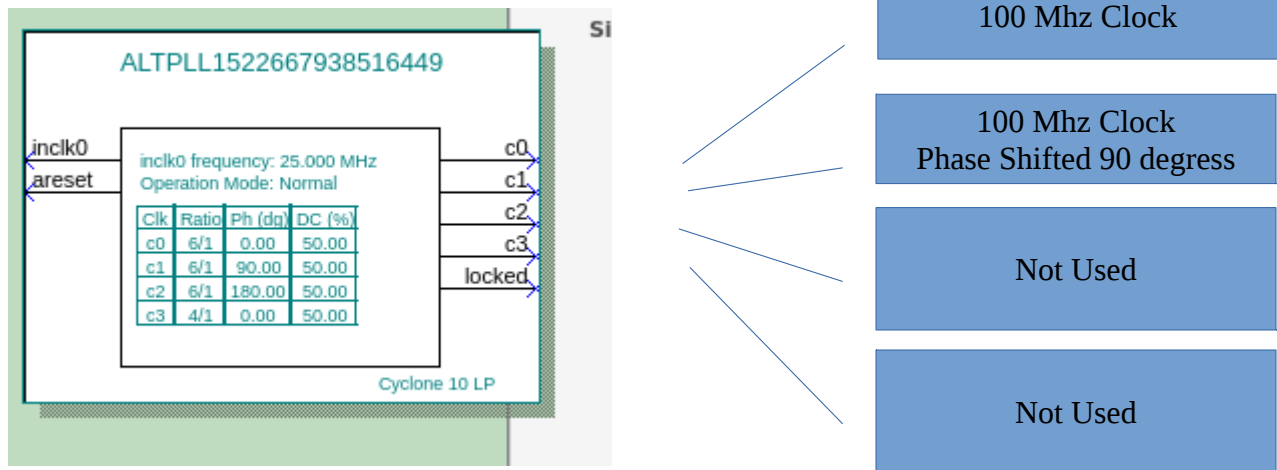
With the external PLL configuration, SLL xSPI-MBMC IP requires 4 clocks :

- **i\_hbus\_clk\_0** : clock driving the Multiple Bus Memory Controller
- **i\_hbus\_clk\_90** : clock for driving some PSRAM/Flash I/O Signals. It operates at the same frequency as **i\_hbus\_clk\_0** but is phase shifted 90 degrees
- **i\_iavs0\_clk** : clock driving the Avalon-MM interface. When SLL xSPI-MBMC IP is configured to run at a single clock speed, this clock is connected to i\_hbus\_clk\_0 clock.

### **3.1.1 Option A – Same Clock for the PSRAM/Flash memory channel an Avalon-MM channel.**

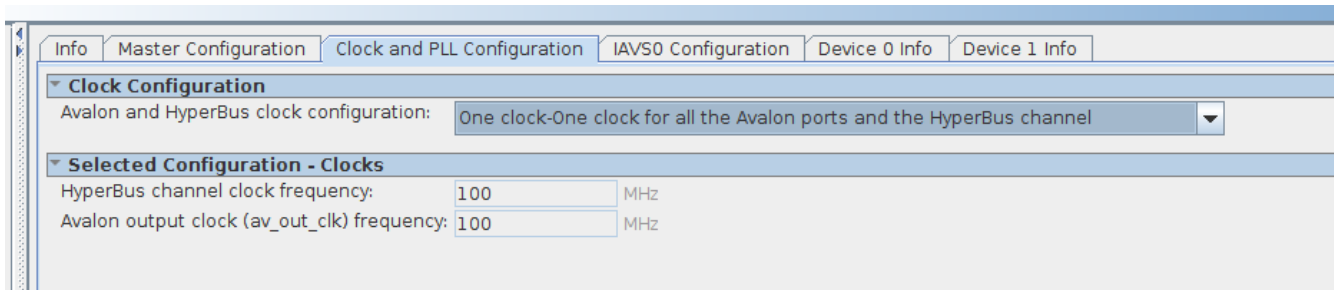
This configuration shows how to connect SLL MBMC IP so that the PSRAM/Flash memory channel operates at the same frequency as the Avalon-MM bus interface. The advantage of this configuration is lower circuit area.

### 3.1.2 Clocking (PLL) Wizard Configuration



The figure below shows a typical example of configuring the Clocking wizard . In this case, the clocks for the PSRAM/Flash channel and Avalon-MM interface channels are all set to 100 Mhz.

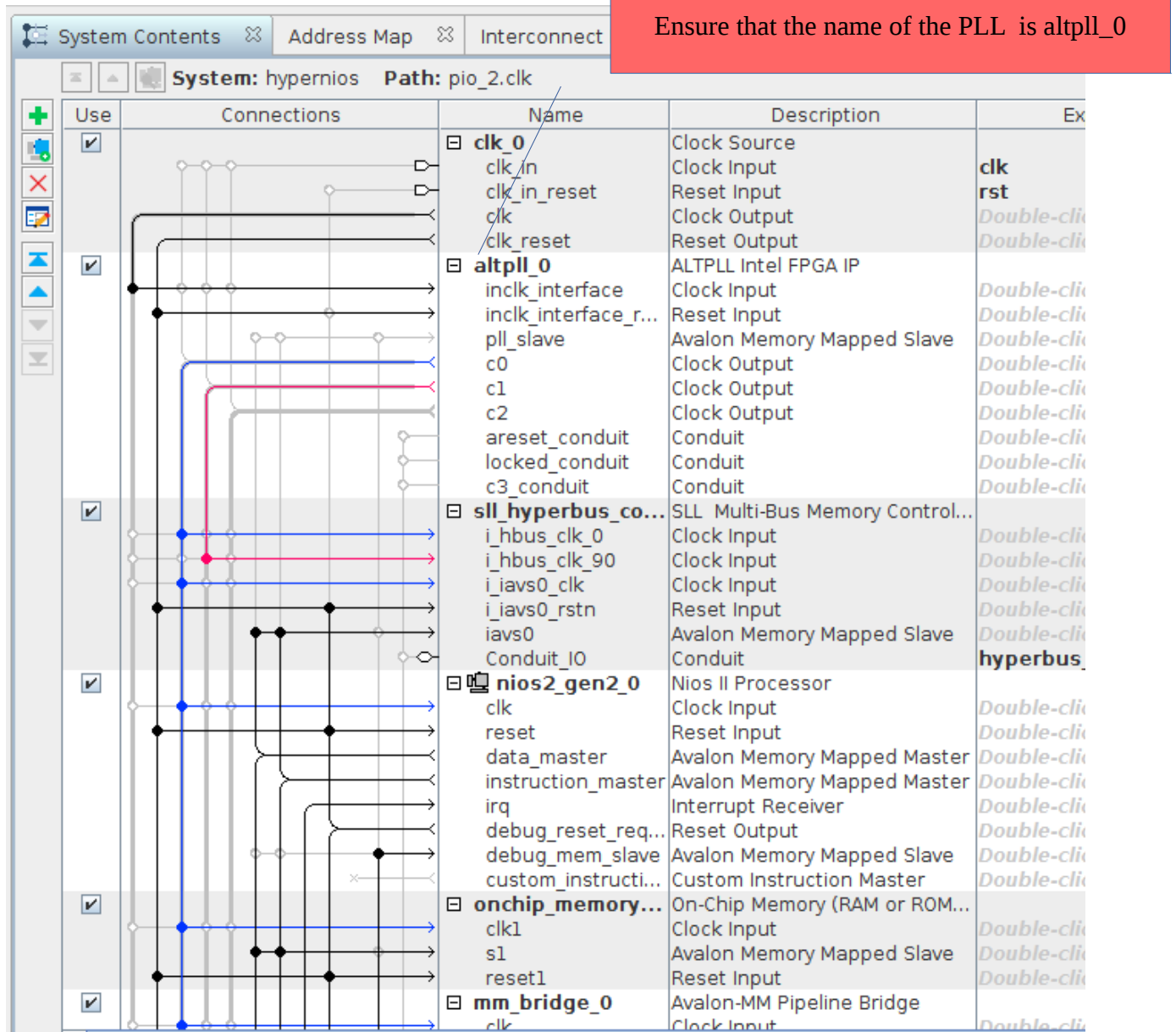
### 3.1.3 SLL xSPI-MBMC Configuration



In this example, SLL xSPI-MBMC IP is configured with :

- **Avalon and Memory clock configuration : One clock**

### 3.1.4 SLL xSPI-MBMC wiring



Altera PLL output clock 0 (c0)

- Connect to i\_hbus\_clk\_0 and i\_iavs0\_clk on SLL xSPI-MBMC IP
- Connect to other Avalon-MM slaves and masters clock sinks

Altera PLL output clock 1 (c1)

- Connect to i\_hbus\_clk\_90 on SLL xSPI-MBMC IP

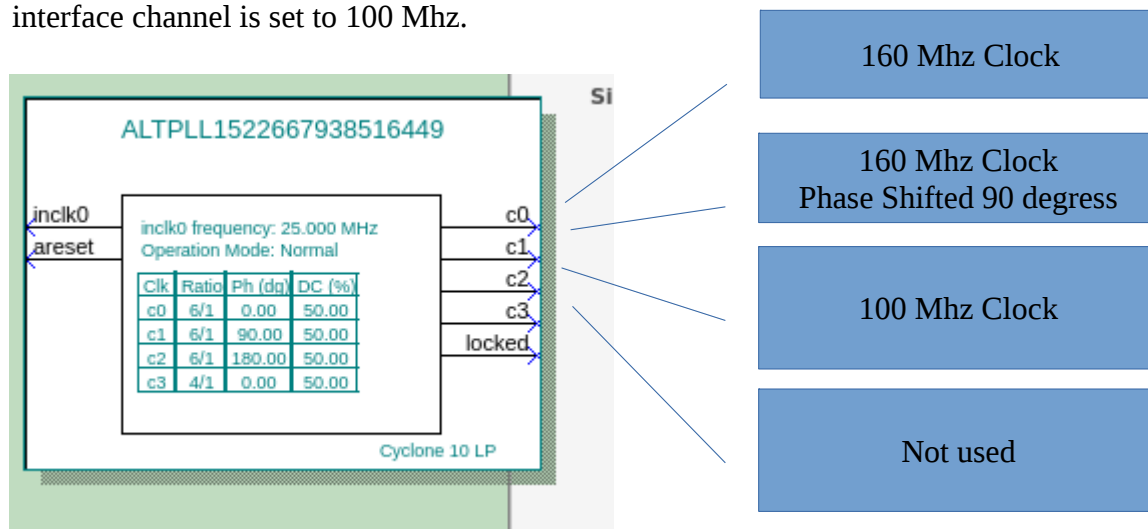
**Note how i\_hbus\_clk\_0 and i\_iavs0\_clk are connected to the same clock.**

## 3.2 Option B – Different Clocks for the PSRAM/Flash memory channel and Avalon-MM channel .

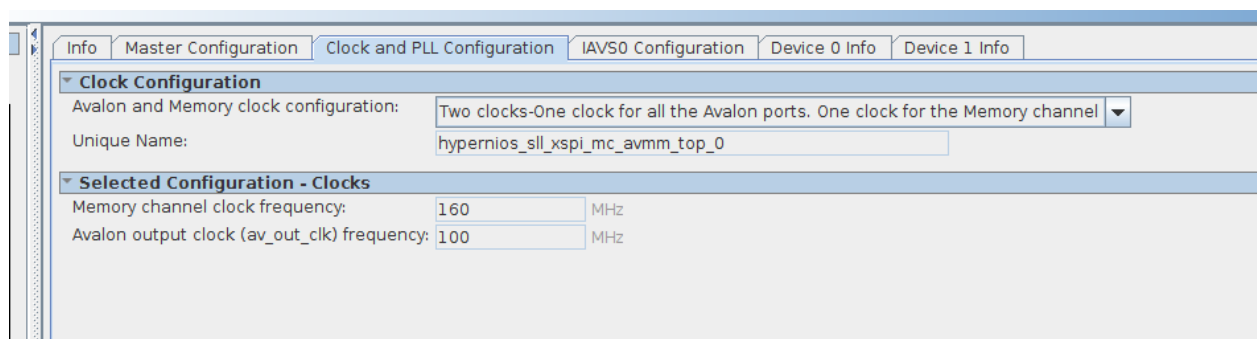
This configuration shows how to connect SLL xSPI-MBMC IP so the PSRAM/Flash memory channel operates at a different clock frequency than the Avalon-MM bus interface.

### 3.2.1 Clocking Wizard Configuration

The figure below shows a typical example of configuring the Clocking wizard . In this case, the clocks for the PSRAM/Flash channel are all set to 160 Mhz, while the clock for the Avalon-MM interface channel is set to 100 Mhz.



### 3.2.2 SLL xSPI-MBMC Configuration

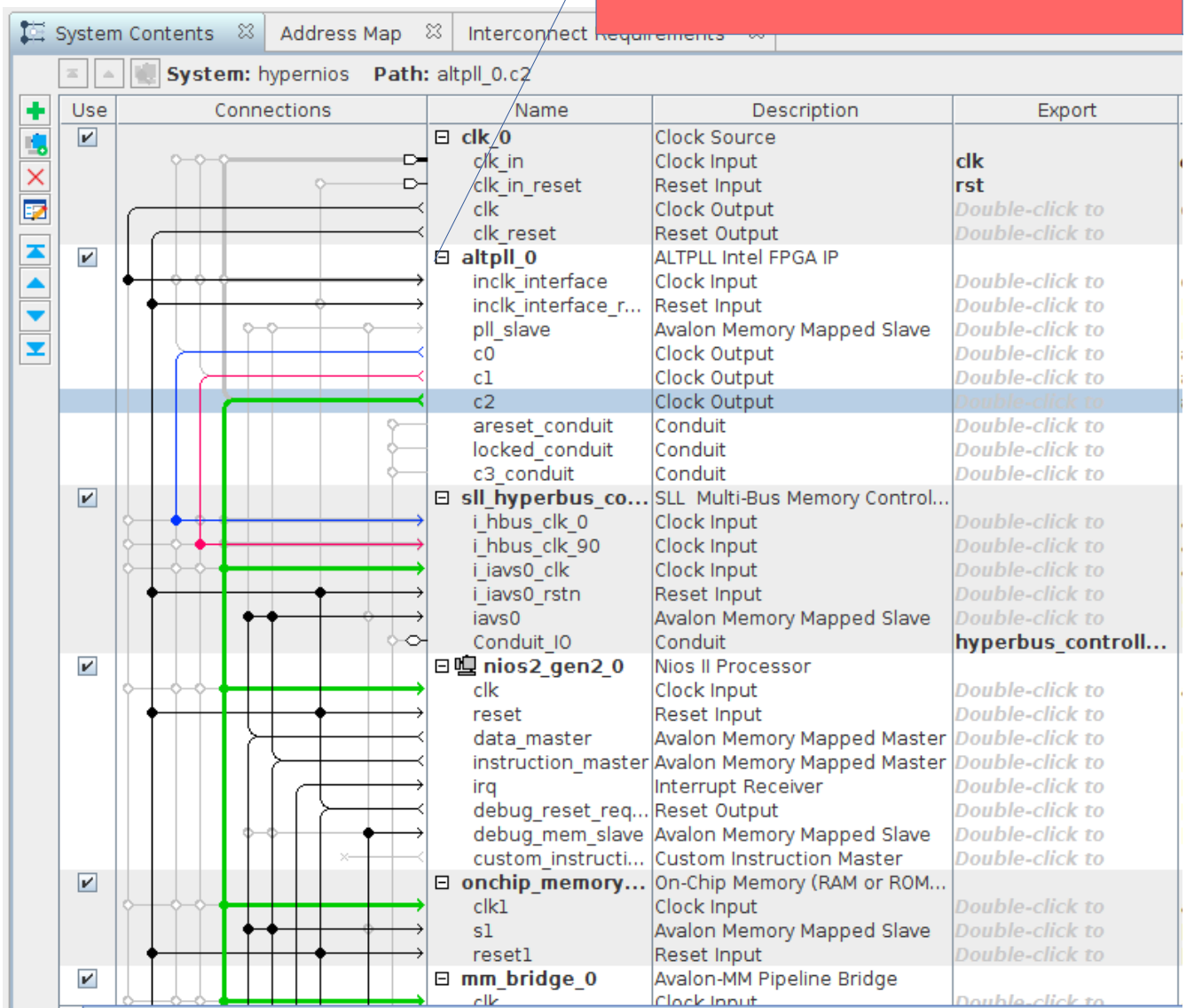


In this example, SLL xSPI MBMC IP is configured with :

- **Avalon and Memory clock configuration : Two clocks**

### 3.2.3 SLL xSPI-MBMC wiring

Ensure that the PLL name is altpll\_0



Altera PLL output clock 0 (c0)

- Connect to i\_hbus\_clk\_0 on SLL xSPI-MBMC IP

Altera PLL output clock 1 (c1)

- Connect to i\_hbus\_clk\_90 on SLL xSPI-MBMC IP

Altera PLL output clock 2 (c2)

- Connect to i\_iavs0\_clk on SLL xSPI-MBMC IP
- Connect to other Avalon-MM slaves and masters clock sinks

Note how i\_hbus\_clk\_0 and i\_iavs0\_clk are connected to a different clock.

## *Important :*

*Please note that SLL MBMC contains a script that sets timing constraints for the PSRAM/Flash IO signals. For the external PLL configuration, this script assumes that in Qsys, Altera PLL's instance name is altpll\_0.*

## 4.0 Signal Description

### 4.1 Memory Bus Signal Interface

Hyperbus Signal	Description	Comments	Board consideration
MB_CLK0p	Differential clock pair 0	To be connected to pin CK on Memory device 0.	Connect to memory CK signal
MB_CLK0n	Differential clock pair 0 (used with 1.8V devices)	To be connected to pin CK# on Memory device 0.	Connect to memory CK# signal when required.
MB_CLK1p	Differential clock pair 1	Additional Clock pair	Not connected
MB_CLK1n	Differential clock pair 1	Additional Clock pair	Not connected
MB_CS0n	Chip select device 0	To be connected to pin CS# on Memory device 0.	Connect to Flash CS# signal. Leave unconnected if Flash not present
MB_CS1n	Chip select device 1	To be connected to pin CS# on Memory device 1.	Connect to PSRAM CS# signal
MB_Wpn	Disables writes to Flash memory devices (Write protect)	To be connected to pin WP# on any Flash device.	Connect to Flash WP# if required. <i>Currently not used.</i>
MB_RWDS	Read strobe/Write mask signal	To be connected to pin RWDS on all memory devices.	Connect to PSRAM / FLASH RWDS signal
MB_DQ	Data bus (8-bit)	To be connected to DDR Data pins on all memory devices.	Connect to PSRAM / Flash Dq signals
MB_INTn	Interrupt from Flash Devices to FPGA.	To be connected to pin INT# on all Flash device.	Connect to Flash INT# signal. <i>Currently not used</i>
MB_RST0	Reset from Flash Devices to FPGA	To be connected to pin RST0 on all Flash devices.	Connect to Flash RST0 signal when required
MB_RSTn	Reset to Memory Devices	To be connected to pin RESET# on all memory devices.	Connect to memory RST# signal when required

**Optional Pullup Resistor** *(not used for the PSRAM Only Configuraton)*

**MB\_INTn and MB\_RST0 signals** signals of the Hyperbus memories are open drain output without a pull-up resistor. Therefore, when these signals are used in the design, it is suggested that a pull-up resistor is inserted either by setting the appropriate constraint on the input FPGA pad or adding a resistor on the board itself.



## 4.2 Memory Clock Signal Interface

PLL Signal	Description	Comments
i_hbus_clk_0	Memory Channel Clock	Memory Clock
i_hbus_clk_90	Memory Channel Clock (90 phase shift)	90 degrees phase shift

## 4.3 Avalon Clock/reset interface

AXI signal	Description	Comments
i_iavs0_clk	Avalon-MM clock source	
i_iavs0_resetsn	Avalon-MM reset source	

## 4.4 Avalon Data Slave Interface

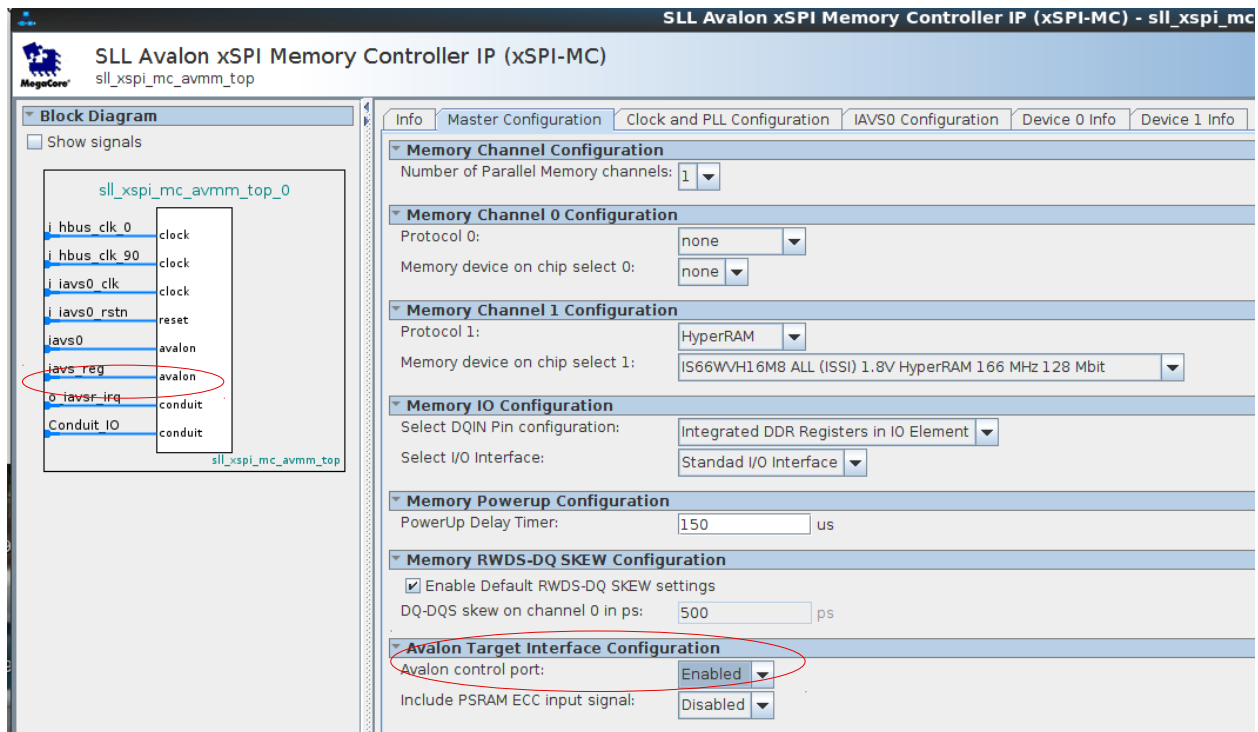
Avalon-MM signal	Description
i_iavs0_do_rd	Avalon Slave Read request
i_iavs0_do_wr	Avalon Slave Write request.
i_iavs0_addr	Avalon Slave Read/Write address.
i_iavs0_burstcount	Burst length. The burst length gives the exact number of transfers in a burst.
o_iavs0_wait_request	This signal indicates that the slave is ready to accept the address and associated control signals
i_iavs0_wdata	Write data
i_iavs0_byteenable	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
o_iavs0_rdata	Read data
o_iavs0_rdata_valid	Read Data valid. This signal indicates that data on o_iavs0_rdata bus is valid

## 4.5 Avalon Control Slave Interface (optional)

Avalon-MM signal	Description
i_iavsr_do_rd	Avalon Control Slave Read request
i_iavsr_do_wr	Avalon Control Slave Write request.
i_iavsr_addr	Avalon Control Slave Read/Write address.
i_iavsr_wdata	Write data
o_iavsr_rdata	Read data. Data is valid 1 cycle after i_iavsr_do_rd is active.

## 5.0 Avalon-MM Register Control Port

When The Avalon Register Control Port is enabled, a secondary Avalon-MM slave interface becomes present (iavs\_reg). This can be used for programming Flash and accessing PseudoRAM registers. Please note that read and write transactions on the Avalon-MM memory port always have precedence over memory transfer requests issued by this control port.

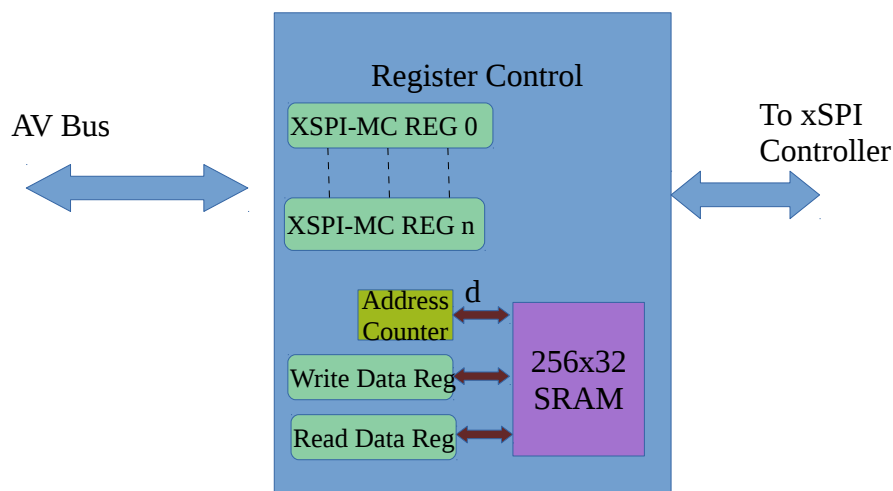


## 6.0 Avalon-MM Register Control Port

SLL xSPI-MBMC includes an Avalon-MM slave interface (iavs\_reg). This can be used for creating memory regions, configuring the xSPI controllers according to the devices present, programming Flash memory devices and also accessing PseudoRAM registers. Please note that read and write transactions over the Avalon-MM memory data port always have precedence over memory transfer requests issued by this control port.

The Register Control Port contains :

- a small bank of Read/Write Register for controlling logic and reading status of the core
- a small memory buffer (128x32 SRAM) for Flash programming



The Register Control Port includes a small memory buffer (128x32 SRAM). This is particularly useful for programming the xSPI flash devices as well as reading multi-word registers from memory devices.

Multiple access to the xSPI-MC\_WDATA Register will fill data into consecutive locations in the SRAM.

Multiple access to the xSPI-MC\_RDATA Register will get data from consecutive locations in the SRAM.

The Address counter for accessing the SRAM memory is

- incremented every time the xSPI-MC\_WDATA Register is accessed
- incremented every time the xSPI-MC\_RDATA Register is accessed.
- cleared automatically when xSPI-MC\_CMD is written
- cleared automatically when xSPI-MC\_CMD\_LEN is written
- cleared automatically when the memory controller acknowledge the operation
- cleared automatically at the end of a read memory operation

## 6.1 Register MAP

### 6.1.1 xSPI-MC Command Address Control registers

Address	Register Name	Direction	Comment
0x10	XSPI-MC-CAD_LEN	Read/Write	Number of 32-bit words to be transferred
0x14	xSPI-MC_CAD_HI	Write Only	Upper 16-bit data CA[47:32] (command/address phase **)
0x18	xSPI-MC_CAD_LOW	Write Only	Lower 32-bit CA[31:0] (command/address phase **)
0x1C	xSPI-MC_WDATA	Write Only	32-bit write data
0x2C	xSPI-MC_RDATA	Read Only	32-bit return read data

### 6.1.2 xSPI-MC Config/status registers

Address	Register Name	Direction	Comment
0x40	xSPI-MC_CMD	Write only	Command register
0x44	xSPI-MC_CFG	Read/Write	Configuration Register
0x48	xSPI-MC_IRQ_MASK	Read/Write	(TBD)
0x4C	xSPI-MC_IRQ_CLR	Write Only	(TBD)
0x4C	xSPI-MC_IRQ_STATUS	Read Only	(TBD)
0x50	XSPI-MC_DEC_ERR	Read Only	Errors related to AXI Decode
0x54	XSPI-MC_FIFO_ERR	Read Only	Errors related to FIFOs
0x5C	xSPI-MC_STATUS	Read only	Status register

### 6.1.3 xSPI-MC Memory Device 0/1 Config registers\*\*\*

\*\* Only used when controller is programmed in generic mode

Address	Register Name	Direction	Comment
0x80	xSPI-MC_DEV_0_CFG	Read/Write	CS0# Memory Configuration Register
0x84	xSPI-MC_DEV_0_TIM	Read/Write	CS0# Memory Timing Register
0x88-0x8C	xSPI-MC_DEV_0_CMD	Read/Write	CS0# Memory Command Register
0x90	xSPI-MC_DEV_1_CFG	Read/Write	CS1# Memory Configuration Register
0x94	xSPI-MC_DEV_1_TIM	Read/Write	CS1# Memory Timing Register
0x98-0x9C	xSPI-MC_DEV_1_CMD	Read/Write	CS1# Memory Command Register

### 6.1.4 xSPI-MC Memory Device 0/1 Memory Partition registers\*\*\*

\*\* Only used when controller is programmed in generic mode

Address	Register Name	Direction	Comment
0xA0	xSPI-MC_DEV_0_MBA	Read/Write	CS0# Memory Base Address Register
0xB0	xSPI-MC_DEV_1_MBA	Read/Write	CS1# Memory Base Address Register

### 6.1.5 xSPI-MC MODE Control registers\*\*\*

\*\* Only used when controller is programmed in generic mode

Address	Register Name	Direction	Comment
0xC0	xSPI-MC_MODE_CTRL	Write Only	Mode Control Register
0xC4	xSPI-MC_DLL_CTRL	Read/Write	DLL delay control

### 6.1.6 xSPI-MC SPI-TO-DDR Control registers\*\*\*

\*\* Only used when controller is programmed in generic mode

Address	Register Name	Direction	Comment
0xC8	xSPI-MC_SPI_TO_DDR0	Write Only	TBD
0xCC	xSPI-MC_SPI_TO_DDR1	Write Only	TBD

### 6.1.7 xSPI-MC ID registers

Address	Register Name	Direction	Comment
0xF0	xSPI-MC_ID_0	Read Only	SLL xSPI-MC ID 0
0xF4	xSPI-MC_ID_0	Read Only	SLL xSPI-MC ID 1

## 6.2 Register Definitions

### 6.2.1 SLL\_xSPI-MC\_CMD Register

<i>bit</i>	<i>Name</i>	<i>Description</i>	<i>Supported Values</i>
0	RW Mode	Read or Write operation	0 : Write 1 : Read
1	RegSpace	Selecting whether access is to the memory address space or register address space. The latter is typically used with PSRAM to program registers	0 : Memory Address Space 1 : Register Address Space
2	CS	Memory Device Selection	0 : Select Device 0 1 : Select Device 1
5:3	CK cycles	number of clocks pulses for Register write command Only	
6	SIZE	Data Transfer Size. Please note that for read operation, xSPI controller always uses 32-bit access.	0 : 16- bit write 1 : 32-bit write / 32-bit read
7	SPI	SPI Mode - not used	0
15:8	LAT	Read latency (used by xSPI controller during Register Read)	

When this register is written, a command is issued to the memory controller.



### 6.2.2 SLL\_xSPI-MC\_IRQ Status Register

<i>bit</i>	<i>Name</i>	<i>Description</i>
0	RdataTimeout	Timeout Error on Read Data Path
1	DecErr	Error at AXI Decode stage detected
2	ExtIRQ	External IRQ Pin status
3	FifoErr	Error on any of the Fifos detected
4	ExtEcc	External ECC Pin status
5	Cad_Ecc	ECC Error on Command Register detected
6	Reg_Ecc	ECC Error on Register RAM read detected
7	Rsv	Reserved

### 6.2.3 SLL\_xSPI-MC\_Fifo Status Register

<i>bit</i>	<i>Name</i>	<i>Description</i>
0	RxInfoPushErr	Read Data Path Info Buffer Push Error
1	RxInfoPopErr	Read Data Path Info Buffer Pop Error
2	RxFifoPushErr	Read Data Path Fifo Push Error
3	RxFifoPopErr	Read Data Path Fifo Pop Error
4	RxFifoEccErr	Read Data Path RAM Ecc Error
7:5	Rsv	Reserved
8	TxInfoPushErr	Write Data Path Command Buffer Push Error
9	TxInfoPopErr	Write Data Path Command Buffer Pop Error
10	TxFifoPushErr	Write Data Path Fifo Push Error
11	TxFifoPopErr	Write Data Path Fifo Pop Error
12	TxFifoEccErr	Write Data Path RAM Ecc Error
15:13	Rsv	Reserved

### 6.2.4 SLL\_xSPI-MC\_Decom Status Register

<i>bit</i>	<i>Name</i>	<i>Description</i>
0	AxiWr	AXI command is write
1	AxiRd	AXI command is read
2	RstActive	Controller in reset state
3	CtrlNotActive	Controller in not in active state
4	SetupNotReady	Controller in not ready to accept any commands
5	NoRegion	AXI address point to no region
6	MultipleRegion	AXI address point to multiple regions

### 6.2.5 SLL\_xSPI-MC\_STATUS Register

<i>bit</i>	<i>Name</i>	<i>Description</i>
9:0	IrqStat	IRQ status
15:10	rsv	Reserved
16	PwrUp	When 1, denotes that memory controller is still powering up
17	ReadValid	Following a single Read command issued through this register port, this bit indicates that the operation is completed and data in the xSPI-MC_RDATA register is valid. This bit will be cleared when SLL_xSPI-MC_STATUS register is read over the APB interface
18	CmdActive	Indicates that the Register Command is still active or pending. This bit will be cleared when SLL_xSPI-MC_STATUS register is read over the APB interface
19	ReadAllValid	Following a multi-word Read command issued through this register port, this bit indicates that the operation is completed and data in the xSPI-MC_RDATA RAM is valid. This bit will be cleared when SLL_xSPI-MC_STATUS register is read over the APB interface
20	SpiCmdActive	When 1, denotes that memory controller is still performing a SPI to DDR command. This status bit is only important when sending SPI commands over the <b>SPI-TO-DDR</b> registers.

It is not necessary to poll the SLL\_xSPI-MC\_STATUS Register bit 16 (PwrUP) to start accessing the HyperFlash / PseudoRAM devices. The xSPI-MC AXI interface forces signals AXI awready and AXI arready low until power-up is completed. The AXI read/write request will be taken once power-up is ready.

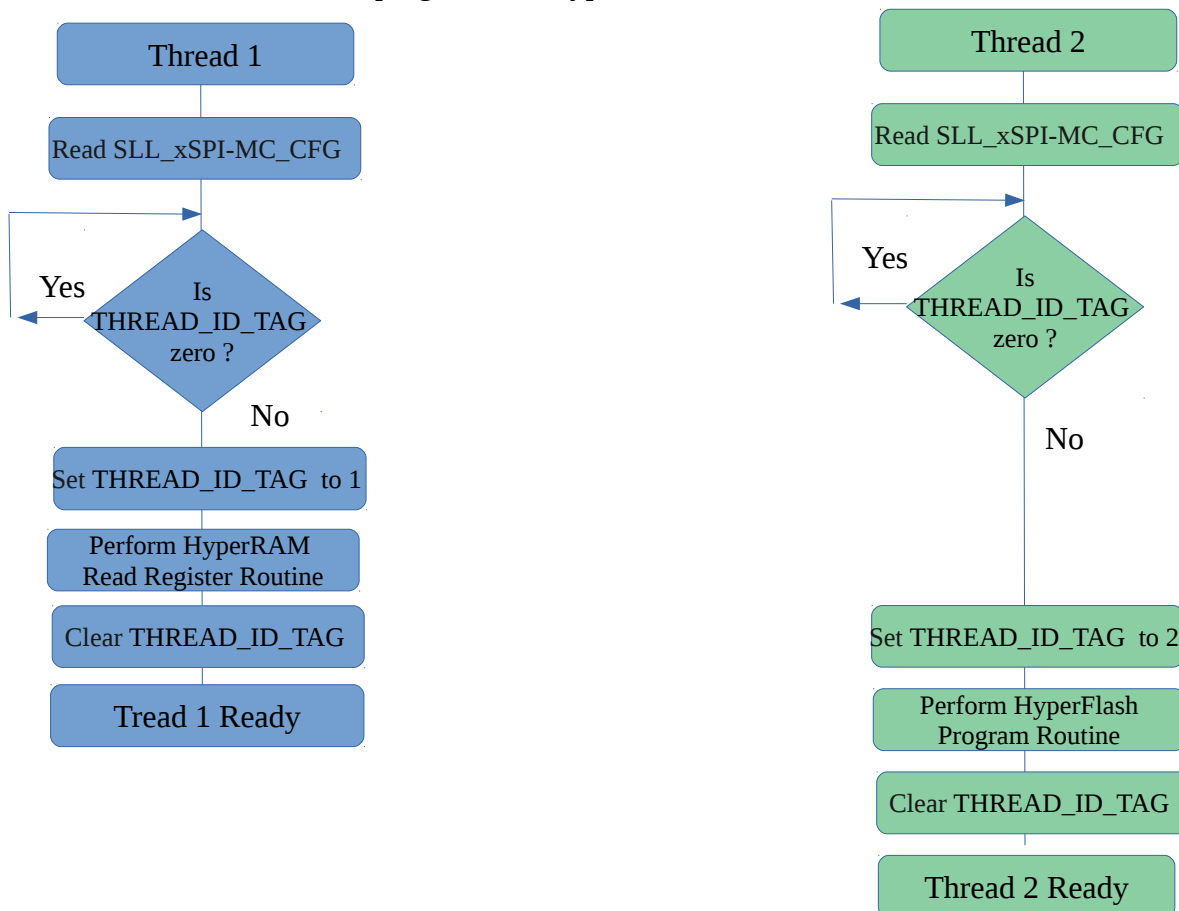
### 6.2.6 SLL\_xSPI-MC\_THREAD Register (to be further defined)

bit	Name	Description
7:0	THREAD_ID_TAG	Can be used to signal which thread is currently in use. Value of 0 means that the Register Control Port is currently not In use.
11:8	Reserved	

Note : It is very important that operations through the Register Control Port are accessed by only one thread at a time. The THREAD\_ID\_TAG field can be used to control different threads that need to perform operations through the Register Control Port.

As an example, let's suppose that there are 2 independent threads running

- Thread 1 wants to read the HyperRAM registers
- Thread 2 wants to program the HyperFlash



### 6.2.7 SLL\_xSPI-MC\_CAD\_LEN Register

This denotes the number of 32-bit words to be transferred during a memory operation issued by the register control module. Maximum supported burst length is 128 words (512 bytes).

### 6.2.8 SLL\_xSPI-MC\_CAD\_HI and SLL\_xSPI-MC\_CAD\_LOW Registers

When accessing the register address space, these 2 register form the 48-bit Command Address field

- SLL\_xSPI-MC\_CA\_HIGH should be set to CA[47:32]
- SLL\_xSPI-MC\_CA\_Low should be set to CA[31:0]

When accessing the memory address Space,

- SLL\_xSPI-MC\_CA\_HIGH should be set to 0
- SLL\_xSPI-MC\_CA\_Low should be set to the half-word aligned address (32-bit address >> 1).

### 6.2.9 SLL\_xSPI-MC\_WDATA Register

Write Data sent during write operation. Multiple writes will fill consecutive locations of the Register Control SRAM memory.

### 6.2.10 SLL\_xSPI-MC\_RDATA Register

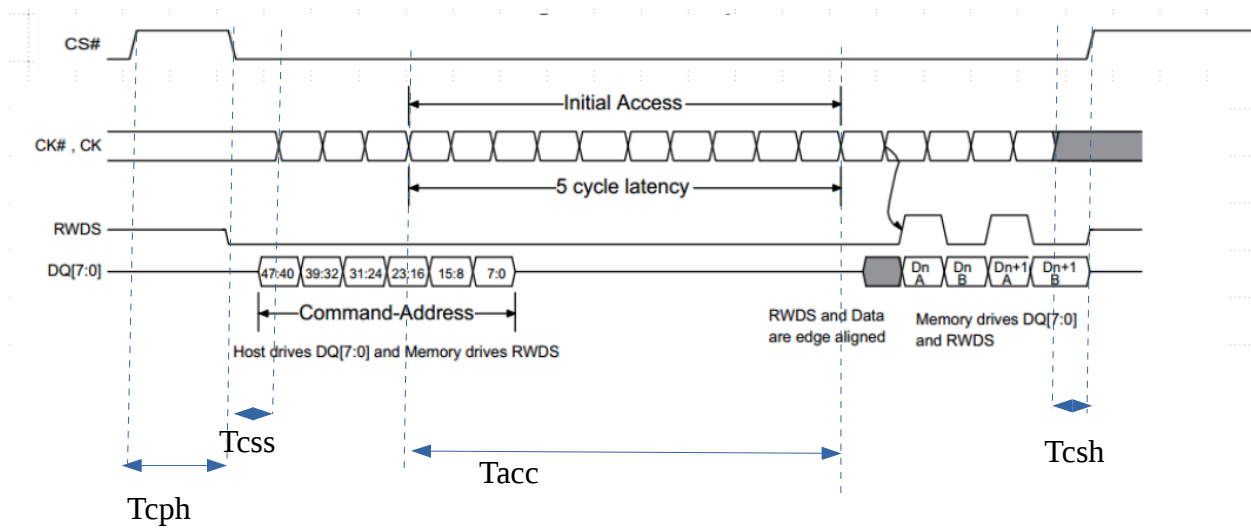
Return Read Data during a Read operation. Multiple reads will read consecutive locations in the Register Control SRAM memory.

### 6.2.11 xSPI-MC\_DEV\_x\_TIM Register (to be further defined)\*\*\*

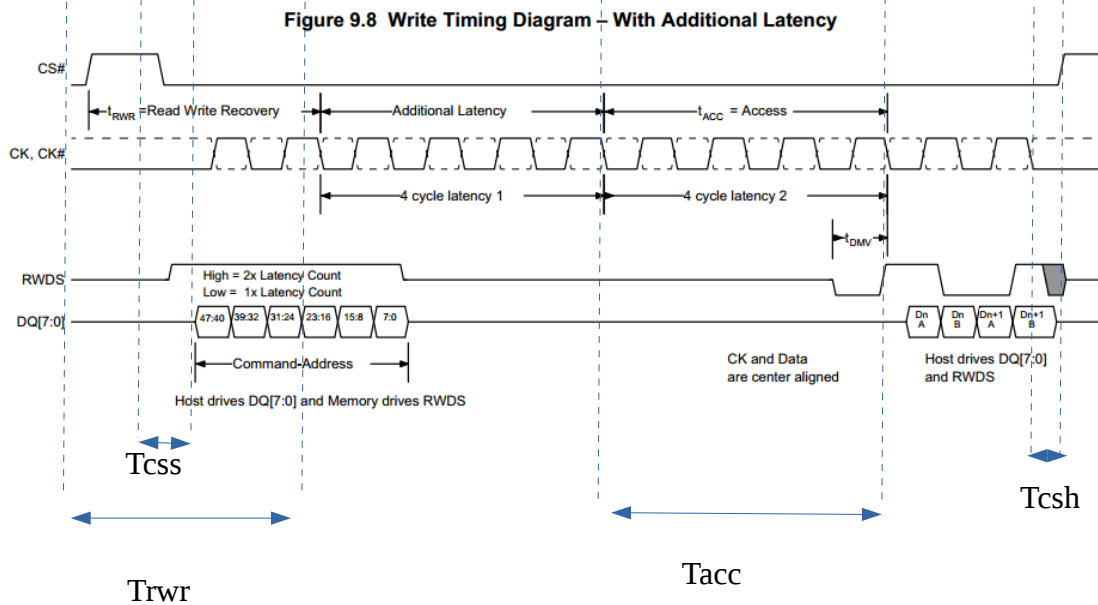
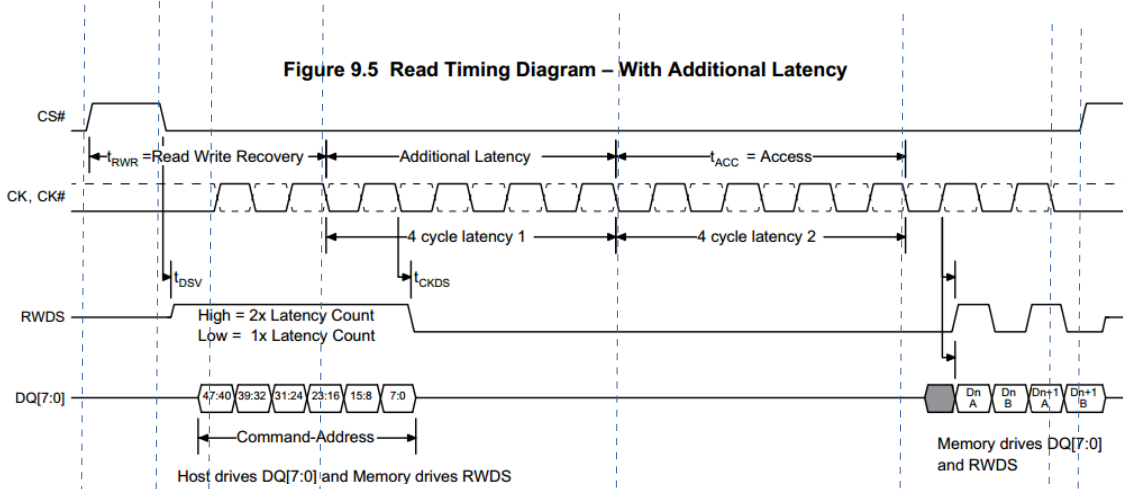
*\*\* Only used when controller is programmed in generic mode*

bit	Name	Description
3:0	T_CSS	Read/Write Chip Select Setup to next CK rising edge. This bit indicates CS# setup time from CS# assertion. 0x0 corresponds to 1 clock cycle, 0xF corresponds to 16 clock cycle.
7:4	T_CSH	Read/Write Chip Select Hold after CK falling edge. This bit indicates CS# hold time to from CS# de-assertion. 0x0 corresponds to 1 clock cycle, 0xF corresponds to 16 clock cycle.
15:8	T_RWR	Chip Select High Between Operations or Read/Write Recovery. This bit controls the CS# high time between successive read and write operations. Select the higher value between the read/write recovery time (Trwr) or Minimum Chip Select Hgh time (Tcph)
23:16	T_RD_ACC	Initial Latency Read Access Cycle. This bit indicates initial latency code for memory read access.
31:24	T_WR_ACC	Initial Latency Write Access Cycle. This bit indicates initial latency code for memory write access.

## HyperFlash protocol Timing

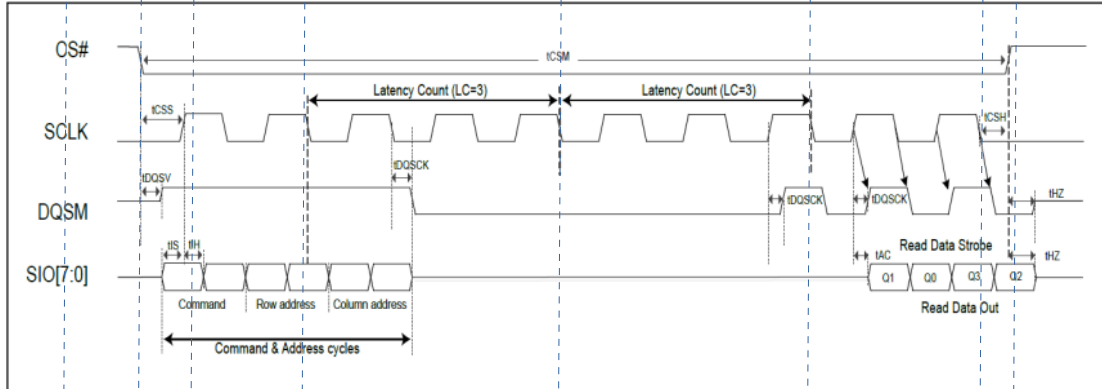


## HyperRAM protocol Timing

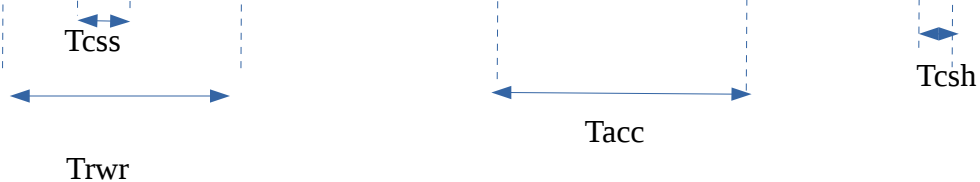
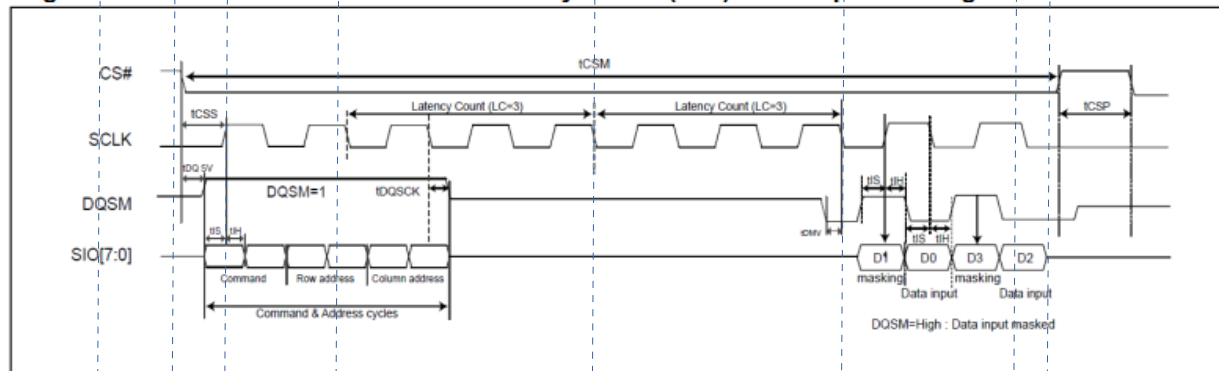


## OctaRAM protocol Timing

**Figure 5.2 Read Timing Diagram - Refresh Collision at Variable-Latency READ (2LC operation)**

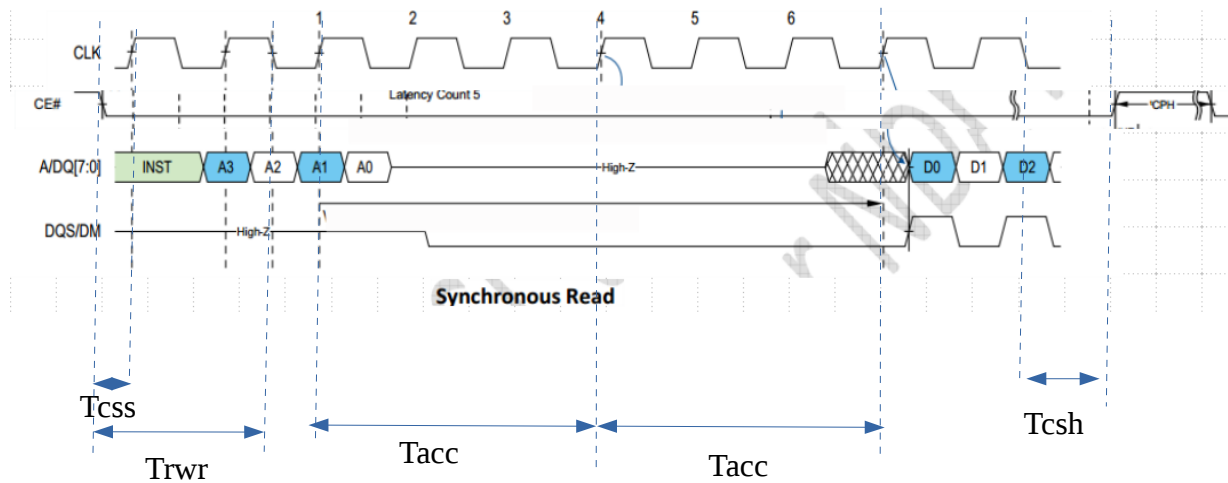
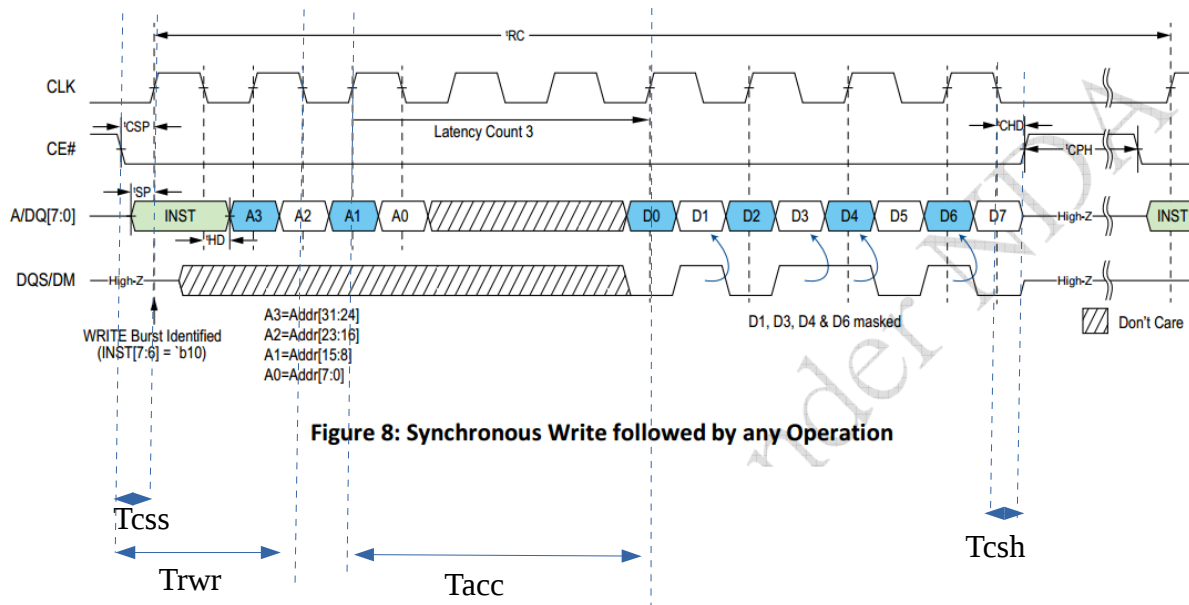


**Figure 5.6 Refresh Collision at Variable Latency WRITE (2LC) / Data Input Masking**

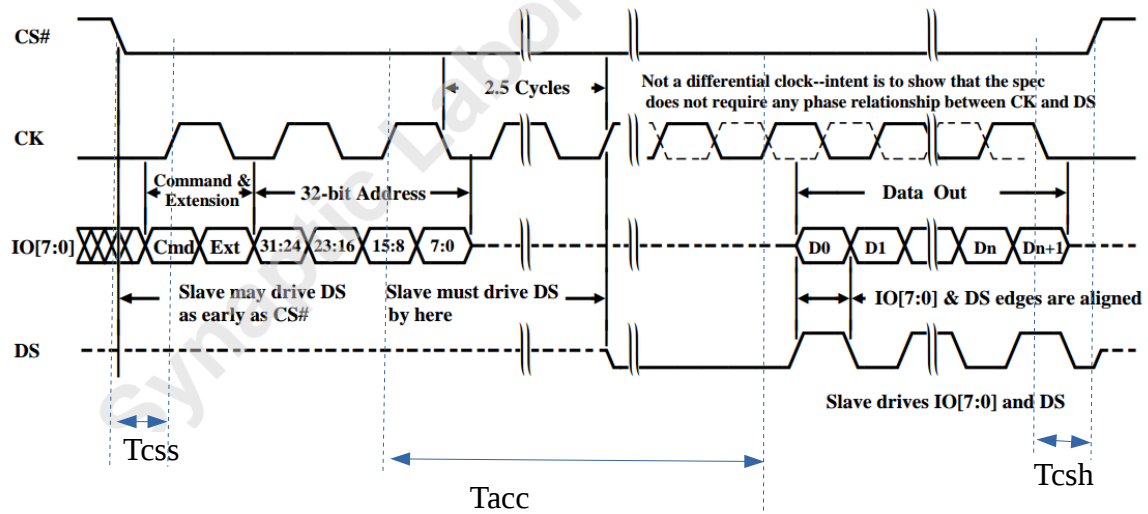




## XccelaRAM protocol Timing



## xSPI Profile 1.0 protocol Timing



## xSPI Profile 2.0 protocol Timing

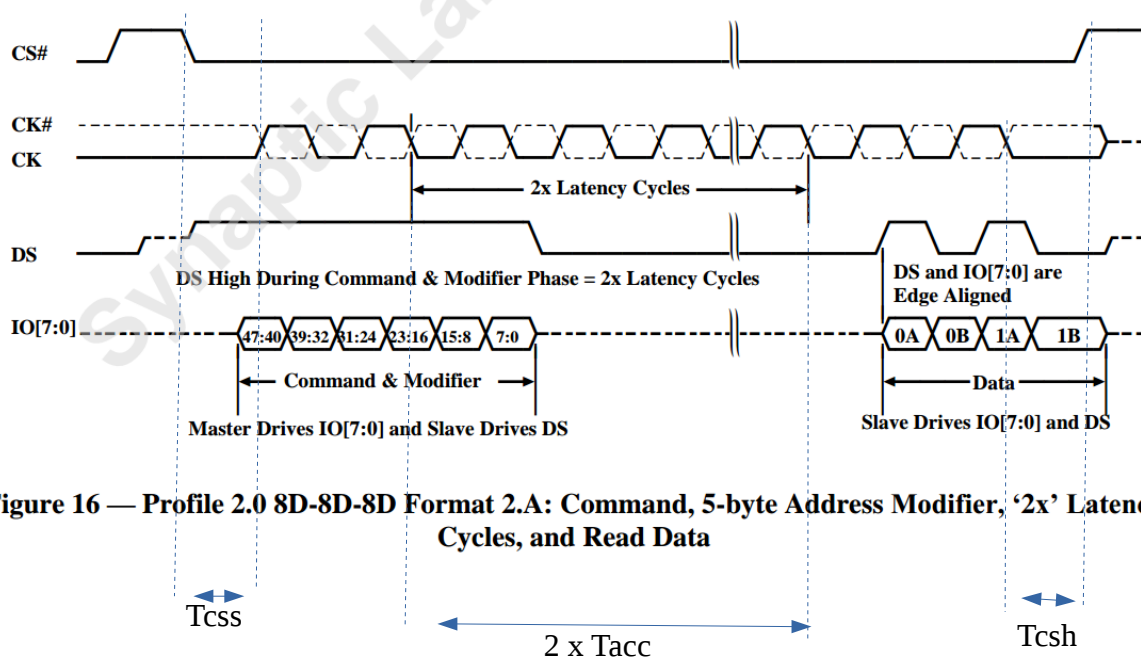


Figure 16 — Profile 2.0 8D-8D-8D Format 2.A: Command, 5-byte Address Modifier, '2x' Latency Cycles, and Read Data

### 6.2.12 xSPI-MC\_DEV\_x\_CMD Register *(to be further defined)*

<i>bit</i>	<i>Name</i>	<i>Description</i>
15:0	CMD_LINEAR_RD	Memory Command for Linear Read Operation
31:16	CMD_WRAP_RD	Memory Command for Wrap Read Operation
47:32	CMD_LINEAR_WR	Memory Command for Linear Write Operation
63:48	CMD_WRAP_WR	Memory Command for Wrap Write Operation

### **6.2.13 xSPI-MC\_DEV\_0\_MBA Register\*\*\***

**\*\* Only used when controller is programmed in generic mode**

This is a 64-bit register containing the start address in the AXI4 Memory Address Space for Device 0. The start address needs to be aligned to the size of the memory device 0.

The memory size of this region is specified in the xSPI-MC\_DEV\_0\_CFG Register.

### **6.2.14 xSPI-MC\_DEV\_1\_MBA Register\*\*\***

**\*\* Only used when controller is programmed in generic mode**

This is a 64-bit register containing the start address in the AXI4 Memory Address Space for Device 1. The start address needs to be aligned to the size of the memory device 1.

The memory size of this region is specified in the xSPI-MC\_DEV\_0\_CFG Register.

### 6.2.15 xSPI-MC\_CTRL Register (to be further defined)\*\*\*

*\*\* Only used when controller is programmed in generic mode*

Bit	value	Name	Description
4:0	0	Core_en	No Core Enabled
	1		qSPI Host Controller Enabled
	2		xSPI Host Controller Enabled
	3		xSPI Slave Controller Enabled
7:4		Reserved	
8		CORE_RST	Core Reset
9		DEV_RST	Device Reset
10		SETUP_RDY	Setup ready bit
11		LOOPBACK	Loopback mode (only when Target SPI core is present)
12		Dev 0 PD Active	Dev 0 Power Down Active
13		Dev 0 PD Wake	Dev 0 Power Down Wakeup
14		Dev 1 PD Active	Dev 1 Power Down Active
15		Dev 1 PD Wake	Dev 1 Power Down Wakeup
16		SPLIT_BURST	Possibility of splitting AXI burst for better management of the minimum Chip Select low device time (Tcsm)
8	other	Set/Clr	Set/Clear control

Bit 8 will set or reset the specified bit

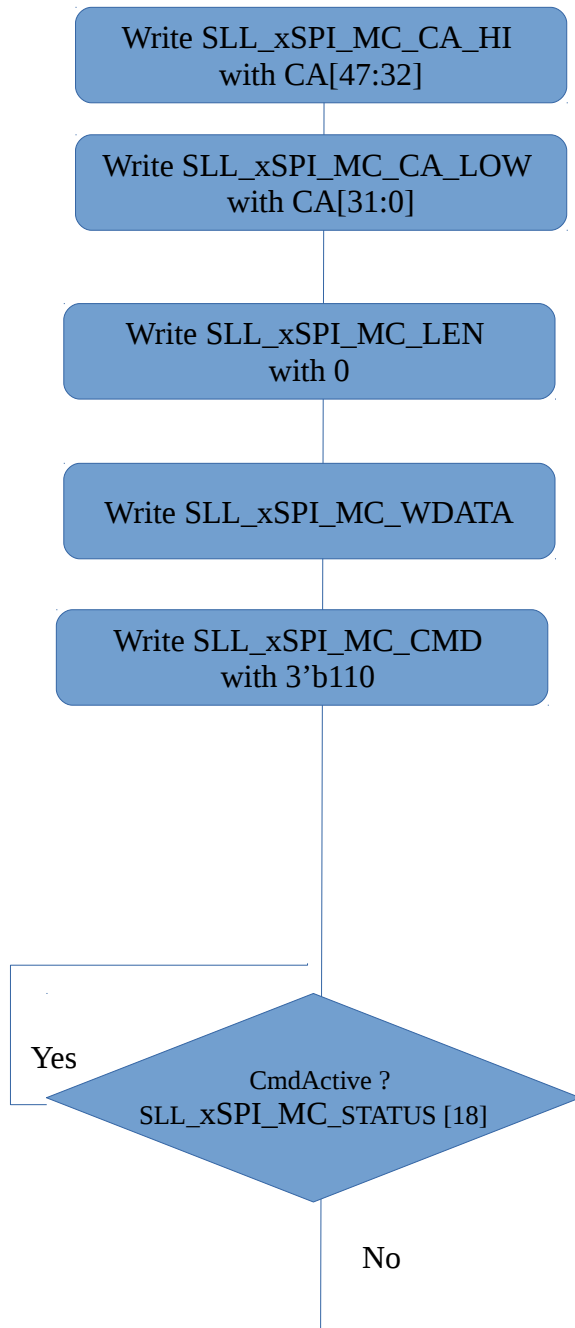
### 6.2.15 xSPI-MC\_DLL\_CTRL Register (to be further defined)

bit	Name	Description	Supported values
3:0	DLL_cntrl_val	Delay control Value for DLL module	0 : 15
7:2	Reserved		

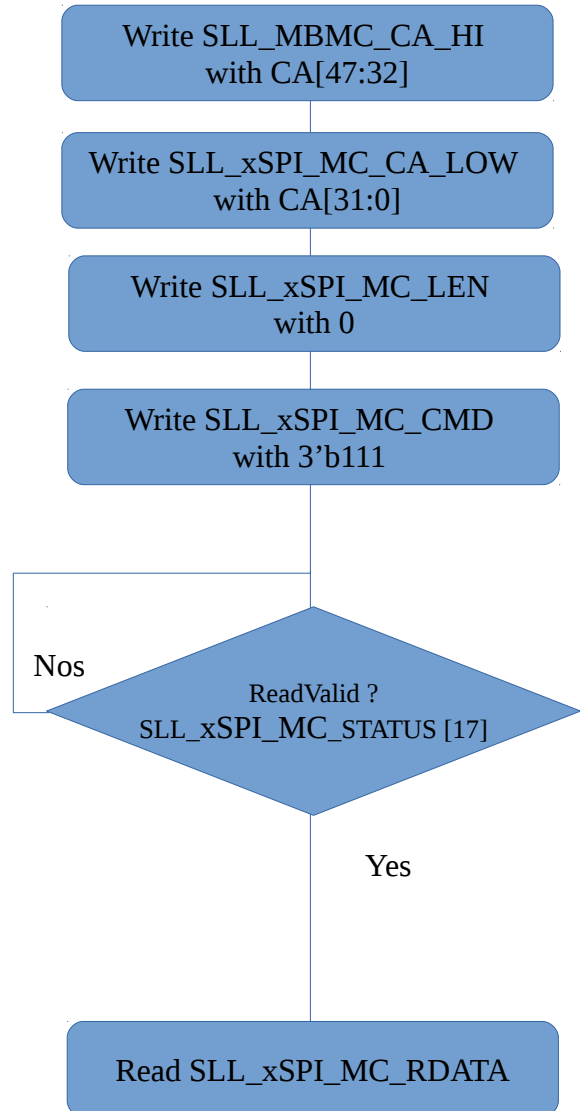
## 6.3 Accessing PsuedoRAM Registers

The follow flowchart shows how to use the control port interface to access the PseudoRAM registers. In this example, it is assumed that the PseudoRAM is connected to chipselect 1 (second chip select).

### Register Write Operation

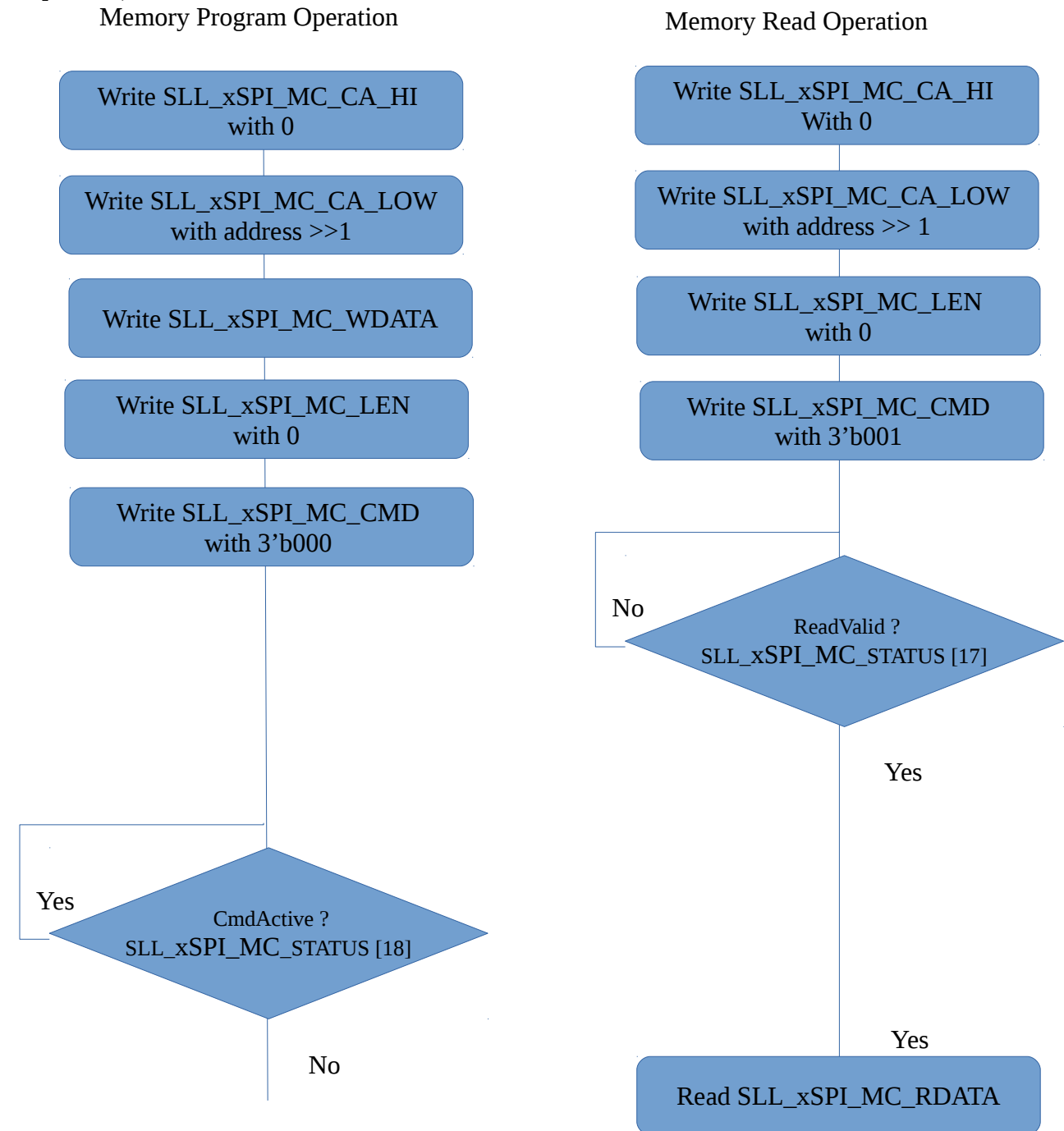


### Register Read Operation



## 6.4 Accessing HyperFlash Memory using the Control Port

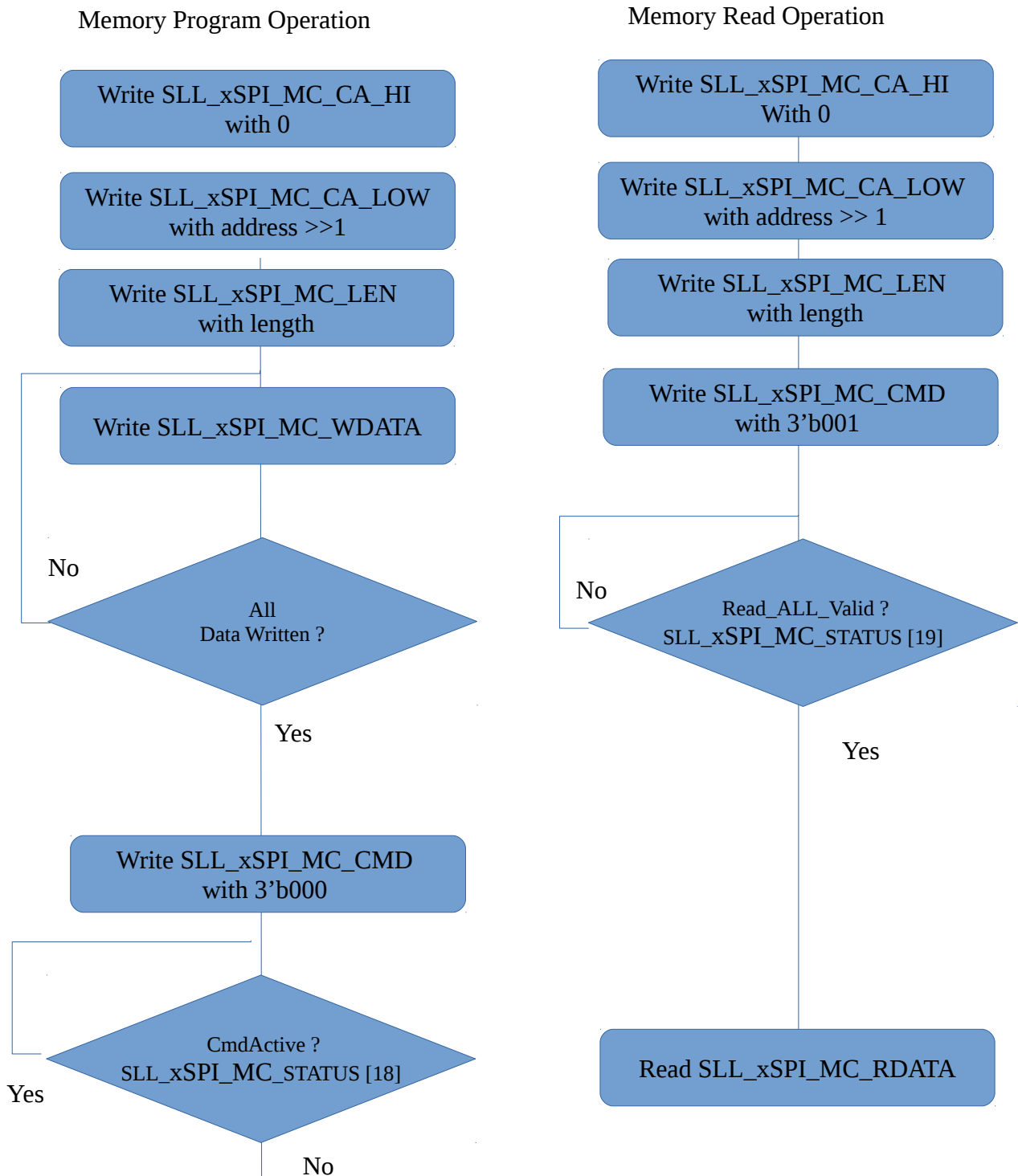
The follow flowchart shows how to use the control port interface to access the HyperFlash memory. In this example, it is assumed that the HyperFlash is connected to chipselect 0 (first chip select).





## 6.5 Accessing xSPI Flash Memory using the Control Port

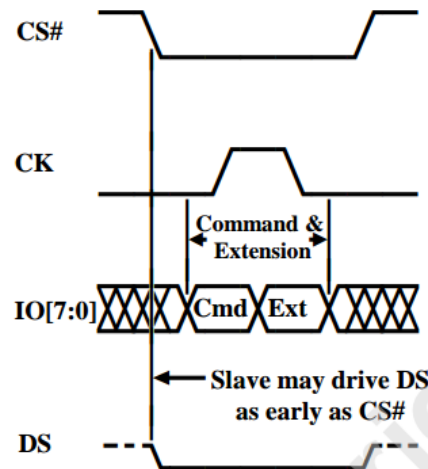
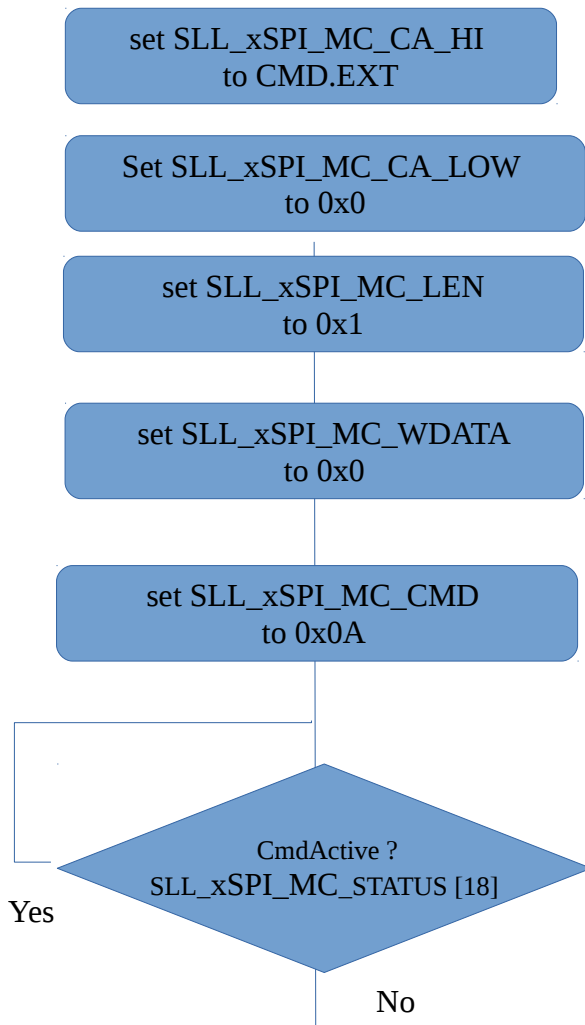
The follow flowchart shows how to use the control port interface to access the xSPI memory. In this example, it is assumed that the xSPI Flash is connected to chipselect 0 (first chip select).



## 6.6 xSPI Profile 1.0 Transaction : Format 1.A

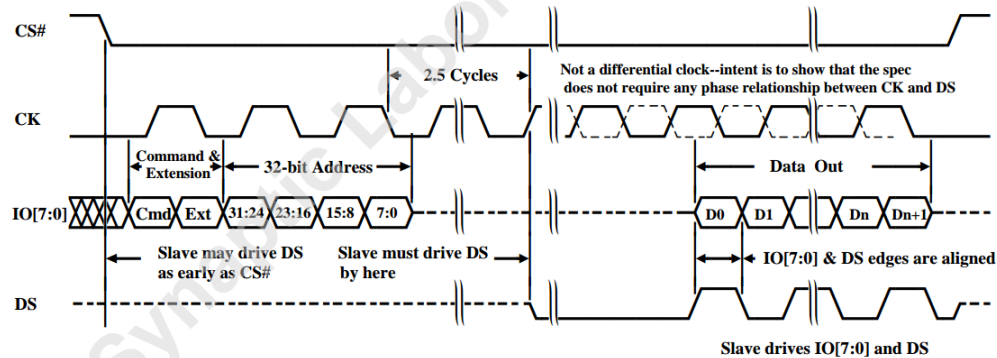
The follow flowchart shows how to use the control port interface to issue a transaction in format 1.A used with xSPI Profile 1.0 memory devices. In this example, it is assumed that the xSPI Flash is connected to chipselect 0 (first chip select).

Format 1.A Transaction

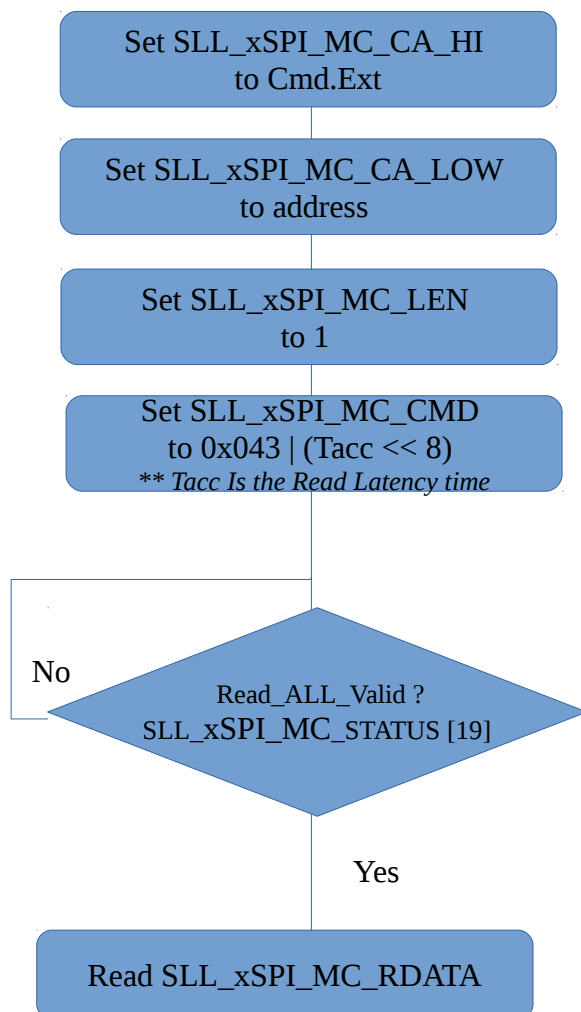


## 6.8 xSPI Profile 1.0 Transaction : Format 1.B

The follow flowchart shows how to use the control port interface to issue a transaction in format 1.B used with xSPI Profile 1.0 memory. In this example, it is assumed that the xSPI Flash is connected to chipselect 0 (first chip select) .



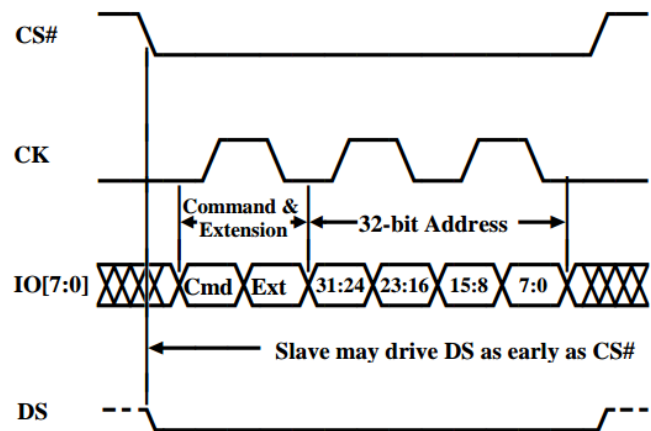
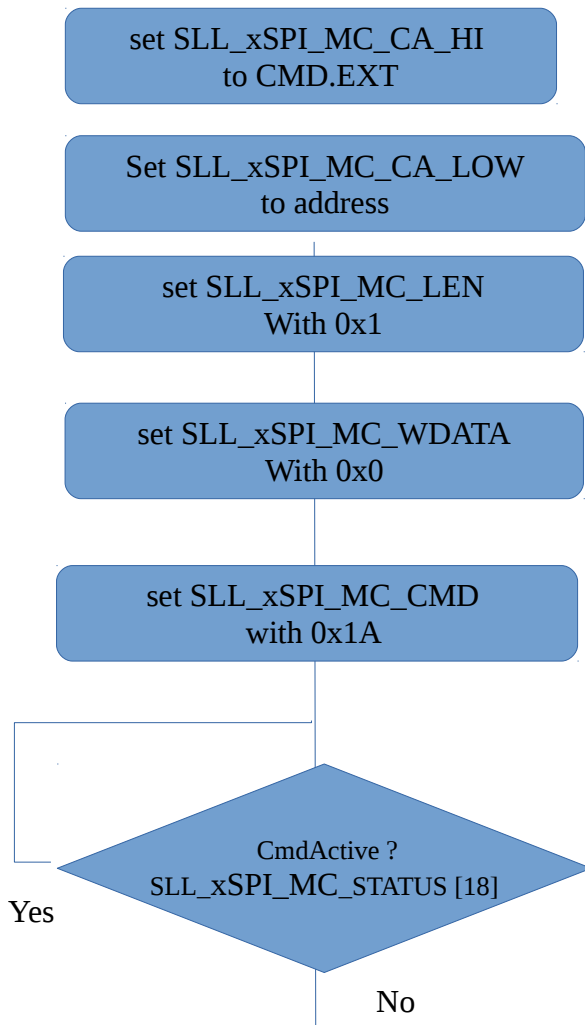
Format 1.B transaction Operation



## 6.9 xSPI Profile 1.0 Transaction : Format 1.C

The follow flowchart shows how to use the control port interface to issue a transaction in format 1.C used with xSPI Profile 1.0 memory. In this example, it is assumed that the xSPI Flash is connected to chipselect 0 (first chip select).

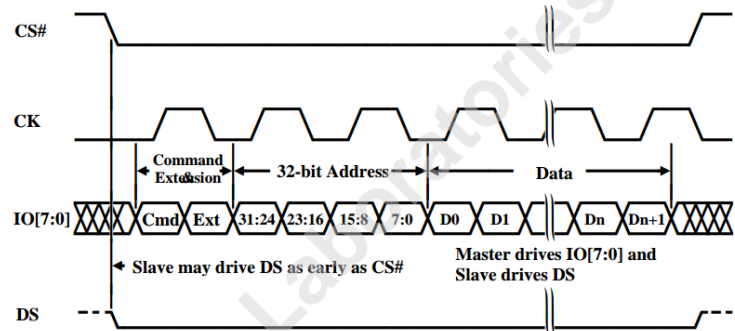
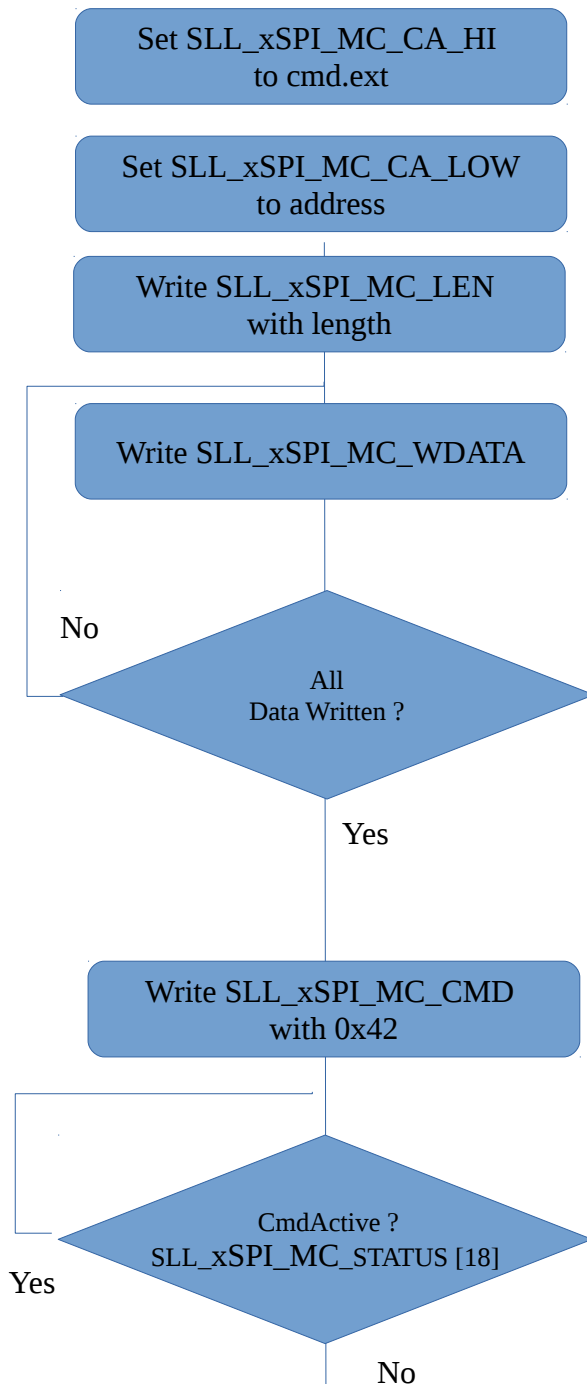
Format 1.C Transaction



## 6.10 xSPI Profile 1.0 Transaction : Format 1.D

The follow flowchart shows how to use the control port interface to issue a transaction in format 1.D used with xSPI Profile 1.0 memory. In this example, it is assumed that the xSPI Flash is connected to chipselect 0 (first chip select).

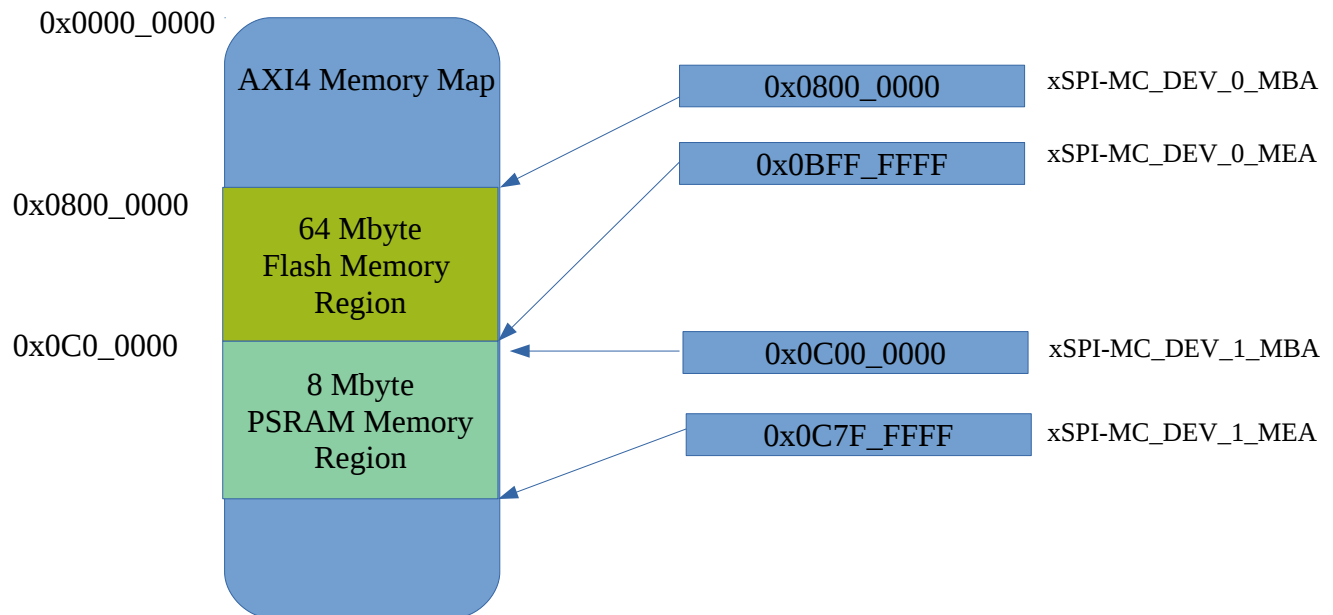
### Memory Program Operation



## 7.0 : AXI4 Memory Address Mapping \*\*\*

**\*\* Only used when controller is programmed in generic mode**

SLL xSPI-MC controller has a single Avalon-MM channel interface. For each memory device present on the system, the user needs to create a memory partition. This is done by programming the memory base address control registers available in the register area through the register control port.



The example below shows a memory partitioning example for a 64 Mbyte HyperFlash and an 8Mbyte HyperRam connected to the system. The first 64 Mbyte memory region is reserved for the HyperFlash memory. The second 8 Mbyte memory region is reserved for the HyperRam memory.

In this example, the base addresses for the HyperFlash and HyperRAM is calculated as follows :

- HyperFlash start address is : 0x0800\_0000
- HyperFlash end address is : 0x08FF\_FFFF (0x0800\_0000 + 0x03FF\_FFFF)
- HyperRAM start address is : 0x0C00\_0000

The memory base address control registers need to be updated as follows :

- Write xSPI-MC\_DEV\_0\_MBA with [0x0800\\_0000](#)
- Write xSPI-MC\_DEV\_1\_MBA with [0x0C00\\_0000](#)

This controller can't assert two Chip Select at the same time.

The way memory devices are accessed is as follows :

CS0# select condition:

- $\text{xSPI-MC\_DEV\_0\_MBA} \leq \text{AxADDR} < \text{xSPI-MC\_DEV\_0\_MBA} + \text{DEV0 Size}$
- Address for Main Controller =  $\text{AxADDR} - \text{xSPI-MC\_DEV\_0\_MBA}$

CS1# select condition:

- $\text{xSPI-MC\_DEV\_1\_MBA} \leq \text{AxADDR} < \text{xSPI-MC\_DEV\_1\_MBA} + \text{DEV01 Size}$
- Address for Main Controller =  $\text{AxADDR} - \text{xSPI-MC\_DEV\_1\_MBA}$

## 8.0 xSPI Controller Core Operation

The xSPI Controller Core is accessed through the

- AXI Read Channel bus
- AXI Write Channel bus
- Internal Memory access generated through APB Register Control Port.

Memory access over the Avalon-MM Read And Write Channels always have precedence over the Internal Memory access generated through Register Control Port. The Avalon-MM Read and Write Channels have the same access priority and a round robin arbiter controls which operation is taken.

- If there is a concurrent Read and Write request, the round robin arbiter controls which operation takes place.
- If there is no read request and there is some write requests, the write request passes through the arbiter.
- If there is no write request and there is some read requests, the read request passes through the arbiter.
- If there is an Internal Memory request generated through Register Control Port, the xSPI Controller will wait until the Avalon-MM Read and Write channel bus are free.



## 9.0 Core Reset

### 9.1 Global Asynchronous Reset

This section describes requirement of **i\_iavs0\_resetsn** signal on the Avalon-MM Data channel and Register Control interface.

When low, **i\_iavs0\_resetsn** signal will reset

- AXI Core logic
- Memory Controller logic
- Forces Registers in the APB region to their default value
- It will force the external xSPI RSTn signal

Reset Pulse width must be greater than > 300us

### 9.2 Software Core Reset *(to be defined)*

The Memory Controller Core can also be reset through software by setting the corresponding bit in the xSPI-MC\_MODE\_CTRL Register. In this case, only the memory controller core is reset. It is recommended that the Reset Pulse width is greater than > 300us.

### 9.3 Software Device Reset *(to be defined)*

The external memory devices can also be reset through software by setting the corresponding bit in the xSPI-MC\_MODE\_CTRL Register. When high, it will force the external xSPI RSTn signal to low. It is recommended that the Reset Pulse width is greater than > 300us

### 9.3 Device Sleep/Power Down *(to be defined)*

The external memory devices can be powered down or set to sleep mode to reduce power consumption. This is done by issuing memory specific register commands to the memory device. After issuing register commands, set the Sleep/PowerDown bit to 1.

To signal the end of sleep/power down, set the Sleep/PowerDown bit needs to be 0.

When this bit changes from high to low, it will force the external xSPI CSn signal to low for a period of 1us to wake up the devices.

Alternatively, a Software Device Reset command can be used to wake up the device.

## 10.0 Powering Up *(generic mode only)*

The process of powering up xSPI-MC is as follows :

Enable qSPI Host Controller  
(xSPI-MC CTRL bit 1 to 1)

Clear Device Reset  
(xSPI-MC CTRL bit 8 to 0)

Detect Flash presence using  
QSPI controller

Enable xSPI Host Controller  
(xSPI-MC CTRL bit 2 to 1)

Set Device 0 configuration register

Set Device 1 configuration register

Set SETUP\_RDY bit  
(xSPI-MC CTRL bit 10 to 1)

## HyperRAM 1.0 and HyperRAM 2.0 Device Configuration (*generic mode only*)

The process to configure a HyperRAM 1.0 and HyperRAM 2.0 device is as follows :

Set Memory Region using registers  
xSPI-MC\_DEV\_x\_MBA

Set Timing Control using register  
xSPI-MC\_DEV\_x\_TIM

Set Config Control using register  
xSPI-MC\_DEV\_x\_CFG

Set RW Commands using registers  
xSPI-MC\_DEV\_x\_CMD

## HyperFlash/SemperFlash with Hyperbus Interface Device Configuration (*generic mode only*)

The process to configure a HyperFlash and SemperFlash with Hyperbus Interface Device is as follows :

Set Memory Region using registers  
xSPI-MC\_DEV\_x\_MBA

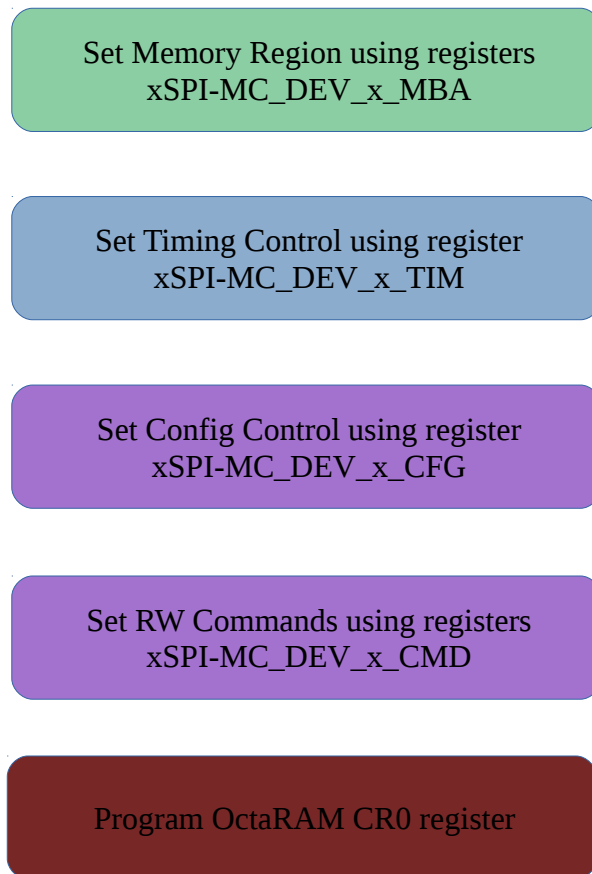
Set Timing Control using register  
xSPI-MC\_DEV\_x\_TIM

Set Config Control using register  
xSPI-MC\_DEV\_x\_CFG

Set RW Commands using registers  
xSPI-MC\_DEV\_x\_CMD

## OctaRAM Device Configuration (*generic mode only*)

The process to configure an OctaRAM memory device is as follows :



\*\* Please note the extra step needed for programming the OctaRAM CR0 register.

## OctaFlash Device Configuration (*generic mode only*)

The process to configure a OctaFlash memory device is as follows :

Set Memory Region using registers  
xSPI-MC\_DEV\_x\_MBA

Set Timing Control using register  
xSPI-MC\_DEV\_x\_TIM

Set Config Control using register  
xSPI-MC\_DEV\_x\_CFG

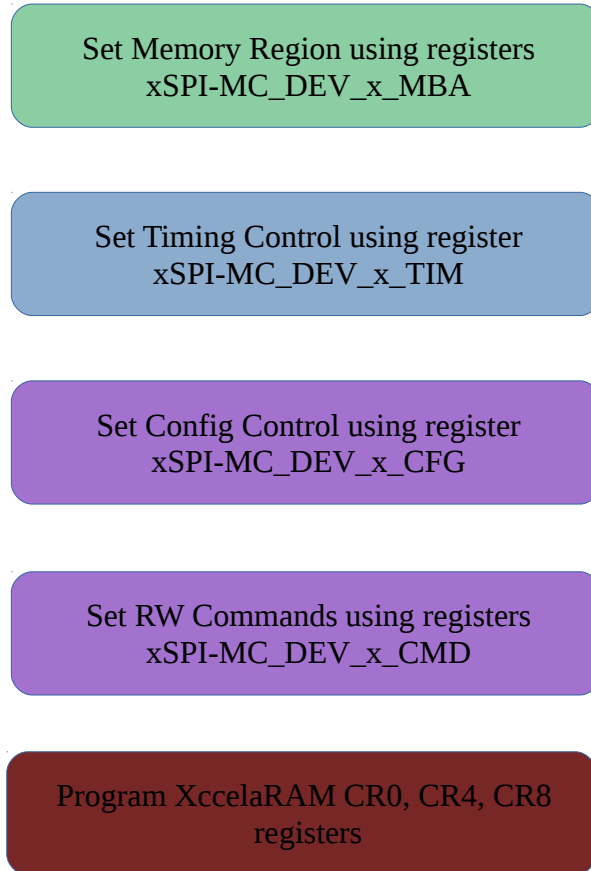
Set RW Commands using registers  
xSPI-MC\_DEV\_x\_CMD

Program OctaFlash Volatile Configuration  
register

\*\* Please note the extra step needed for programming the OctaFlash Volatile Configuration register.

## XccelaRAM Device Configuration (*generic mode only*)

The process to configure a XccelaRAM memory device is as follows :



\*\* Please note the extra step needed for programming the XccelaRAM CR0,CR4 and CR8 registers.

## **xSPI Profile 1.0 Flash Device Configuration (*generic mode only*)**

The process to configure the xSPI Profile 1.0 memory device is as follows :

Set Memory Region using registers  
xSPI-MC\_DEV\_x\_MBA

Set Timing Control using register  
xSPI-MC\_DEV\_x\_TIM

Set Config Control using register  
xSPI-MC\_DEV\_x\_CFG

Set RW Commands using registers  
xSPI-MC\_DEV\_x\_CMD

Program xSPI Profile 1.0  
Volatile Configuration registers

\*\* Please note the extra step needed for programming the xSPI Profile 1.0 volatile configuration registers.



## **xSPI Profile 1.0 Flash Device Configuration (*generic mode only*)**

The process to configure the xSPI Profile 1.0 memory device is as follows :

Set Memory Region using registers  
xSPI-MC\_DEV\_x\_MBA

Set Timing Control using register  
xSPI-MC\_DEV\_x\_TIM

Set Config Control using register  
xSPI-MC\_DEV\_x\_CFG

Set RW Commands using registers  
xSPI-MC\_DEV\_x\_CMD

Program Flash Volatile Configuration  
registers

**\*\* Please note the extra step needed for programming the appropriate Flash Volatile configuration registers.**

## 11.0 CycloneV Support

Targeting Higher clock speeds on Intel's Cyclone V (and Cyclone V with HPS) FPGA family.

On the Cyclone V, there are two different ways to interface with PSRAM/Flash enabled memories:

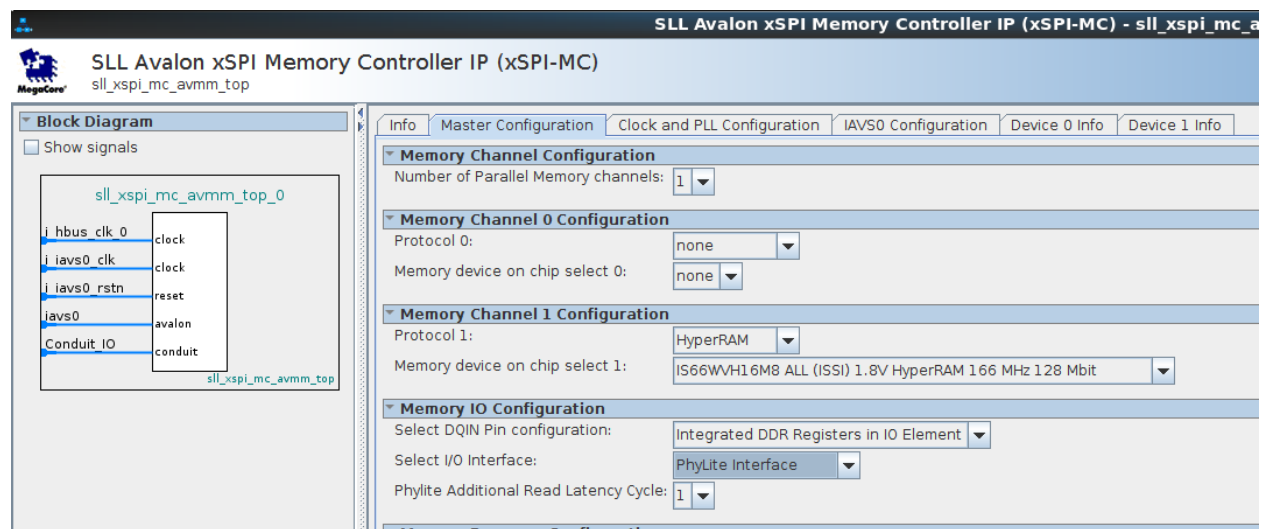
- Using a General Purpose I/O (GPIO) pin library

This technique works on all Intel FPGA device families. However, the top clock speed achievable for accessing DDRx memories using GPIO libraries varies depending on the choice of pins, and on the FPGA device family. The GPIO library is not well suited to driving SDRAM on Cyclone V families. Maximum speed is around 50 Mhz

- Using Intel's integrated DDR/IO registers.

These registers integrated in the I/O pads permit considerably higher clock speeds over the GPIO direct implementation on the same FPGA device. The new version of SLL MBMC IP (v3.3.x upward) now provide support this configuration.

## 11.1 SLL xSPI-MBMC Configuration for Integrated DDIO Support



To enable Integrated DDIO support on SLL xSPI-MBMC IP :

- Open SLL xSPI-MBMC IP
- Locate the **Master Configuration** tab
- Locate the **Memory I/O configuration** section
- Set the **Select DQIN Pin Configuration** to **Integrated DDR/IO DQ input**
- Set the **Select I/O interface** to **Phylite Interface**

### Additional QSF constraint

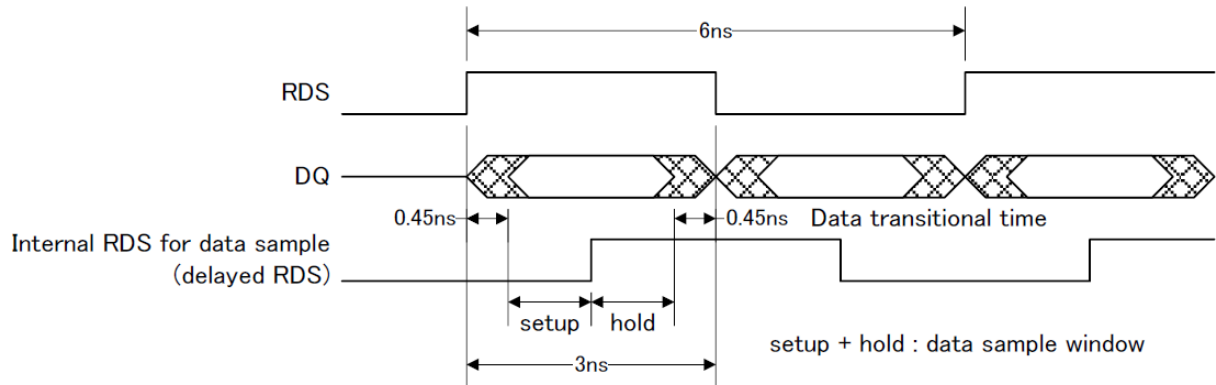
\*\* Very important : The user needs to add the following constraint in the top level QSF file.

```
set_global_assignment -name VERILOG_MACRO "CYCLONE_V_DDIO=1"
```

Kindly ask for a reference design in case static timing is not achieved.

## 12.0 Static Timing Consideration

### 12.1 Data Input Timing Constraint



The input read strobe (MB\_RWDS) signal is edge aligned to the data signal (MB\_DQ).

### 12.2 Timing Constraints

A timing constraint script (**sll\_ca\_xsapi\_mc\_top.sdc**) will be automatically generated by Synaptic Labs' xSPI-Multi-Bus memory Controller Qsys component. This script will be placed in Quartus Prime Synthesis directory and executed automatically during the Quartus Prime compilation process.

## 12.3 Pin Clustering

Please ensure all PSRAM/Flash channel pins are clustered physically close together in the programmable FPGA fabric. The FPGA board designer will need to balance the place-and-route requirements of the Multi-Bus memory Controller Logic against the ideal placement of pins from the board layout perspective to minimize skew across pins and to minimise pin-to-pin wire latency delay.

The signals received on the MB\_DQ pins are fed as an input into a single on-chip SRAM (e.g. M9K). The parallel capture of those signals is clocked by MB\_RWDS. Hence the location of the MB\_DQ/MB\_RWDS pins must be placed in a way to also ensure low wire latencies to that single on-chip SRAM. From the perspective of the PSRAM/Flash memory controller, try to ensure that the data being transported over all DQ signals arrive as close as possible, with as little skew, at the I/O pads”

The following output signals { MB\_CLK0, MB\_CLK0n, MB\_CLK1 and MB\_CLK1n }

- use an **altera\_gpio\_lite/altera\_ddio output** pad configured in DDR register mode.

The following bi-direction signals MB\_DQ and MB\_RWDS

- (Output mode) - use an **altera\_gpio\_lite/altera\_ddio output** pad with output enable control (oe) and configured in DDR register mode.
- (Input mode) - use an **altera\_gpio\_lite/altera\_ddio input** pad registered buffer mode

Please ensure these DDR signals are mapped to IO Elements with DDR capabilities.

## 12.4 Using 3V PSRAM and Flash devices

### 12.4.1 Selecting the correct operating frequency in Qsys

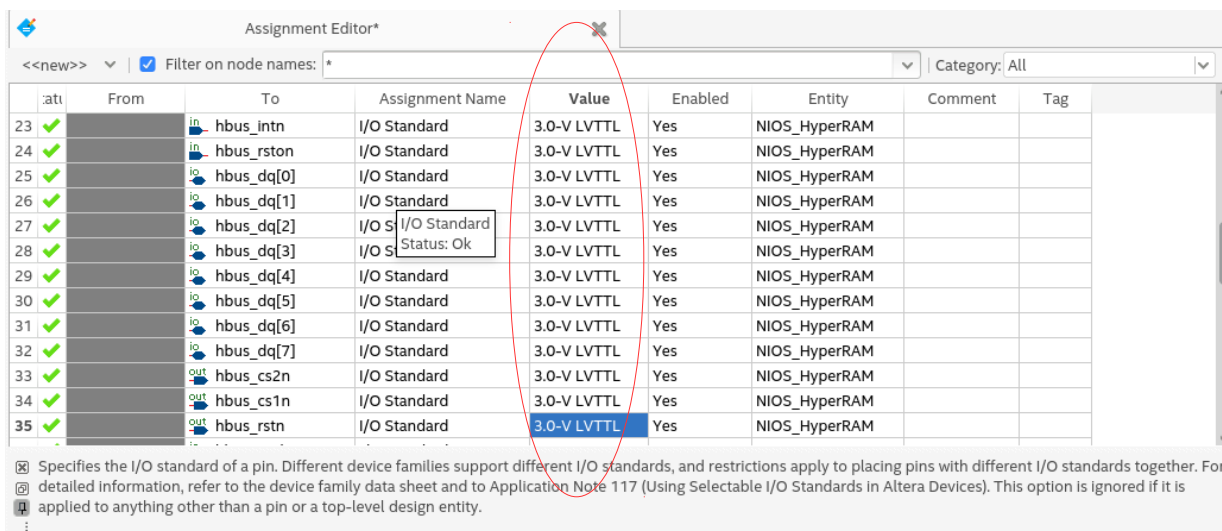
The 3V PSRAM/HyperRAM and HyperFlash memories supports a lower frequency than the 1.8V memory devices. Hence the designer needs to select a frequency that is supported by the memory device.

In Qsys, open SLL xSPI-MBMC IP component. Locate the Clock and PLL configuration Tab. Ensure that **The memory channel clock Frequency** is 100Mhz or less.

We suggest that the designer use the One Clock operation for both the Avalon Port and the PSRAM/Flash Channel Port.

### 12.4.2 Selecting the correct voltage in Quartus

The designer needs to set the HyperRAM/Flash voltage level signals to 3V. In Quartus, open the Assignment editor (Quartus → Assignment → Assignment Editor). Set the PSRAM/Flash memory signals voltage level to 3V.



Assignment Editor\*

<<new>> | Filter on node names: \* | Category: All

Pin	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
23	in	hbus_intn	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
24	in	hbus_rstn	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
25	io	hbus_dq[0]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
26	io	hbus_dq[1]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
27	io	hbus_dq[2]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
28	io	hbus_dq[3]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
29	io	hbus_dq[4]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
30	io	hbus_dq[5]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
31	io	hbus_dq[6]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
32	io	hbus_dq[7]	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
33	out	hbus_cs2n	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
34	out	hbus_cs1n	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		
35	out	hbus_rstn	I/O Standard	3.0-V LVTTTL	Yes	NIOS_HyperRAM		

☒ Specifies the I/O standard of a pin. Different device families support different I/O standards, and restrictions apply to placing pins with different I/O standards together. For detailed information, refer to the device family data sheet and to Application Note 117 (Using Selectable I/O Standards in Altera Devices). This option is ignored if it is applied to anything other than a pin or a top-level design entity.

### 12.4.3 Connecting the PSRAM signals to the FPGA I/O

The 3V PSRAM and Flash devices do NOT require a differential clock pair. MB\_CLK0n is left unconnected.

The figure below shows a typical top level connection for a 3V HyperRAM only design. Note that some signals are left unconnected since they are not used in the design.

```
-----  
-- Instantiation of main QSys system  
-----  
u0_main : component lab2  
  port map (  
    clk_clk      => c10_clk50m,  
    reset_reset_n => c10_resetrn,  
  
    sll_hyperbus_HB_RSTn  => hbus_rstn,  -- .HB_RSTn  
    sll_hyperbus_HB_CLK0  => hbus_clk0p,  -- .HB_CLK0  
    sll_hyperbus_HB_CLK0n => open,       -- .HB_CLK0n  
    sll_hyperbus_HB_CLK1  => open,       -- .HB_CLK1  
    sll_hyperbus_HB_CLK1n => open,       -- .HB_CLK1n  
    sll_hyperbus_HB_CS0n  => open,       -- .HB_CS0n  
    sll_hyperbus_HB_CS1n  => hbus_cs2n,  -- .HB_CS1n  
    sll_hyperbus_HB_WPn   => open,       -- .HB_WPn  
    sll_hyperbus_HB_RWDS  => hbus_rwds,  -- .HB_RWDS  
    sll_hyperbus_HB_dq    => hbus_dq,    -- .HB_dq  
    sll_hyperbus_HB_RST0n => '1',       -- .HB_RST0n  
    sll_hyperbus_HB_INTn  => '1',       -- .HB_INTn  
  
    pio_led_export => LEDs_debug,  
  );
```

#### Additional QSF constraint

Sometimes the user needs to manually add delay constraints in the top level QSF file for the Read Strobe signal (RWDS).

```
set_instance_assignment -name PAD_TO_CORE_DELAY 2 -to hbus_rwds
```

Kindly ask for a reference design in case static timing is not achieved.