MICROTRONIX AVALON MULTI-PORT SDRAM CONTROLLER

USER MANUAL V3.11

126-4056 Meadowbrook Drive
London, ON
Canada N5L 1E3
www.microtronix.com

Document Revision History

This user guide provides basic information about using the Microtronix ***Avalon Multi-port SDRAM Controller IP core***, (PN:6240-xx-xx). The following table shows the document revision history.

Date	Description
August 2008	Version 2.0 – Added Stratix III support, SOPC Builder setup Version 2.1 – Added Arria GX support
November 2008	Version 2.2 – Increased ports to 10
February 2009	Version 2.3 – Added TimeQuest SDC script support
March 2009	Version 2.4 – Added address/command clock option. Increased ports to 16. Re-added SDR support.
December 2009	Version 2.5 – Updated to new SOPC Builder GUI system. Added support for Hardcopy 2 & 3 devices without dedicated DDR hardware blocks.
February 2010	Version 2.6 – Added Cyclone IV support
July 2010	Version 2.7 – Added Arria II GX support
July 2011	Version 2.8 – Added Qsys support
January 2012	Version 2.9 – Added double-width option to ports. Added Random port buffer flush mechanism and control port.
April 2012	Version 3.0 – Added support for MDDR ¾ and ¼ drive. GUI modifications.
July 2013	Version 3.2 – Added support for Stratix IV devices.
Aug 2013	Version 3.3 – Added support for Cyclone V devices.
Mar 2014	Version 3.4 – Changed clocking recommendations and added TimeQuest constraint generation for single data rate sdram.
August 2014	Version 3.5 – Added support for Quartus version 14. Made the DQ delay chain length settable in the GUI.
December 2015	Version 3.6 – Added SDR memory support for MAX 10 devices.
January 2017	Version 3.7 – Add simulation library support for Cyclone V devices.
January 2017	Version 3.8 – Added support for Quartus 16.0, 16.1. Update Cyclone V in the performance table.
May 2017	Version 3.9 – Added support for Cyclone 10 LP devices
July 2017	Version 3.10 – Added support for DDR, DDR2, MDDR for MAX 10 devices

Nov 2017	Correct Table 3 DDR Fmax column. The numbers for Cyclone 10 LP devices were in reverse order.

How to Contact Microtronix

E-mail

Sales Information: sales@microtronix.com

Support Information: support@microtronix.com

Website

General Website: http://www.microtronix.com

FTP Upload Site: http://microtronix.leapfile.com

Phone Numbers

General: (001) 519-690-0091

Fax: (001) 519-690-0092

Typographic Conventions

Path/Filename	A path/filename
`[SOPC Builder]$ <cmd>`	A command that should be run from within the Cygwin Environment.
`Code`	Sample code.
↵	Indicates that there is no break between the current line and the next line.

Table of Contents

Features

- Support for Single-Data-Rate (SDR), Double-Data-Rate (DDR & DDR2) and Mobile DDR (mDDR) SDRAM memory devices in all supported FPGA families.
- Advanced Performance Architecture
    - Supports Avalon burst transfers
    - Up to 16 Avalon bus port interfaces
    - User configurable FIFO cache for each system port
    - Operates with memory clock independent of system clock
- Qsys support
- Support for Altera OpenCore Plus evaluation
- Reference Designs
- TimeQuest Timing Analyzer support
- Supported devices:
    - Cyclone II, III, IV E, IV GX, V, 10 LP
    - Stratix II, II GX, III, IV E, IV GX & IV GT
    - Arria GX, Arria II GX
    - MAX 10

Introduction

The Microtronix ***Avalon Multi-port SDRAM Memory Controller IP-Core*** (6240-XX-XX) provides a complete, easy-to-use solution to interface the Avalon bus with a wide variety of SDRAM memory devices. Both single data rate (SDR) and double data rate (DDR, DDR2 and Mobile DDR) are supported.

The memory controller is optimized for Altera Stratix and Cyclone families of programmable logic devices. The SDRAM Memory Controller is Altera SOPC Builder and Qsys ready and integrates easily into any SOPC Builder or Qsys generated system. It is designed to optimize the performance of Avalon bus based streaming data systems.

The SDRAM Memory Controller handles all memory tasks, including refresh and device initialization cycles. The IP Core is designed to operate asynchronous to the system clock. Clocking the IP-Core at the same frequency as the SDRAM memory maximizes system performance.

The DDR Memory Controller IP uses a DQS source-synchronous design architecture to capture the high-speed double-data rate data from the memory device independent of the memory round-trip. This technology simplifies memory interface design by removing the need for extra resynchronization clocks and maximizes the performance of DDR/DDR2 memory system.

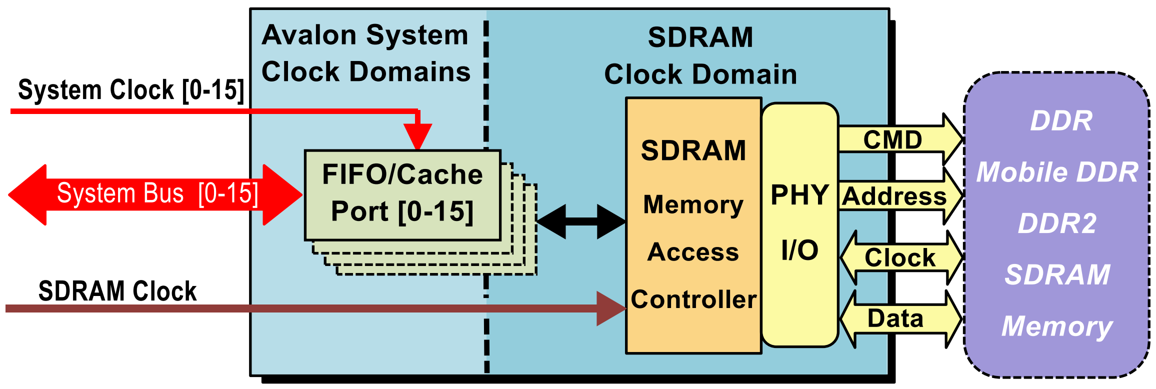


Figure 1: Block diagram of Avalon Multi-port SDRAM IP Core

Local Ports

The SDRAM memory controller has up to sixteen Avalon local ports. Each port has an internal FIFO, which acts as a bridge between the local interface and the SDRAM memory. Each port can be configured as an Avalon Random Slave, or an Avalon Burst Slave.

NOTE: The FIFOs of different ports do not maintain data coherency between each other.

The data width of the Avalon ports depends on the memory architecture used. When configured for SDR memory, the default Avalon width is the same as the memory width. This can be doubled to twice the memory width on a per-port basis by selecting the “Double the Avalon data width” option. When the controller is configured for DDR, Mobile DDR or DDR2 memory, the default Avalon width is twice the memory width. This can be doubled to four times the memory width on a per-port basis by selecting the “Double the Avalon data width” option.

The memory controller also has an optional command port. This port contains a register for flushing the random port caches, if that option is enabled.

Avalon Random Slave

The Avalon Random Slave is used for processor access. Each random port contains two buffers, one for the read direction and one for the write direction.

The internal FIFO is implemented as a direct mapped cache memory. On a cache hit, data is available immediately with one cycle of read latency. On a cache miss, the SDRAM memory controller inserts wait states and synchronizes the cache with the SDRAM memory. The cache maintains full data coherency, meaning a read cycle to the last written address produces the last written data. The port achieves this by writing the write buffer contents to memory before filling the read buffer if the write buffer contains modified data that would end up in the read buffer on a read.

It is important to note that with the default settings, write data is only written to memory on a read, or a write to an address that is not in the cache. Depending on the data access patterns and the use of multiple ports, it is possible for write data to remain in the cache of a particular port for an indefinite amount of time. Each random port can be configured to enable cache flushing through the command port. See the section on the command port for more information.

The size of the cache can be set in the SOPC Builder GUI. A larger cache size increases the probability of a cache hit, but also increases the waitrequest assert time.

Figure 2 shows a Random read transfer. The cache size is set to eight words and it holds addresses A0 through A7. Address A8 results in a cache miss and the waitrequest is asserted.

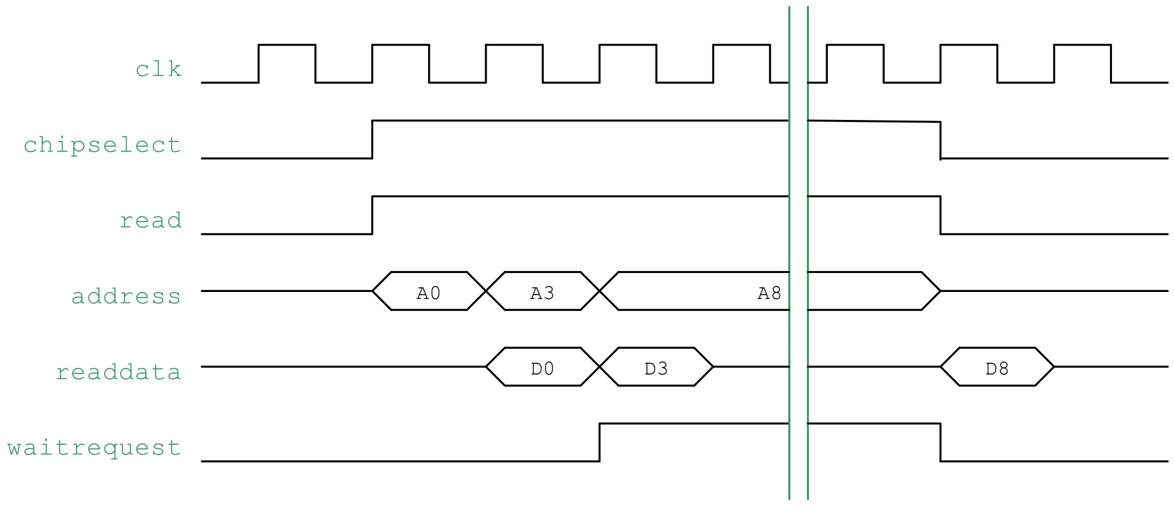


Figure 2: Random Read Transfer

Avalon Burst Slave

The Avalon Burst Slave maximizes the data throughput. It handles a transfer as a unit instead of multiple single transfers. The Burst Slave supports multiple posted write and read transfers.

Figure 3 shows a read transfer. The address and the burstcount are latched at the beginning of the transfer. As soon as the first read data is available, the datavalid is asserted. During a transfer the burst slave can de-assert the datavalid signal when no valid read data is available.

To get the most optimal read performance, the Avalon Master should post read transfers as soon as possible. It enables the Burst Slave to fetch the read data from the SDRAM memory. For more information on burst transfers see the Avalon Interface Specification document published by Altera.

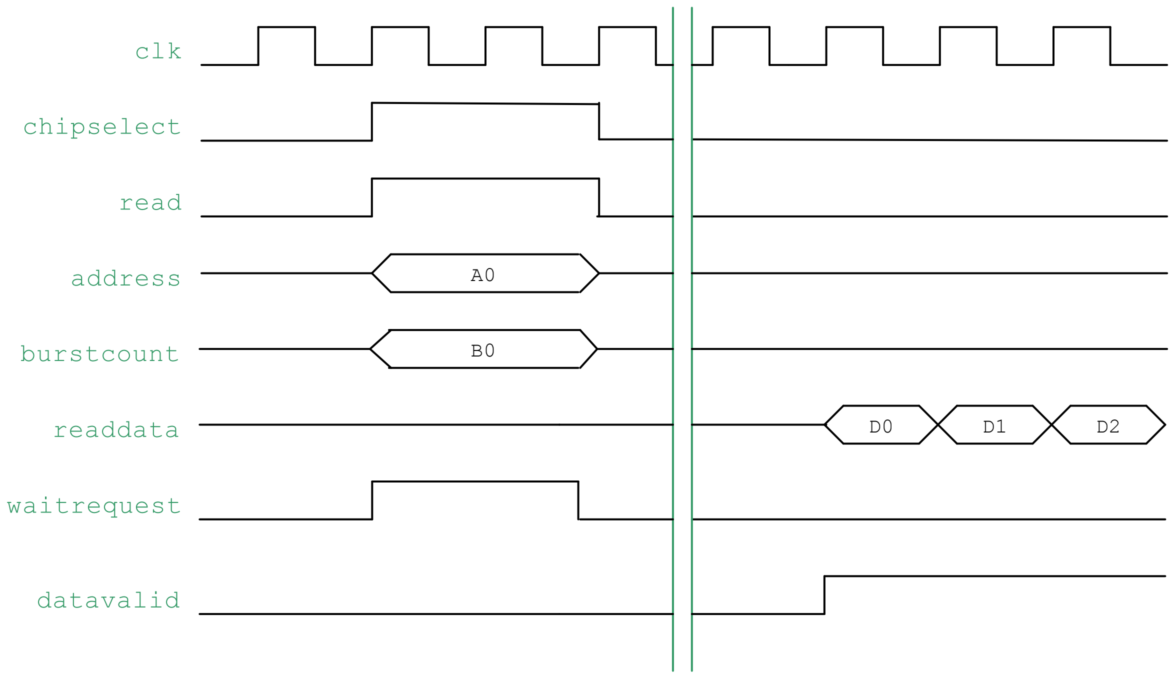


Figure 3: Burst Read Transfer

For a write burst transfer the Avalon master can interrupt the transfers by driving the write signal low. The original Avalon burst specification doesn't provide a way to interrupt a read burst transfer for the master.

When selecting the tick box 'Enable Read Data Flow Control' in the SOPC Builder an extra (non-Avalon) called dataenable is created. This signal allows the master to stall an active read transfer.

Figure 4 shows an interrupted burst read transfers. During a read transfer the Avalon master drives the dataenable signal low to indicate it wants to temporarily stall the read transfer. The burst port responds by driving the datavalid signal low. When the master re-asserts the dataenable signal, the burst port enables the datavalid and starts transferring read data.

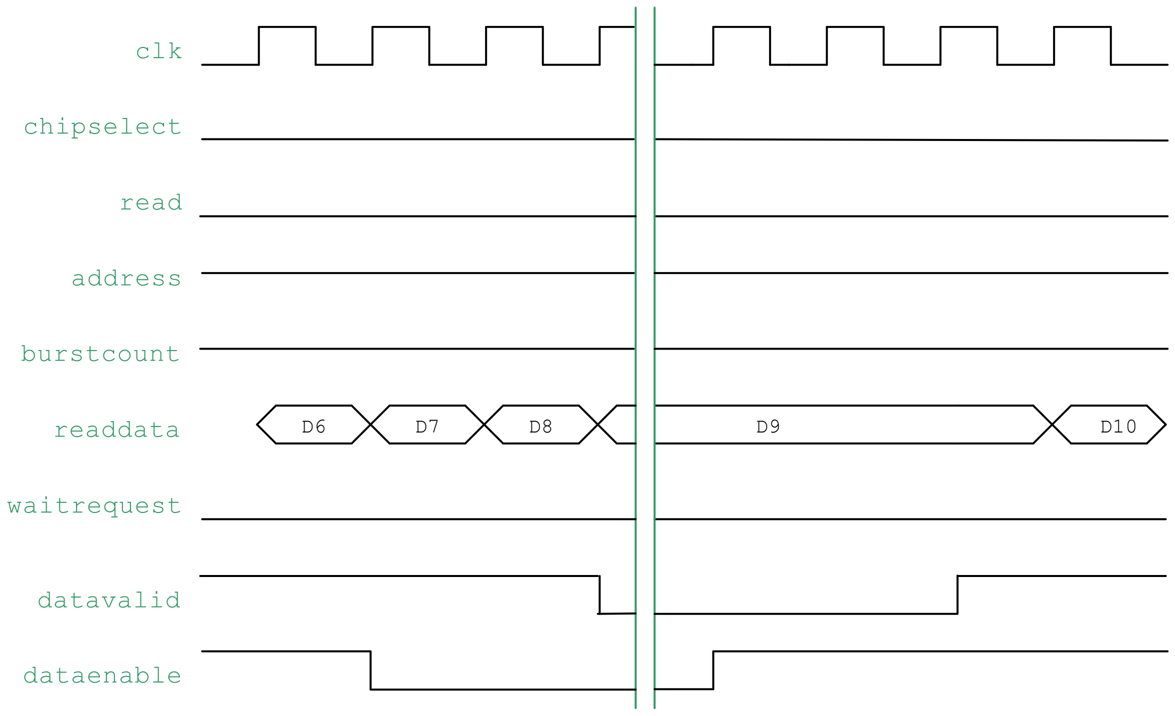


Figure 4: Interrupted Burst Read Transfer

Avalon Control Port

The Avalon Control Port is used to (by Nios software) control some of the memory controller's internal functions. This control port is enabled if any of the ports configured in Random mode have the "Enable the buffer flush interface" option selected.

The Control Port consists of a single 32-bit register, with each bit corresponding to an Avalon port of the memory controller.

Table 1: Control Port Registers

Register	Bit Assignments																
	31-16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0 – Port Flush	Unused	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

When a '1' is written to a bit in the Port Flush Register, the control port signals to the corresponding local port to flush any unwritten data to memory. Once the port has been signaled, the bit will be cleared. If the port is configured in burst mode, or the "Enable the buffer flush interface" option of the port is not selected, the flush signal is ignored.

Port Arbitration

Ports use a round-robin arbitration scheme to acquire access to the SDRAM memory. Ports generate arbitrator request for access to the SDRAM based on the half-empty or half-full status of their port FIFO.

1. When configured as an Avalon Random port, it will transfer the number of words required to fill or empty the FIFO buffer up to the maximum depth of the buffer. Control is then passed to the next port requesting service.
2. When configured as an Avalon Burst port, it will transfer the number of words required to fill or empty the FIFO buffer up to the maximum burst length specified for the port. Upon completion, control is passed to the next port requesting service.

When a port is active, and another port requests access to the memory the first port can only retain access to the SDRAM up to the maximum number of words specified for the FIFO depth or the burst length. By this means, each port is assured access to the SDRAM memory system.

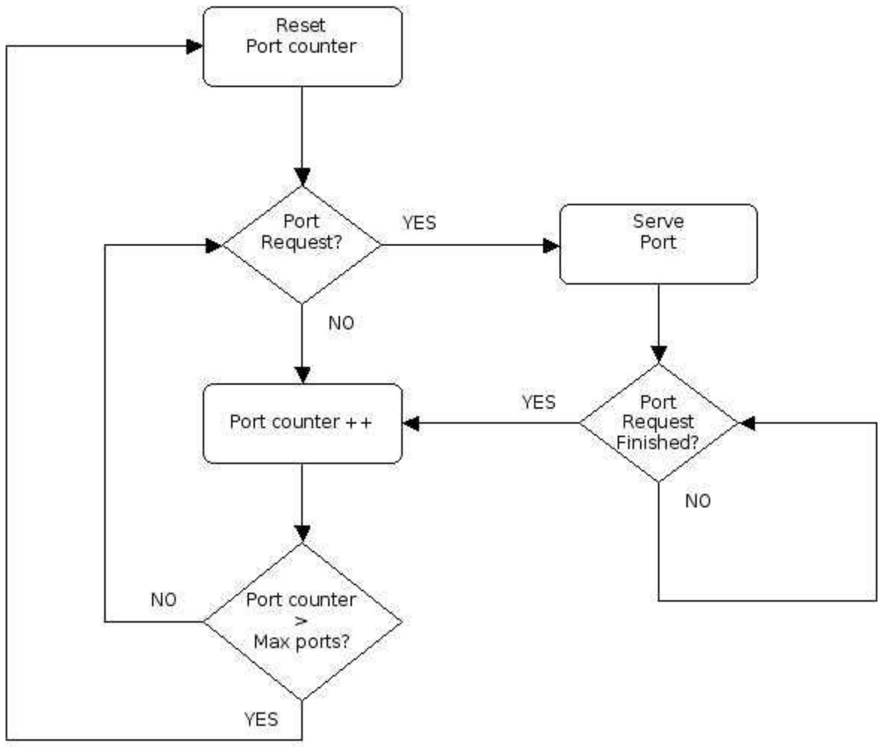


Figure 5: SDRAM port arbitration flow chart

Design Flow

The following steps describe how to integrate the SDRAM controller in a Qsys or SOPC Builder system.

- Create or open a Quartus project.
- Launch Qsys or SOPC Builder from the Quartus Tools menu.
- Build your system by adding components from the Component Library list.
- Choose the Multi-port SDRAM Controller from Library > Microtronix, then click Add.

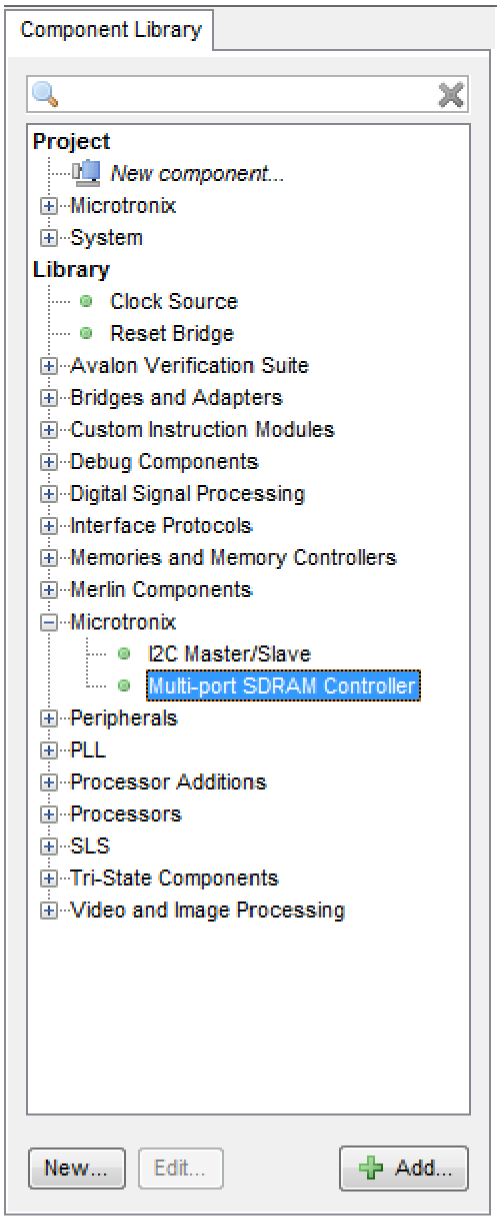


Figure 6: Qsys Component Library

Avalon Ports Tab

The Port tab is used to configure the Avalon slave interfaces.

- Click on the first Port tab.
- Ports can be configured as Avalon-MM Random or Avalon-MM Burst slaves, or they can be disabled.

It should be noted that no memory protection or management schemes are provided. If needed, a master peripheral connected to the local Avalon port must take care that data is only written in its own address space and no data is overwritten in the other address spaces.

CONFIGURING PORTS FOR A NIOS II PROCESSOR

To maximize Nios II processor performance, it is recommended to connect the Nios II Instruction and Data Master ports to separate ports on the SDRAM Memory Controller. In this configuration, connect the Instruction Master to a Random port. Do not enable bursts on the Instruction Master as the Microtronix SDRAM controller does not support this configuration. For the Nios II Data Master port, if bursts are enabled, the recommended port type is Burst. Otherwise, the Random type is recommended.

If both the Instruction Master and Data master are to be connected to the same port on the SDRAM controller, the Random port type is recommended.

Avalon Random Port Settings

BUFFER SIZE

This specifies the size of the port's internal buffer in Avalon words. For the best Nios II processor performance, the buffer size should be limited to 8 or less words.

DOUBLE THE AVALON DATA WIDTH

Enabling this option will set the Avalon data width to twice the standard data width. The standard data width for SDR is equal to the memory width. The standard data width for DDR, Mobile DDR and DDR2 is twice the memory width. Best performance is achieved when the port's data width matches the width of the Avalon masters connected to it, where possible.

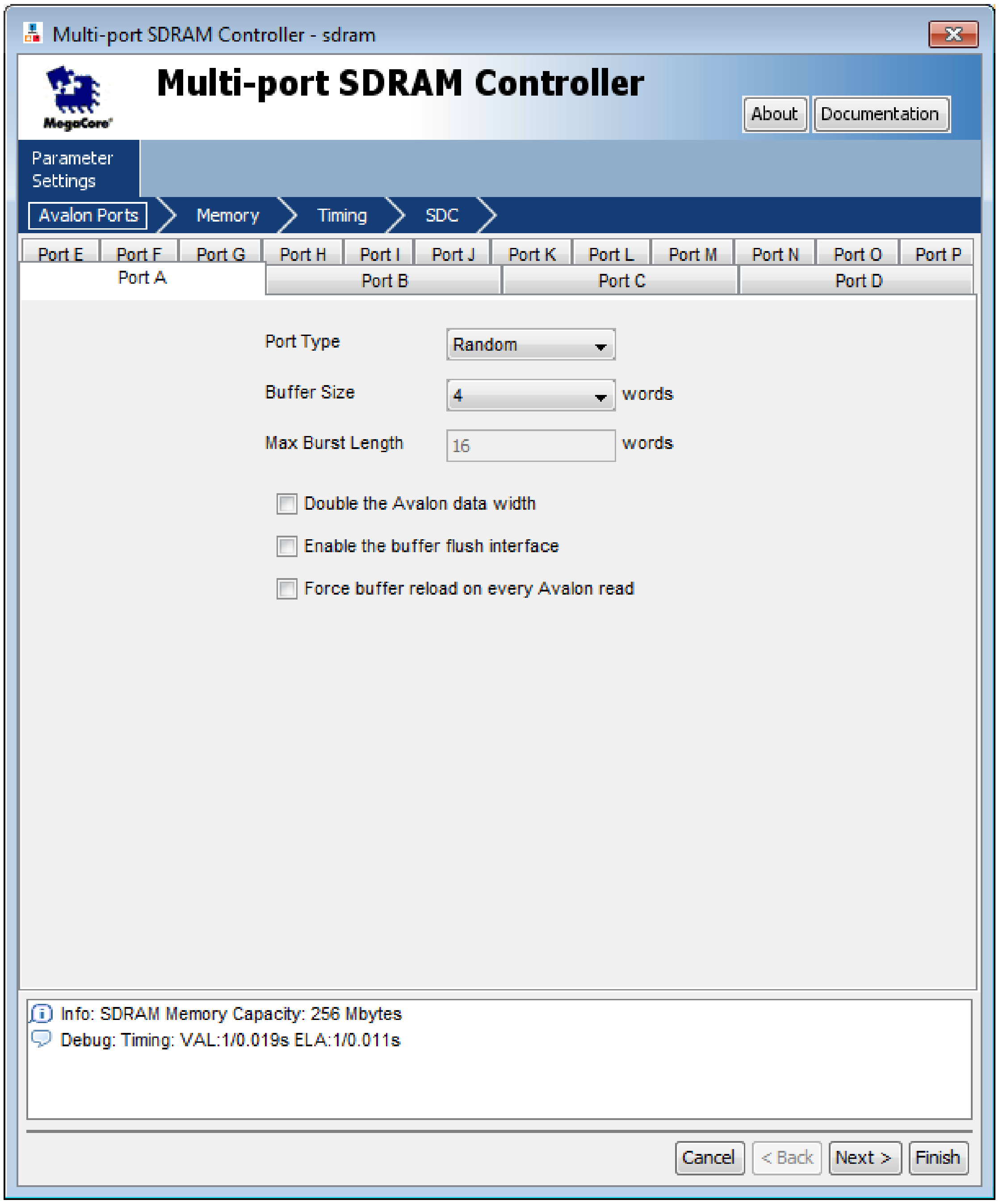
ENABLE THE BUFFER FLUSH INTERFACE

Adds the Avalon control slave to the memory controller and allows it to signal this port to flush any unwritten data from its buffer to memory.

FORCE BUFFER RELOAD ON EVERY AVALON READ

Enabling this option will disable any read caching. Whenever an Avalon master reads from this port, it will fill its internal buffer from memory.

NOTE: This setting will significantly decrease memory performance



Port Type: Random

Buffer Size: 4 words

Max Burst Length: 16 words

☐ Double the Avalon data width

☐ Enable the buffer flush interface

☐ Force buffer reload on every Avalon read

```
Info: SDRAM Memory Capacity: 256 Mbytes
Debug: Timing: VAL:1/0.019s ELA:1/0.011s
```

Figure 7: Avalon Ports Tab – Avalon Random Settings

Avalon Burst Port Settings

BUFFER SIZE

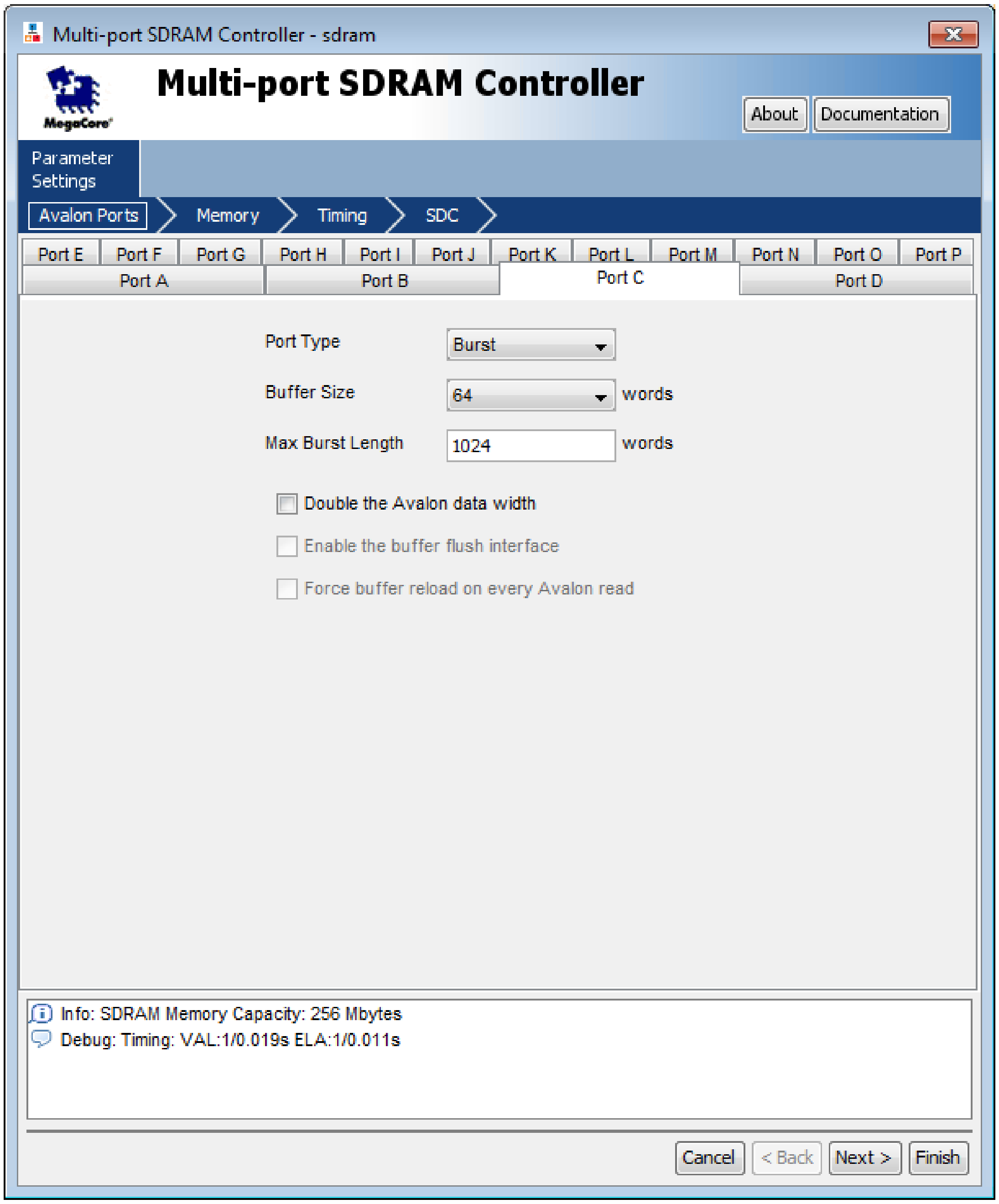
This specifies the size of the port's internal FIFOs in Avalon words. There are two FIFOs in a burst port, one for reads and one for writes.

MAX BURST LENGTH

This specifies the maximum length of Avalon burst transfers to this port. This value should be at least as large as the maximum burst size of any connected Avalon masters.

DOUBLE THE AVALON DATA WIDTH

Enabling this option will set the Avalon data width to twice the standard data width. The standard data width for SDR is equal to the memory width. The standard data width for DDR, Mobile DDR and DDR2 is twice the memory width. Best performance is achieved when the port's data width matches the width of the Avalon masters connected to it, where possible.



Port Type: Burst

Buffer Size: 64 words

Max Burst Length: 1024 words

☐ Double the Avalon data width

☐ Enable the buffer flush interface

☐ Force buffer reload on every Avalon read

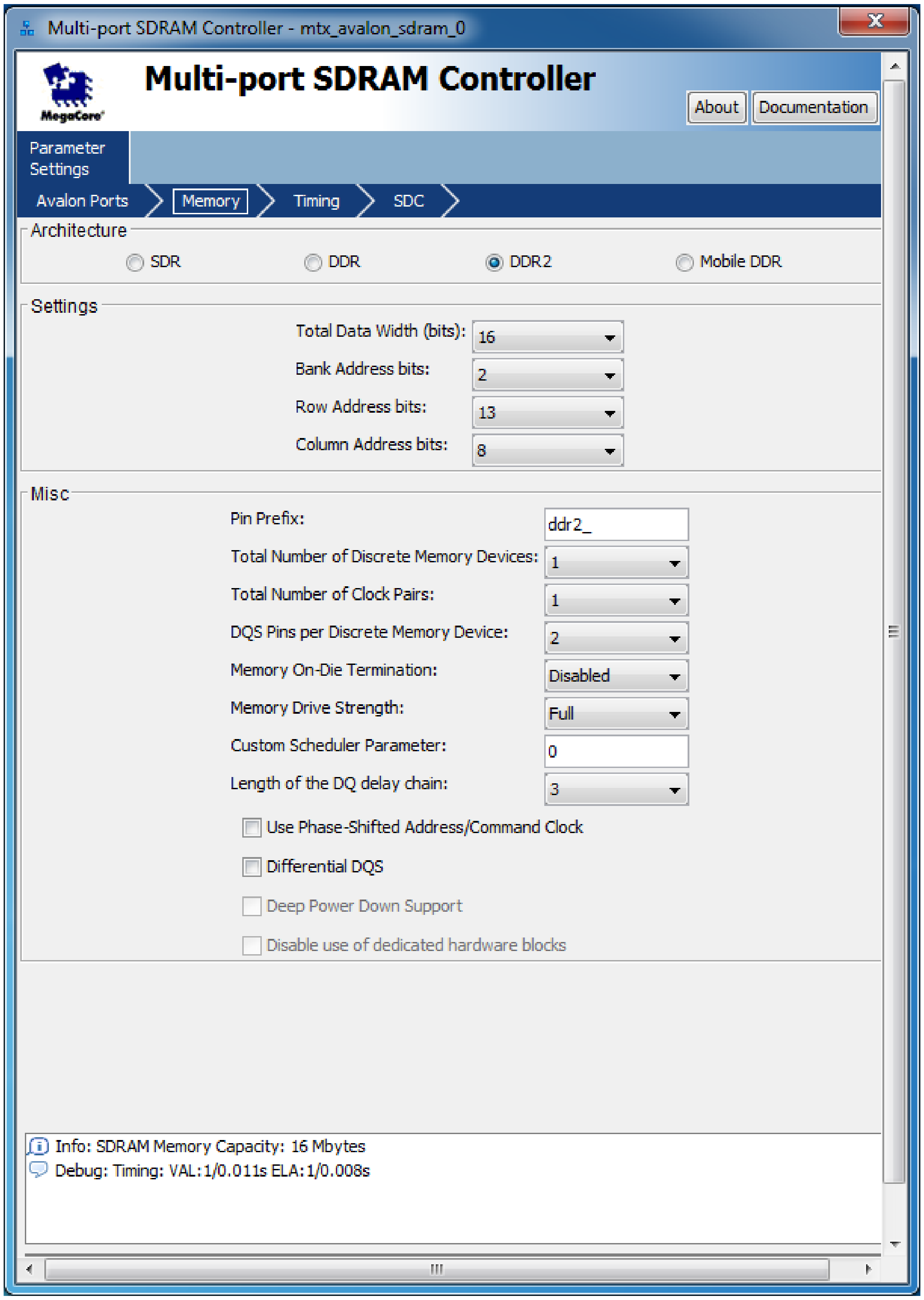
```
Info: SDRAM Memory Capacity: 256 Mbytes
Debug: Timing: VAL:1/0.019s ELA:1/0.011s
```

Figure 8: Avalon Ports Tab – Avalon Burst Settings

Memory Tab

The Memory tab is used to select the memory architecture and device specific properties of the targeted SDRAM memory system.

- Click on the Memory tab to select the SDRAM memory properties of the targeted memory device. The SDRAM settings can be found in the SDRAM datasheet.


Multi-port SDRAM Controller - mtx_avalon_sdram_0
Multi-port SDRAM Controller
MegaCore
About
Documentation
Parameter Settings
Avalon Ports
Memory
Timing
SDC

Architecture: SDR [✓ DDR2] DDR Mobile DDR

Settings

Total Data Width (bits):	16
Bank Address bits:	2
Row Address bits:	13
Column Address bits:	8

Misc

Pin Prefix:	ddr2_
Total Number of Discrete Memory Devices:	1
Total Number of Clock Pairs:	1
DQS Pins per Discrete Memory Device:	2
Memory On-Die Termination:	Disabled
Memory Drive Strength:	Full
Custom Scheduler Parameter:	0
Length of the DQ delay chain:	3

- [ ] Use Phase-Shifted Address/Command Clock
- [ ] Differential DQS
- [ ] Deep Power Down Support
- [ ] Disable use of dedicated hardware blocks

```
Info: SDRAM Memory Capacity: 16 Mbytes
Debug: Timing: VAL:1/0.011s ELA:1/0.008s
```

Figure 9: Memory Tab

DATA BITS

The total number of data bits across all memory devices on the interface (e.g. four 16-bit DDR devices have a total of 4x16 = 64 bits).

ADDRESS BITS

The number of bank, row and column address bits required by a single memory device on the interface. All devices must be the same size.

TOTAL DEVICES

The number of discrete memory devices on the interface. DDR modules usually have 8 devices.

CLOCK PAIRS

Configure the 'Clock Pairs' to match the number required by the memory architecture. For example, DIMM modules typically require three clock pairs.

DQS SIGNALS PER SDRAM DEVICE

The number of DQS signals on each discrete memory device. This is typically one per 8 bits of data width (e.g. when using 16-bit devices, this value will be 2). Each discrete device on the interface must have the same data width.

MEMORY DRIVE STRENGTH

The option sets the output drive strength of the memory devices.

ON-DIE TERMINATION

The 'On-Die Termination Support' option is used to support DDR2 memory devices incorporating on-die termination. The available termination options are 50, 75 or 150 ohms.

CUSTOM SCHEDULER PARAMETER

This is an integer value passed to the scheduler portion of the memory controller. It is not used by the default scheduler, but it can be utilized by custom schedulers to modify their behavior or allow different scheduling algorithms for multiple memory controllers instantiated in the same project. See the scheduler manual for more information.

LENGTH OF THE DQ DELAY CHAIN

This is the length of a delay chain used in the data path for DDR, DDR2, and MDDR memory types. This parameter is not applicable to Cyclone II, Cyclone III or Cyclone IV devices. The default value is 3. It may be possible to achieve timing closure at higher frequencies by adjusting the value. The optimal value will change depending on the device used and other aspects of the design.

Phase-Shifted Address/Command Clock

Enabling this option creates an additional clock input to the memory controller called “cmd_clk”. This clock is used to output the address and command signals to the memory device. Its phase can be selected to improve setup and hold margins at the memory device. The phase shift can be any value between 180 and 359 degrees.

Differential DQS

The ‘Differential DQS’ option enables support for DDR2 differential DQS pins. Each DQS pin will be replaced with a DQS_P and DQS_N pair. This option is only supported on Stratix III, IV, and Cyclone V devices.

Deep Power Down Support

The ‘Deep Power Down Support’ option is used to enable support for the deep power down mode of Mobile DDR. This option adds two additional signals to the top-level SOPC block. A high level on the DPD input is used to signal the memory controller to put the Mobile DDR into deep power down mode. The POWER_STATE output indicates when the SDRAM device is powered and initialized. When the device is in deep power down mode the POWER_STATE output will be low. After deasserting DPD to return to normal power mode, the user must wait until POWER_STATE returns to a high level before attempting to access SDRAM.

NOTE: Mobile DDR does not retain data in deep power down mode.

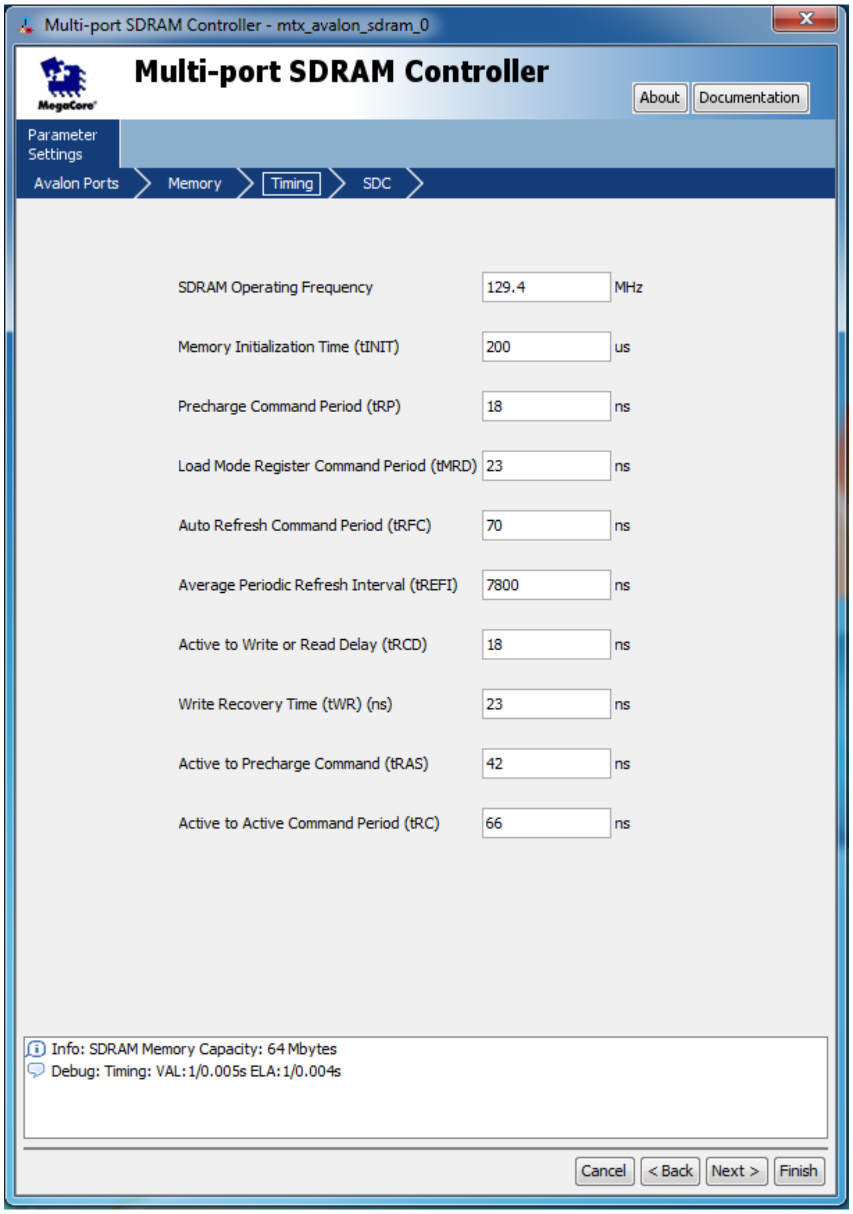
Disable use of dedicated hardware blocks

This option prevents the use of certain hardware features of devices. For HardCopy 2/3 devices that lack dedicated DDR hardware blocks used by most memory controllers, enabling this option will bypass the dedicated DDR blocks and allow the memory controller to support these devices.

Timing Tab

The Timing tab is used to enter the memory specific timing parameters found in the datasheet supplied by the vendor of the memory device.

- Click the Timing tab.
- Enter the memory timing parameters. SDRAM Operating Frequency should match the frequency of the memory clock being used in the design. All other timing parameters are found in the SDRAM memory datasheet.



SDRAM Operating Frequency	129.4	MHz
Memory Initialization Time (tINIT)	200	us
Precharge Command Period (tRP)	18	ns
Load Mode Register Command Period (tMRD)	23	ns
Auto Refresh Command Period (tRFC)	70	ns
Average Periodic Refresh Interval (tREFI)	7800	ns
Active to Write or Read Delay (tRCD)	18	ns
Write Recovery Time (tWR) (ns)	23	ns
Active to Precharge Command (tRAS)	42	ns
Active to Active Command Period (tRC)	66	ns

```
Info: SDRAM Memory Capacity: 64 Mbytes
Debug: Timing: VAL:1/0.005s ELA:1/0.004s
```

Figure 10: Timing Tab

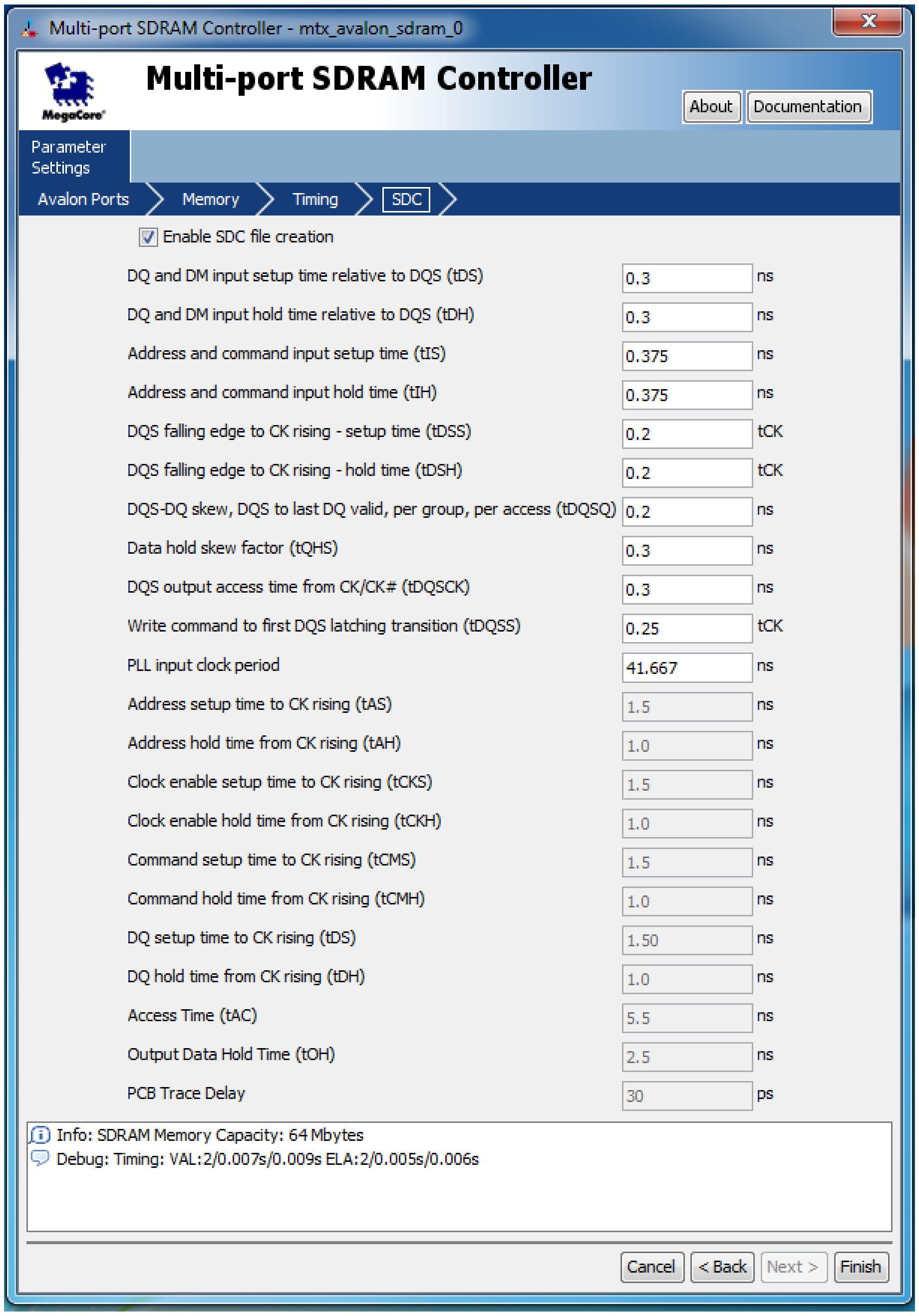
SDC Tab

The SDC tab is used to enter the memory device specific timing parameters found in the datasheet supplied by the vendor of the memory device. This tab is an extension to the timing tab. Not all fields on this tab are applicable to all types of memory. Depending on the type of memory select, certain fields will be grayed out.

The information entered under this tab is used in the Synopsys Design Constraint (SDC) script required by TimeQuest. The SDC script defines the timing constraints and specifications to validate the timing performance of the memory controller logic in the FPGA. The script is written in TCL command language and will be automatically generated when the SDRAM controller is added to the SOPC Builder system.

The parameter 'PLL Input Clock Period' is the input frequency to the PLL that generates the clocks for the memory controller. It is typically a different frequency than the memory clock generated by the PLL. This parameter is used in the SDC script to generate a base clock for timing analysis.

- Click the SDC tab.
- Check the Enable SDC file creation option.
- Enter the values for the SDC-specific parameters.
- Click Finish to add the custom variation SDRAM controller to the SOPC Builder system.
- Qsys and SOPC Builder name the new component mtx_sdram_classic_0, but you can rename it to any name.
- Connect SDRAM ports to the desired Avalon bus masters.
- Create the rest of your Qsys or SOPC Builder system and click Generate to generate the system.
- Before the project can be compiled a top-level design must be created that instantiates the SOPC Builder system and PLL. Also the SDRAM pin locations must be defined using the Quartus assignment editor or pin planner. The script generated by the SDRAM expects that the SDRAM interface pins have fixed pin names. See also Table 1. If the SDC file creation option was checked, then generated script file will be located in your project directory. The name of this file is <component_name>.sdc where ‘component_name’ is the name of your SDRAM module in SOPC. Add it to your project by clicking on Project -> Add/Remove files in Project and then selecting the SDC file from the appropriate directory.



[✓] Enable SDC file creation

DQ and DM input setup time relative to DQS (tDS)	0.3	ns
DQ and DM input hold time relative to DQS (tDH)	0.3	ns
Address and command input setup time (tIS)	0.375	ns
Address and command input hold time (tIH)	0.375	ns
DQS falling edge to CK rising - setup time (tDSS)	0.2	tCK
DQS falling edge to CK rising - hold time (tDSH)	0.2	tCK
DQS-DQ skew, DQS to last DQ valid, per group, per access (tDQSQ)	0.2	ns
Data hold skew factor (tQHS)	0.3	ns
DQS output access time from CK/CK# (tDQSCK)	0.3	ns
Write command to first DQS latching transition (tDQSS)	0.25	tCK
PLL input clock period	41.667	ns
Address setup time to CK rising (tAS)	1.5	ns
Address hold time from CK rising (tAH)	1.0	ns
Clock enable setup time to CK rising (tCKS)	1.5	ns
Clock enable hold time from CK rising (tCKH)	1.0	ns
Command setup time to CK rising (tCMS)	1.5	ns
Command hold time from CK rising (tCMH)	1.0	ns
DQ setup time to CK rising (tDS)	1.50	ns
DQ hold time from CK rising (tDH)	1.0	ns
Access Time (tAC)	5.5	ns
Output Data Hold Time (tOH)	2.5	ns
PCB Trace Delay	30	ps

```
Info: SDRAM Memory Capacity: 64 Mbytes
Debug: Timing: VAL:2/0.007s/0.009s ELA:2/0.005s/0.006s
```

Figure 11: SDC Tab

SDRAM Pin Names

The SDRAM GUI allows the selection of a prefix (e.g. sdram_) on the Memory tab for the names of all memory interface pins and it expects that these pins have a fixed name as shown in Table 2. If the pins names used in the design are different from the expected values it will be necessary to edit the generated SDC file to correct the names.

Table 2: SDRAM Pins

SDRAM Pin Function	SDRAM Pin Name (without prefix)
Clock Enable	cke
Bank Address	ba
Address	a
Chip Select *(Note 1)*	cs
Row Address Strobe *(Note 1)*	ras
Column Address Strobe *(Note 1)*	cas
Write Enable *(Note 1)*	we
Data	dq
Data Mask	dqm
Data Strobe	dqs
Differential Data Strobe Positive	dqs_p
Differential Data Strobe Negative	dqs_n
SDR Clock Out	clk_out
DDR Clock Out Positive	clk_out_p
DDR Clock Out Negative	clk_out_n

Note 1: These signals are active low.

DQ/DQS Pin Assignments

Unlike most SDRAM cores, the Microtronix memory controller is more flexible in the use of IO for the memory interface. For Stratix and Arria families, the dedicated DQ / DQS pins must be used and all Quartus DQ-group rules must be obliged. For Cyclone families only the dedicated DQS pins must be used. In this device family, the dedicated FPGA DQ IO pins are not required to be used.

WARNING:

On Cyclone III, IV, and V devices the DQS signal requires the use a digital delay element. Therefore, use only the pins in the data sheet labeled DQS / DPCLK as these have a digital delay element. *Note: In all cases, a Quartus project should be compiled prior to PCB layout to verify pin assignments.*

Running the TCL assignment script

The Qsys or SOPC Builder generation project creates the following scripts that contain assignments required for Quartus to properly compile the memory controller:

Qsys – <Qsys system>/synthesis/submodules/<Qsys system>_<memory controller>_assignments.tcl

SOPC Builder – <memory controller>_assignments.tcl

From the Tools menu, select TCL Scripts. Select the assignment script from the Libraries list and click Run.

Adding the SDC file to the project

If the Enable SDC file creation box is checked on the SDC tab in the GUI, Qsys or SOPC Builder generation project creates a timing constraints file for the memory controller:

Qsys – <Qsys system>/synthesis/submodules/<Qsys system>_<memory controller>.sdc

SOPC Builder – <memory controller>sdc

This file must be added to the project via Project->Add/Remove File in Project.

External PLL

DDR, DDR2 and Mobile DDR

In a double-data rate system the SDRAM clocks are generated by a PLL. The first PLL output drives the SDRAM clock and it runs at the maximum SDRAM frequency. The second PLL output is connected to the SDRAM write clock and it runs at the same SDRAM frequency. An optional third PLL output can be used to generate a system clock for other logic in the design.

Use the Altera MegaWizard to generate the SDRAM PLL. The SDRAM clock is not phase shifted and the SDRAM write clock is phase shifted –90 degrees.

Add the PLL to the top level and start compilation (don’t forget to assign the SDRAM pin locations before starting the compilation).

When targeting a Cyclone II device, the first compilation runs an automatically generated script that locates the most critical DDR cells and moves them to fixed locations close to the SDRAM pins. This ensures the maximum performance and best timing. Recompile the design to apply the new DDR locations and the SDRAM controller is ready for use.

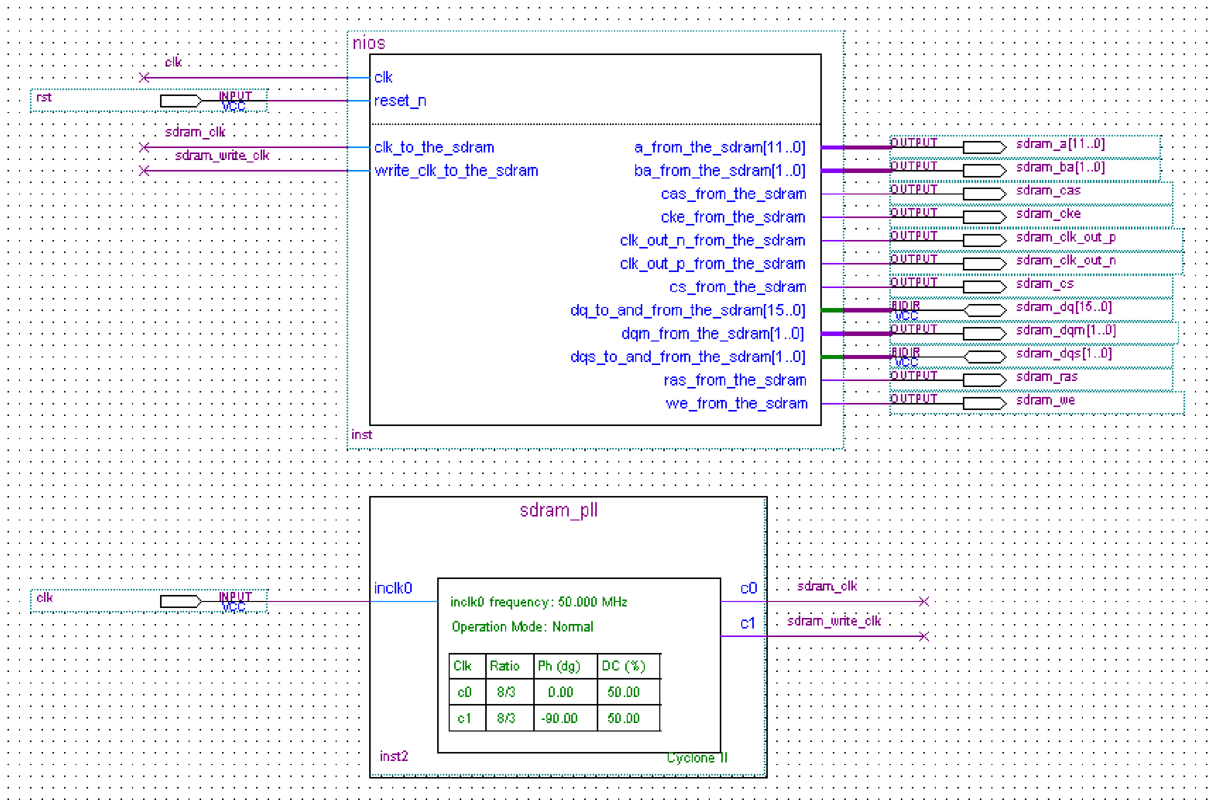


Figure 12: Clocking for DDR, MDDR, DDR Memory

SDR

In a single-data rate SDRAM system, a PLL is used to generate the various SDRAM clocks.

The first PLL output generates the SDRAM controller clock and it runs at up to the maximum SDRAM frequency (depending on the speed grade of the programmable logic and SDRAM device). Set the phase shift of this clock to zero degrees.

The second PLL output generates the capture clock for the SDRAM controller. This output must have the same frequency as the SDRAM controller clock and a phase shift of 180 degrees. The capture clock is used by the SDRAM controller to correctly read the high-speed data from the SDRAM device.

The third PLL output generates the output clock to the SDRAM chip(s). This clock has the same frequency as the first two outputs and a nominal phase shift of 90 degrees. The phase of this clock can be adjusted to optimize SDRAM timing. If the TimeQuest analysis shows timing failures ***to*** the sdram pins (address, data or control pins), try increasing the phase shift of this clock. If the TimeQuest analysis shows timing failures ***from*** the sdram_dq[*] pins, then try reducing the phase shift. The phase shift must always be greater than 0 degrees and less than (not equal to) the phase shift of the sdram_capture_clk for the timing analysis to be valid.

An optional fourth PLL output can be used to generate a system clock for other logic in the design.

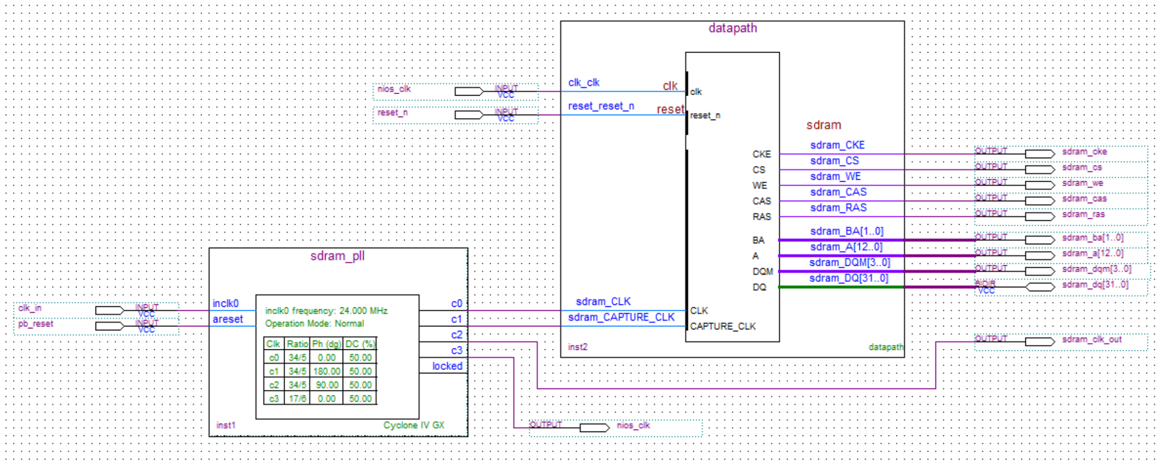


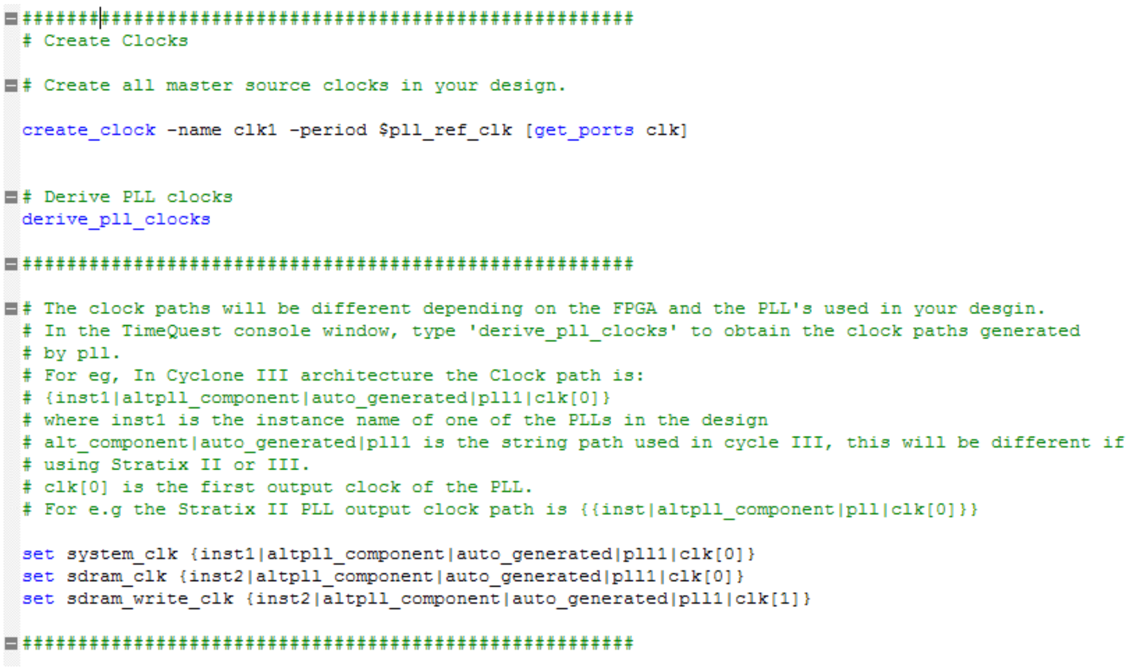
Figure 13: Clocking for SDR Memory

Use the MegaWizard Plug-in Manager in Quartus to generate the SDRAM PLL. Set the first three outputs to have the same frequency and for the first PLL output, select a phase shift of zero degrees. For the second PLL output, select a phase shift of 180 degrees, and for the third PLL, select a phase shift of 90 degrees.

If other system clocks are generated by the same PLL they can have any phase shift required by the design.

Modifying the SDC script – DDR, MDDR, DDR2

- Open the SDC file that was generated by SDRAM GUI. This script is a generic script and will need modification depending on the overall system design and configuration. The part of the code that may need modification is shown below:



```
#######################################################
# Create Clocks

# Create all master source clocks in your design.

create_clock -name clk1 -period $pll_ref_clk [get_ports clk]


# Derive PLL clocks
derive_pll_clocks

#######################################################

# The clock paths will be different depending on the FPGA and the PLL's used in your desgin.
# In the TimeQuest console window, type 'derive_pll_clocks' to obtain the clock paths generated
# by pll.
# For eg, In Cyclone III architecture the Clock path is:
# {inst1|altpll_component|auto_generated|pll1|clk[0]}
# where inst1 is the instance name of one of the PLLs in the design
# alt_component|auto_generated|pll1 is the string path used in cycle III, this will be different if
# using Stratix II or III.
# clk[0] is the first output clock of the PLL.
# For e.g the Stratix II PLL output clock path is {{inst|altpll_component|pll|clk[0]}}

set system_clk {inst1|altpll_component|auto_generated|pll1|clk[0]}
set sdram_clk {inst2|altpll_component|auto_generated|pll1|clk[0]}
set sdram_write_clk {inst2|altpll_component|auto_generated|pll1|clk[1]}

#######################################################
```

- All the master clocks in the design have to be constrained in the script. In the Figure above, there is only one master clock in the system (clk1). If your design has more clock ports, then each of them has to be specified using a create_clock command. Refer to Altera's TimeQuest Timing Analyzer and SDC support documents for more information on properly constraining the clocks.

- PLL output clock path has to be correctly defined. This path is different for Stratix and Cyclone devices. To obtain the correct PLL output clock path, type 'derive_pll_clocks' in the TimeQuest console window. Then assign the PLL output clocks to the appropriate clock variables:

  - system_clk: NIOS system clock.

  - sdram_clk: This is the SDRAM clock. The frequency of this clock is selected in the Timings tab.

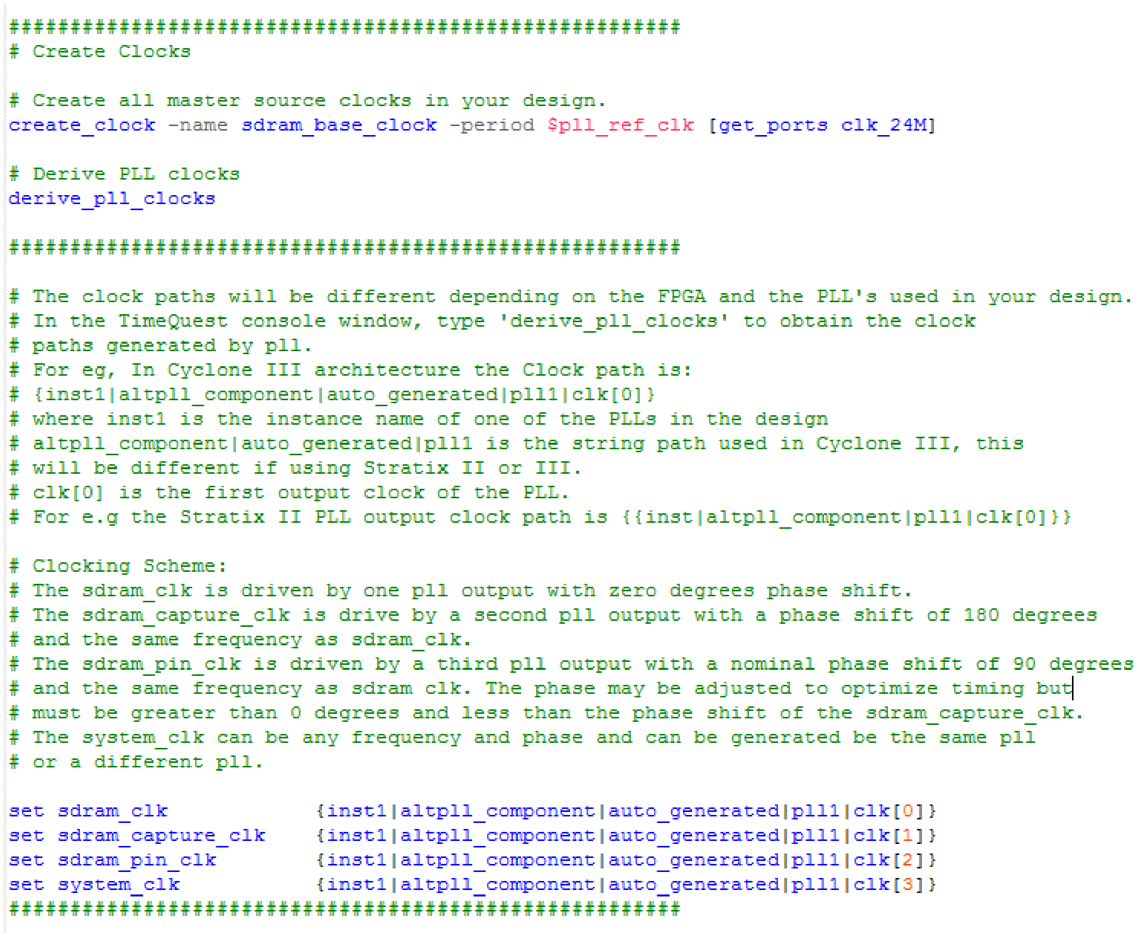
  - sdram_write_clk: -90 degree phase shifted sdram_clk.

- If there are additional clock ports or I/Os then they have to be constrained as well. There are various comments through-out the script that will guide you in constraining and setting the necessary false paths between these ports and other clocks in the system.

- The pin names in the script have to match the memory controller pin names defined in the system top level file. Make sure all the pin names are correctly defined; otherwise TimeQuest will report timing violations and the controller will not function properly.

Modifying the SDC script – SDRAM

- Open the SDC file that was generated by SDRAM GUI. This script is a generic script and will need modification depending on the overall system design and configuration. The part of the code that may need modification is shown below:



```
#######################################################
# Create Clocks

# Create all master source clocks in your design.
create_clock -name sdram_base_clock -period $pll_ref_clk [get_ports clk_24M]

# Derive PLL clocks
derive_pll_clocks

#######################################################

# The clock paths will be different depending on the FPGA and the PLL's used in your design.
# In the TimeQuest console window, type 'derive_pll_clocks' to obtain the clock
# paths generated by pll.
# For eg, In Cyclone III architecture the Clock path is:
# {inst1|altpll_component|auto_generated|pll1|clk[0]}
# where inst1 is the instance name of one of the PLLs in the design
# altpll_component|auto_generated|pll1 is the string path used in Cyclone III, this
# will be different if using Stratix II or III.
# clk[0] is the first output clock of the PLL.
# For e.g the Stratix II PLL output clock path is {{inst|altpll_component|pll1|clk[0]}}

# Clocking Scheme:
# The sdram_clk is driven by one pll output with zero degrees phase shift.
# The sdram_capture_clk is drive by a second pll output with a phase shift of 180 degrees
# and the same frequency as sdram_clk.
# The sdram_pin_clk is driven by a third pll output with a nominal phase shift of 90 degrees
# and the same frequency as sdram clk. The phase may be adjusted to optimize timing but
# must be greater than 0 degrees and less than the phase shift of the sdram_capture_clk.
# The system_clk can be any frequency and phase and can be generated be the same pll
# or a different pll.

set sdram_clk            {inst1|altpll_component|auto_generated|pll1|clk[0]}
set sdram_capture_clk    {inst1|altpll_component|auto_generated|pll1|clk[1]}
set sdram_pin_clk        {inst1|altpll_component|auto_generated|pll1|clk[2]}
set system_clk           {inst1|altpll_component|auto_generated|pll1|clk[3]}
#######################################################
```

- All the master clocks in the design have to be constrained in the script. In the Figure above, there is only one master clock in the system (clk_24M). If your design has more clock ports, then each of them has to be specified using a create_clock command. Refer to Altera's TimeQuest Timing Analyzer and SDC support documents for more information on properly constraining the clocks.

- PLL output clock path has to be correctly defined. This path is different for different FPGA devices. To obtain the correct PLL output clock path, type 'derive_pll_clocks' in the TimeQuest console window. Then assign the PLL output clocks to the appropriate clock variables:

  - system_clk: NIOS system clock.

    - sdram_clk: This is the SDRAM controller clock. The frequency of this clock is set in the Timings tab and must match the settings in the PLL.
    - sdram_capture_clk: 180 degree phase shifted clock. sdram_pin_clk: This is the clock that drives the sdram chip. It has a nominal 90 degree phase shift.

- If there are additional clock ports or I/Os then they have to be constrained as well. There are various comments through-out the script that will guide you in constraining and setting the necessary false paths between these ports and other clocks in the system.
- The pin names in the script must match the memory controller pin names defined in the system top level file. Make sure all the pin names are correctly defined; otherwise TimeQuest will not correctly analyze the timing and the controller will not function properly.

Optimizing Performance

Depending on the memory architecture and FPGA device family, the TCL assignment script may include delay assignments. The delay values in the TCL script will be typical values. To achieve the maximum possible operating frequency it may be necessary to customize the delay values for each design. The Quartus Assignment Editor can be used to change the delay values.

Pin Assignment Suggestions

Before PCB layout, the project should be compiled in Quartus to verify pin assignments and timing closure. The choice of pins will affect the maximum memory performance that can be achieved.

The dual function VREF / IO available on some device families typically have lower performance than dedicated pins. It is recommended not to use these pins for the memory interface. If they are used, the Memory clock speeds may have to be reduced to achieve timing closure.

MAX10 devices may have low speed IO banks on the top or left edges of the device. Refer to the device handbook for the locations of these banks. The low speed IO pins are not recommended for use in the memory interface. If they are used, it will not be possible to achieve the maximum performance numbers provided in this manual.

On Cyclone IV and V devices the DQS signal requires the use a digital delay element. Therefore, use only the pins in the data sheet labeled DQS as these have a digital delay element.

Performance

Table 3 shows the maximum performance results for the SDRAM controller. Actual performance may be affected by the memory width, system LE count, Quartus version, and the FPGA pin assignments.

Table 3: Maximum SDRAM Performance

Device	Speed Grade (Commercial)	System Fmax (MHz)		
		DDR	DDR2	Mobile DDR
Arria II GX	-4	200	267	200
	-5	200	233	200
	-6	200	200	200
Arria GX	-6	200	200	200
Stratix IV	-2	200	333	200
	-3	200	300	200
	-4	200	267	200
Stratix III	-2	200	333	200
	-3	200	300	200
	-4	200	267	200
Stratix II	-3	200	300	200
	-4	200	267	200
	-5	167	233	167
MAX 10	-6	144	144	144
	-7	136	136	136
	-8	128	128	128
Cyclone 10 LP	-6	200	200	200
	-7	175	180	180
	-8	167	167	167
Cyclone V	-6	167	167	167
	-7	156	156	156
	-8	140	140	140
Cyclone IV E	-7	167	200	167
	-8L	167	167	167
	-9L	133	133	133
Cyclone IV GX	-6	190	220	175
	-7	180	200	167
	-8	167	167	150

Cyclone III	-6	200	200	200
	-7	167	167	167
	-8	167	167	167
Cyclone II	-6	200	200	200
	-7	180	180	180
	-8	160	160	160

Device	Speed Grade (Commercial)	System Fmax (MHz)
		SDR
Cyclone II	-6	167 MHz
	-7	156 MHz
	-8	144 MHz
Cyclone IV E	-6	167 MHz
	-7	156 MHz
	-8	144 MHz
Stratix IV	-2	167 MHz
	-3	162 MHz
	-4	153 MHz
MAX 10	-7	116 MHz
Cyclone V	All	Not Recommended
Arria V	All	Not Recommended
Stratix V	All	Not Recommended

Resource Requirements

Table 4 shows the typical size in logic elements (LE) for the various SDRAM Controller modules. The actual number of logic elements may vary depending on the device family and Quartus settings. The table shows the minimal M4K RAM blocks usage. If a large cache size is selected, the number of M4K RAM blocks may increase.

Table 4: FPGA Resource Requirements

Module	LE*	M4K RAM Blocks*
SDRAM Controller	700	-
Avalon Random Port	500	2
Avalon Burst Port	500	2

***NOTE:** The number of logic element resources depends on the memory architecture, data width and cache settings. The numbers shown in the table are for a 16-bit DDR SDRAM implementation.

Simulation

A precompiled simulation library is provided for performing simulations using ModelSim. The library is located in the <install_dir>/simulation directory. Perform the following steps to simulate your design with the sdram memory controller.

1. Launch ModelSim
2. Map the sdram memory controller library. At the ModelSim prompt type;
   ```
   vmap mtx_sdram <install_dir>/simulation/mtx_sdram
   ```
   If you use a newer version of ModelSim, you must refresh the precompiled library. At the Modelsim prompt type;
   ```
   vcom -refresh -work mtx_sdram
   ```
3. Compile the sdram top level. Eg. If the sdram controller was named `sdram` in the SOPC Builder, then type;
   ```
   vcom -93 sdram.vhd
   ```
4. Compile all of the design files
5. Start the ModelSim simulation by typing;
   ```
   vsim -t ps -L mtx_sdram <top_level>
   ```

This procedure assumes all of Altera's simulation libraries are installed and available to ModelSim. If this is not the case, use the following procedure to generate the libraries with Quartus.

1. From the Tools menu, select Launch EDA Simulation Library Compiler.
2. Select ModelSim from the Tool name list. Browse to and select the location of the ModelSim executable.
3. Select the following device families: Cyclone II, Cyclone III, Cyclone IV E, Cyclone IV GX, Cyclone V, Stratix II, Stratix III and Stratix IV.
4. Select the desired language and output directory.
5. Click Start Compilation.
6. Once the libraries are generated, they must be mapped into ModelSim.

Verification

The SDRAM controller has been verified on Altera and other FPGA development boards. Table 5 shows the hardware platforms on which the IP core has been tested and the memory devices contained on each board.

Table 5: Tested Hardware Platforms

Development Board	Altera Device	SDRAM Device
Altera Arria II GX FPGA Development Board	EP2AGX125EF35	Micron MT8HTF12864HZ-800H1 (DDR2)
Altera Arria GX Development Kit	EP1AGX60DF780	Micron MT47H32M16 (DDR2)
Altera Stratix III FPGA Development Kit	EP3SL150F1152C2	Micron MT47H32M8BP-3 (DDR2)
Altera Audio Video Development Kit, Stratix II GX Edition	EP2GX90FF40C3	Micron MT9HTF6472AY-53EB3 (DDR2)
ViClaro III	EP3C120F780C7	Micron MT47H32M16BN-3 (DDR2)
ViClaro II	EP2C35F484C6	Micron MT47H16M16BG-3 (DDR2)
Altera Cyclone III Starter Kit	EP3C25F324C8	PowerChip A2S56D40CTP-G5 (DDR)
ViClaro	EP2C20F256C7	ISSI IS42S16800B-7TL (SDR)
FireFly II	EP2C20F484C8	Micron MT48LC8M32B2B5-7 (SDR)
Sendero	EP2C35F672C6	ISSI IS43R16160A-6T (DDR)
Sendero	EP2C35F672C6	Infineon HYB25D256160CE-5 (DDR)
Vivien	EP2C5T144C6	ISSI IS42S16100C1-6TL (SDR)
Altera Nios II Development Kit, Cyclone II Edition*	EP2C35F672C8	Micron MT46V16M16-6T (DDR)
Sasco Holz Pablo	EP2C20F484C6	Micron MT46H8M32LFB5-6 (Mobile DDR)
Altera DE4 Development and Education Board	EP4SGX230KF40C2	Hynix H5PS1G83EFA (DDR2)

*Requires board modification, contact support@microtronix.com for information

Installation

Follow these steps to install the Microtronix SDRAM controller on your computer.

1. Insert the Microtronix SDRAM Memory Controller Installation CD into your CD-ROM (or equivalent)
2. The setup program for the package should start. If it doesn't, browse to the CD using Windows Explorer and double-click on the setup icon.
3. Follow all the prompts. The setup program will attempt to auto-detect the installation location of the Quartus II and Nios II Embedded Processor. Please correct the specified paths if the setup program doesn't or incorrectly detects them.

License

A valid IP core license is required from Microtronix to generate program files incorporating the SDRAM IP-core. These licenses are generated based on a NIC or Guard ID supplied by the user. They can be either server or workstation based.

After purchasing a license you receive your license file. Copy the license file (license.dat) to your current Quartus license file and the SDRAM controller (CC21_6240) will show in the Quartus License Setup.

With the free OpenCore Plus feature the SDRAM controller can be evaluated in real hardware. If during compilation no valid license is detected, the OpenCore Plus feature is enabled.

Please contact Microtronix for licensing details.

FAQ

Q1. Why does Quartus report the error 'Error: DQS I/O pin does not feed a Clock Delay Control block'?

A1. For Cyclone II devices the DQS signals must be assigned to the dedicated DQS pins.

Q2. Do you have any recommendations for layout of my SDRAM memory board?

A2. Microtronix does not have specific layout recommendations. Please follow the layout guidelines of your SDRAM manufacturer. For example, Micron's recommendations for DDR/DDR2 design can be found at

http://www.micron.com/support/designsupport/tools/ddrtoolbox/ddrtoolbox.aspx

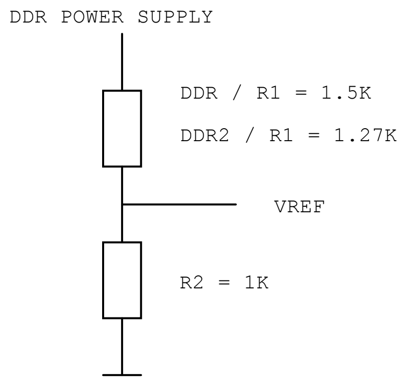
Q2. My DDR / DDR2 SDRAM memory board layout includes external parallel termination resistors. Is this supported?

A3. During idle the parallel termination resistor pulls the DQ/DQS signal to the VTT voltage level, which is the same level as the VREF voltage. The SSTL input buffer is a comparator and this causes unwanted glitches inside the programmable device.

For DDR2 memory architectures using On-Die Termination (ODT) is preferable and this feature gives better SI results than external parallel termination.

To support external parallel termination one of the following steps must be applied to the board.

1. In design using a point-to-point memory architecture (short DQ/DQS traces) and single parallel termination resistors, remove the resistor connected to DQS0.
2. In designs incorporating more complex memory architecture (longer traces, DIMM sockets) or resistors packages, adjust the VREF voltage by using the voltage divider.


DDR POWER SUPPLY
DDR / R1 = 1.5K
DDR2 / R1 = 1.27K
VREF
R2 = 1K