RAMBUS

Rambus Memory Controller

April 3, 1998

Written by: *Frederick A. Ware*
Rambus Inc.

Rambus Inc.
2465 Latham Street
Mountain View, California USA
94040

Telephone: 650-944-8000
Fax: 650-950-8080

DL0030-03

Version 3.0

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

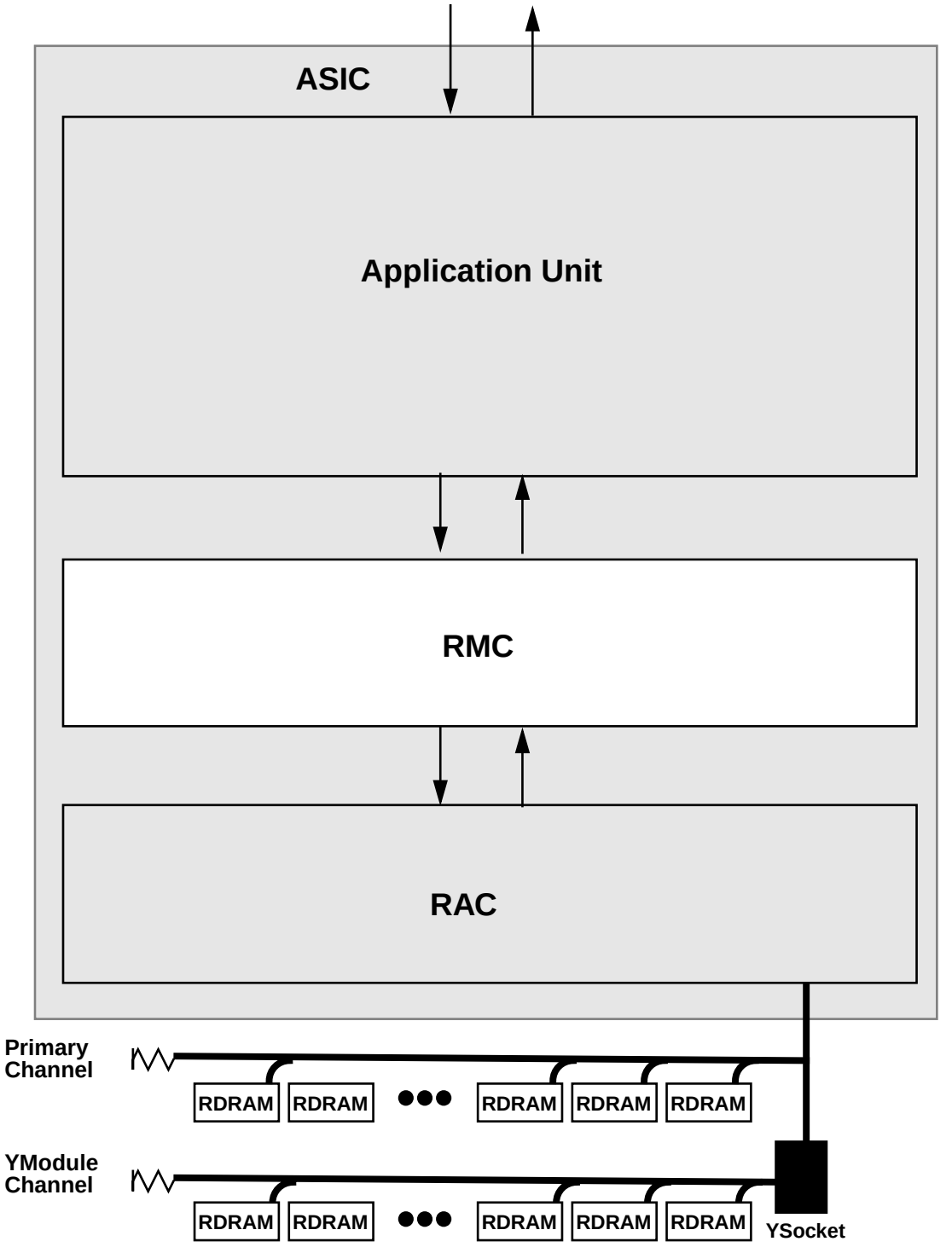
The Rambus Memory Controller (referred to hereafter as RMC) is a block of digital logic residing on an Application Specific Integrated Circuit (ASIC) which connects to a Rambus memory subsystem. A block diagram of the ASIC and subsystem are shown in the following figure.

The ASIC contains a block of logic with the storage and processing functions needed by the application (this will be called the Application Unit). The application requires further storage in the external Rambus DRAM (RDRAM) components. A Rambus ASIC Cell (RAC) is included as an Input / Output cell in the libraries of Rambus partners. The RAC provides the basic multiplexing / demultiplexing functions for converting from a byte-serial bus with a 600 MHz signaling rate to an eight byte wide bus with a 75 MHz signaling rate. The RAC manages the physical layer of the Rambus subsystem.

The RMC manages the logical layer of the Rambus subsystem. The RMC sits between the RAC and the Application Unit and provides a simple intermediate protocol for performing read and write transactions to RDRAMs. It serves as a reference design for system customers, and it may be used as-is, or it may be modified for a particular system application.

The RMC supports from one to sixteen Base or Concurrent RDRAMs on a Primary Channel and (optional) Y-Module Channel. It accepts read and write transactions from the controller and manages the operation of the RAC and RDRAMs in an optimal manner; no Channel bandwidth is wasted and the RMC adds minimal latency to read transactions. The RMC also supports interleaved transactions, permitting a RAS access to be started in one RDRAM while a CAS access is performed to another.

Figure 1: Block diagram of the RMC Environment



Chapter 2 RMC/Application Unit Interface

2.1 Signal Description

The following table and figure summarize the signals used at the interface between the RMC and the Application Unit. The signals between the RMC and the RAC are not visible to the application, and will not be discussed in this document.

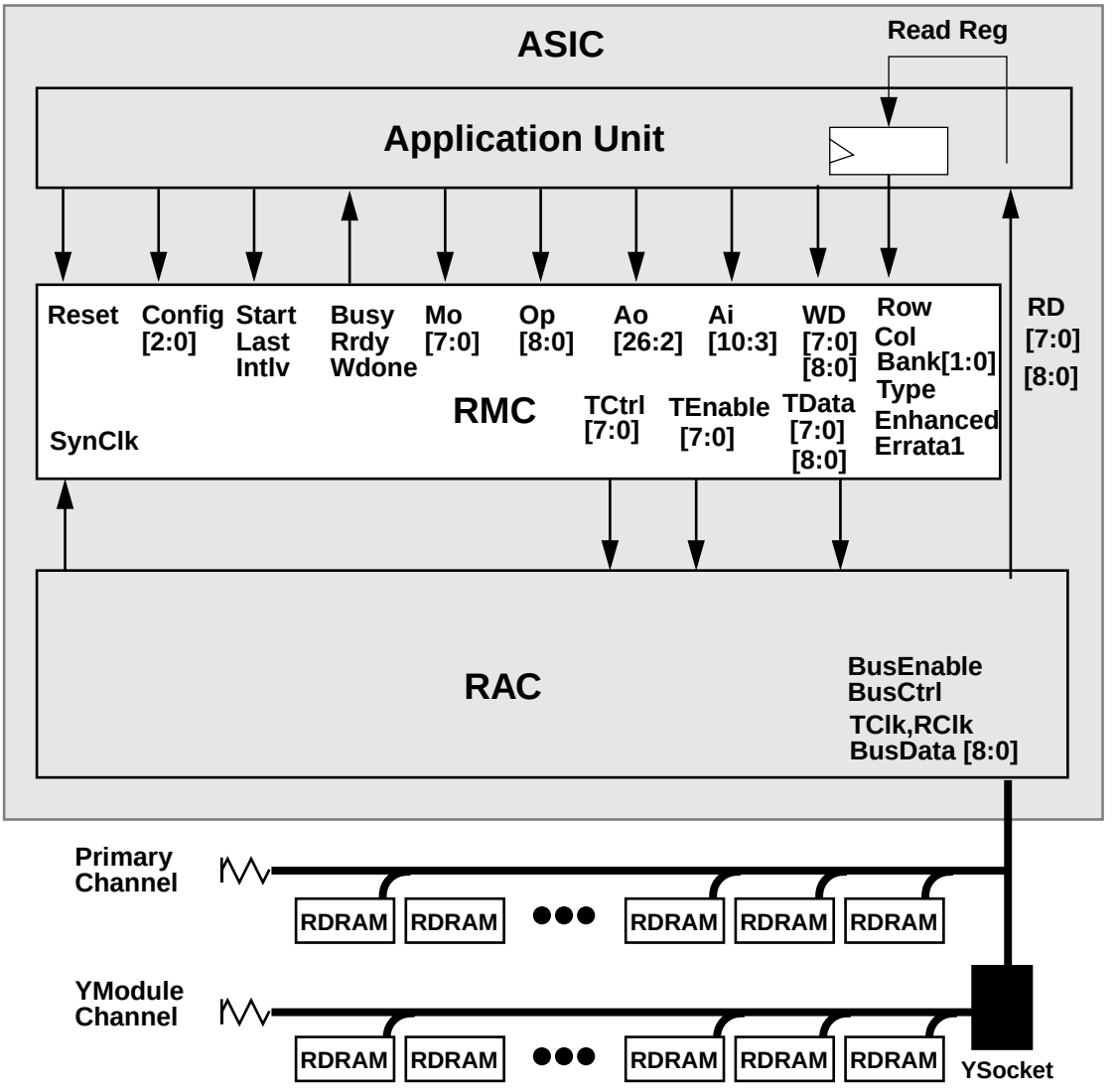
Table 1: RMC Input/Output Signals

Signal	In/Out	Description
Config[2:0]	input	This bus is normally left at a static value that is determined at initialization. It selects timing and byte masking options.
SynClk	Input	Clock from RAC for RMC and Application Unit
Reset	input	This signal forces most of the RMC's storage elements into known states.
Start	input	This signal is asserted for one cycle to indicate the start of a transaction. The Start, Last and Busy signals form a handshake between the RMC and the controller, permitting optimal packing of transactions.
Busy	output	This signal is asserted while the RMC is busy with the current transaction. It is deasserted when the RMC is ready to accept the next transaction.
Intlv	input	This signal is asserted to perform interleaved transactions.
Op[8:0]	input	This bus is valid in the cycle in which Start is asserted, and is ignored in all other cycles. It contains the opcode for the transaction that is starting.
Mo[7:0]	input	This bus is valid in the cycle in which Start is asserted, and is ignored in all other cycles. It contains the byte mask for the first octbyte of write data. This bus should be set to all 1's for normal write.
Ao[26:2]	input	This bus is valid in the cycle in which Start is asserted, and is ignored in all other cycles. It contains the base address for the transaction that is starting.
Ai[10:3]	input	This bus is valid in the cycle in which Start is asserted, and in each subsequent cycle until Last is asserted. It contains the column address for all data octbytes (except the first, which is contained in Ao[10:3]).
Last	input	This signal is asserted for one cycle to indicate the final column address (and data octbyte in the case of write transactions) has been transferred.

Table 1: RMC Input/Output Signals

Signal	In/Out	Description
WD[7:0][8:0]	input	This bus is valid in the cycle in which Start is asserted, and in each subsequent cycle until Last is asserted. It contains the octbytes (eight nine-bit bytes) of write data for write transactions.
Wdone	output	This signal is asserted for one cycle by the RMC for each octbyte of write data that has been retired.
Rrdy	output	This signal is asserted for one cycle by the RMC for each octbyte of read data that is returned. This signal serves as a read strobe.
Type	input	Indicates protocol type (0->Base, 1->Concurrent)
Row	input	row size(0 -> 2^9, 1 -> 2^{10})
Column	input	column size (0 -> 2^{10}, 1 -> 2^{11})
Bank	input	Bank sice(0 ->2^0, 1 -> 2^1, 2 -> 2^2)
Enhanced	input	Base subtype (0 -> B70, 1 -> B60)
Errata1	input	B70 $t_{POSTWRITEDELAY}$ errata (o -> errata, 1 -> no errata)

Figure 2: RMC Input/Output Signals

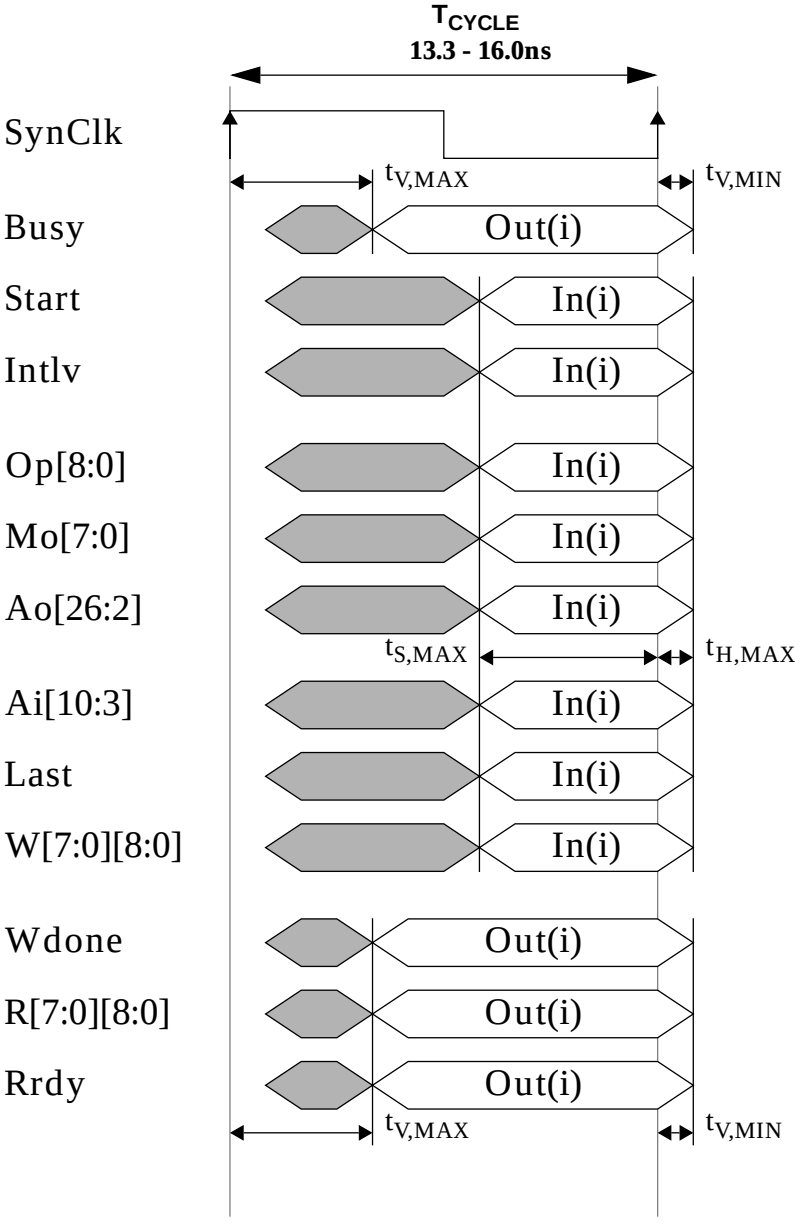


Note: The Row, Col, Bank[1:0], Type, Enhanced, and Errata1 information are read from the RDRAM to initialize the RMC.

2.2 RMC/Controller Interface – Signal Timing

The eight control wires and seven buses which form the RMC / controller interface are synchronous; they are sampled or driven by the rising edge of SynClk:

Figure 3: RMC Timing Windows



The following table format is equivalent to the timing diagram. This format will be used in the next sections to describe the RMC / Application Unit interface.

Table 2: Table Format for Intermediate Protocol

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD/ RD	Wdone /Rrdy
In(i)	Out(i)	In(i)	In(i)	In(i)	In(i)	In(i)	In(i)	In/Out(i)	Out(i)

Most input signals have minimum setup and hold parameters of 2.0ns and 0.5ns. Most output signals have maximum and minimum valid times of 3.0ns and 1.0ns. The Start and Last input signals can require more setup for some contiguration options. Likewise, the maximum valid time of RD[7:0][8:0] can also become larger with some configuration options.

Table 3: RMC Timing Parameters

Signal	In/Out	$T_{S,MAX}$	$T_{H,MAX}$	$T_{V,MAX}$	$T_{V,MIN}$
Config [2:0]	input	2.0ns	0.5ns	–	–
Reset	input	8.0ns	0.5ns	–	–
Start	input	4.0/8.0n(1)	0.5ns	–	–
Busy	output	–	–	5.0/12.0ns(1)	1.0ns
Intlv	input	4.0ns	0.5ns	–	–
Op [8:0]	input	4.0ns	0.5ns	–	–
Mo [7:0]	input	4.0ns	0.5ns	–	–
Ao [26:2]	input	4.0ns	0.5ns	–	–
Ai [10:3]	input	6.0/10.0ns(1)	0.5ns	–	–
Last	input	4.0/8.0ns(1)	0.5ns	–	–
WD [7:0][8:0]	input	2.0ns	0.5ns	–	–
Wdone	output	–	–	12.0ns	1.0ns
Rrdy	output	–	–	5.0ns	1.0ns
RD [7:0][8:0}	output	–	–	11.0/7.0ns(1)	1.0ns

(1) There are two select options (Z / A) which can be statically chosen for the RMC. Option Z sets BDSel, BCSel and BESel to 0010 and sets RDSel and RCSel to 1000. Option A sets BDSel, BCSel and BESel to 0001 and sets RDsel and RCsel to 0100. Notice that the timing numbers are referenced to the SynClk rising edge. The clock tree delay between the SynClk and the clock to the RMC registers are estimated to be 2-4 ns which is included in this timing table for Option A.

Option Z is the recommended timing option for the RMC. This minimizes the setup times of all inputs. It is also recommended that the receive data bus RD be immediately driven into a register clocked by the rising edge of RMC clock.

Chapter 3 Non-Interleaved Memory Transactions

3.1 Overview

This section describes non-interleaved memory read and write transactions. Non-interleaved means that the Busy signal is not released until the data transfer is (nearly) complete. This means that the transaction includes both a RAS/CAS access and a data transfer. A non-interleaved transaction is specified by deasserting the Intlv input on the first cycle (when Start is asserted). The following figure shows the timing of a four octbyte non-interleaved read or write transaction. The information associated with a single transaction [i] is highlighted so that it may be followed more easily.

Table 4: Non-Interleaved Four Octbyte Read/Write Transaction

Start	Busy	Intlv	Op Mo Ao	Ai Last WD	Wdone	Rrdy	RD	RAS/ CAS access	Channel transfer
...	...	...	...	...	...	...	...	...	
1 (a)	0	0	OMA[i]	ALW[i]	[i-1]	0	[i-1]	–	[i-1]
0 (b)	1	0	–	ALW[i]	0	0	[i-1]	–	[i-1]
0 (c)	1	0	–	ALW[i]	0	0	[i-1]	–	Req[i]
0 (d)	1	0	–	ALW[i]	0	0	–	RAS[i]	–
0	1	0	–	–	0	0	–	RAS[i]	–
0	1	0	–	–	0	0	–	RAS[i]	–
0	1	0	–	–	0	0	–	RAS[i]	–
0	1	0	–	–	0	R[i]	–	CAS[i]	–
0	1	0	–	–	W[i]	R[i]	–	CAS[i]	–
0	1	0	–	–	W[i]	R[i]	–	–	D[i]
0 (e)	1	0	–	–	W[i]	R[i]	D[i]	–	D[i]
1 (f)	0	0	[i+1]	[i+1]	W[i]	0	D[i]	–	D[i]
0	1	0	–	[i+1]	0	0	D[i]	–	D[i]
0	1	0	–	[i+1]	0	0	D[i]	–	Req[i+1]
0	1	0	–	[i+1]	0	0	–	[i+1]	[i+1]
...	...	...	...	...	...	...	...	...	

Transaction [i] begins in cycle (a), when Busy and Intlv are deasserted and Start is asserted. In cycles (a) through (d) (since this is a four octbyte transaction), the Op[8:0], Mo[7:0], Ao[26:2], Ai[10:3], Last and WD[7:0][8:0] input buses are driven with all the information needed to perform the read or write access. A detailed description of the values driven on these buses will follow in a later section.

While the input buses for transaction [i] are received by the RMC, parallel activity is started on the Channel and within the selected RDRAM. Cycles (b) and (c) are used by the RMC to wake up the RDRAM (standby to activate transition) and to transmit the request packet. A RAS / CAS access to the specified bank will start at the beginning of cycle (d). Six cycles (the worst case for Concurrent RDRAMs) is shown for the RAS / CAS access. This time depends upon the RDRAM state, the RDRAM type, and the transaction type. This dependency is described in the next section.

While the RAS / CAS access is completing, the RMC will signal when each data octbyte access is about to take place. This is done with the Wdone signal for write transactions, and with the Rrdy signal for read transactions. When the RAS / CAS access is complete, the four octbyte data transfer actually takes place (at the rate of one octbyte per cycle). In the case of a read transaction, read data is returned on the RD bus exactly three cycles after the Rrdy strobe.

While the last data octbytes are being transferred, the RMC will signal that it is ready to start the next transaction by deasserting Busy. In the example above, this occurs in cycle (f). The next transaction may be started at the end of the cycle (if it is ready) by asserting Start and driving the input buses. Note that there is a configuration option which causes the Busy to be deasserted a cycle earlier than is shown in the example (in cycle (e) in other words). This has some advantages and disadvantages, and is described in a subsequent section.

Although this example shows transactions with a four octbyte size, the RMC will support transactions with any mix of data sizes. The minimum size is one octbyte, and the maximum is size is determined by the size of the buffers in the PreDelay and ALWDelay modules (the default size is eight octbytes).

3.2 RAS/CAS Access Times

The RAS / CAS access times needed for the six read / write cases for Concurrent RDRAMs are given in the following table. The controller decides whether a Close command is used in each transaction. If a bank is Closed after a transaction, then the next transaction to that bank will result in the RowEmpty case. If a bank is not Closed after a transaction, then either a RowHit or RowMiss will result upon the next transaction to that bank, depending upon the base address Ao in the transaction. The RMC keeps track of the status of each RDRAM bank (whether it is closed or open, and if it is open, which row is sensed), and decides whether each transaction is a RowHit, RowEmpty, or RowMiss.

Note that in the case of RowHit, no RAS operations (precharge or sense) are needed. In the RowEmpty case, the bank was left precharged and only needs a sense operation (two

cycles). In the Rowmiss case, the bank has sensed the wrong row and needs to be precharged and resensed (four cycles). The CAS access for a read transaction requires two cycles and the CAS access for a write transaction requires (effectively) zero cycles. These times are automatically tracked by the RMC, and the proper delay is added.

Table 5: Effective RAS/CAS Access Time for Concurrent and Base RDRAMs

Type	Concurent RDRAM			Base RDRAM (B-60/B-70)		
Op	**RowHit**	**RowEmpty**	**RowMiss**	**RowHit**	**RowMiss Clean**	**RowMiss Dirty**
write	0 cycles	2 cycles	4 cycles	0/0 cycles	5/6 cycles	-/8 cycles
read	2 cycles	4 cycles	6 cycles	2/2 cycles	7/8 cycles	-/10 cycles

3.3 Non-Interleaved Memory Write Transactions

This section describes non-interleaved write transactions. A non-interleaved transaction is specified by deasserting the Intlv input on the first cycle. Using the table format described in a previous section, the RMC intermediate protocol can be specified. The first table shows a basic memory write transaction in which four octbytes are transferred.

Table 6: Basic Memory Write Transaction (4 octbytes with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	A1	0	W0	0
0	1 (c)	0	–	–	–	A2	0	W1	0
0	1 (c)	0	–	–	–	A3	0	W2	1 (e)
0	1 (c)	0	–	–	–	–	1 (d)	W3	1 (e)
0	1 (c)	0	–	–	–	–	0	–	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

Time flows vertically downward in this format, rather than left to right. In the first cycle the Busy output (a) is deasserted, so the transaction may be started by asserting the Start input (b). In this first cycle, the controller also drives the opcode WMem on the Op[8:0] input, the base address Adbr / A0 on the Ao[26:2] input, the second column address A1 on the Ai[10:3] bus, and the first octbyte of write data W0 on the W[7:0][8:0] input. The base address Adbr / A0 consists of a device field Ao[26:21], a bank field Ao[20], a row field Ao[19:11], and a column field Ao[10:3] (Ao[2] is only for registers). On the second and third cycles, the controller drives the second and third column addresses (A2 and A3) on the Ai[10:3] bus. On the second, third and fourth cycles, the controller drives the

second, third, and fourth octbytes of write data (W1, W2, and W3) on the W[7:0][8:0] bus. The Last input is asserted in the fourth cycle (d) to indicate that column address A3 and write data octbyte W3 represent the end of the transaction.

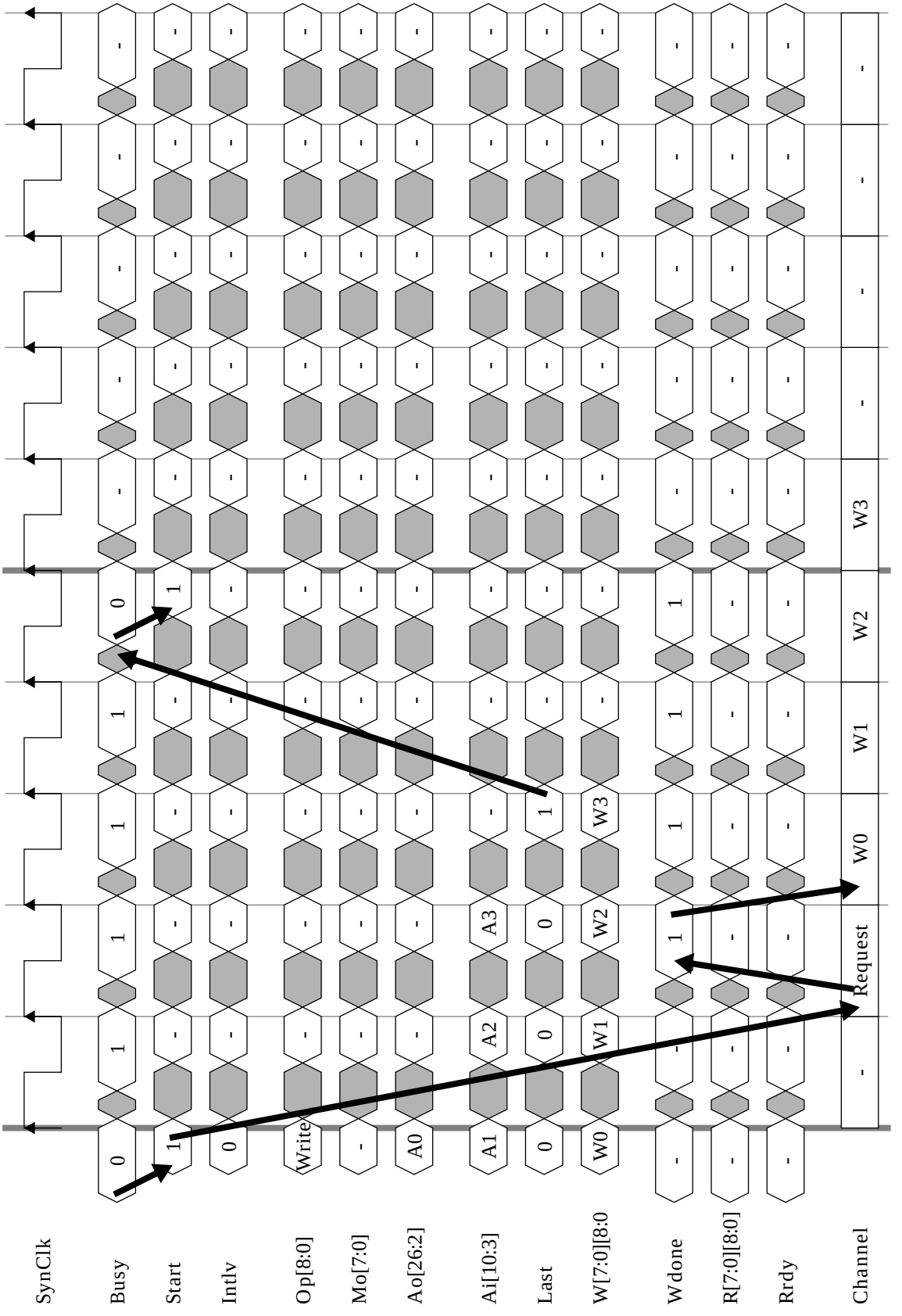
This sequence of information on the Start, Op[8:0], Mo[7:0], Ao[26:2], Ai[10:3], Last and WD[7:0][8:0] inputs is always the same, regardless of the state of the RMC and regardless of whether a RAS access (RowEmpty or RowMiss) needs to be performed. This means that the controller may implement a “fire-and-forget” approach when using Rambus technology; the RMC presents a simple intermediate protocol which hides the details of the technology and which performs low-level optimization that allows the controller to get the maximum performance from the Rambus subsystem.

The Busy output (c) is asserted in the second through fifth cycles to indicate that the RMC is processing the transaction. It is deasserted in cycle six (f) when the transaction has completed and the RMC is ready for the next transaction. The controller should assert Start in the next cycle (g) (if the next transaction is ready) in order to avoid any performance loss—the RMC will pack the two transactions together on the Channel as tightly as possible. This gives the controller approximately one half of a cycle from the time Busy is deasserted to when Start for the next transaction must be asserted. There is a configuration option which permits the Busy to be asserted one cycle earlier, possibly helping with critical paths in the controller (this is described in a later section).

The Wdone output (e) is asserted in the third through sixth cycles indicating when each octbyte of write data is written to the RDRAM.

This four octbyte non-interleaved write example is shown in a conventional timing diagram format in the following figure. The interface signals are shown, as well as an indication of the Channel activity. Note that this example assumes a RowHit state for the RDRAM (either Base or Concurrent RDRAM type).

Figure 4: NonInterleaved Write Timing - Four Octbyte RowHit Case



The previous example showed the timing of a memory write transaction in which there was a RowHit. This means that the row which is requested in the transaction address matches the row which is currently sensed in the RDRAM bank. The following example shows the same transaction with two cycles of delay. This delay is due to the fact that the memory bank to which the write transaction is directed has a RowEmpty state (the bank was precharged by a Close command in a previous transaction). This case is shown below. Note that the sequence of information on the Start, Op[8:0], Mo[7:0], Ao[26:2], Ai[10:3], Last, and WD[7:0][8:0] inputs is unchanged. The Wdone output (e) and Busy output deassertion (f) are each delayed by two cycles. The Start input (g) for the next transaction is also delayed two cycles.

Table 7: Basic Memory Write Transaction (4 octbytes with RowEmpty)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	A1	0	W0	0
0	1 (c)	0	–	–	–	A2	0	W1	0
0	1	0	–	–	–	A3	0	W2	0
0	1	0	–	–	–	–	1 (d)	W3	0
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

The case of a RowMiss is shown in the next figure (the sensed row in the bank and requested row on the Ao bus do not match). There are four extra cycles inserted by the RMC for this case. Note that the RMC keeps track of whether a transaction is a RowHit, RowEmpty, or RowMiss. However, the controller decides whether a Close command is used in each transaction. If a bank is Closed after a transaction, then the RowEmpty case will result. If a bank is not Closed after a transaction, then either a RowHit or RowMiss will result, depending upon the base address Ao in the transaction.

Table 8: Basic Memory Write Transaction (4 octbytes with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	A1	0	W0	0
0	1 (c)	0	–	–	–	A2	0	W1	0
0	1	0	–	–	–	A3	0	W2	0
0	1	0	–	–	–	–	1 (d)	W3	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

The shortest possible data transfer for a memory write transaction is one octbyte. This is shown in the following figure for a RowHit.

Table 9: Short Memory Write Transaction (1 octbyte with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	–	1 (d)	W0	0
0	1	0	–	–	–	–	0	–	0
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

A one octbyte write transaction with RowEmpty is shown in the next figure.

Table 10: Short Memory Write Transaction (1 octbyte with RowEmpty)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	–	1 (d)	W0	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

The one octbyte write transaction with RowMiss is shown in the next figure.

Table 11: Short Memory Write Transaction (1 octbyte with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	–	1 (d)	W0	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

The longest possible data transfer for a memory write transaction is eight octbytes (shown in the next figure). This is a limit imposed by the size of the data and address buffers in the PreDelay and ALWDelay modules. The buffers can be increased or decreased to accommodate other maximum sizes. This is discussed further in section on RMC Modifications and Extensions.

Table 12: Long Memory Write Transaction (8 octbytes with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	A1	0	W0	0
0	1 (c)	0	–	–	–	A2	0	W1	0
0	1	0	–	–	–	A3	0	W2	1 (e)
0	1	0	–	–	–	A4	0	W3	1 (e)
0	1	0	–	–	–	A5	0	W4	1 (e)
0	1	0	–	–	–	A6	0	W5	1 (e)
0	1	0	–	–	–	A7	0	W6	1 (e)
0	1	0	–	–	–	–	1 (d)	W7	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

The eight octbyte write transaction with a RowMiss is shown in the next figure. There are four delay cycles added by the RMC. The RowEmpty case is similar, but only two delay cycles are added. Note that every transaction length between the minimum (one octbyte) and the maximum (eight, as imposed by the PreDelay and ALWDelay modules) is permitted.

Table 13: Long Memory Write Transaction (8 octbytes with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	WMem	M0	Adbr/A0	A1	0	W0	0
0	1 (c)	0	–	–	–	A2	0	W1	0
0	1	0	–	–	–	A3	0	W2	0
0	1	0	–	–	–	A4	0	W3	0
0	1	0	–	–	–	A5	0	W4	0
0	1	0	–	–	–	A6	0	W5	0
0	1	0	–	–	–	A7	0	W6	1 (e)
0	1	0	–	–	–	–	1 (d)	W7	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

3.4 Non-interleaved Memory Read Transactions

This section describes non-interleaved read transactions. The protocol for memory read transactions is similar to what has already been presented for memory writes. The following table shows a basic memory read transaction in which four octbytes are transferred. Time flows vertically downward in this format, rather than left to right. In the first cycle the Busy output (a) is deasserted, so the transaction may be started by asserting the Start input (b). In this first cycle, the controller also drives the opcode RMem on the Op[8:0] input, the base address Adbr / A0 on the Ao[26:2] input, and the second column address A1 on the Ai[10:3] bus. On the second and third cycles, the controller drives the second and third column addresses (A2 and A3) on the Ai[10:3] bus. The Last input is asserted in the fourth cycle (d) to indicate that column address A3 represents the end of the transaction.

This sequence of information on the Start, Op[8:0], Mo[7:0], Ao[26:2], Ai[10:3], Last and WD[7:0][8:0] inputs is always the same, regardless of the state of the RMC and regardless of whether a RAS access (RowEmpty or RowMiss) needs to be performed.

The Busy output (c) is asserted in the second through seventh cycles to indicate that the RMC is processing the transaction. It is deasserted in cycle eight (f) when the transaction has completed and the RMC is ready for the next transaction. The controller can assert Start in the next cycle (g).

Table 14: Basic Memory Read Transaction (4 octbytes with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	A1	0	–	0
0	1 (c)	0	–	–	–	A2	0	–	0
0	1	0	–	–	–	A3	0	–	0
0	1	0	–	–	–	–	1 (d)	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	R0	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						R1	–
----------Next Transaction ----------								R2	–
----------Next Transaction ----------								R3	–
...	...	...	...	...	...	...	...	...	...

The Rrdy output (e) is asserted in the fourth through seventh cycles indicating when each octbyte of write data is written to the RDRAM. This output serves as a data strobe for the octbytes of read data in cycles seven through ten (Rrdy always appears three cycles ahead of the read data RD). Note that the read data overlaps into the next transaction. This will not cause a problem since the RD[7:0][8:0] bus is unidirectional.

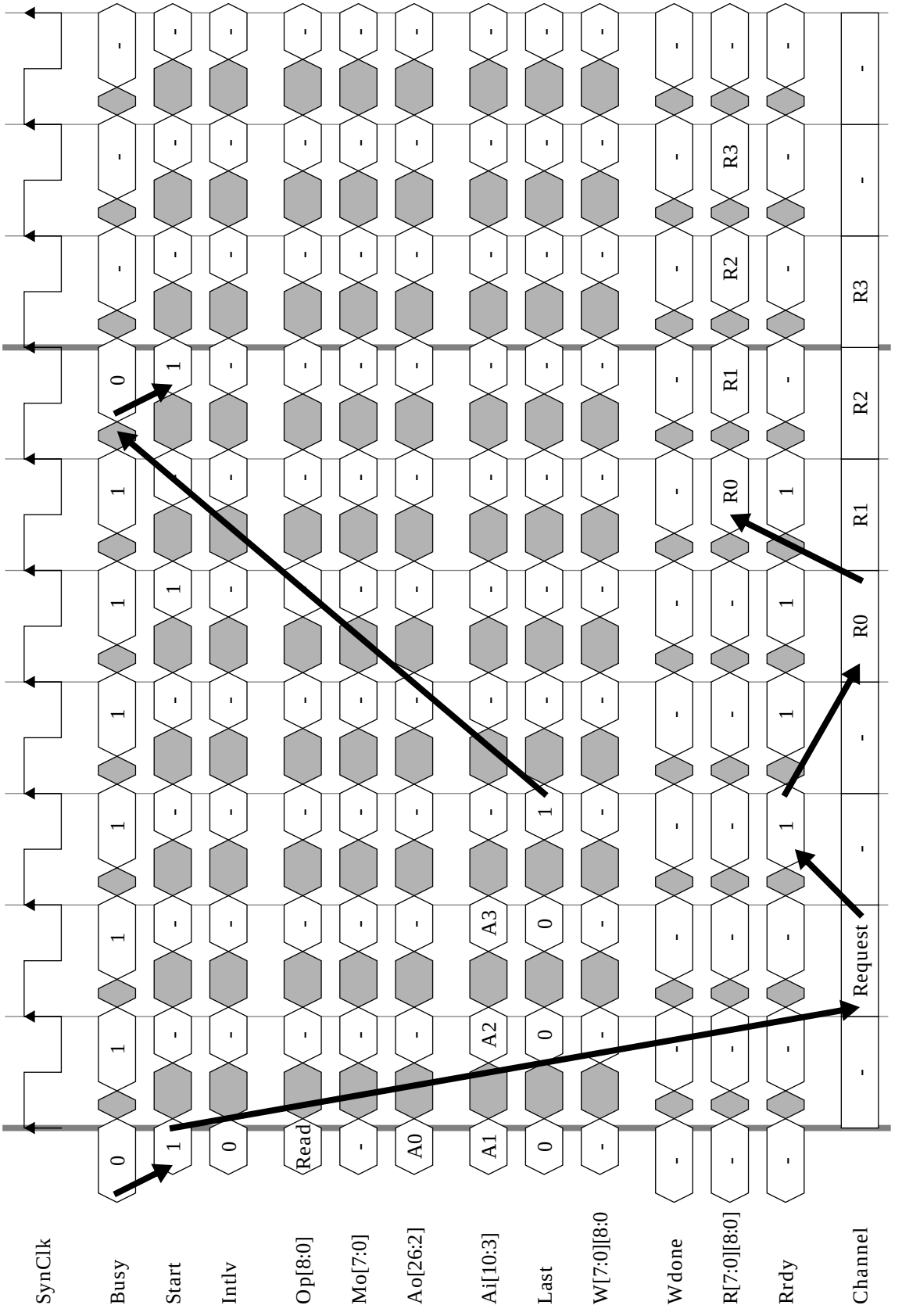
The read latency from the assertion of the Start input to the first RD octbyte becoming valid is (13ns + 5 cycles). The RMC contributes 4ns to that total. The breakdown of the access latency follows:

Table 15: RMC Read Latency

6ns	RMC setup time (with RAC transmit) for Start input
1.0cycle	RDRAM suspend-to-enable wake up
0.75cycle	Request packet on channel
2.25cycle	RDRAM read delay
1.0cycle	Data octbyte on channel
7ns	RMC valid time (with RAC receive) for RD outputs
13ns + 5 cycles	Total

This four octbyte non-interleaved read example is shown in a conventional timing diagram format in the following figure. The interface signals are shown, as well as an indication of the Channel activity. Note that this example assumes a RowHit state for the RDRAM (either Base or Concurrent RDRAM type).

Figure 5: Non-Interleaved Read Timing - Four Octbyte RowHit Case



The previous example showed the timing of a memory read transaction in which there was a RowHit. The following example shows the same transaction with two cycles of delay. This delay is due to the fact that the memory bank to which the read transaction is directed has a RowEmpty state (the bank was precharged by a Close command in a previous transaction). This case is shown below. Note that the sequence of information on the Start, Op[8:0], Mo[7:0], Ao[26:2], Ai[10:3], and Last inputs is unchanged. The Rrdy output (e) and Busy output deassertion (f) are each delayed by two cycles. The Start input (g) for the next transaction is also delayed two cycles.

Table 16: Basic Memory Read Transaction (4 octbytes with RowEmpty)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	A1	0	–	0
0	1 (c)	–	–	–	–	A2	0	–	0
0	1	0	–	–	–	A3	0	–	0
0	1	0	–	–	–	–	1	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	R0	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						R1	–
----------Next Transaction ----------								R2	–
----------Next Transaction ----------								R3	–
...	...	...	...	...	...	...	...	...	...

The case of a RowMiss is shown in the next table (the sensed row in the bank and requested row on the Ao bus do not match). There are four extra cycles inserted by the RMC for this case. Note that the RMC keeps track of whether a transaction is a RowHit, RowEmpty, or RowMiss. However, the controller decides whether a Close command is used in each transaction. If a bank is Closed after a transaction, then the RowEmpty case will result. If a bank is not Closed after a transaction, then either a RowHit or RowMiss will result, depending upon the base address Ao in the transaction.

Table 17: Basic Memory Read Transaction (4 octbytes with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	A1	0	–	0
0	1 (c)	0	–	–	–	A2	0	–	0
0	1	0	–	–	–	A3	0	–	0
0	1	0	–	–	–	–	1	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	0
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	–	0	–	1 (e)
0	1	0	–	–	–	-	0	-	1 (e)
0	1	0	-	-	-	-	0	R0	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						R1	-
----------Next Transaction ----------								R2	-
----------Next Transaction ----------								R3	-
...	...	...	...	...	...	...	...	...	...

The shortest possible data transfer for a memory read transaction is one octbyte. This is shown in the following table for a RowHit.

Table 18: Short Memory Read Transaction (1 octbyte with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	-	1 (d)	-	0
0	1 (c)	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						-	-
----------Next Transaction ----------								-	-
----------Next Transaction ----------								R0	-
...	...	...	...	...	...	...	...	...	...

The case of a RowEmpty with a one octbyte read is shown below - two delay cycles are inserted by the RMC.

Table 19: Short Memory Read Transaction (1 octbyte with RowEmpty)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	-	1 (d)	-	0
0	1 (c)	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						-	-
----------Next Transaction ----------								-	-
----------Next Transaction ----------								R0	-

The case of a RowMiss with a one octbyte read is shown below - four delay cycles are inserted by the RMC.

Table 20: Short Memory Read Transaction (1 octbyte with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	-	1 (d)	-	0
0	1 (c)	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						-	-
----------Next Transaction ----------								-	-
----------Next Transaction ----------								R0	-
...	...	...	...	...	...	...	...	...	...

The longest possible data transfer for a memory read transaction is eight octbytes (shown in the next figure). This is a limit imposed by the size of the address buffers in the PreDelay and ALWDelay modules. The buffers can be increased or decreased to accommodate other maximum sizes. This is discussed further in the section on RMC Modifications and Extensions.

Table 21: Long Memory Read Transaction (8 octbytes with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	A1	0	-	0
0	1 (c)	0	-	-	-	A2	0	-	0
0	1	0	-	-	-	A3	0	-	0
0	1	0	-	-	-	A4	0	-	1 (e)
0	1	0	-	-	-	A5	0	-	1 (e)
0	1	0	-	-	-	A6	0	-	1 (e)
0	1	0	-	-	-	A7	0	R0	1 (e)
0	1	0	-	-	-	-	1 (d)	R1	1 (e)
0	1	0	-	-	-	-	0	R2	1 (e)
0	1	0	-	-	-	-	0	R3	1 (e)
0	1	0	-	-	-	-	0	R4	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						R5	-
----------Next Transaction ----------								R6	-
----------Next Transaction ----------								R7	-
...	...	...	...	...	...	...	...	...	...

The eight octbyte read transaction with a RowMiss is shown in the next figure. There are four delay cycles added by the RMC. The RowEmpty case is similar, but only two delay cycles are added. Note that every transaction length between the minimum (one octbyte) and the maximum (eight, as imposed by the PreDelay and ALWDelay modules) is permitted.

Table 22: Long Memory Read Transaction (8 octbytes with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RMem	M0	Adbr/A0	A1	0	-	0
0	1 (c)	0	-	-	-	A2	0	-	0
0	1	0	-	-	-	A3	0	-	0
0	1	0	-	-	-	A4	0	-	0
0	1	0	-	-	-	A5	0	-	0
0	1	0	-	-	-	A6	0	-	0
0	1	0	-	-	-	A7	0	-	0
0	1	0	-	-	-	-	1 (d)	-	1 (e)
0	1	0	-	-	-	-	0	-	1 (e)
0	1	0	-	-	-	-	0	-	1 (e)
0	1	0	-	-	-	-	0	R0	1 (e)
0	1	0	-	-	-	-	0	R1	1 (e)
0	1	0	-	-	-	-	0	R2	1 (e)
0	1	0	-	-	-	-	0	R3	1 (e)
0	1	0	-	-	-	-	0	R4	1 (e)
1 (g)	0 (f)	----------Next Transaction ----------						R5	-
----------Next Transaction ----------								R6	-
----------Next Transaction ----------								R7	-
...	...	...	...	...	...	...	...	...	...

Chapter 4 Interleaved Memory Transactions

4.1 Overview

An interleaved transaction is one in which the information on the input buses (Op[8:0], Mo[7:0], Ao[26:2], Ai[10:3], Last and WD[7:0][8:0]) is accepted and stored by the RMC, but only a RAS/CAS access is performed in the current transaction. The data transfer on the Channel does not happen until the next transaction. Thus, the current transaction overlaps the previous transaction and the next transaction. The following table shows this overlapping for a steady state situation with four octbyte interleaved read and write transactions (read and write may be mixed in any way).

Table 23: Steady State - Interleaved Four Octbyte Read/Write Transactions

Start	Busy	Intlv	Op Mo Ao	Ai Last WD	Wdone	Rrdy	RD	RAS/ CAS access	Channel transfer
...	...	...	...	...	...	...	...	...	
1 (a)	0	1	OMA[i]	ALW[i]	[i-2]	-	[i-2]	[i-1]	[i-2]
0 (b)	1	0	-	ALW[i]	0	[i-1]	[i-2]	[i-1]	[i-2]
0 (c)	1	0	-	ALW[i]	[i-1]	[i-1]	[i-2]	[i-1]	Req[i]
0 (d)	1	0	-	ALW[i]	[i-1]	[i-1]	-	RAS[i]	[i-1]
0 (e)	1	0	-	-	[i-1]	[i-1]	[i-1]	RAS[i]	[i-1]
1 (f)	0	1	[i+1]	[i+1]	[i-1]	0	[i-1]	RAS[i]	[i-1]
0 (g)	1	0	-	[i+1]	0	R[i]	[i-1]	CAS[i]	[i-1]
0 (h)	1	0	-	[i+1]	W[i]	R[i]	[i-1]	CAS[i]	Req[i+1]
0 (i)	1	0	-	[i+1]	W[i]	R[i]	-	[i+1]	D[i]
0 (j)	1	0	-	-	W[i]	R[i]	D[i]	[i+1]	D[i]
1 (k)	0	1	[i+2]	[i+2]	W[i]	-	D[i]	[i+1]	D[i]
0 (l)	1	0	-	[i+2]	0	[i+1]	D[i]	[i+1]	D[i]
0 (m)	1	0	-	[i+2]	[i+1]	[i+1]	D[i]	[i+1]	Req[i+2]
0	1	0	-	[i+2]	[i+1]	[i+1]	-	[i+2]	[i+1]
0	1	0	-	-	[i+1]	[i+1]	[i+1]	[i+2]	[i+1]
...	...	...	...	...	...	...	...	...	

The information associated with a single transaction [i] is highlighted so that it may be followed more easily. The sequence is also divided into five-cycle blocks to emphasize the identical timing slots for each transaction.

Transaction [i] enters the memory pipeline in cycle (a), when Busy is deasserted and Start and Intlv are both asserted. The Op[8:0], Mo[7:0], Ao[26:2], Ai[10:3], Last and WD[7:0][8:0] input buses are also driven with the same information sequence previously described for non-interleaved transactions. The table also shows (roughly) the RAS/CAS activity in the RDRAM (precharge and sense for RowMiss and sense for RowEmpty) and the activity on the Channel (request and data transfers).

Transaction [i] is received by the RMC at the end of cycle (a). Cycles (b) and (c) are used by the RMC to wake up the RDRAM (standby to activate transition) and to transmit the request packet. A RAS/CAS access to the specified bank will start at the beginning of cycle (d). With a four octbyte data transfer in transaction [i-1], five cycles are available for the RAS/CAS access by transaction [i] as long as there is no bank conflict (the bank conflict case will be discussed in detail later in this section).

Although this example shows transactions with the same four octbyte size, the RMC will support interleaved transactions with any mix of data sizes. The minimum size is one octbyte, and the maximum is size is determined by the size of the buffers in the PreDelay and ALWDelay modules (the default size is eight octbytes).

The data transfer for transaction [i] is scheduled on the Channel immediately after the transaction [i+1] is received in cycle (f) (when Start is asserted for the next transaction). The D[i] octbytes of data will be transferred on the Channel immediately after the Req[i+1] request packet.

In the case of a write transaction, the RMC indicates when the data is being transferred by asserting Wdone. This occurs in cycles (h) through (k) for transaction [i]. In the case of a read transaction, the RMC indicates when the data is being transferred by asserting Rrdy. This occurs in cycles (g) through (j) for transaction [i]. The Rrdy strobe precedes the read data on the RD bus by exactly three cycles, meaning the read data for transaction [i] is valid in cycles (j) through (m). Transaction [i] is completed when the last Wdone is asserted in cycle (k) or the last RD octbyte is driven in cycle (m).

4.2 RAS/CAS Access Times

The RAS/CAS access times needed for the six read/write cases are given in the following table. Note that in the case of RowHit, no RAS operations (precharge or sense) are needed. However, a read RowHit needs two cycles for a CAS access, and a write RowHit needs (effectively) zero cycles for a CAS access. These times have been included in the table.

These times are automatically tracked by the RMC. Delay is added if the data transfer for transaction [i+1] is shorter than the RAS/CAS access for transaction [i]. It can be seen that with four octbyte data transfers by transaction [i-1] (giving five cycles for a RAS/CAS access), that there will be no extra delay added by the RMC except for the case of a read RowMiss. For shorter data transfers in transaction [i-1], more delay will be added by the RMC..

Table 24: Effective RAS/CAS Access Time for Concurrent and Base RDRAMs

Type	**Concurent RDRAM**			**Base RDRAM (B-60/B-70)**		
Op	**RowHit**	**RowEmpty**	**RowMiss**	**RowHit**	**RowMiss Clean**	**RowMiss Dirty**
write	0 cycles	2 cycles	4 cycles	0/0 cycles	5/6 cycles	-/8 cycles
read	2 cycles	4 cycles	6 cycles	2/2 cycles	7/8 cycles	-/10 cycles

4.3 Bank Conflict

If transaction [i] is a RowEmpty or RowMiss and is directed to the same bank as transaction [i-1], then the RDRAM cannot interleave the transactions (because of resource conflicts within the core). In this case, the RMC will automatically delay Req[i] to the next transaction slot. This will cause approximately one transaction's worth of Channel bandwidth to be lost. This is shown in the following table.

The timing slots labeled with an "x" have been inserted as a result of the bank conflict. Note that after transaction [i] is started in cycle 1, the RMC keeps Busy asserted from cycle 2 through cycle 10. Transaction [i+1] is held off during this time, and is not able to start until cycle 11. By adding five cycles of delay, the RMC ensures that the data transfer for transaction [i-1] and the RAS/CAS access for transaction [i] do not overlap.

Note also that the RAS/CAS access for transaction [i] is shown as five cycles. In fact, this time will be one of the four RowEmpty/RowMiss entries of the table in the previous section (remember that a bank conflict will not occur if transaction [i] is a RowHit). This means that the actual RAS/CAS access time will be 2, 4, or 6 cycles.

Table 25: Bank Conflict - Interleaved Four Octbyte Read/Write Transactions

Start	Busy	Intlv	Op Mo Ao	Ai Last WD	Wdone	Rrdy	RD	RAS/ CAS access	Channel transfer
...	...	...	...	...	...	...	...	...	
1	0	1	OMA[i]	ALW[i]	[i-2]	-	[i-2]	[i-1]	[i-2]
0	1	0	-	ALW[i]	0	[i-1]	[i-2]	[i-1]	[i-2]
0	1	0	-	ALW[i]	[i-1]	[i-1]	[i-2]	[i-1]	x
0	1	0	-	ALW[i]	[i-1]	[i-1]	-	x	[i-1]
0	1	0	x	x	[i-1]	[i-1]	[i-1]	x	[i-1]
0	1	0	x	x	[i-1]	x	[i-1]	x	[i-1]
0	1	0	x	x	x	x	[i-1]	x	[i-1]
0	1	0	x	x	x	x	x	x	Req [i]
0	1	0	x	x	x	x	x	RAS[i]	x
0	1	0	x	x	x	x	x	RAS[i]	x
1	0	1	[i+1]	[i+1]	x	-	x	RAS[i]	x
0	1	0	-	[i+1]	0	R[i]	x	CAS[i]	x
0	1	0	-	[i+1]	W[i]	R[i]	x	CAS[i]	Req[i+1]
0	1	0	-	[i+1]	W[i]	R[i]	-	[i+1]	D[i]
0	1	0	-	-	W[i]	R[i]	D[i]	[i+1]	D[i]
1	0	1	[i+2]	[i+2]	W[i]	-	D[i]	[i+1]	D[i]
0	1	0	-	[i+2]	0	[i+1]	D[i]	[i+1]	D[i]
0	1	0	-	[i+2]	[i+1]	[i+1]	D[i]	[i+1]	Req[i+2]
0	1	0	-	[i+2]	[i+1]	[i+1]	-	[i+2]	[i+1]
0	1	0	-	-	[i+1]	[i+1]	[i+1]	[i+2]	[i+1]
...	...	...	...	...	...	...	...	...	

4.4 Empty/Fill Interleave Pipeline

If another transaction is not available, the data from transaction [i] can be transferred by deasserting Intlv and asserting Start in the cycle in which the next transaction would have begun. In the following example, this is cycle (a).

The timing for transaction [i] is unchanged, but there will be no request transmitted for transaction [i+1]. The timing slots which are left vacant (because there is no transaction available in cycle (a)) are filled with a “y” to make them more visible. Note that the Busy signal is asserted from cycle (b) through (e) (just as it would if a transaction had started in cycle (a)) because the RMC is busy transferring data on the Channel.

When Busy is deasserted in cycle (f), another transaction may be started (if one is available). It may be an interleaved or non-interleaved transaction. If it is interleaved (as the example shows), the Intl signal is asserted along with the Start signal in cycle (f). The memory pipeline is refilled (again with “y” indicating the vacant timing slots).

Table 26: Empty/Fill Pipeline - Interleaved Four Octbyte Read/Write Transactions

Start	Busy	Intlv	Op Mo Ao	Ai Last WD	Wdone	Rrdy	RD	RAS/ CAS access	Channel transfer
...	...	...	...	...	...	...	...	...	
1	0	1	OMA[i]	ALW[i]	[i-2]	-	[i-2]	[i-1]	[i-2]
0	1	0	-	ALW[i]	0	[i-1]	[i-2]	[i-1]	[i-2]
0	1	0	-	ALW[i]	[i-1]	[i-1]	[i-2]	[i-1]	Req[i]
0	1	0	-	ALW[i]	[i-1]	[i-1]	-	RAS[i]	[i-1]
0	1	0	-	-	[i-1]	[i-1]	[i-1]	RAS[i]	[i-1]
1 (a)	0	0	y	y	[i-1]	-	[i-1]	RAS[i]	[i-1]
0 (b)	1	0	-	y	0	R[i]	[i-1]	CAS[i]	[i-1]
0 (c)	1	0	-	y	W[i]	R[i]	[i-1]	CAS[i]	y
0 (d)	1	0	-	y	W[i]	R[i]	-	y	D[i]
0 (e)	1	0	-	-	W[i]	R[i]	D[i]	y	D[i]
1 (f)	0	1	[i+1]	[i+1]	W[i]	-	D[i]	y	D[i]
0	1	0	-	[i+1]	y	y	D[i]	y	D[i]]
0	1	0	-	[i+1]	y	y	D[i]	y	Req[i+1]
0	1	0	-	[i+1]	y	y	y	[i+1]	y
0	1	0	-	-	y	y	y	[i+1]	y
1	0	1	[i+2]	[i+2]	y	y	y	[i+1]	y
0	1	0	-	[i+2]	0	[i+1]	y	[i+1]	y
0	1	0	-	[i+2]	[i+1]	[i+1]	y	[i+1]	Req[i+2]
0	1	0	-	[i+2]	[i+1]	[i+1]	-	[i+2]	[i+1

4.5 Interleaved Memory Write Transactions

This section includes examples of interleaved write transactions. In all cases, two interleaved write transactions are presented to an idle RMC. This will show all three states of the RMC during an interleaved sequence (fill, continue, and drain). Longer interleaved sequences simply repeat the continue state. The different sequences include the RowHit, RowEmpty, RowMiss cases, and four, eight, and one octbyte transfer lengths.

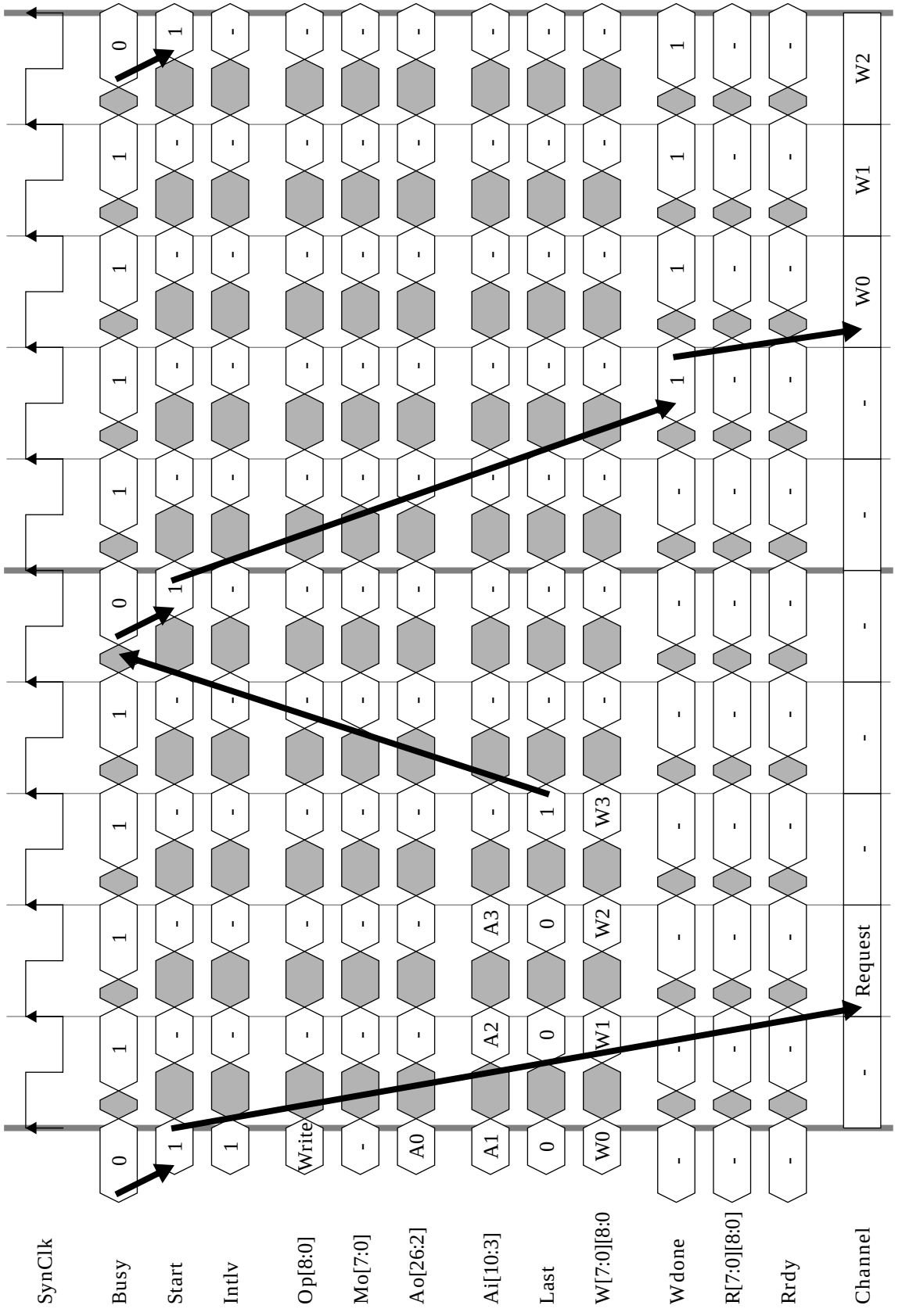
The first table below shows the RowHit, RowEmpty and RowMiss cases with two four-octbyte transactions. RAS/CAS accesses for the two transactions are performed in cycles (a) through (d) and in cycles (e) through (i). The Channel transfers for the two transactions are performed in cycles (e) through (i) and in cycles (j) through (n). With the four octbyte transfer size, the RAS/CAS access times are completely overlapped.

Table 27: Interleaved Write Transaction (4 octbytes with RowHit/Empty/Miss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	WMem	M0	Adbr/A0	A1	0	W0	0
0 (b)	1	0	-	-	-	A2	0	W1	0
0 (c)	1	0	-	-	-	A3	0	W2	0
0 (d)	1	0	-	-	-	-	1	W3	0
1 (e)	0	1	WMem	M0	Adbr/A4	A5	0	W4	0
0 (f)	1	0	-	-	-	A6	0	W5	0
0 (g)	1	0	-	-	-	A7	0	W6	1
0 (h)	1	0	-	-	-	-	1	W7	1
0 (i)	1	0	-	-	-	-	0	-	1
1 (j)	0	0	-	-	-	-	0	-	1
0 (k)	1	0	-	-	-	-	0	-	0
0 (l)	1	0	-	-	-	-	0	-	1
0 (m)	1	0	-	-	-	-	0	-	1
0 (n)	1	0	-	-	-	-	0	-	1
1	0	---------- Next Transaction ----------							1
...	...	...	...	...	...	...	...	...	...

The above example is shown in a timing diagram format in the following figure:

Figure 6: Interleaved Write Timing - Four Octbytes with RowHit/Empty/Miss



The next sequence is below, and shows two eight-octbyte transactions.

Table 28: Interleaved Write Transaction (8 octbytes with RowHit/RowEmpty/RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	WMem	M0	Adbr/A0	A1	0	W0	0
0 (b)	1	0	-	-	-	A2	0	W1	0
0 (c)	1	0	-	-	-	A3	0	W2	0
0 (d)	1	0	-	-	-	A4	0	W3	0
0 (e)	1	0	-	-	-	A5	0	W4	0
0 (f)	1	0	-	-	-	A6	0	W5	0
0 (g)	1	0	-	-	-	A7	0	W6	0
0 (h)	1	0	-	-	-	-	1	W7	0
1 (i)	0	1	WMem	M0	Adbr/A8	A9	0	W8	0
0 (j)	1	0	-	-	-	A10	0	W9	0
0 (k)	1	0	-	-	-	A11	0	W10	1
0 (l)	1	0	-	-	-	A12	0	W11	1
0 (m)	1	0	-	-	-	A13	0	W12	1
0 (n)	1	0	-	-	-	A14	0	W13	1
0 (o)	1	0	-	-	-	A15	0	W14	1
0 (p)	1	0	-	-	-	-	1	W15	1
0 (q)	1	0	-	-	-	-	0	-	1
1 (r)	0	0	-	-	-	-	0	-	1
0 (s)	1	0	-	-	-	-	0	-	0
0 (t)	1	0	-	-	-	-	0	-	1
0 (u)	1	0	-	-	-	-	0	-	1
0 (v)	1	0	-	-	-	-	0	-	1
0 (w)	1	0	-	-	-	-	0	-	1
0 (x)	1	0	-	-	-	-	0	-	1
0 (y)	1	0	-	-	-	-	0	-	1
0 (z)	1	0	-	-	-	-	0	-	1
1	0	----------Next Transaction ----------							1

In cycles (a) through (h), the input values are fed into the RMC, with the asserted Intlv input in (a) indicating that this will be an interleaved transaction. The RAS/CAS access for this transaction is started as the input values are stored. When Last is asserted in cycle (h), the RMC immediately deasserts Busy in cycle (i), permitting the second transaction to be fed in

During cycles (i) through (q), the RAS/CAS access of the second transaction takes place. Simultaneously, the data transfer for the first transaction occurs, as indicated when Wdone is asserted. Busy is deasserted in cycle (r) when this transfer is complete.

The controller decides that there are no more transactions available for interleaving, and so asserts Start while simultaneously deasserting Intlv in cycle (r). This instructs the RMC to drain the interleave pipeline, meaning that the data transfer for the second transaction is carried out, but there will not be a RAS/CAS access for a third transaction. Busy is deasserted after this transfer, indicating the RMC idle and ready when another transaction becomes available.

It should be noted that the RowHit/RowEmpty/RowMiss cases are identical for a memory write when the transfer length of the previous transaction is three octbytes or greater. This is because the RAS/CAS access times for these cases are 0/2/4 cycles. These times are short enough to be completely shadowed by the previous data transfer.

If the controller had more than two transactions to interleave, then cycles (i) through (q) would be repeated one or more times. If the controller had chosen to interleave in cycle (a), and then discovered that only a single transaction was available, then cycles (i) through (q) would be dropped in the above diagram. This fill/drain sequence of cycles (a) through (h) followed by cycles (r) through (z) would not be as efficient in Channel bandwidth as a single non-interleaved equivalent, since the RMC would have to hold off the transfer on the Channel until Start was asserted. If the transfer size is smaller, the RAS/CAS time will limit the speed of the access, and a single interleaved transaction (consisting of a fill transaction and an drain transaction) will take the same number of cycles as the equivalent non-interleaved transaction.

The shortest transfer length permitted for interleaved transactions is one octbyte. The RowHit case for one octbyte transfers is shown in the following diagram.

Table 29: Interleaved Write Transaction (1 octbytes with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	WMem	M0	Adbr/A0	-	1	W0	0
0 (b)	1	0	-	-	-	-	0	-	0
1 (d)	0	1	WMem	M0	Adbr/A1	-	1	W1	0
0 (e)	1	0	-	-	-	-	0	-	0
1 (g)	0	0	-	-	-	-	0	-	1
0 (h)	1	0	-	-	-	-	0	-	0
1	0	---------- Next Transaction ----------							1
...	...	...	...	...	...	...	...	...	...

The RowEmpty case for one octbyte transfers is shown in the following diagram. A single delay cycle has been inserted, relative to the RowHit case.

Table 30: Interleaved Write Transaction (1 octbytes with RowEmpty)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	WMem	M0	Adbr/A0	-	1	W0	0
0 (b)	1	0	-	-	-	-	0	-	0
0 (c)	1	0	-	-	-	-	0	-	0
1 (d)	0	1	WMem	M0	Adbr/A1	-	1	W1	0
0 (e)	1	0	-	-	-	-	0	-	0
0 (f)	1	0	-	-	-	-	0	-	1
1 (g)	0	0	-	-	-	-	0	-	0
0 (h)	1	0	-	-	-	-	0	-	0
1	0	---------- Next Transaction ----------							1
...	...	...	...	...	...	...	...	...	...

The RowMiss case for one octbyte transfers is shown in the following diagram. Three delay cycles have been inserted, relative to the RowHit case.

Table 31: Interleaved Write Transaction (1 octbytes with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	WMem	M0	Adbr/A0	-	1	W0	0
0 (b)	1	0	-	-	-	-	0	-	0
0 (c)	1	0	-	-	-	-	0	-	0
0 (d)	1	0	-	-	-	-	0	-	0
0 (e)	1	0	-	-	-	-	0	-	0
1 (f)	0	1	WMem	M0	Adbr/A1	-	1	W1	0
0 (g)	1	0	-	-	-	-	0	-	0
0 (h)	1	0	-	-	-	-	0	-	1
0 (i)	1	0	-	-	-	-	0	-	0
0 (j)	1	0	-	-	-	-	0	-	0
1 (k)	0	0	-	-	-	-	0	-	0
0 (l)	1	0	-	-	-	-	0	-	0
1	0	---------- Next Transaction ----------							1
...	...	...	...	...	...	...	...	...	...

These examples have shown sequences of identical length write transactions with the same RowHit / RowEmpty / RowMiss case. The RMC will accept interleaved transactions with any mix of read / write accesses, with any mix of transfer length (subject to buffer size restrictions), and with any mix of RowHit, RowEmpty, and RowMiss cases. The RMC will keep track of the transaction mixture, and the Busy handshake signal will assure that transactions are properly spaced on the Channel.

4.6 Interleaved Memory Read Transactions

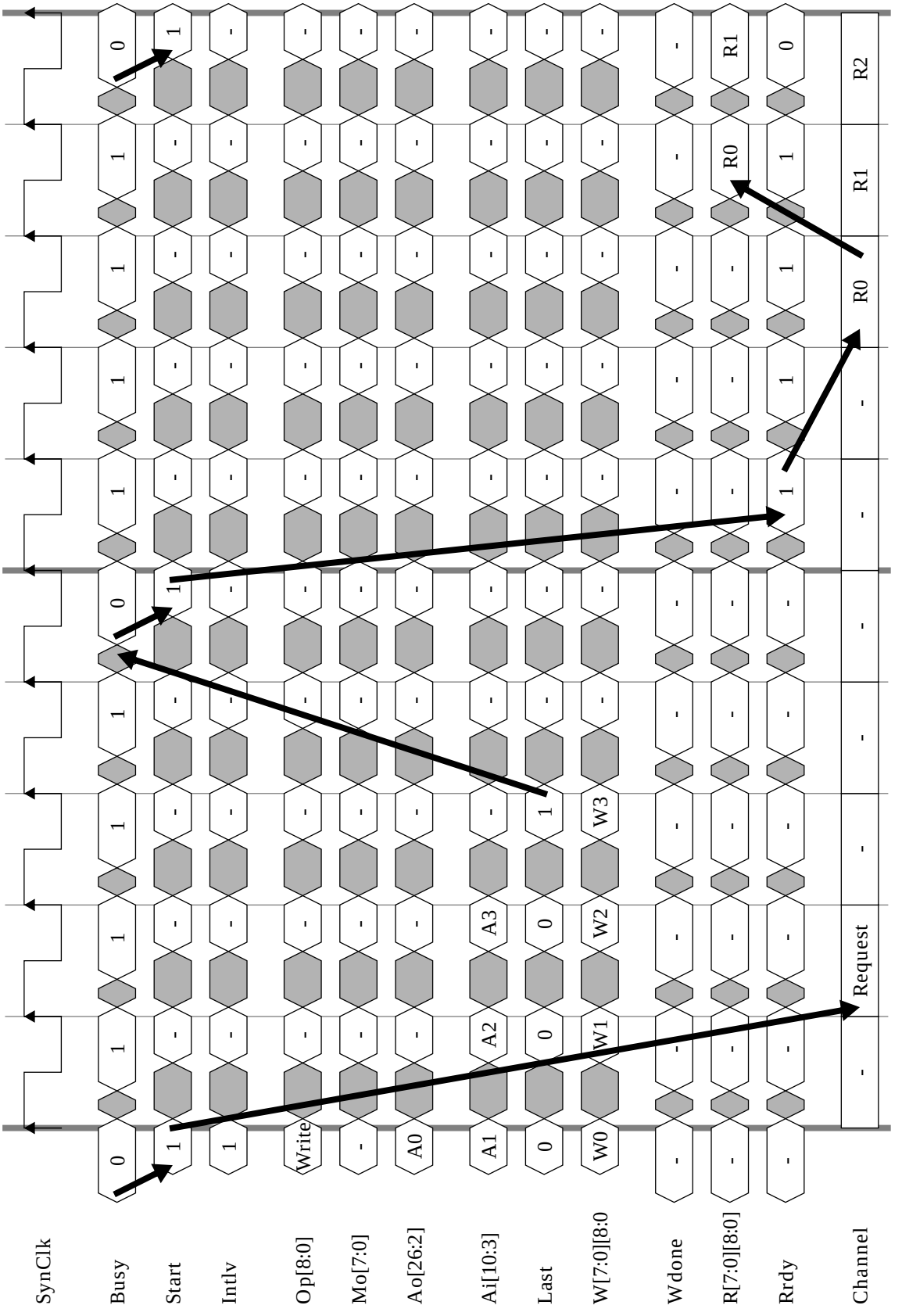
This section shows examples of interleaved read transactions. In all cases, two interleaved read transactions are presented to an idle RMC. All three states (RowHit, RowEmpty, RowMiss) of the RMC during an interleaved sequence (fill, continue, and drain) will be presented. Longer interleaved sequences simply repeat the continue state. The different sequences include four, eight, and one octbyte transfer lengths.

The next diagram shows the RowHit and RowEmpty cases with a four octbyte transfer size. RAS/CAS accesses for the two transactions are performed in cycles (a) through (d) and in cycles (e) through (i). The Channel transfers for the two transactions are performed in cycles (e) through (i) and in cycles (j) through (n). With the four octbyte transfer size, the RAS/CAS access times are completely overlapped.

Table 32: Interleaved Read Transaction (4 octbytes with RowHit/RowEmpty)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	RMem	M0	Adbr/A0	A1	0	-	0
0 (b)	1	0	-	-	-	A2	0	-	0
0 (c)	1	0	-	-	-	A3	0	-	0
0 (d)	1	0	-	-	-	-	1	-	0
1 (e)	0	1	RMem	M0	Adbr/A4	A5	0	-	0
0 (f)	1	0	-	-	-	A6	0	-	1
0 (g)	1	0	-	-	-	A7	0	-	1
0 (h)	1	0	-	-	-	-	1	-	1
0 (i)	1	0	-	-	-	-	0	R0	1
1 (j)	0	0	-	-	-	-	0	R1	0
0 (k)	1	0	-	-	-	-	0	R2	1
0 (l)	1	0	-	-	-	-	0	R3	1
0 (m)	1	0	-	-	-	-	0	-	1
0 (n)	1	0	-	-	-	-	0	R4	1
1	0	---------- Next Transaction ----------						R5	-
---------- Next Transaction ----------								R6	-
---------- Next Transaction ----------								R7	-
...	...	...	...	...	...	...	...	...	...

Figure 7: Interleaved Read Timing



In the case of a RowMiss, the RAS/CAS time will not be fully overlapped, and the RMC will need to insert two delay cycles. This is cycles (e) and (f) in the following diagram.

Table 33: Interleaved Read Transaction (4 octbytes with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	RMem	M0	Adbr/A0	A1	0	-	0
0 (b)	1	0	-	-	-	A2	0	-	0
0 (c)	1	0	-	-	-	A3	0	-	0
0 (d)	1	0	-	-	-	-	1	-	0
0 (e)	1	0	-	-	-	-	0	-	0
0 (f)	1	0	-	-	-	-	0	-	0
1 (g)	0	1	RMem	M0	Adbr/A4	A5	0	-	0
0 (h)	1	0	-	-	-	A6	0	-	1
0 (i)	1	0	-	-	-	A7	0	-	1
0 (j)	1	0	-	-	-	-	1	-	1
0 (k)	1	0	-	-	-	-	0	R0	1
0 (l)	1	0	-	-	-	-	0	R1	0
1 (m)	0	0	-	-	-	-	0	R2	0
0 (n)	1	0	-	-	-	-	0	R3	1
0 (o)	1	0	-	-	-	-	0	-	1
0 (p)	1	0	-	-	-	-	0	-	1
0 (q)	1	0	-	-	-	-	0	R4	1
1	0	---------- Next Transaction ----------						R5	-
---------- Next Transaction ----------								R6	-
---------- Next Transaction ----------								R7	-
...	...	...	...	...	...	...	...	...	...

The first sequence is below, and uses an eight octbyte transfer size. In cycles (a) through (h), the input values are fed into the RMC, with the asserted Intlv input in (a) indicating that this will be an interleaved transaction. The RAS/CAS access for this transaction is started as the input values are stored. When Last is asserted in cycle (h), the RMC immediately deasserts Busy in cycle (i), permitting the second transaction to be fed in

.

Table 34: Interleaved Read Transaction(8 octbytes with RowHit/Empty/Miss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
1 (a)	0	1	RMem	M0	Adbr/A0	A1	0	-	0
0 (b)	1	0	-	-	-	A2	0	-	0
0 (c)	1	0	-	-	-	A3	0	-	0
0 (d)	1	0	-	-	-	A4	0	-	0
0 (e)	1	0	-	-	-	A5	0	-	0
0 (f)	1	0	-	-	-	A6	0	-	0
0 (g)	1	0	-	-	-	A7	0	-	0
0 (h)	1	0	-	-	-	-	1	-	0
1 (i)	0	1	RMem	M0	Adbr/A8	A9	0	-	0
0 (j)	1	0	–	-	-	A10	0	-	1
0 (k)	1	0	-	-	-	A11	0	-	1
0 (l)	1	0	-	-	-	A12	0	-	1
0 (m)	1	0	-	-	-	A13	0	R0	1
0 (n)	1	0	-	-	-	A14	0	R1	1
0 (o)	1	0	-	-	-	A15	0	R2	1
0 (p)	1	0	-	-	-	-	1	R3	1
0 (q)	1	0	-	-	-	-	0	R4	1
1 (r)	0	0	-	-	-	-	0	R5	0
0 (s)	1	0	-	-	-	-	0	R6	1
0 (t)	1	0	-	-	-	-	0	R7	1
0 (u)	1	0	-	-	-	-	0	-	1
0 (v)	1	0	-	-	-	-	0	R8	1
0 (w)	1	0	-	-	-	-	0	R9	1
0 (x)	1	0	-	-	-	-	0	R10	1
1 (y)	1	0	-	-	-	-	0	R11	1
1 (z)	1	0	-	-	-	-	0	R12	1
1	0	---------- Next Transaction ----------						R13	-
---------- Next Transaction ----------								R14	-
---------- Next Transaction ----------								R15	-

During cycles (i) thorugh (q), the RAS/CAS access of the second transaction takes place. Simultaneously, the data transfer for the first transaction occurs, as indicated when Rrdy is asserted. Busy is deasserted in cycle (r) when this transfer is complete. The controller decides that there are no more transactions available for interleaving, and so asserts Start while simultaneously deasserting Intlv in cycle (r).

This instructs the RMC to drain the interleave pipeline, meaning that the data transfer for the second transaction is carried out, but there will not be a RAS/CAS access for a third transaction.Busy is deasserted after this transfer, indicating the RMC idle and ready when another transaction becomes available. The Rrdy strobe precedes the read data on the RD bus by three cycles, just as it does in the non-interleaved case. It should be noted that the RowHit/RowEmpty/RowMiss cases are identical when the transaction length is eight octbytes. This is because the RAS/CAS access times for these cases are 2/4/6 cycles. These times are short enough to be completely shadowed by the data transfer of the previous transaction.

If the controller had more than two transactions to interleave, then cycles (i) through (q) would be repeated one or more times. If the controller had chosen to interleave in cycle (a), and then discovered that only a single transaction was available, then cycles (i) through (q) would be dropped in the above diagram. This drain/fill sequence of cycles (a) through (h) followed by cycles (r) through (z) would not be as efficient in Channel bandwidth as a single non-interleaved equivalent, since the RMC would have to hold off the transfer on the Channel until Start was asserted. If the transfer size is smaller, the RAS/CAS time will limit the speed of the access, and a single interleaved transaction (consisting of a fill sequence and an drain sequence) will take the same number of cycles as the equivalent non-interleaved transaction.

The shortest transfer length permitted for interleaved transactions is one octbyte. The RowHit case for one octbyte transfers is shown in the following diagram.

Table 35: Interleaved Read Transaction (1 octbytes with RowHit)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
1 (a)	0	1	RMem	M0	Adbr/A0	-	1	-	0
0 (b)	1	0	-	-	-	-	0	-	0
1 (c)	0	1	RMem	M0	Adbr/A1	-	1	-	0
0 (d)	1	0	-	-	-	-	0	-	1
1 (e)	0	0	-	-	-	-	0	-	0
0 (f)	1	0	-	-	-	-	0	-	1
1	0	---------- Next Transaction ----------						R0	-
---------- Next Transaction ----------								-	-
---------- Next Transaction ----------								R1	-

The RowEmpty case for one octbyte transfers is shown in the following diagram. Two delay cycles have been inserted, relative to the RowHit case.

Table 36: Interleaved Read Transaction (1 octbytes with RowEmpty)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
1 (a)	0	1	RMem	M0	Adbr/A0	-	1	-	0
0 (b)	1	0	-	-	-	-	0	-	0
0 (c)	1	0	-	-	-	-	0	-	0
0 (d)	1	0	-	-	-	-	0	-	0
1 (e)	0	1	RMem	M0	Adbr/A1	-	1	-	0
0 (f)	1	0	-	-	-	-	0	-	1
0 (g)	1	0	-	-	-	-	0	-	0
0 (h)	1	0	-	-	-	-	0	-	0
1 (i)	0	0	-	-	-	-	0	R0	0
0 (j)	1	0	-	-	-	-	0	-	1
1	0	---------- Next Transaction ----------						-	-
---------- Next Transaction ----------								-	-
---------- Next Transaction ----------								R1	-

The RowMiss case for one octbyte transfers is shown in the following diagram. Four delay cycles have been inserted, relative to the RowHit case. These examples have shown sequences of identical length read transactions with the same RowHit / RowEmpty / RowMiss case. The RMC will accept interleaved transactions with any mix of read / write accesses, with any mix of transfer length (subject to buffer size restrictions), and with any mix of RowHit, RowEmpty, and RowMiss cases.

Table 37: Interleaved Read Transaction (1 octbytes with RowMiss)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (a)	0	1	RMem	M0	Adbr/A0	-	1	-	0
0 (b)	1	0	-	-	-	-	0	-	0
0 (c)	1	0	-	-	-	-	0	-	0
0 (d)	1	0	-	-	-	-	0	-	0
0 (e)	1	0	-	-	-	-	0	-	0
0 (f)	1	0	-	-	-	-	0	-	0
1 (g)	0	1	RMem	M0	Adbr/A1	-	1	-	0
0 (h)	1	0	-	-	-	-	0	-	1
0 (i)	1	0	-	-	-	-	0	-	0
0 (j)	1	0	-	-	-	-	0	-	0
0 (k)	1	0	-	-	-	-	0	R0	0
0 (l)	1	0	-	-	-	-	0	-	0
1 (m)	0	0	-	-	-	-	0	-	0
0 (n)	1	0	-	-	-	-	0	-	1
1	0	---------- Next Transaction ----------						-	-
---------- Next Transaction ----------								-	-
---------- Next Transaction ----------								R1	-
...	...	...	...	...	...	...	...	...	...

Chapter 5 Register Operations

5.1 Register Write

The RDRAM control registers are written with the Wreg opcode. A single octbyte of data is the default transfer length. The following figure shows the transaction with no delay. It is identical to the one octbyte memory write with RowHit that was discussed in a previous section.

The example shows a directed register write (only one RDRAM responds). A broadcast register write (WregB) has an identical format.

It is necessary for the controller to provide the $t_{REGISTERWRITEOVERHEAD}$ (4 synclk cycles). This is the time required after a register write for the side-effects of the write to take effect. After a register write to an RDRAM, no other transactions may be directed to the RDRAM for that interval.

Table 38: Basic Register Write Transaction (1 octbyte with no delay)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	WD	Wdone
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	Wreg	M0	Adbr/A0	-	1 (d)	W0	0
0	1	0	-	-	-	-	0	-	0
1 (g)	0 (f)	---------- Next Transaction ----------							1 (e)
...	...	...	...	...	...	...	...	...	...

5.2 Register Read

The RDRAM control registers are read with the Rreg opcode. A single octbyte of data is the default transfer length. The following figure shows the transaction with no delay. It is identical to the one octbyte memory read that was discussed in a previous section.

Table 39: Basic Register Read Transaction (1 octbyte with no delay)

Start	Busy	Intlv	Op	Mo	Ao	Ai	Last	RD	Rrdy
...	...	...	...	...	...	...	...	...	...
1 (b)	0 (a)	0	RReg	M0	Adbr/A0	-	1 (d)	-	0
0	1 (c)	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	0
0	1	0	-	-	-	-	0	-	1 (e)
1 (g)	0 (f)	---------- Next Transaction ----------						-	-
---------- Next Transaction ----------								-	-
---------- Next Transaction ----------								R0	-
...	...	...	...	...	...	...	...	...	...

Chapter 6 Opcode Description

The Op[8:0] input bus specifies the transaction command. This is a decoded bus, meaning that a number of the 512 combinations are invalid; for example, the bit-masking and byte-masking options are only defined for write operations. In general, the opcode combinations which are not listed in the Opcode Table below are reserved, and should not be used.

Table 40: Opcode Table - all combinations of Op[8:0] not listed below are reserved

Op[8:6]	Op[5:4]	Op[3:0]	Command	Description
x00	bb	0001	WMsk	Memory write with bytemask and bitmask bb
x00	00	0100	RMem	Memory read
x00	bb	0101	WMem	Memory write with bitmask bb
000	00	0110	Rreg	Register read
000	00	0111	Wreg	Register write
000	00	1111	WregB	Broadcast register write to all RDRAMs
xx1	xx	xxxx	Rsrv	Reserved
x1x	xx	xxxx	Rsrv	Reserved
100	00	1x01	WmemB	Broadcast memory write
1xx	xx	xxxx	Close	Close (precharge) the bank
x00	bb	1001	WMsk	Broadcast Memory write with bytemask and bit-mask bb
x00	bb	1101	WMem	Broadcast Memory write with bitmask bb

The above encodings are based on the assignment of a particular, independent function to each of the nine opcode bits. These functions are summarized in the following table.

Table 41: Opcode Bit Functions

Bit	Name	Description
Op[0]	Write	0 Read transaction.
		1 Write transaction.
Op[1]	Reg	0 Memory space transaction.
		1 Control-register space transaction.
Op[2]	NoByte	0 Byte-masking (memory write only).
		1 No byte-masking.
Op[3]	Bct	0 Transaction to a single RDRAM.
		1 Broadcast transaction to all RDRAMs (register write only).
Op[5:4]	BitMask	00 Npb - no bit masking.
		01 Dpb - persistent bit-masking (memory write only).
		10 Bpb - dynamic bit-masking (memory write only).
		11 Mpb - color-masking (memory write only).
Op [6]	Rsrv	Reserved
Op [7]	Rsrv	Reserved
Op[8]	Close	0 Bank is left open (sensed) at end of transaction.
		1 Bank is left closed (precharged) at end of transaction.

6.1 Memory Space Read Transactions

The description of the memory space transactions requires the definition of several arrays. These are:

Table 42 Arrays for Transaction Descriptions

Mem[2^24-1:0][7:0][8:0]	Memory space in [octbytes][bytes][bits]
Ao[26:11]	Device / bank / row transaction base address
Ac[L-1:0][10:3]	Column address for transaction octbytes
Adr[L-1:0][26:3]	Composite octbyte addresses for transaction
RD[L-1:0][7:0][8:0]	Read data in [octbytes][bytes][bits]
L	Number of octbytes transferred
WD[L-1:0][7:0][8:0]	Write mask / data in [octbytes][bytes][bits]
D[L-1:0][7:0][8:0]	Write data value in [octbytes][bytes][bits]
M[L-1:0][7:0][8:0]	Write BitMask value in [octbytes][bytes][bits]
BytM[L-1:0][7:0]	Write ByteMask value in [octbytes][bytes]
MD[7:0][8:0]	MaskData register in [bytes][bits]

Here Ac[L-1:1][10:3] are the values transferred on the Ai[10:3] bus and Ac[0][10:3] is the value on Ao[10:3] on the first clock edge. Ao[26:11] is the value on Ao[26:11] on the first clock edge. First, the array of Adr[L-1:0][10:3] addresses is formed from a composite of these other two arrays. Then the RMem transaction performs the indicated action for each octbyte that is transferred.

```
Ac[0][10:3] = Ao[10:3];                                  // Column address zero
for (i=0; i<L; i=i+1)                                    // Octbyte loop
        begin
        Adr[i][26:3] = {Ao[26:11],Ac[i][10:3]};          // Form octbyte addresses
        end
for (i=0; i<L; i=i+1)                                    // Octbyte loop
        begin
for (j=0; j<8; j=j+1)                                    // Byte loop
                begin
                for (k=0; k<9; k=k+1)                    // Bit loop
                        begin
                        RD[i][j][k] = Mem[Adr[i][26:3]][j][k];   // Memory to read data
                         end
                end
        end
```

6.2 Memory Space Write Transactions

All the write transactions perform the same basic action on each octbyte of data - the difference between the various commands is where the address, data, and mask values come from. The write loop follows:

```
for (i=0; i<L; i=i+1)                                   // Octbyte loop
        begin
        for (j=0; j<8; j=j+1)                           // Byte loop
                begin
                for (k=0; k<9; k=k+1)                   // Bit loop
                        begin
                        if (BytM[i][j] && M[i][j][k])   // Write the bit to memory if
                         Mem[Adr[26:3]][j][k] = D[i][j][k];  // bytemask and bitmask are set
                        end
                end
        end
```

Note that for each bit of write data D[i][j][k] there is a bit of BitMask M[i][j][k], and for each byte of write data D[i][j][8:0] there is a bit of ByteMask BytM[i][j].

6.2.1 WMem Transactions

Here Ac[L-1:1][10:3] are the values transferred on the Ai[10:3] bus and Ac[0][10:3] is the value on Ao[10:3] on the first clock edge. Ao[26:11] is the value on Ao[26:11] on the first clock edge. The array of Adr[L-1:0][10:3] addresses is formed from a composite of these other two arrays, as shown below:

```
Ac[0][10:3] = Ao[10:3];                                 // Column address zero
for (i=0; i<L; i=i+1)                                   // Octbyte loop
        begin
        Adr[i][26:3] = {Ao[26:11],Ac[i][10:3]};         // Octbyte addresses
        end
for (i=0; i<L; i=i+1)                                   // Octbyte loop
        begin
        for (j=0; j<8; j=j+1)                           // Byte loop
                begin
                BytM[i][j] = 1;                         // ByteMask of all ones
                end
        end
```

6.2.2 WMsk Transactions

Here Ac[L-1:1][10:3] are the values transferred on the Ai[10:3] bus and Ac[0][10:3] is the value on Ao[10:3] on the first clock edge. Ao[26:11] is the value on Ao[26:11] on the first clock edge. The array of Adr[L-1:0][10:3] addresses is formed from a composite of these other two arrays, as shown below:

```
Ac[0][10:3] = Ao[10:3];                          // Column address zero
for (i=0; i<L; i=i+1)                            // Octbyte loop
        begin
        Adr[i][26:3] = {Ao[26:11],Ac[i][10:3]};  // Octbyte addresses
        end
if (Config[2])                                   // Config[2] = 1 uses
        begin                                    // Base-style bytemask
        for (i=0; i<L; i=i+1)                    // Octbyte loop
                begin
                for (j=0; j<8; j=j+1)            // Byte loop
                        begin
                        BytM[i][j] = WD[0][i][j]; // Octbyte of bytemasks
                        end
                end
        end
else                                             // Config[2] = 0 uses
        begin                                    // Concurrent bytemask
        BytM[0][7:0] = Mo[7:0];                  // First bytemask
        for (i=1; i<L; i=i+1)                    // Octbyte loop
                begin
                for (j=0; j<8; j=j+1)            // Byte loop
                        begin
                        BytM[i][j] = WD[i-1][j][8]; // Bit[9] of each byte
                        end                      // contains bytemask
                end                              // for next byte
        end
```

Normally, the Config[2] bit is set to a static value at initialization. The Config[2] bit selects between two different byte mask mechanisms. When it is zero, the Concurrent mechanism is chosen: each bytemask bit is contained in bit[8] of the previous octbyte, with the first eight bytemask bits contained on the Mo[7:0] input bus. When the Config[2] bit is a one, the Base mechanism is chosen: the 64 bytemask bits for eight octbytes of data are assembled into an octbyte which precedes the data (this format is shown in detail in a later section).

6.2.2.1 Npb Write Transactions

This is the default bit-masking option - a bit mask of all ones is effectively formed for all octbytes of data.

```
for (i=0; i<L; i=i+1)                          // Octbyte loop
        begin
        for (j=0; j<8; j=j+1)                  // Byte loop
                begin
                for (k=0; k<9; k=k+1)          // Bit loop
                        begin
                        M[i][j][k] = 1;        // BitMask of all ones
                        D[i][j][k] = WD[i][j][k];  // Write data from WD bus
                        end
                end
        end
```

6.2.2.2 Dpb Write Transactions

A constant BitMask from the MDReg is used for all octbytes of data, with the write data coming from the WD input bus.

```
for (i=0; i<L; i=i+1)                          // Octbyte loop
        begin
        for (j=0; j<8; j=j+1)                  // Byte loop
                begin
                for (k=0; k<9; k=k+1)          // Bit loop
                        begin
                        M[i][j][k] = MDReg[j][k];  // BitMask from MDReg
                        D[i][j][k] = WD[i][j][k];  // Write data from WD bus
                        end
                end
        end
```

6.2.2.3 Mpb Write Transactions

Constant data from the MDReg is used for all octbytes, with the bitmask coming from the WD input bus.

```
for (i=0; i<L; i=i+1)                          // Octbyte loop
        begin
        for (j=0; j<8; j=j+1)                  // Byte loop
                begin
                for (k=0; k<9; k=k+1)          // Bit loop
                        begin
                        M[i][j][k] = WD[i][j][k];  // BitMask from WD bus
                        D[i][j][k] = MDReg[j][k];  // Write data from MDReg
                        end
                end
        end
```

6.2.2.4 Bpb Write Transactions

Both the write data and the BitMasks come from the WD input bus. The effective transfer length L is halved, as a result:

```
for (i=0; i<L/2; i=i+1)                          // Octbyte loop
        begin
        for (j=0; j<8; j=j+1)                    // Byte loop
                begin
                for (k=0; k<9; k=k+1)            // Bit loop
                        begin
                        M[i][j][k] = WD[2*i][j][k];   // BitMask from WD bus
                        D[i][j][k] = WD[2*i+1][j][k]; // Write data from WD bus
                        end
                end
        end
```

6.2.3 WMemNpb, WMemDpb, WMemMpb, WMemBpb Transaction Tables

The following table shows the values on the Ai[10:3] and WD[7:0][8:0] input buses for transfers which are 8 octbytes long for the WMemNpb, WMemDpb, WMemMpb transactions.

Table 43: Input buses for WMemNpb, WMemDpb, WMemMpb Transactions

i	Ai[10:3]	WMemNpb WD[7:0][8:0]	WMemDpb WD[7:0][8:0]	WMemMpb WD[7:0][8:0]
0	Ac[1][10:3]	D[0][7:0][8:0]	D[0][7:0][8:0]	M[0][7:0][8:0]
1	Ac[2][10:3]	D[1][7:0][8:0]	D[1][7:0][8:0]	M[1][7:0][8:0]
2	Ac[3][10:3]	D[2][7:0][8:0]	D[2][7:0][8:0]	M[2][7:0][8:0]
3	Ac[4][10:3]	D[3][7:0][8:0]	D[3][7:0][8:0]	M[3][7:0][8:0]
4	Ac[5][10:3]	D[4][7:0][8:0]	D[4][7:0][8:0]	M[4][7:0][8:0]
5	Ac[6][10:3]	D[5][7:0][8:0]	D[5][7:0][8:0]	M[5][7:0][8:0]
6	Ac[7][10:3]	D[6][7:0][8:0]	D[6][7:0][8:0]	M[6][7:0][8:0]
7	–	D[7][7:0][8:0]	D[7][7:0][8:0]	M[7][7:0][8:0]

The following table shows the values on the Ai[10:3] and WD[7:0][8:0] input buses for a transfer which is 8 octbytes long for the WMemBpb transaction.

Table 44: Input buses for WMemBpb Transactions

i	Ai[10:3]	WMemBpb WD[7:0][8:0]
0	Ac[0][10:3]	M[0][7:0][8:0]
1	-	D[0][7:0][8:0]
2	Ac[1][10:3]	M[1][7:0][8:0]
3	-	D[1][7:0][8:0]
4	Ac[2][10:3]	M[2][7:0][8:0]
5	-	D[2][7:0][8:0]
6	Ac[3][10:3]	M[3][7:0][8:0]
7	-	D[3][7:0][8:0]

6.2.4 WMskNpb, WMskDpb, WMskMpb Transaction Tables (Config[2]=0)

The following table shows the values on the Ai[10:3] and WD[7:0][8:0] input buses for transfers which are 8 octbytes long for the WMskNpb, WMskDpb, WMskMpb transactions when the Config[2] bit is a zero (Concurrent-style byte mask).

Table 45: Ai and WD Buses for Write Transaction with Config[2] = 0

i	Ai[10:3]	WMskNpb WD[7:0][8:0]	WMskDpb WD[7:0][8:0]	WMskMpb WD[7:0][8:0]
0	Ac[1][10:3]	D[0][7:0][8:0]	D[0][7:0][8:0]	M[0][7:0][8:0]
1	Ac[2][10:3]	D[1][7:0][8:0]	D[1][7:0][8:0]	M[1][7:0][8:0]
2	Ac[3][10:3]	D[2][7:0][8:0]	D[2][7:0][8:0]	M[2][7:0][8:0]
3	Ac[4][10:3]	D[3][7:0][8:0]	D[3][7:0][8:0]	M[3][7:0][8:0]
4	Ac[5][10:3]	D[4][7:0][8:0]	D[4][7:0][8:0]	M[4][7:0][8:0]
5	Ac[6][10:3]	D[5][7:0][8:0]	D[5][7:0][8:0]	M[5][7:0][8:0]
6	Ac[7][10:3]	D[6][7:0][8:0]	D[6][7:0][8:0]	M[6][7:0][8:0]
7	-	D[7][7:0][8:0]	D[7][7:0][8:0]	M[7][7:0][8:0]

The byte mask bit BytM[i][j] for each byte of data comes from D[i-1][j][8] (the previous octbyte). The byte mask for the first octbyte of data comes from the Mo[7:0] bus.

The bit and byte detail of an eight octbyte transaction is shown in the next table. It shows how the byte mask bit for a particular data byte comes from bit[8] exactly eight bytes earlier. For example, write data byte WD[4][2][8:0] is “vDDDDDDDD”. WD[4][2][8] (the value “v”) is equivalent to BytM[5][2], and controls whether data byte D[5][2][7:0] (the value “vvvvvvvv”) is written. The ninth bit of the data bytes in the last octbyte transferred (D[7][7:0][8:0] in this example) are not used, and are shown as “-” in the figure.

Using this notation, the value of the Mo[7:0] bus on the first clock edge of the transaction would contain the value “456789@$” (the byte mask bits BytM[0][7:0]), which controls whether each of the bytes from the first data octbyte D[0][7:0][8:0] are written or not written.

Table 46: Mapping of ByteMask Control Bit to Data Bytes (Config[2]=0)

k =	876543210 WD[7][k]	876543210 WD[6][k] \	876543210 WD[5][k]	876543210 WD[4][k]	876543210 WD[3][k]	876543210 WD[2][k]	876543210 WD[1][k]	876543210 WD[0][k]
i=								
0	W44444444	X55555555	Y66666666	Z77777777	088888888	199999999	2@@@@@@@@	3$$$$$$$$
1	owwwwwwww	PXXXXXXXX	QYYYYYYYY	RZZZZZZZZ	S00000000	T11111111	U22222222	V33333333
2	GOOOOOOOO	HPPPPPPPP	IQQQQQQQQ	JRRRRRRRR	KSSSSSSSS	LTTTTTTTT	MUUUUUUUU	NVVVVVVVV
3	yGGGGGGGG	zHHHHHHHH	AIIIIIIII	BJJJJJJJJ	CKKKKKKKK	DLLLLLLLL	EMMMMMMM	FNNNNNNNN
4	qyyyyyyyy	rzzzzzzzz	sAAAAAAAA	tBBBBBBBB	uCCCCCCCC	vDDDDDDDD	wEEEEEEEE	xFFFFFFFF
5	iqqqqqqqq	jrrrrrrrr	ksssssss	lttttttt	muuuuuuuu	nvvvvvvvv	owwwwwwww	pxxxxxxxx
6	aiiiiiiii	bjjjjjjjj	ckkkkkkkk	dlllllll	emmmmmmmm	fnnnnnnnn	goooooooo	hpppppppp
7	-aaaaaaaa	-bbbbbbbb	-cccccccc	-dddddddd	-eeeeeeee	-ffffffff	-gggggggg	-hhhhhhhh

In general, if the byte masking command WMsk is used, the ninth bit of each byte is not available for storage (even if it is present in the RDRAM). If individual bytes within 72-bit octbytes need to be written, then there are three possible alternatives.

1. A read-modify-write operation is used.
2. The WMemBpb permits bytemasking (using dynamic bitmasking).
3. WMsk permits one octbyte writes with bytemasking, since the byte mask for the first octbyte comes form the Mo[7:0] bus.

Which of these is chosen depends upon the characteristics of the application. All three provide bytemasking and nine-bit bytes at a bandwidth that is about half of the bandwidth of the unmasked writes or masked writes with 8-bit bytes.

6.2.5 WMskNpb, WMskDpb, WMskMpb Transaction Tables (Config[2]=1)

The following table shows the values on the Ai[10:3] and WD[7:0][8:0] input buses for transfers which are 9 octbytes long for the WMskNpb, WMskDpb, WMskMpb transactions. Base-style byte masking (Config[2]=1) is used.

Table 47: Ai and WD Buses for Write Transaction with Config[2] = 1

i	Ai[10:3]	WMskNpb WD[7:0][8:0]	WMskDpb WD[7:0][8:0]	WMskMpb WD[7:0][8:0]
0	Ac[0][10:3]	BytM[7:0][8*:0]	BytM[7:0][8*:0]	BytM[7:0][8*:0]
1	Ac[1][10:3]	D[0][7:0][8:0]	D[0][7:0][8:0]	M[0][7:0][8:0]
2	Ac[2][10:3]	D[1][7:0][8:0]	D[1][7:0][8:0]	M[1][7:0][8:0]
3	Ac[3][10:3]	D[2][7:0][8:0]	D[2][7:0][8:0]	M[2][7:0][8:0]
4	Ac[4][10:3]	D[3][7:0][8:0]	D[3][7:0][8:0]	M[3][7:0][8:0]
5	Ac[5][10:3]	D[4][7:0][8:0]	D[4][7:0][8:0]	M[4][7:0][8:0]
6	Ac[6][10:3]	D[5][7:0][8:0]	D[5][7:0][8:0]	M[5][7:0][8:0]
7	Ac[7][10:3]	D[6][7:0][8:0]	D[6][7:0][8:0]	M[6][7:0][8:0]
8	-	D[7][7:0][8:0]	D[7][7:0][8:0]	M[7][7:0][8:0]

* Note - BytM[7:0][8] is unused

The bit and byte detail of the ByteMask is shown in the last table. It shows the mapping from each bit of the BytM[7:0][8:0] mask to each byte of the eight octbytes of write data WD[8:1][7:0]. For example, ByteMask bit “J” (WD[0][3][4]) controls the byte “JJJJJJJJ” (WD[4][4][8:0]).

Table 48: Mapping of ByteMask Control Bit to Data Bytes (Config[2]=1)

k =	**876543210 WD[7][k]**	**876543210 WD[6][k**	**876543210 WD[5][k]**	**876543210 WD[4][k]**	**876543210 WD[3][k]**	**876543210 WD[2][k]**	**876543210 WD[1][k]**	**876543210 WD[0][k]**
i=								
0	-abcdefgh	-ijklmnop	-qrstuvwx	-yzABCDEF	-GHIJKLMN	-OPQRSTUV	-WXYZ0123	-456789@$
1	444444444	555555555	666666666	777777777	888888888	999999999	@@@@@@@@	$$$$$$$$$
2	wwwwwwwww	XXXXXXXXX	YYYYYYYYY	ZZZZZZZZZ	000000000	111111111	222222222	333333333
3	OOOOOOOOO	PPPPPPPPP	QQQQQQQQQ	RRRRRRRRR	SSSSSSSSS	TTTTTTTTT	UUUUUUUUU	VVVVVVVVV
4	GGGGGGGGG	HHHHHHHHH	IIIIIIIII	JJJJJJJJJ	KKKKKKKKK	LLLLLLLLL	MMMMMMMM	NNNNNNNNN
5	yyyyyyyyy	zzzzzzzzz	AAAAAAAAA	BBBBBBBBB	CCCCCCCCC	DDDDDDDDD	EEEEEEEEE	FFFFFFFFF
6	qqqqqqqqq	rrrrrrrrr	sssssssss	ttttttttt	uuuuuuuuu	vvvvvvvvv	wwwwwwwww	xxxxxxxxx
7	iiiiiiiii	jjjjjjjjj	kkkkkkkkk	lllllllll	mmmmmmmm	nnnnnnnnn	ooooooooo	ppppppppp
8	aaaaaaaaa	bbbbbbbbb	ccccccccc	ddddddddd	eeeeeeeee	fffffffff	ggggggggg	hhhhhhhhh

6.3 Register Space Transactions

Note that address bit Ao[2] is needed to address the control registers. All register transactions are one octbyte in length.

Chapter 7 RMC Modifications and Extensions

The RMC directory includes several versions of certain modules. These variations permit a trade-off between cost and performance over a limited range to be made.

7.1 RowTrack Module

The RowTrack module contains a RowCache with a storage array. The following table shows available options.

Table 49: RowTrack Modules

Filename	Gates	Description
RowTrackR.4.v	462	RowCache with 4 entries
RowTrackR.16.v	1302	RowCache with 16 entries

The filename also indicates the number of entries in the RowCache. Currently, 16-entry RowCaches are available and the 4-entry RowCaches model is still under development. Using these two examples, it is easy to implement RowCaches with other storage array sizes. Note that if the number of RDRAM banks exceeds the number of RowCache entries, the RMC will still operate correctly, but at a lower performance level, since the RowCache will make worst case assumptions (RowMiss) about a bank that has been displaced.

7.2 ALWDelay Module

The ALWDelay module contains a delay buffer needing a storage array.

Table 50: ALWDelay Modules

Filename	Gates	Description
ALWDelayR.9.8.v	5407	Buffer for 9-octbyte write, 8-octbyte read

The filename indicates the number of entries in the delay buffer. Currently, only 9.8-entry (9 octbyte write transactions and 8 octbyte read transactions) delay buffers are supported.

7.3 PreDelay Module

The PreDelay module contains a delay buffer needing a storage array.

Table 51: PreDelay Modules

Filename	Gates	Description
PreDelayR.9.8.v	5407	Buffer for 9-octbyte write, 8-octbyte read

The filename indicates the number of entries in the delay buffer. Currently, only 9.8-entry (9 octbyte write transactions and 8 octbyte read transactions) delay buffers are supported.

Chapter 8 RMC Performance Calculations

Performance can be measured in two ways. Latency is a measure of the round- trip delay seen by read data. It is measured from when the Start input is asserted to begin the transaction to when the first octbyte of read data becomes valid on the RD[7:0][8:0] bus. This is broken down below:

Table 52: Latency of a Read Transaction (RowHit)

Delay	Description
6ns	RMC setup time (with RAC transmit) for Start input
1.0cycle	RDRAM standby-to-activate wake up
0.75cycle	Request packet on channel
2.25cycle	RDRAM read delay
1.0cycle	Data octbyte on channel
7ns	RMC valid time (with RAC receive) for RD outputs
13ns + 5 cycles	Total

If there is a RowEmpty or RowMiss, then 2 or 4 cycles are added to the above latency total.

Performance is also measured in effective bandwidth. This is calculated with the following formula:

BWeff = BWpeak * D/(Ov+D)

where:

BWeff	Effective bandwidth
BWpeak	Peak bandwidth (500, 533, or 600 Mbyte/s)
D	Length of transaction data packet (measured in octbytes)
Ov	Transaction overhead (measured in cycles)

The overhead value can be found in the following table:

Table 53: Transaction Overhead (in cycles)

Op	Ov	Conditions	Ov	Conditions
	Concurrent - No Interleave		Base-60(1) - No Interleave	
RMem	3/5/7	RowHit/RowEmpty/RowMiss	3/8(1)	RowHit/RowMiss
WMem	1/3/5	RowHit/RowEmpty/RowMiss	1/6(1)	RowHit/RowMiss
WMsk	1/3/5(2)	RowHit/RowEmpty/RowMiss	2/7(1)(3)	RowHit/RowMiss
	Concurrent - Interleave		Base-60 - Interleave	
RMem	1(4)		1/5(4)	RowHit/RowMiss
WMem	1(4)		1/3(4)	RowHit/RowMiss
WMsk	1(2)(4)		2/4(3)(4)	RowHit/RowMiss

(1)For Base-70, add +1/3 for clean/dirty RowMiss
(2)Config[2]=0 (Concurrent-style byte masking is assumed)
(3)Config[2]=1 (Base-style byte masking is assumed)
(4)For the interleaved cases, the data length of the previous transaction is assumed to be large enough to completely hide the RAS/CAS access time of the current transaction.

For example, with eight octbyte data transfers, the non-interleaved RMem RowHit bandwidth is 533*8/11 or 388 Mbyte/s and the WMem Rowhit bandwidth is 533*8/9 or 474 Mbyte/s.

Chapter 9 RMC Cost Estimates

The RMC cost is calculated in gates. The gate cost of some primitive functions are shown in the table below. This table can be used to adjust the gate count figures for other gate array families.

Table 54: Primitive Functions - Gate Cost

Function	Gates	Description
Nand2	1.0	2-input Nand gate
Nand3	1.5	3-input Nand gate
Nand4	2.0	4-input Nand gate
Nor2	1.0	2-input Nor gate
Nor3	1.5	3-input Nor gate
Nor4	2.0	4-input Nor gate
Xor4	2.7	2-input Xor gate
Xnor4	2.7	2-input Xnor gate
AOI2222	3.3	And-Or-Invert gate (4 2-input Ands are Nored together)
Decode2-4	5.7	Decode 2 wires to 4 wires
Decode3-8	14.0	Decode 3 wires to 8 wires
MuxE2x4	9.0	2-1 Multiplexer with enable (4 bits wide)
Latx8	20.3	Level-gated latch (8 bits wide)
Regx4	16.3	Edge-triggered register (4 bits wide)

Using the above figures, the gate cost of the five modules of the RMC can be calculated. this is shown in the first table below. The gate cost of the other module options for the RowTrack, ALWDelay, and AddrMap modules are summarized next.

Table 55: RMC Modules - Gate Cost

Module	Gates	Comments
RMCSys.v	0	
Control.v	1196	
Request.v	801	
RowTrackR.4.v	462	4-entry RowCache - see other options below
ALWDelayR.5.8.v	3157	4/8 octbyte write/read - see other options below
PreDelayL.0.0v	0	0/0 octbyte write/read - see other options below
AddrMap.v	744	No mapping option - see other options below
Total	6360	

Table 56: Other Module Options

Filename	Gates	Description
RowTrackR.4.v	462	RowCache with 4 entries
RowTrackR.16.v	1302	RowCache with 16 entries
ALWDelayR.5.8.v	3157	Buffer for 5-octbyte write, 8-octbyte read
ALWDelayR.9.8.v	5407	Buffer for 9-octbyte write, 8-octbyte read
PreDelayR.5.8.v	3157	Buffer for 5-octbyte write, 8-octbyte read
PreDelayR.9.8.v	5407	Bufferfor 9-octbyte write, 8-octbyte read

Chapter 10 Designing with the RMC

10.1 RAC Interface

The key signals in the interface between the RAC and the RMC were described in an earlier section. Refer to Section 2.1 "Signal Description" on page 3. There are other RAC signals which do not connect to the RMC, the following figure shows some of these signals. Refer to Figure 8 "Rambus Subsystem Block Diagram" on page 68.

The remaining RAC I/O signals are connected to static values or are used for test purposes. Connect all the unused input pins to ground, except for SSTReq which should be connected to Vdd. Also, connect the SRCtrl to the inverse of RESET. All the output signals can be left unconnected. Please contact the chip manufacture for the usage of seven test interface pins (SCANIn, SCANOut, SCANEn, SCANMode, SCANClk, BIST-Mode, BISTFlag). More detailed information on the RAC interface can be found in the *RAC Test Methodology* Document (DL0015) and the *Rambus ASIC Cell User's Guide and Specification* (DL0005).

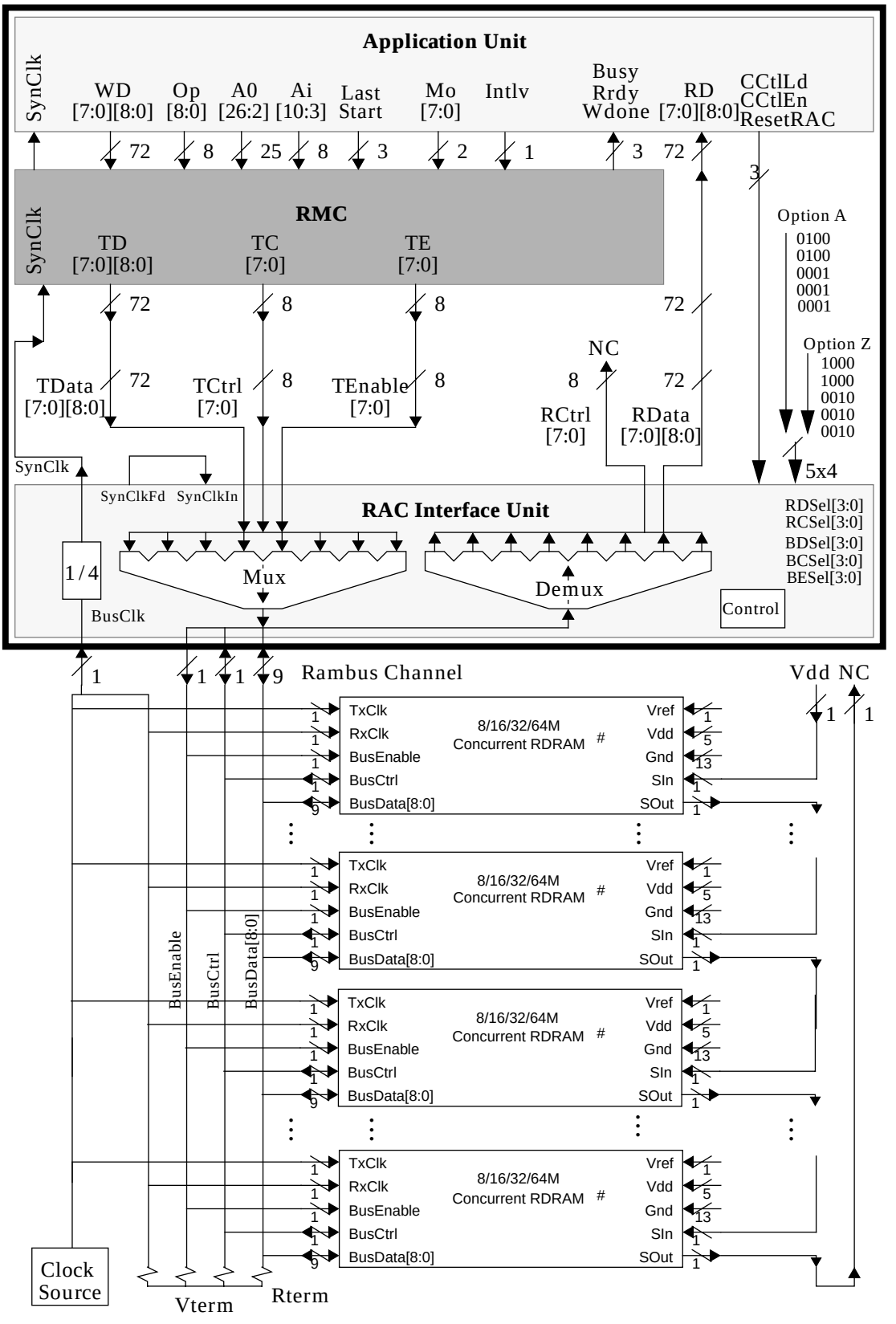
10.2 Initialization

The initialization of an RDRAM system involves three steps; first the reset signal of the RAC and RMC blocks is asserted, second a reset pulse is asserted on the BusCtrl wire causing all RDRAMs to reset to an initial configuration, and finally a series of register write and read operations is made to the RDRAMs to set the final configuration. Chapter 9 of the *Concurrent Design Guide* (DL0032) includes an initialization summary and an initialization routine.

10.3 Refresh and Current Control

RDRAMs (like any DRAM memory technology) require a periodic refreshing of their storage cell contents. This function is managed by a block of logic in the Application Unit of the ASIC. RDRAMs also require that the I_{OL} current level used by the RDRAM output drivers also be periodically updated. Another block of logic in the Application Unit accomplishes this. Chapter 8 of the *Concurrent Design Guide* (DL0032) shows examples of these two logic blocks.

Figure 8: Rambus Subsystem Block Diagram



10.4 Config[2:0] Input Summary

The Config[2:0] input bus permits the controller to specify a number of options in the RMC. The Config[2:0] bus is normally driven with a static value.

10.4.1 Config[0] - Busy control

0 - Normal Busy. Start may be asserted in the same cycle in which Busy is deasserted (this is the mode used in the examples throughout this document).
1 - This setting is not supported.

10.4.2 Config[1] - PreDelay control

0 - PreDelay enabled. The PreDelay buffer is used during interleaved transactions and is automatically bypassed for non-interleaved transactions (this is the mode used in the examples throughout this document).
1 - PreDelay bypassed. The PreDelay buffer is bypassed for both interleaved and non-interleaved transactions. This is equivalent to using the PreDelay.0.0 module. The benefit of this option is that the storage bits of the PreDelay buffers are not included in the RMC gate total. The cost of this option is that the information on the Ai[7:0], Last, and Wd[7:0][8:0] buses must be delayed by one transaction when interleaved transactions are used. The following timing diagram shows the steady-state four octbyte interleaved case with this option. This is appropriate for applications such as 3D graphics when data must be read, modified and written back while the accessed page remains open. Contact Rambus for more information about this.

Table 57: Steady State - Interleaved Four Octbyte Read/Write Transactions – with Config[1]=1

Start	Busy	Intlv	Op Mo Ao	Ai Last WD	Wdone	Rrdy	RD	RAS/ CAS access	Channel transfer
...	...	...	...	...	...	...	...	...	
1 (a)	0	1	OMA[i]	[i-1]	[i-2]	-	[i-2]	[i-1]	[i-2]
0 (b)	1	0	-	[i-1]	0	[i-1]	[i-2]	[i-1]	[i-2]
0 (c)	1	0	-	[i-1]	[i-1]	[i-1]	[i-2]	[i-1]	Req[i]
0 (d)	1	0	-	[i-1]	[i-1]	[i-1]	-	RAS[i]	[i-1]
0 (e)	1	0	-	-	[i-1]	[i-1]	[i-1]	RAS[i]	[i-1]
1 (f)	0	1	[i+1]	ALW[i]	[i-1]	-	[i-1]	RAS[i]	[i-1]
0 (g)	1	0	-	ALW[i]	0	R[i]	[i-1]	CAS[i]	[i-1]
0 (h)	1	0	-	ALW[i]	W[i]	R[i]	[i-1]	CAS[i]	Req[i+1]
0 (i)	1	0	-	ALW[i]	W[i]	R[i]	-	[i+1]	D[i]
0 (j)	1	0	-	-	W[i]	R[i]	D[i]	[i+1]	D[i]
1 (k)	0	1	[i+2]	[i+1]	W[i]	-	D[i]	[i+1]	D[i]
0 (l)	1	0	-	[i+1]	0	[i+1]	D[i]	[i+1]	D[i]
0 (m)	1	0	-	[i+1]	[i+1]	[i+1]	D[i]	[i+1]	Req[i+2]
0 (n)	1	0	-	[i+1]	[i+1]	[i+1]	-	[i+2]	[i+1]
0 (o)	1	0	-	-	[i+1]	[i+1]	[i+1]	[i+2]	[i+1]
...	...	...	...	...	...	...	...	...	

The difference that results from this option is that the Ai, Last, and WD information for transaction [i] is given to the RMC in cycles (f) through (i) rather than cycles (a) through (d), as had been the case with Config[1]=0. All other information is received and transmitted in the same timing slots. Non-interleaved transactions are not affected at all.

10.4.3 Config[2] - Bytemask control

0 - Concurrent bytemask. The Concurrent RDRAM bytemask mechanism is used: the bytemask for the first octbyte of data is in the request packet, the bytemask for all remaining octbytes are contained within the ninth bits of the previous octbyte. This option should be used if only Concurrent RDRAMs will be hooked up to the RMC.

1 - This setting is not supported.

10.4.4 Timing Options for the RMC IO signals.

The RMC has the Option A and The Option Z to adjust the RMC I/O signals Start, Busy, Last and RD[7:0][8:0]. In Option Z, the transmit and receive windows used by the RAC are delayed by 1/4 SynClk (compared to Option A).

The RMC IO timing adjustment is performed by programing certain input signals to the RAC including BDSel, BCSel, BESel, RDSel and RCSel as shown in the block diagram 10.3.1.

Chapter 11 File Description

This directory contains test files and Verilog files. The text files include:

Table 58 Text File Description

RMC-README	An overview document for the RMC
RMC-Errata	Deviations of the Verilog RMC model relative to the specification for the RMC

There is a high level Verilog module called RcpuSys.v. It uses the following module. (The RAC.v, RDRAM.v and RDRAMC.v are not part of RMC code and subject to update – contact Rambus for the most updated models.):

Table 59 Verilog File Description

Rcpu.v	RMC protocol unit
RAC.v	RAC interface unit
RDRAMB.v	RDRAM 16M Base RDRAM
RDRAMC.v	RDRAM 16/64M Concurrent RDRAM model
RcpuTasks.v	Tasks used by initialization and verification suites
MuxLatReg.v	Multiplexer, latch, and register primitives
PatternsInitialize.v	Patterns to initialize RDRAM control registers
VerifyShort.v	Patterns to verify RMC design (~2k transactions)
VerifyMedium.v	Patterns to verify RMC design (~96k transactions)
VerifyLong.v	Patterns to verify RMC design (~320k transactions)
VerifyTests.v	Patterns for debugging

RMC.v includes seven modules Control.v, Request.v, RowTrackL.16.v, ALWDelayR.9.8.v, PreDelayR.9.8.v, and AddrMap.v.

Several command files are used for running Verilog and Undertow:

Table 60 Command Files

govs	Launches Verilog using the RMCSys.v module
gous	Launches Undertow using the RMCSys.dump file
UndertowSignals-ForDebug	Undertow signal list for internal and external signals

Document Change History

Date	Version	Page	Description
10-15-96	DL0030-02		Previous update
04-03-98	-03	Title	Copyright updated, ® added, phone numbers changed and version updated to 3.0
	-03	3	Table 1, Mo[7:0] Description: last sentence added.
	-03	7	Table 3, $T_{S,MAX}$ and $T_{V,MAX}$ values all changed. Table footnote1 text changed including Option B changed to Option Z: Table footnote 2 deleted since Early Busy is not supported.
	-03	47	Third paragraph, first sentence, (16 synclk) changed to (4 synclk).
	-03	49	Table 40, last two rows added.
	-03	68	Figure 8, Option A, last three values changed. Option B changed to Option Z; all values changed.
	-03	69	10.4.1 Config[0] - Busy control, last paragraph (1-Early Busy) deleted.
	-03	71	Section 10.4.4. Option B changed to Option Z. Last two sentences of first paragraph replaced.