MEMOIR SYSTEMS

Renaissance
DATA SHEET

Renaissance™ Product Family mRnW Memory Cores

This datasheet provides functional information for the multiport Read-Write (mRnW) Algorithmic Memory® cores.

Release Date: Nov 5th 2012

Contents

1. Introduction

This datasheet describes the functioning of Memoir's Renaissance product family memory cores. The cores are created by wrapping RTL-IP around physical memories in a target library. The RTL implements the algorithm specified memory core type. The memory cores are generated by the user using the MemoGen™ software made available by Memoir Systems. The Renaissance family currently offers memory cores which provide 2X, 4X performance acceleration.

This datasheet gives an overview of a generic 'mRnW' memory core, a multi-port read-write core with 'm' read-ports and 'n' write-ports. Table 1-1 lists the various combinations of 'm' and 'n' offered in the Renaissance product family. Depending on the availability (or non-availability) of certain physical memories in the target library some of the IP cores listed in the table may NOT be available.

This datasheet describes a sample memory core with four read and four write ports. Functionality of cores with different values of 'm' and 'n' can be inferred similarly from this description. Specific information about a memory core - address/data bus-widths, clock-frequency, timing parameters, area, power, underlying physical memory components, refresh-scheme details (if applicable), etc. - is made available to the user in a separate datasheet which is auto-generated by the MemoGen software.

Each mRnW memory core supports the following:

- Parameterized number of address-words, data widths
- Clock-synchronous, SRAM-like interface with pipelined random memory access for all ports
- Pass-through interface for diagnostic and testability (incl. BIST) support

Table 1-1: Renaissance Memory Core Types

Core Designation	Physical Read Ports	Physical Write Ports	Notes
Renaissance 2X			
1RW	1	1	Single port read or write memory
2Ror1W	2	1	Reads and write are mutually exclusive.
1R1W	1	1	Read and write operations can occur simultaneously.
1RW1W	1	2	First port is read/write; second port is write-only.
1R1RW	2	1	First port is read-only; second port is read/write.
2RW	2*	2*	*Each port is either a read or a write port in a given cycle. Address bus is common.
Renaissance 4X			
1R2W	1	2	Read and write operations can occur simultaneously.
1R3W	1	3	Read and write operations can occur simultaneously.
2R1W	2	1	Read and write operations can occur simultaneously.
2R2W	2	2	Read and write operations can occur simultaneously.
3Ror1W	3	1	Reads and write are mutually exclusive.
3R1W	3	1	Reads and write operations can occur simultaneously.
4Ror1W	4	1	Reads and write are mutually exclusive.
4R4W	4	4	Reads and writes can occur simultaneously.

2. Block Diagram and Interface Description

Figure 2-1 illustrates the block diagram of a 4 read port and 4 write port Renaissance memory core. For compactness, subscripts are appended to the port signal names to imply four independent ports. Subscripts {0, 1, 2, 3} imply four read ports numbered port-0, port-1, port-2, and port-3. Similarly subscripts {4, 5, 6, 7} imply four write ports numbered port-4, port-5, port-6, and port-7. These subscripts should NOT be confused as bit-numbers of a bus.

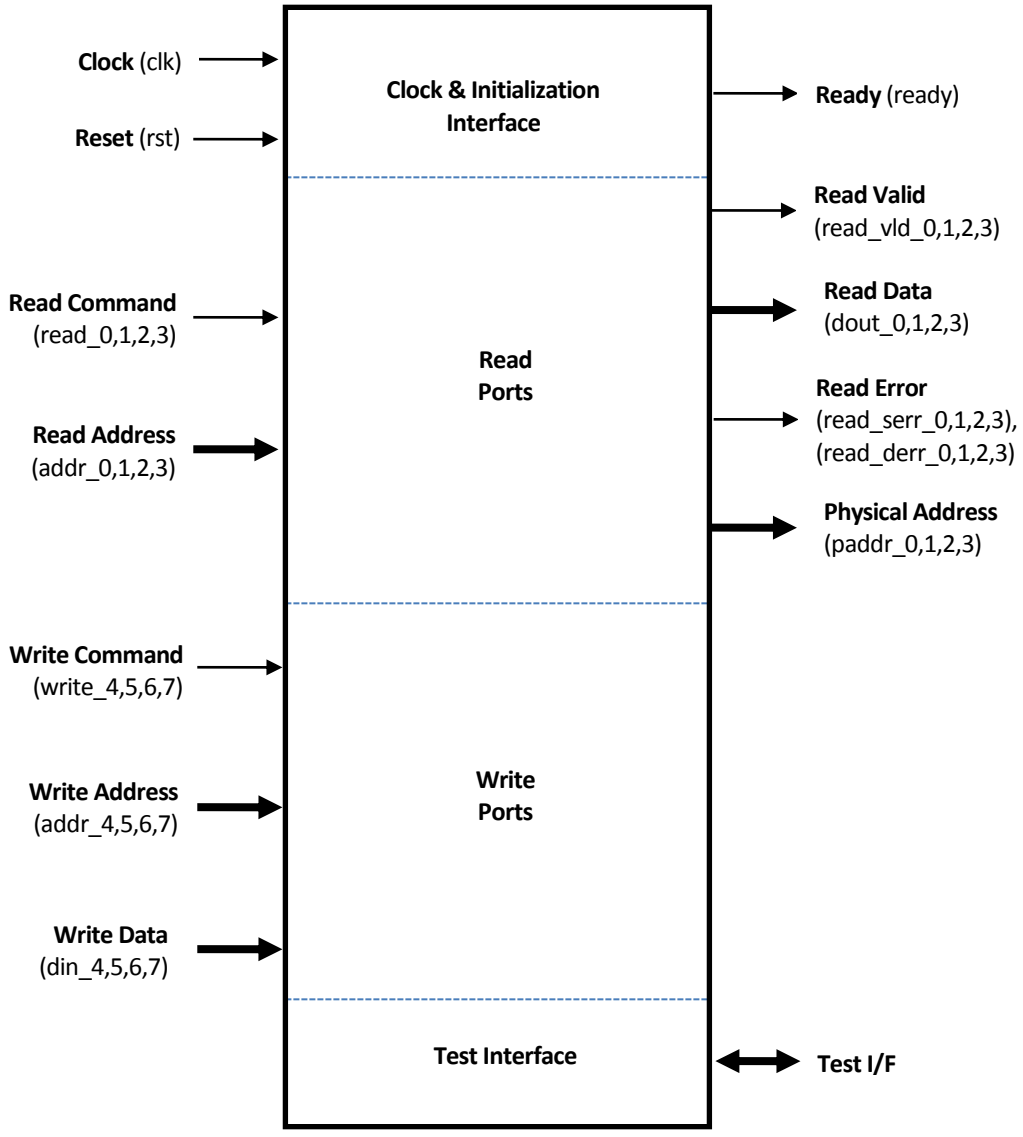

Figure 2-1: Memory Core Block Diagram

2.1. Interface Description

Table 2-1 describes the interface pins of a memory core shown in Figure 2-1.

Table 2-1: Interface Pins

Pin	Pin Type	Description	Notes
Clock and Initialization			
clk	Input	Clock	Clock input pin
rst	Input	Reset	Reset pin
ready	Output	Ready	Assertion implies memory core ready for functional operation
Read Ports			
addr_{0,1,2,3}[1][2]	Input	Read Address	Address bits (width N bits) to denote up to 2^N addresses; N is parameter. Same value for all ports
read_{0,1,2,3}	Input	Read Valid	Denotes that a Read command is valid
dout_{0,1,2,3}	Output	Read Data	Data (width W) associated with a read address; W is parameter. Same value for all ports
read_vld_{0,1,2,3}	Output	Read Data Valid	Denotes valid read data
read_serr_{0,1,2,3}, read_derr_{0,1,2,3}	Output	Read Error	Denotes read data error. Data on *dout* pins is invalid
paddr_{0,1,2,3}	Output	Physical Address	Physical Address (width P bits) associated with the read. Parameter P depends on the physical address space of the memory core
Write Ports			
addr_{4,5,6,7}[3]	Input	Write Address	Address bits (width N bits). Parameter N is the same as that used for Read address.
write_{4,5,6,7}	Input	Write Valid	Denotes that a write command is valid
din_{4,5,6,7}	Input	Write Data	Data (width W bits) associated with a write command; Parameter W is same that for Read data.
Test Interface			
Test I/F	I/O	Test Interface pins	Test/Diagnostic pins associated with base memories

[1] For compactness, port signal names are appended with '_{w, x, y, z}' . **Each appended signal name implies four independent port signal names.** This nomenclature is followed throughout the document.

[2] For 'n' RW cores (n=2,3,4), the port numbering is same for read and write ports. Further, the address busses are common. E.g. for a 2RW core, only *addr_0*, *addr_1* are offered. The other signals are *read_0,1, write_0,1, dout_0,1, paddr_0,1* and *din_0,1*.

[3] For cores with less than 4 read ports, the write port numbering begins where the read port numbering ends. E.g. for a core with 2 read ports and 1 write port, the numbering scheme would be read_0, read_1, write_2.

The different interfaces are described next.

- **Clock and Initialization pins:** This set of signals is common to all ports. Clock is provided to the memory core via the *clk* input pin. The frequency is user determined. The clock pin is common to all the ports indicating full synchronous operation across the memory core (Multi-clock domain operation across ports is currently not supported).

  The memory core is initialized by asserting the *rst* pin for a minimum of ten clocks. The completion of reset process is indicated by assertion of the *ready* pin. The mRnW memory core is ready for normal operations on the next clock after this assertion.

- **Read Interface:** On each port, the interface accepts a Read command (read_{0,1,2,3}) and a corresponding Read address (*addr*_{0,1,2,3}). The read data (*dout*_{0,1,2,3}) is available after a fixed latency along with the read valid signal (*read_vld*_{0,1,2,3}). The latency of a read access is fixed irrespective of operations on other ports. The *read_vld* signal is provided as a timing signal to enable the host chip to accept valid data. This way the host chip does not have to hardwire the latency value in its logic.

  Read Error: The assertion of read error signal indicates detection of a parity error in the internal data structures implemented in the core. Their occurrences are reported with the assertion of *read_serr* (single-bit error), or *read_derr* (double-bit error) outputs. The read physical address (described next) points to the physical memory location where the error occurred. The user is required to reset the memory core and initiate the reset process. If the error was 'soft' in nature then normal operations should ensue after reset. If errors repeat then user could choose to do BIST testing via the pass-through test interface.

  Read error events happen only for certain memory cores, depending on the underlying algorithm. The core-specific datasheet states if the Read error outputs are used for that core. If not, these outputs are held inactive (low) throughout the course of operation.

  Read Physical Address: ECC implementation on the user data is the responsibility of the user. The user can choose to do this as part of data itself. Due to address mapping schemes deployed within the core, the physical location of data for a given user read address may change from access to access. The physical address of the read location (*read_padr*_{0,1,2,3}) is provided to help with address logging in case the user chooses to maintain statistics on ECC error events. The mapping between the physical address and physical memories (reference-designators) is detailed in the memory core-specific datasheet.

  Note: The read Physical address is mainly used during normal operation to help the user track any ECC error events seen in the data. But for those cores where a Read error event can happen, the physical address also points to the erring memory location during such an event.

- **Write Interface:** The interface accepts a write command (*write*_{4,5,6,7}), a write address (*addr*_{4,5,6,7}) and the corresponding write data (*din*_{4,5,6,7}).

- **Test I/F:** This set of signals is common to all ports. Memoir memory cores support all Test and Diagnostics features (including M-BIST) offered by the vendor of the physical library. All the associated pins are direct passed-thru to the physical memories from the Test Interface.

3. Interface Timing

Timing information for the various operations is presented in this section.

3.1. Initialization

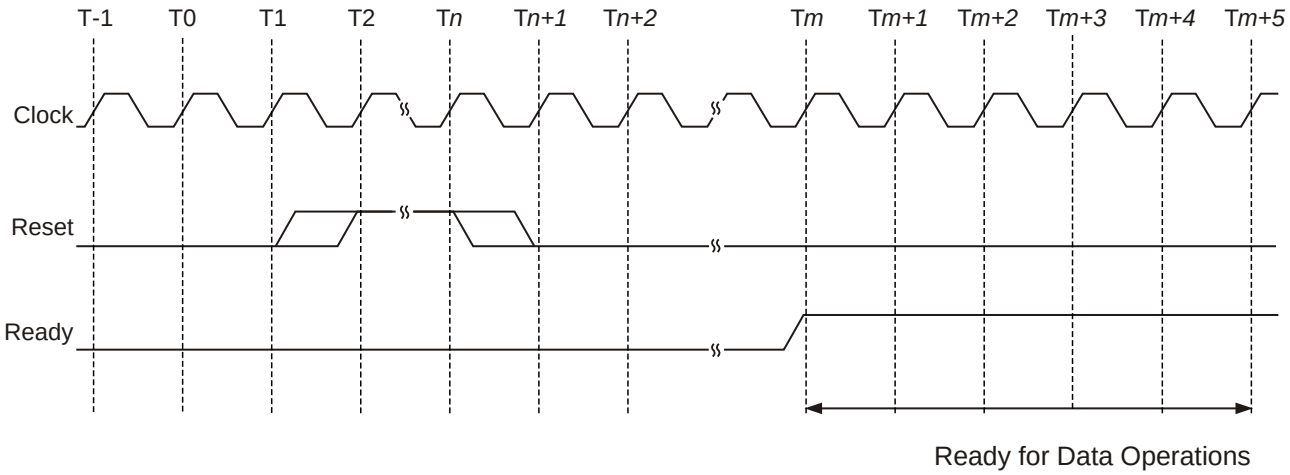

Figure 3-1: Reset Sequence

The reset sequence presented in Figure 3-1 brings the mRnW memory core out of reset. The *reset* pin is asserted for a minimum of 10 clock cycles.

The internal reset process begins after the de-assertion of reset. It completes a fixed time delay later. The completion is indicated by the assertion of *ready* pin. The *reset* to *ready* delay will depend on the size of memory core and the algorithm implemented. Its value is specified in the memory core-specific datasheet (auto-generated by the MemoGen software). The host logic should wait until *ready* is asserted. The mRnW memory core is ready to accept data operations on the next clock after *ready* is asserted. The *ready* pin is kept asserted throughout the operation of the memory core. This is shown in the timing diagrams for the read and write operations in the following sections.

3.2. Read Timing

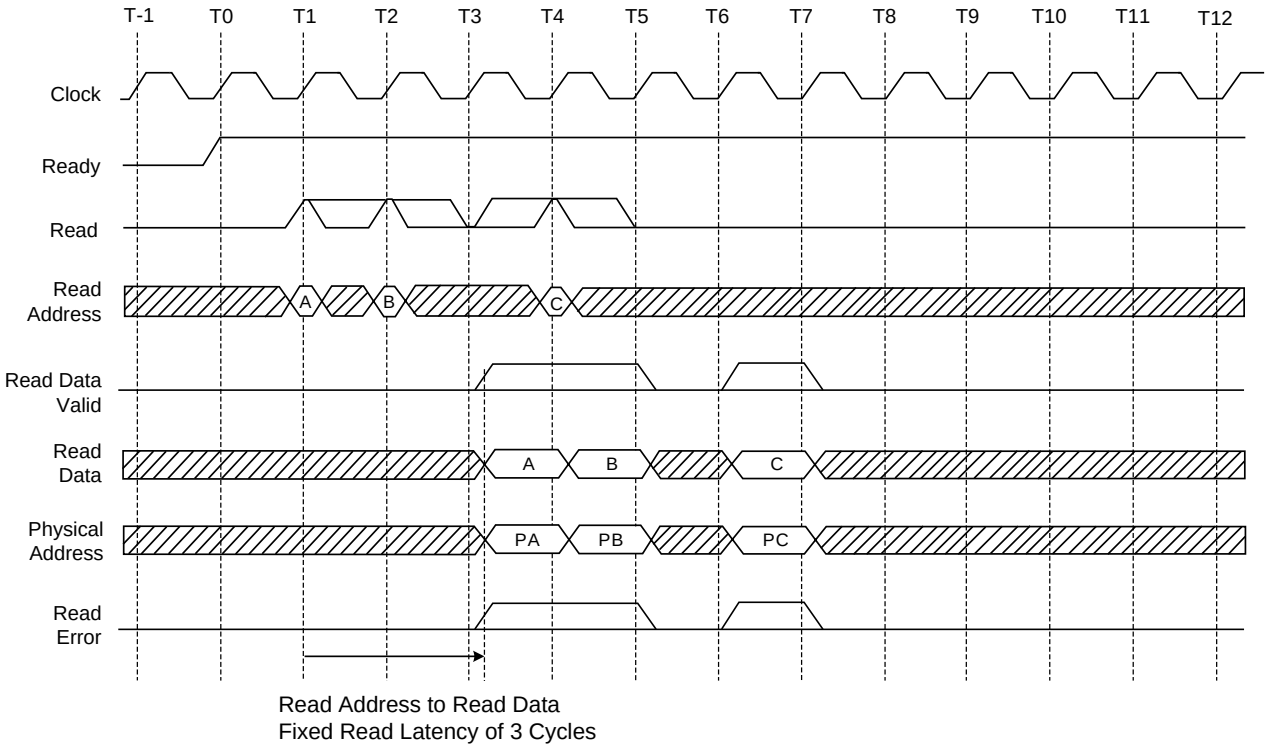

Figure 3-2: Read Interface Timing Diagram

The timing diagram in Figure 3-2 shows three read requests - A, B and C respectively. The first read operation reads address A in clock cycle 1. Data for A is returned in clock cycle 3. Similar operations happen for addresses B and C in clock cycle 2 and 4 respectively. Data for B is returned in clock cycle 4 while data for C is returned in clock cycle 6. The three read ports operate independently of each other. The signal names in the figure represent signals for ALL the read ports of the memory core.

A read latency of '3' clocks is assumed in the illustrations above. The latency of a given memory core is always fixed, irrespective of port operations on other ports. It depends on the inherent latencies of the base-memories and the algorithm type implemented. The validity of data for requests A, B, and C is indicated by the assertion of *read_vld* in clock cycles 3, 4 and 6.

The mRnW memory core does NOT provide error-protection for user data. The user is responsible for incorporating any error-detection/correction schemes as part of the data itself. The physical address (*paddr_{0,1,2,3}*) output is provided to help the user with any address/data logging for ECC statistics maintenance on user data. The mapping between physical address and the underlying physical memories is provided in the memory core-specific datasheet. The physical addresses (PA, PB, and PC) for each read request are returned alongside their data output – in cycles 3, 4 and 6 respectively.

3.3. Write Timing

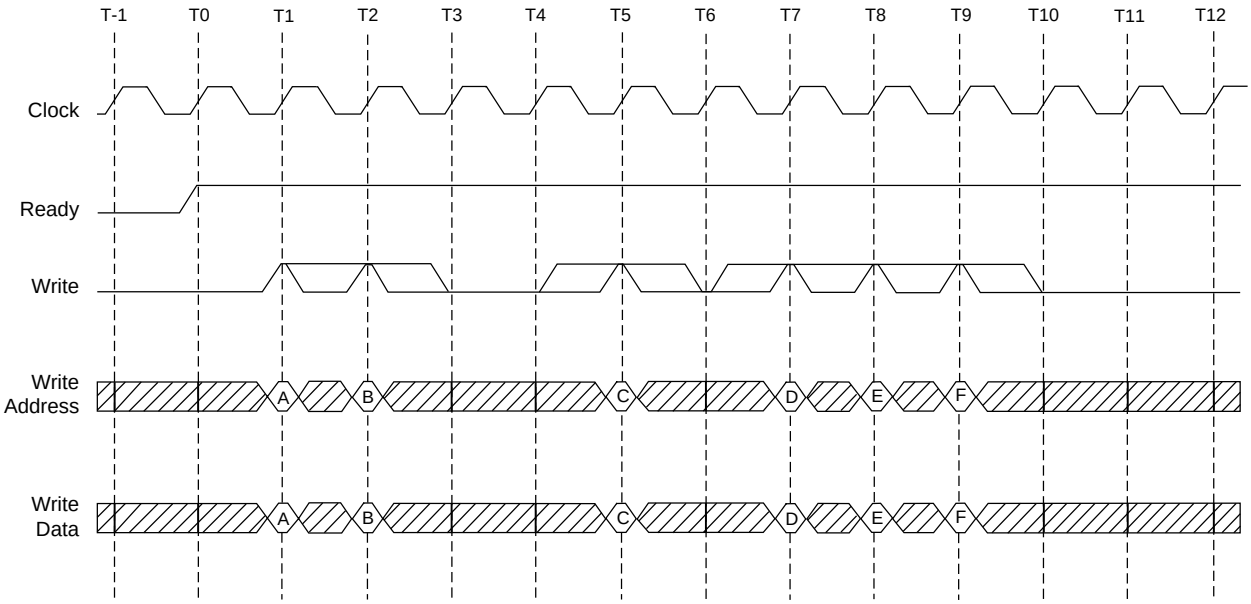

Figure 3-3: Write Interface Timing Diagram

The timing diagram in Figure 3-3 shows six write requests – A, B, C, D, E, and F. The first write operation writes to address A in clock cycle 1 with data for A being written in the same cycle. The second write operation writes to address B in clock cycle 2. Similarly, the write operations to addresses C, D, E, and F happen in clock cycles 5, 7, 8, 9 respectively with the corresponding data being written in the same cycles.

Like the read ports, the write ports operate independently of each other. The signal names in the figure represent signals for ALL the write ports.

3.4. Read and Write Interaction

As stated earlier, the read and write interfaces operate independently of each other unless the memory core is of type 'mR**or**nW'. For these memory cores the read operations are mutually exclusive with write operations.

When independent operation of read and write is allowed (memory core type mRnW), and if a read and a write transaction occur to the same address in the same clock cycle, the old value associated with the read address is returned on *dout* and the new value on the write data bus is posted to the address. In this sense, the read is considered to be executed *before* the write.

For memory cores with multiple write ports, when simultaneous writes to the same address from multiple ports occur then the data from the highest port number gets written to that location.

3.5. Refresh Timing

This section is **optional**. It applies to only those memory cores which are built with eDRAMs as base memories. Please refer to Table 1-1 for the applicable memory core types.

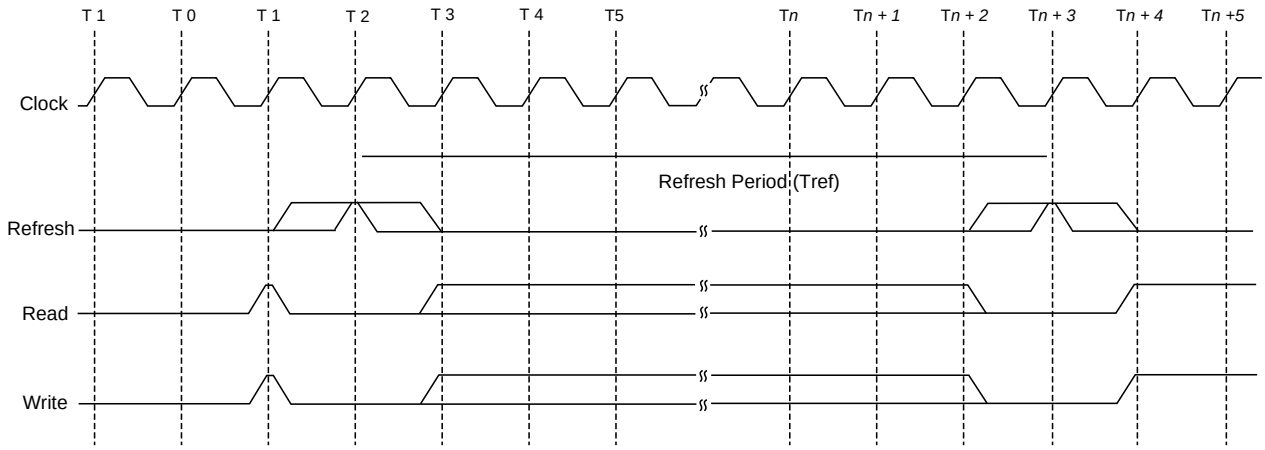

Figure 3-4: One-in-N Refresh Timing

Figure 3-4 shows the refresh operation for a One-in-N refresh scheme. Other refresh schemes for an eDRAM based memory core are possible. If implemented, they would be specific to a given memory core implementation and will be described in the memory core-specific datasheet.

A *refr* command to the mRnW memory core initiates a refresh operation on the eDRAM instantiated inside. The refresh-controller implemented in the memory core manages the refresh bank address sequencing and issues it internally on the eDRAM's address bus. The user is NOT required to issue any addresses externally at the memory core interface.

The only requirement on the user's side is to provide one refresh command within the Tref window. A variation of this scheme is to provide a minimum number of refresh commands (Nminref) within the Tref window. Nminref depends on the size of the instantiated eDRAM memory core. The values of Nminref and Tref are specified in the memory core-specific datasheet which is auto-generated by the MemoGen software.

The user should NOT assert either the *read* or *write* commands on any port when *refr* is asserted. Doing so will result in erroneous read or write operation.

4. Revision Change Log

Version	Date of Release	Notes
V1.0	March 17th 2012	Initial Release.
V1.1	Aug 17th 2012	Renaissance 4X, Read_Error functionality addition
V1.2	Nov 5th 2012	Added 4R4W and clarified that not all IP cores may be available depending on the target library

Memoir Systems Inc.

2350 Mission College Blvd. # 1275, Santa Clara, CA 95054

Phone: 1-408-550-2382

Email: support@memoir-systems.com