Overview

Rambus XDR™ technology has been designed to support a broad range of applications and component implementations spanning multiple technology generations. The XDR architecture comprises three primary semiconductor components:

- XDR DRAM
- XDR IO Cell (XIO)
- XDR Memory Controller (XMC)

The XDR DRAM is a device-level product offered by Rambus memory partners containing an XDR-compliant interface specified by Rambus. The XDR DRAM interface module serves as an interface between the XDR Channel and the XDR DRAM core. Each XDR DRAM component is described by its component datasheet. However, a memory controller may need to accommodate a range of XDR DRAM configurations. Alternatively, a memory controller may need to accommodate an XDR DRAM configuration which has not yet been implemented, and thus has no component datasheet.

The XIO consists of a collection of hard and soft macros providing the physical layer interface between the XMC or customer-designed memory controller and the XDR channel. XIO hard macros are planned for availability from a variety of foundry sources across multiple process technology generations.

The XMC is a configurable soft macro reference design provided by Rambus. Customers may integrate the XMC directly or use it as a reference in designing and verifying their own memory controller.

The purpose of this architecture datasheet is to specify the full range of the system-level features and parameters that can vary from one XDR configuration to another. For example, with this document and one or more component datasheets, a controller can be designed to accommodate a wide range of XDR DRAM configurations.

XDR DRAM Core Organization

XDR DRAMs will support a variety of core organizations over the lifetime of the architecture. Table 1 lists the range of device densities, bank counts, and page sizes planned for the architecture as well as a more limited range which is recommended for initial memory controller implementations. Verification priority is also listed to emphasize the most important modes for initial shipping products.

Table 1 **XDR DRAM Core Organization Parameters**

Parameter	Architectural Range	Recommended Initial Range for Memory Controller	
		Design	Verification Priority
Device Density	256Mb - 8Gb	256Mb - 2Gb	512Mb - 1Gb
Device Bank Count	4, 8, 16, 32 independent banks	4, 8, 16 independent banks	8 independent banks
Device Page Size	1, 2, 4, 8 KB	1, 2, 4 KB	2, 4 KB

Request Packets

Request packet formatting is discussed in each XDR DRAM component-level datasheet, but this section describes the packet formatting across multiple device implementations. Please consult the component datasheets for full details.

Request Packet Formats

There are five types of request packets:

1. ROWA — specifies an ACT command
2. COL — specifies RD and WR commands
3. COLM — specifies a WRM command
4. ROWP — specifies PRE and REF commands
5. COLX — specifies the remaining commands

Table 2 describes fields within different request packet types along with the planned architectural range of each field, a recommended range for initial product designs, and a recommended list of the highest priority modes for verification. Request packet formats are illustrated in Figure 1.

Netlist EX2009-p. 1
Samsung v Netlist
IPR2023-00454

In the ROWA packet, the positions of the opcode (OP), and command delay (DELA) are fixed, but there is a flexible boundary or "sliding bar" between the bank address (BA) and row address (R) fields. The fewer the number of bank address bits required, the more that are available for row addressing, and vice-versa.

In the COL and COLM packets, there is a sliding bar between the bank address (BC) and the column address (C), allowing flexibility between device bank count and maximum page size. There is also a sliding bar between the C and SC fields, which is set by the native width of the device. This is described in the section on Column Addressing below.

The ROWP and COLX packets do not contain any sliding bars, so all bit positions are fixed, although the number of bank bits (BP) used may vary by device implementation.

Table 2 **Request Packet Field Description and Range**

Field	Packet Types	Architectural Range	Recommended Initial Range for Memory Controller		Description
			Design	Verification Priority	
OP	ALL types	4 bits			Operation code that specifies packet format.
DELA	ROWA	1 bit			Delay the associated row activate command by 0 or 1 t_{CYCLE}.
BA	ROWA	2-5 bits	2-4 bits	3 bits	Bank address for row activate command.
R	ROWA	9-17 bits	11-15 bits	12-13 bits	Row address for row activate command.
SR	ROWA	2 bits			Sub-Row address for sub-page activation.
WRX	COL	1 bit			Specifies RD (=0) or WR (=1) command.
DELC	COL	1 bit			Delay the column read or write command by 0 or 1 t_{CYCLE}.
BC	COL/COLM	2-5 bits	2-4 bits	3 bits	Bank address for column read or write command.
C	COL/COLM	All modes listed in Table 4	Modes listed in Table 4 for 1, 2,4, KB page size	Modes listed in Table 4 for 2,4 KB page size	Column address for column read or write command.
SC	COL/COLM				Sub-column address for read or write command when DQ width is less than maximum.
M	COLM	8 bits			Mask key for masked-write command WRM.
POP	ROWP	3 bits			Operation code that specifies row precharge command with a delay of 0 to 3 t_{CYCLE}.
BP	ROWP	2-5 bits	2-4 bits	3 bits	Bank address for row precharge command.
ROP	ROWP	3 bits			Operation code that specifies refresh commands.
RA	ROWP	8 bits			Refresh address field (specifies BR bank address, delay value, and REFr load value).
XOP	COLX	4 bits			Extended operation code for column preload, calibration and powerdown commands.
MX	COLX	8 bits			Write mask key for column preload command (only used for systems which support a write buffer).

Netlist EX2009-p. 2
Samsung v Netlist
IPR2023-00454

Table of Contents

Overview **1**
Request Packets **1**
Request Packet Formats 1
Table of Contents **3**
List of Tables **3**
List of Figures **3**
Addressing Considerations 6
Dynamic Point-to-Point Configurations 10
Memory System Addressing 11
XDR Architecture Features **15**
XDR DRAM Width 16
Sub-Page Activation 16
ECC/Chipkill Support 17
Module Upgradability (DPP) 17
Write Masking 17
W/R Turnaround Improvement 18
Refresh Support 20
Calibration Support 23
Timing Parameters **24**
Frequency Scalability 24
XDR DRAM Timing Parameters 25
Rules for Setting MC Timing Parameters 25
XIO Interface Timing Parameters 25
List of Tables **35**
List of Figures **37**

List of Tables

XDR DRAM Core Organization Parameters **1**
Request Packet Field Description and Range **2**
Sub-Column Decoding for COL I/O Selection and Datapath Routing **5**
Column Addressing for Various Device Page Sizes and Widths (BL = 16)* **6**
Sub-Page Addressing Support **7**
Planned Range of Sub-Page Support for XDR Architecture **7**
Device Connections Usage Versus System Configuration and Module Device Count **9**
XDR Architectural Features **14**
System Turnaround Bubble Sizes (Early Read Option) **18**
Architectural Range for Refresh **19**
XDR Calibration Operations **22**
Frequency Scalability for XDR Architecture **23**
XDR DRAM Timing Parameters **26**
Rules and Guidelines for Setting MC Timing Parameters **28**
XIO Interface Timing Parameters **29**

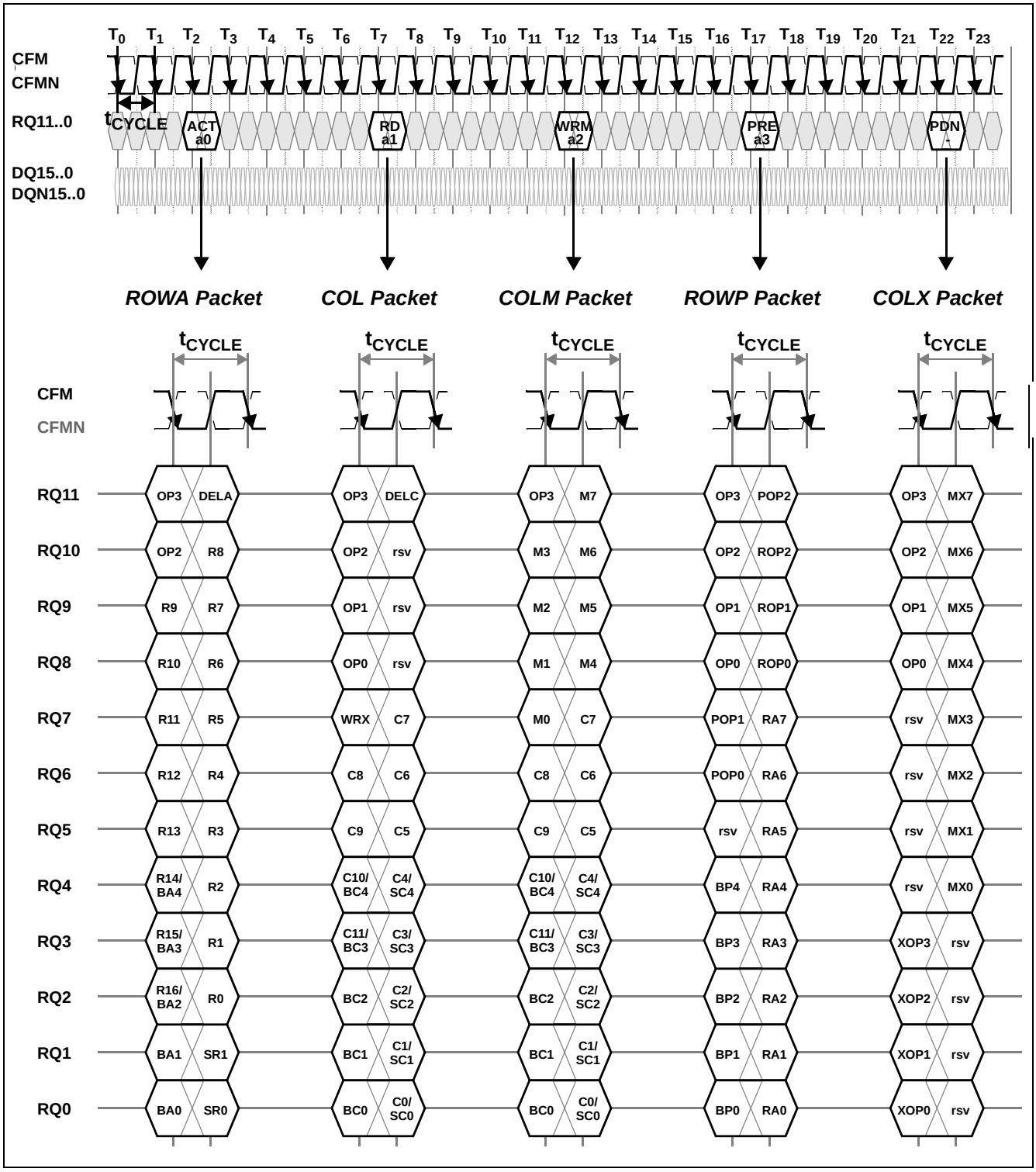
List of Figures

Request Packet Formats for Full XDR Architectural Range **4**
Example Half-Page Activation for x16/x8/x4 XDR DRAM **8**
Dynamic Point-to-Point Topology with Separate RQ Channel Per Module (x32 DQ Channel) **9**
Example XDR Module Addressing **10**
Module Addressing for 16d x 16 Module (BL=16) **11**
Example Address Remapping Logic **13**
Simultaneous Refresh Transactions **21**
XDR Frequency Roadmap **23**

Netlist EX2009-p. 3
Samsung v Netlist
IPR2023-00454

Netlist EX2009-p. 4
Samsung v Netlist
IPR2023-00454

Figure 1 Request Packet Formats for Full XDR Architectural Range



Netlist EX2009-p. 5
Samsung v Netlist
IPR2023-00454

Addressing Considerations

Native Versus Programmable Width

XDR DRAMs are defined as having a native width, which is the maximum width of the DQ interface supported, and an optional range of narrower programmed widths. The following sections describe the addressing considerations as a function of the native and programmed device widths.

Column Addressing

The point-to-point nature of the XDR memory architecture allows all memory devices in the system to be accessed in parallel. Consequently, system page size (typically) grows with the number of devices installed in the system. Column addressability, in turn, must proportionally scale with the number of devices in the system and, therefore, inversely with the programmed device width. The incremental column address bits used to achieve this increased addressability are used within the DRAM to select the appropriate target subset of the column for the current transaction and the datapath routing to/from that section.

For this reason, these incremental column address bits are referred to as "sub-column" address bits (SC). To simplify the XDR DRAM interface design (that is, the block that adapts the channel interface to the core interface), while minimizing latency impact, the sub-column address bits will be assigned to fixed positions within the COL/COLM packet across all device densities and organizations. The SC bits are used to select the target section of the page (COL I/O amps) as shown in Table 3 below.

If a device does not support programmable device width, it will not use sub-column (SC) bits — only column address bits (C). Similarly, if a device supports a limited range of programmable widths, only those SC bits corresponding to valid device widths will be allocated in the Request packet. For example, if a x16 native device supports x8 and x4 programmable widths, then it would use SC[3:2] only. SC[1:0] would not be used and would be documented as "reserved" in the datasheet for that device. The column address bits decoded for a given page size and programmed device width are shown in Table 4 for all native device widths.

Table 3 **Sub-Column Decoding for COL I/O Selection and Datapath Routing**

Native Device Width	Programmed Device Width	SC Bits Decoded for Datapath Routing Control	Fraction of Column I/O Amps Selected	Native Device Width	Programmed Device Width	SC Bits Decoded for Datapath Routing Control	Fraction of Column I/O Amps Selected
x32	x32	None	All	x8	x8	None	All
	x16	SC[4]	1 / 2		x4	SC[2]	1 / 2
	x8	SC[4:3]	1 / 4		x2	SC[2:1]	1 / 4
	x4	SC[4:2]	1 / 8		x1	SC[2:0]	1 / 8
	x2	SC[4:1]	1 / 16	x4	x4	None	All
	x1	SC[4:0]	1 / 32		x2	SC[1]	1 / 2
x16	x16	None	All		x1	SC[1:0]	1 / 4
	x8	SC[3]	1 / 2	x2	x2	None	All
	x4	SC[3:2]	1 / 4		x1	SC[0]	1 / 2
	x2	SC[3:1]	1 / 8	x1	x1	None	All
	x1	SC[3:0]	1 / 16				

Netlist EX2009-p. 6
Samsung v Netlist
IPR2023-00454

Table 4 **Column Addressing for Various Device Page Sizes and Widths (BL = 16)***

Device Page Size	Max Bank Count	Prog Device Width	Column Address Bits Decoded					
			x32 Native Device	x16 Native Device	x8 Native Device	x4 Native Device	x2 Native Device	x1 Native Device
1 KB	32	x32	C[8:5]					
		x16	C[8:5], SC[4]	C[8:4]				
		x8	C[8:5], SC[4:3]	C[8:4], SC[3]	C[8:3]			
		x4	C[8:5], SC[4:2]	C[8:4], SC[3:2]	C[8:3], SC[2]	C[8:2]		
		x2	C[8:5], SC[4:1]	C[8:4], SC[3:1]	C[8:3], SC[2:1]	C[8:2], SC[1]	C[8:1]	
		x1	C[8:5], SC[4:0]	C[8:4], SC[3:0]	C[8:3], SC[2:0]	C[8:2], SC[1:0]	C[8:1], SC[0]	C[8:0]
2 KB	32	x32	C[9:5]					
		x16	C[9:5], SC[4]	C[9:4]				
		x8	C[9:5], SC[4:3]	C[9:4], SC[3]	C[9:3]			
		x4	C[9:5], SC[4:2]	C[9:4], SC[3:2]	C[9:3], SC[2]	C[9:2]		
		x2	C[9:5], SC[4:1]	C[9:4], SC[3:1]	C[9:3], SC[2:1]	C[9:2], SC[1]	C[9:1]	
		x1	C[9:5], SC[4:0]	C[9:4], SC[3:0]	C[9:3], SC[2:0]	C[9:2], SC[1:0]	C[9:1], SC[0]	C[9:0]
4 KB	16	x32	C[10:5]					
		x16	C[10:5], SC[4]	C[10:4]				
		x8	C[10:5], SC[4:3]	C[10:4], SC[3]	C[10:3]			
		x4	C[10:5], SC[4:2]	C[10:4], SC[3:2]	C[10:3], SC[2]	C[10:2]		
		x2	C[10:5], SC[4:1]	C[10:4], SC[3:1]	C[10:3], SC[2:1]	C[10:2], SC[1]	C[10:1]	
		x1	C[10:5], SC[4:0]	C[10:4], SC[3:0]	C[10:3], SC[2:0]	C[10:2], SC[1:0]	C[10:1], SC[0]	C[10:0]
8 KB	8	x32	C[11:5]					
		x16	C[11:5], SC[4]	C[11:4]				
		x8	C[11:5], SC[4:3]	C[11:4], SC[3]	C[11:3]			
		x4	C[11:5], SC[4:2]	C[11:4], SC[3:2]	C[11:3], SC[2]	C[11:2]		
		x2	C[11:5], SC[4:1]	C[11:4], SC[3:1]	C[11:3], SC[2:1]	C[11:2], SC[1]	C[11:1]	
		x1	C[11:5], SC[4:0]	C[11:4], SC[3:0]	C[11:3], SC[2:0]	C[11:2], SC[1:0]	C[11:1], SC[0]	C[11:0]

* For BL=32 devices, the most-significant bit of the column address is one less than what is shown in Table 4. Because twice as much data is transferred per access, one less column address bit is required for BL=32 devices relative to BL=16 devices.

Netlist EX2009-p. 7
Samsung v Netlist
IPR2023-00454

Row Addressing

Row addressing is simpler than column addressing, although it is also affected by programmable/variable width support. Because an XDR memory system accesses all devices in parallel, system-level page size would typically grow in proportion to the number of devices installed. For large capacity configurations, system page sizes can grow quite large relative to operating system page sizes and it is unlikely that the system would derive much benefit from a system page size this large. It is generally desirable to place a maximum limit on the system page size that will be supported. By limiting system-level page size, row activation power is commensurately reduced, minimizing the burden on the power management/cooling solution and power supply decoupling network.

To limit system-level page size, XDR architecture supports the use of sub-page activation, in which only a portion of the page is sensed during an Activate command. Column operations must then be directed to the sensed portion of the page. Before accessing the remainder (unsensed portion) of the page, a precharge must be performed to the open bank followed by a sub-page activation to the new sub-page. The XDR architecture supports both half-page and quarter-page activation, depending upon the device width selected relative to the native device width.

Many DRAM core designs already provide sufficient partitioning of the wordlines to achieve sub-page activation. As device width is narrowed, transactions can be routed to the activated sub-page.

Sub-page addressing will be handled via the use of Sub-Row (SR) address fields within the ROWA packet. The XDR DRAM protocol provides support for up to two SR bits, allowing page size to be reduced by a factor of two (half-page activation) or four (quarter-page activation). Row addressing for sub-page activation is shown in Table 5. Sub-Row fields would typically be mapped so as to carry the same address bits as the the most-significant Sub-Column address bits, since these are used for datapath routing control.

The planned range of sub-page activation for XDR architecture is illustrated in Table 6. Letters in parentheses correspond to Examples A, B, C, and D illustrated in Figure 2 of how half-page activation could be applied within the XDR DRAM. In these examples, a 4-bit slice of a x16/x8/x4 XDR DRAM is shown. When using half-page activation as shown, SR[1] must be equal to SC[3] to ensure that the transaction is routed to the correct set of column I/O amps for the activated sub-page.

Please refer to the Refresh section later in this document for details on how sub-page activation affects refresh.

Table 5 **Sub-Page Addressing Support**

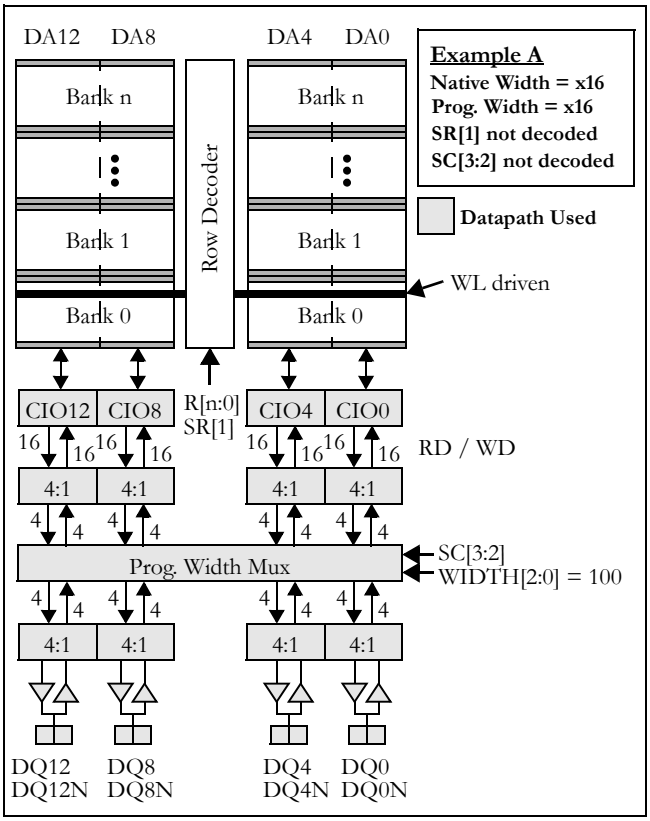
Smallest Page Activation Supported	Page Activation Selected	Row Address Bits Decoded
Full	Full	R[n:0]
Half	Full (A)	R[n:0]
	Half (B,C,D)	R[n:0], SR[1]
Quarter	Full	R[n:0]
	Half	R[n:0], SR[1]
	Quarter	R[n:0], SR[1:0]

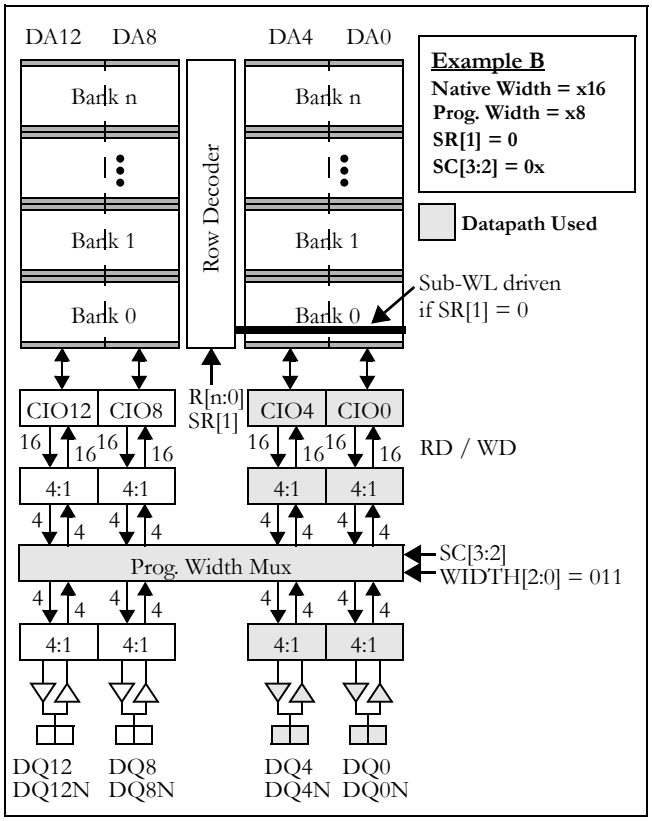
Table 6 **Planned Range of Sub-Page Support for XDR Architecture**

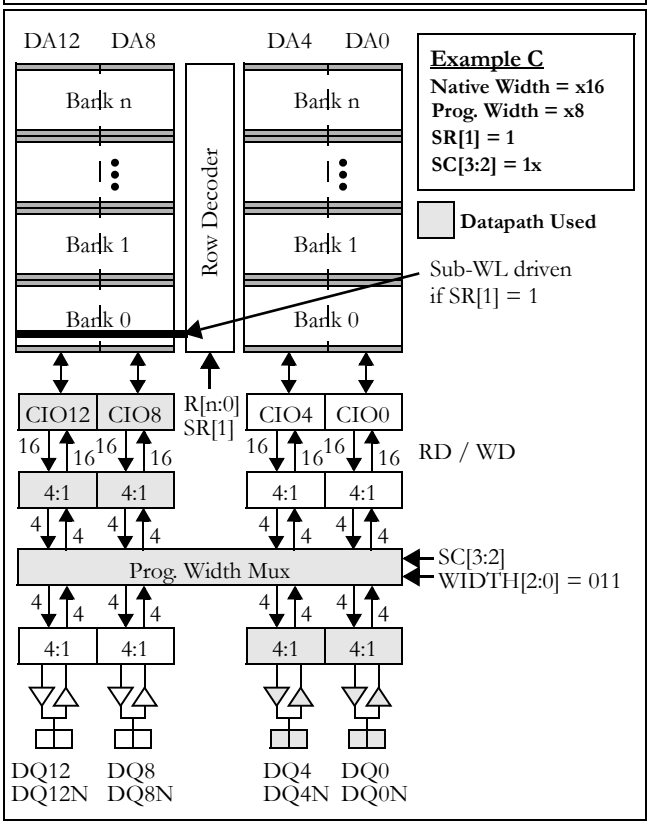
Programmed Device Width	Page Activation Modes Supported (R = Required O = Optional)					
	x32 Native Device	x16 Native Device	x8 Native Device	x4 Native Device	x2 Native Device	x1 Native Device
x32	Full					
x16	TBD	Full (A)				
x8	TBD	Full O: Half (B,C)	Full			
x4	TBD	Full O: Half (D)	Full O: Half	Full		
x2	TBD	R: Half O: Quarter	R: Half	R: Half	Full	
x1	TBD	R: Half O: Quarter	R: Half O: Quarter	R: Half O: Quarter	R: Half	Full

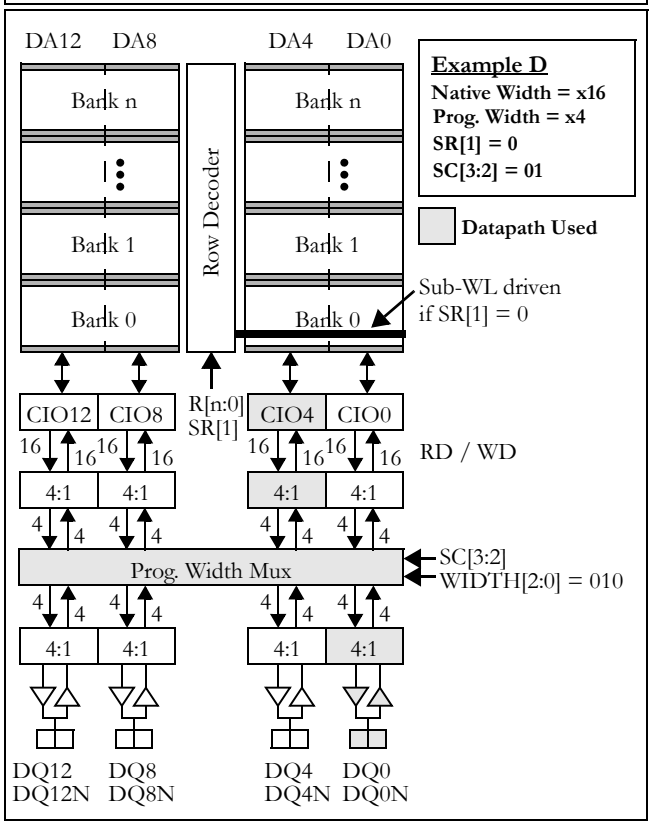
Netlist EX2009-p. 8
Samsung v Netlist
IPR2023-00454

Figure 2 Example Half-Page Activation for x16/x8/x4 XDR DRAM







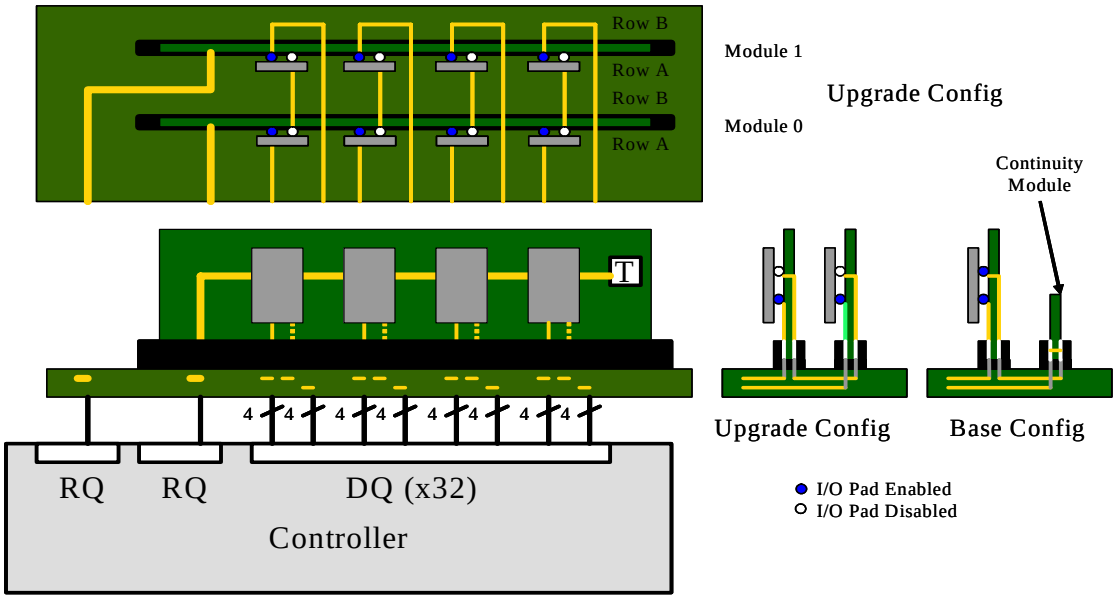


Dynamic Point-to-Point Configurations

The XDR architecture makes exclusive use of point-to-point data connections to deliver maximum performance while still allowing module upgradability. The upgradability solution is called a Dynamic Point-to-Point (DPP) topology. An example x32 DPP topology with one RQ channel per module is shown in Figure 3.

In a DPP system, the base configuration uses a single memory module and a single continuity module. The modules may be installed in any order. For the base configuration, the installed memory module uses the full 32-bit data path for data transactions. When the system is upgraded, the continuity module is removed and a second memory module is installed. The system detects the installation of the second memory module and programs each of the modules to half its normal data width or 16 bits. In order for the system to exploit the full bandwidth of the memory system, accesses must be paired between the two modules. Access pairing and alternate RQ channel topologies will be covered in a future revision of this document.

Figure 3 Dynamic Point-to-Point Topology with Separate RQ Channel Per Module (x32 DQ Channel)



The DRAM device data connections that will be used depend upon the number of devices per module and the number of modules per system. Assuming a 32-bit wide module (36-bits wide with ECC), Table 7 indicates the device data connections used for each module and system configuration currently planned for the XDR architecture.

Table 7 **Device Connections Usage Versus System Configuration and Module Device Count**

Devices Per Module	Modules Per DQ Channel	Module DQ Width	Device Width	Device Data Connections	Row A Connections	Row B Connections
2d	1	x32	x16	DQ0-15	DQ0-7	DQ8-15
	2	x16	x8	DQ0-7		
4d/5d	1	x32/x36	x8	DQ0-7	DQ0-3	DQ4-7
	2	x16/x18	x4	DQ0-3		
8d/9d	1	x32/x36	x4	DQ0-3	DQ0-1	DQ2-3
	2	x16/x18	x2	DQ0-1		
16d/18d	1	x32/x36	x2	DQ0-1	DQ0	DQ1
	2	x16/x18	x1	DQ0		

Netlist EX2009-p.10
Samsung v Netlist
IPR2023-00454

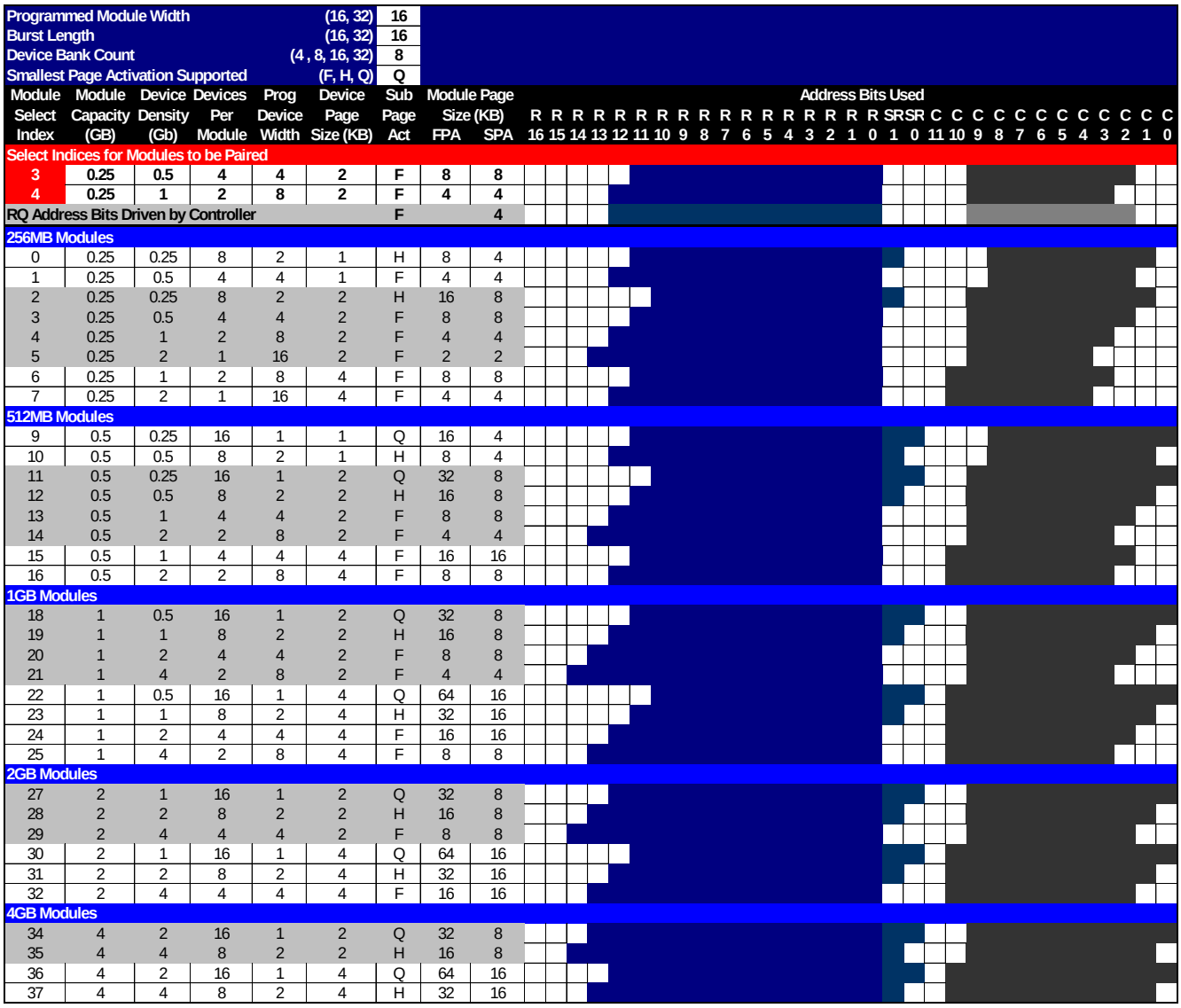
Memory System Addressing

In the XDR architecture, as the number of memory devices per request channel is increased, device width is reduced to maintain a fixed datapath width. DPP module topologies allow the module width to be cut in half for upgraded configurations, allowing paired access between two modules. The XDR architecture supports up to 18 devices per module for ECC configurations and up to 16 devices per module for non-ECC configurations.

The memory controller must generate an appropriate address factoring in the number of devices, device width, device bank/row/col organization, sub-page activation mode, and burst length. Figure 4 shows the address bits which must be generated for an example set of these parameters (Module Width = x16, Bank Count = 8, Page Size = 1KB/2KB/4KB, Burst Length = 16, Full-Page Activation). There are obviously many combinations of these parameters across the architectural range. Rambus can provide additional tools and information to analyze addressing solutions for a particular target application.

Figure 4 Example XDR Module Addressing

Programmed Module Width					(16, 32)	16		
Burst Length					(16, 32)	16		
Device Bank Count					(4 , 8, 16, 32)	8		
Smallest Page Activation Supported					(F, H, Q)	Q		
Module Select Index	Module Capacity (GB)	Device Density (Gb)	Devices Per Module	Prog Device Width	Device Page Size (KB)	Sub Page Act	Module Page Size (KB) FPA	Module Page Size (KB) SPA
Select Indices for Modules to be Paired								
3	0.25	0.5	4	4	2	F	8	8
4	0.25	1	2	8	2	F	4	4
RQ Address Bits Driven by Controller						F		4
256MB Modules								
0	0.25	0.25	8	2	1	H	8	4
1	0.25	0.5	4	4	1	F	4	4
2	0.25	0.25	8	2	2	H	16	8
3	0.25	0.5	4	4	2	F	8	8
4	0.25	1	2	8	2	F	4	4
5	0.25	2	1	16	2	F	2	2
6	0.25	1	2	8	4	F	8	8
7	0.25	2	1	16	4	F	4	4
512MB Modules								
9	0.5	0.25	16	1	1	Q	16	4
10	0.5	0.5	8	2	1	H	8	4
11	0.5	0.25	16	1	2	Q	32	8
12	0.5	0.5	8	2	2	H	16	8
13	0.5	1	4	4	2	F	8	8
14	0.5	2	2	8	2	F	4	4
15	0.5	1	4	4	4	F	16	16
16	0.5	2	2	8	4	F	8	8
1GB Modules								
18	1	0.5	16	1	2	Q	32	8
19	1	1	8	2	2	H	16	8
20	1	2	4	4	2	F	8	8
21	1	4	2	8	2	F	4	4
22	1	0.5	16	1	4	Q	64	16
23	1	1	8	2	4	H	32	16
24	1	2	4	4	4	F	16	16
25	1	4	2	8	4	F	8	8
2GB Modules								
27	2	1	16	1	2	Q	32	8
28	2	2	8	2	2	H	16	8
29	2	4	4	4	2	F	8	8
30	2	1	16	1	4	Q	64	16
31	2	2	8	2	4	H	32	16
32	2	4	4	4	4	F	16	16
4GB Modules								
34	4	2	16	1	2	Q	32	8
35	4	4	8	2	2	H	16	8
36	4	2	16	1	4	Q	64	16
37	4	4	8	2	4	H	32	16



Netlist EX2009-p. 11
Samsung v Netlist
IPR2023-00454

Maximum Capacity Configurations

To understand the maximum number of address bits required across the architectural range, it is helpful to consider the worst-case scenario of a maximum capacity module. A 16d x16 (or 18d x18 with ECC) module configuration represents the scenario requiring the maximum number of address bits in the protocol.

Figure 5 shows the number of Bank, Column (plus Sub-Column), and Row Address bits for a 16d x16 module using x1 devices and burst length (BL) = 16 for all device densities and page sizes supported within the XDR architectural range. For BL = 32 devices, the number of Column (plus Sub-Column) address bits is one less than the number shown in Figure 5.

The number of Row Address bits required is a function of device density, page size, bank count, and burst length, and therefore is not a direct function of device width or device count.

The boxes shown in dark gray with white numerals represent configurations which are not supported by the XDR protocol due to a limit of 15 address bits in the COL/COLM packets for BC, C, and SC fields. This means that a 16-bank device can support a maximum of a 4KB page and a 32-bank device can support a maximum of a 2KB page. Devices with bank counts this large are not anticipated to require page sizes beyond the limits specified above.

For devices which support Sub-Page Activation, additional SR bits may be used to decrease the effective module page size. In the extreme case of an 8Gb device with a 1KB page size (viewed to be an unlikely scenario due to die size overhead), SR[0] could potentially be re-mapped as the most-significant row address bit, since quarter-page activation would be unnecessary for a device with such a small page size.

Figure 5 Module Addressing for 16d x 16 Module (BL=16)

16d x16 Module Organizations Supported																	
Devices per module		16	(excluding ECC)														
Programmed Device Width		x1	(upgraded DPP configuration)														
Log Transfer Size (bits)		5	(32 bytes per column access)														
Device Page Size (KB)			1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8
Module Page Size (KB)			16	32	64	128	16	32	64	128	16	32	64	128	16	32	64	128
C+SC address bits			9	10	11	12	9	10	11	12	9	10	11	12	9	10	11	12
BANK address bits			2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5
Density (Gb)	**Module Capacity (GB)**	**Addr Bits**	**Row Address Bits**															
0.25	0.5	**29**	13	12	11	10	12	11	10	9	11	10	9	8	10	9	8	7
0.5	1	**30**	14	13	12	11	13	12	11	10	12	11	10	9	11	10	9	8
1	2	**31**	15	14	13	12	14	13	12	11	13	12	11	10	12	11	10	9
2	4	**32**	16	15	14	13	15	14	13	12	14	13	12	11	13	12	11	10
4	8	**33**	17	16	15	14	16	15	14	13	15	14	13	12	14	13	12	11
8	16	**34**	18	17	16	15	17	16	15	14	16	15	14	13	15	14	13	12

Netlist EX2009-p.12
Samsung v Netlist
IPR2023-00454

Module Addressing For Shared Request Channel

The XDR architecture supports a maximum of two device types per request channel. This allows a request channel to be shared between two memory modules (for pin-savings) or for two different memory device organizations on a single module (for optimized ECC support, for example). To support this feature, some special addressing logic is required in the memory controller to generate a single address which is compatible with both device types on the request channel.

Referring back to Figure 4, an example is provided in which we desire to address two identical capacity modules using different density devices. In this case, the modules are referenced by the indices 3 and 4 in the table. Module 3 is a 256MB module comprised of four 512Mb x4 devices. Module 4 is a 256MB module comprised of two 1Gb x8 devices. In this example, both modules use devices having eight banks, burst length = 16, 2KB page size, and full-page activation. The addresses required for each module are as follows:

Module 3 Address: {R[11:0], C[9:2]}
Module 4 Address: {R[12:0], C[9:3]}

In this case, the controller must generate a single address which is compatible with both modules. Because Module 3 does not decode R[12] and Module 4 does not decode C[2], it is possible for the controller to generate the following address which works for both modules:

Controller Address: {R[12:0], C[9:2]}, C[2] == R[12]

In this case, the memory controller can be programmed for the module configuration requiring the greatest number of row address bits — in this case, Module 4 — while allowing the most-significant row bit (R[12]) to be re-mapped to the least-significant column address bit (C[2]). The module having the greatest number of row bits will also have the smallest module page size. For closed-page policy memory controllers, the difference in module page size will not generally affect normal access. For open-page policy controllers, the controller can be simplified by setting the effective module page size to the smaller of the two — in this case, 4KB. Using this scheme, both modules will be activated and precharged at the same time, simplifying the controller design.

As a second example, consider the case when Module 3 and Module 6 are paired. The primary difference now is that Module 6 uses devices having a 4KB page size, while Module 3 uses devices having a 2KB page size. For this example, the following addresses are required:

Module 3 Address: {R[11:0], C[9:2]}
Module 6 Address: {R[11:0], C[10:3]}

In this example, instead of remapping the most-significant row bit, the most-significant column bit should be remapped as follows:

Controller Address: {R[11:0], C[10:2]}, C[2] == C[10]

Address Optimization for ECC Configurations

As a third example, consider the case where it is desired to combine two different device types on one module. This might be desirable when using a x36/x18 ECC module with five devices. For example, consider the following module configuration:

Device Type A:	512Mb x4
Device Type B:	256Mb x2
Module Cfg:	4 Device A + 1 Device B
Device A Address:	{R[11:0], C[9:2]}
Device B Address:	{R[10:0], C[9:1]}

In this example, the Device A organization matches that of the devices used in Module 2 of Figure 4, while the Device B organization matches that of the devices used in Module 3. The controller address can be generated as follows:

Controller Address: {R[11:0], C[9:1]}, C[1] == R[11]

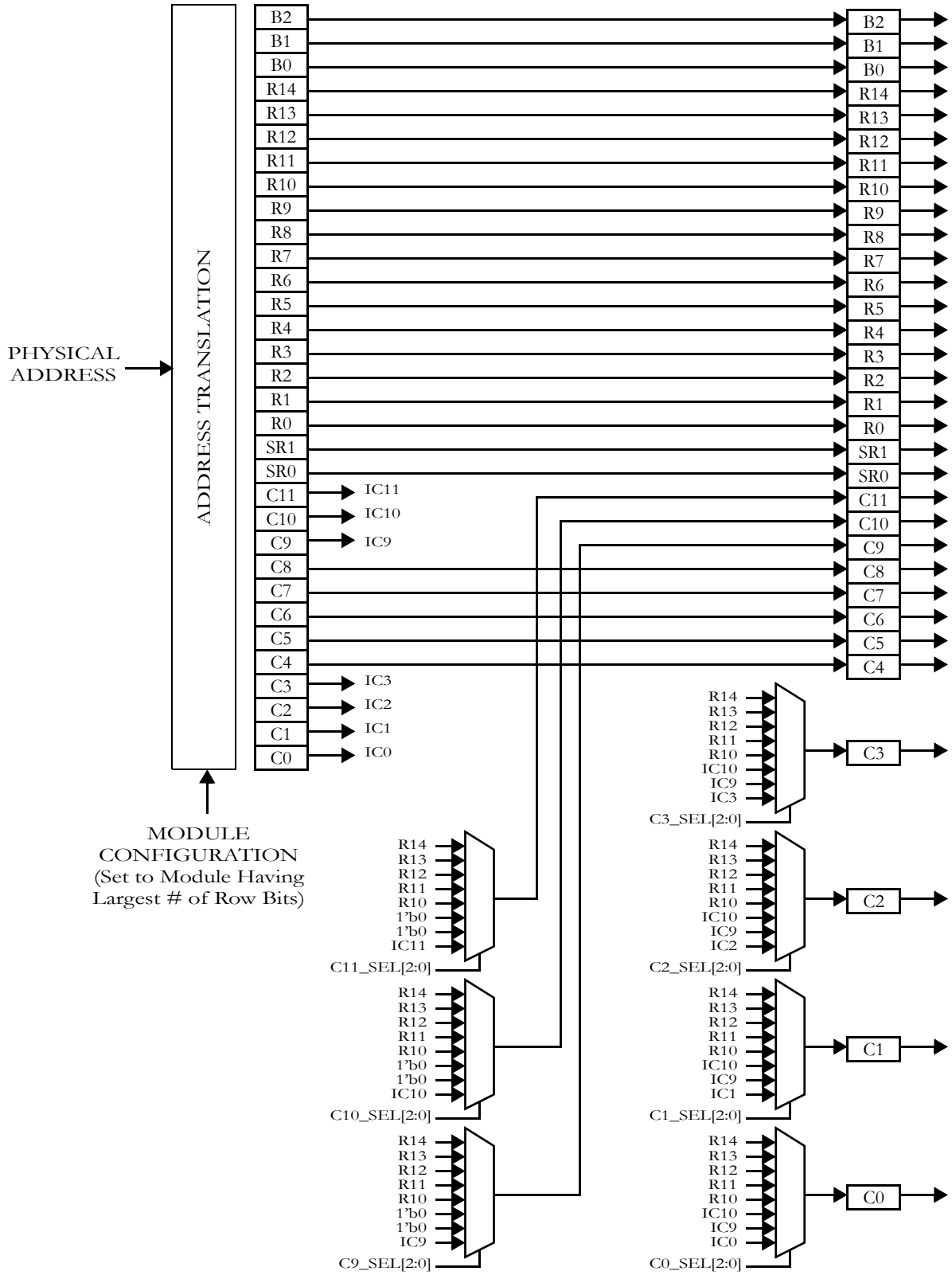
Memory Controller Address Remapping

In each of the examples above, the address generated by the controller is a superset address requiring either the most-significant row address bit or most-significant column address bit to be remapped to one of the least-significant column address bits. It is possible to define some simple hardware in the memory controller to support this type of remapping function. Depending upon the requirements of the particular application, the optimal number of address bits to remap will vary.

Figure 6 shows an example diagram of remapping logic which allows a high degree of flexibility for address remapping. This level of support is likely sufficient for most applications which mix device types on a single request channel. In this example, configuration software can select independent mappings for each of the seven address bits which support remapping. Further optimizations are possible to reducing the amount of muxing hardware. Latency impact should be negligible for closed-page policy controllers and minimal for page hits using open-page policy controllers.

Netlist EX2009-p. 13
Samsung v Netlist
IPR2023-00454

Figure 6 Example Address Remapping Logic



Netlist EX2009-p.14
Samsung v Netlist
IPR2023-00454

XDR Architecture Features

The XDR architecture supports a base set of features and a limited number of optional features. Rambus logic customers and memory partners can work together with Rambus to determine the appropriate set of optional features to include in a particular device. Table 8 lists the features supported by the XDR architecture and a recommended set of features for initial product implementations.

Table 8 **XDR Architectural Features**

Feature Class	Features Supported by Architecture	Recommended Initial Features	
		XDR DRAMs	Memory Controllers
XDR DRAM Width	x32/x16/x8/x4/x2/x1	Type A: x16/x8/x4 Type B: x4/x2/x1	Consumer: x16/x8/x4 Main Memory: x16/x8/x4/x2/x1
Sub-Page Activation	Full, Half, Quarter	Type A: Full Type B: x4-Full, x2-Half, x1-Quarter	Full, Half, Quarter
ECC/Chipkill Support	9th device per 8 device group: 5d x8/x16* 9d x4/x8 18d x2/x4 36d x1/x2 * 90% capacity utilization	See XDR DRAM width recommendations above	Support only those modes required for the product
Module Upgradability	Dynamic Point-to-Point (DPP) Modules	Minimum of two register-programmable width options per device type	Provide addressing support if upgradable modules are a product requirement
Write Masking	- WRM with 8-bit Mask Key - Read-Modify-Write	- WRM with 8-bit Mask Key - RD/WR commands	- WRM with 8-bit Mask Key - Read-Modify-Write
Efficiency	- Early Read (Disable option) - 4-Deep Write Buffer (Disable option) - Simultaneous Activate - Simultaneous Precharge	- Early Read - Simultaneous Precharge	- Early Read (Disable option) - Simultaneous Precharge
Refresh	- Self-Refresh (Powerdown) - Single-Bank Refresh (Active) - Simultaneous Refresh (Active)	- Self-Refresh (Powerdown) - Single-Bank Refresh (Active)	- Self-Refresh (Powerdown) - Single-Bank Refresh (Active)
Timing Calibration (TCAL)	- Initial XIO TCAL - Periodic XIO TCAL	- Initial XIO TCAL (WDSL)	- Initial XIO TCAL
Current Calibration (CCAL)	- Initial/Periodic XIO RQ CCAL - Initial/Periodic XDR DRAM CCAL - Continuous XIO DQ CCAL	- Initial/Periodic XDR DRAM CCAL	- Initial/Periodic XIO RQ CCAL - Initial/Periodic XDR DRAM CCAL
Termination Impedance Calibration (ZCAL)	- Initial/Periodic XIO ZCAL - Initial/Periodic XDR DRAM ZCAL - XDR DRAM Resistor Trimming	- XDR DRAM Resistor Trimming	- Initial/Periodic XIO ZCAL - Initial/Periodic XDR DRAM ZCAL

Netlist EX2009-p. 15
Samsung v Netlist
IPR2023-00454

XDR DRAM Width

XDR DRAM Width Considerations

Probably the most important feature for XDR memory systems is the variable width of the XDR DRAM devices. The XDR architecture supports x32, x16, x8, x4, x2, and x1 devices.

This variable width allows maximum performance across a wide range of system capacities while providing excellent opportunities for device-level cost reduction and system-level power reduction. A particular XDR DRAM die will typically be designed to support a range of widths. The width may be either selected via a register setting, bond/fuse options, metal mask options, or some combination of these.

There are a number of factors to consider when determining the optimal range of widths for a particular device. Designing a common die for multiple widths helps to save design and validation resources, although reducing the range of width options in a shipping product can help to reduce the test time and therefore cost. However, providing more width options in a shipping product can help improve inventory management, reduce reliance upon customer demand mix projections, and increase the number of applications for a given device. Furthermore, by incorporating at least two register-programmable width options in a device, module upgrade capabilities are enhanced.

For example, an XDR DRAM die could be designed to support x16/x8/x4 widths. It could be fuse-programmed to be either a x16-only device or a x8/x4 device and sold with two separate part numbers. The x16 device could be sold into a fixed configuration consumer device while the x8/x4 device could be sold into an upgradable main memory application.

For applications requiring high capacity, cost savings can be realized by reducing the native width of the device and therefore the peak core bandwidth of the device. For example, a x1 XDR device at 3.2Gbps signaling rate only requires 400MB/sec of core bandwidth — very low by modern standards. However, 64 of these devices can be accessed in parallel to yield a memory subsystem having 25.6GB/sec of peak bandwidth — extremely high by modern standards. The use of sub-column I/O gating along with sub-page activation in conjunction with narrow device widths helps to reduce device and system power to very reasonable levels.

Memory Controller Considerations for Width

Because the memory controller is mainly concerned with addressing a rank of devices instead of a single device, the memory controller will need to vary the number of Sub-Row (SR) and/or Sub-Column (SC) address bits within the Request Packet as device width and device count are varied. This is described in the section above on Addressing Considerations.

Sub-Page Activation

As described above, the XDR architecture supports the use of sub-page activation as device width is reduced (that is, as system capacity is increased) to minimize peak activation power across multiple parallel devices. In addition to power savings, this helps to decrease the burden on the system cooling solution and power supply decoupling network. This may also help to improve yield for certain core timing parameters.

XDR DRAM Sub-Page Considerations

To support sub-page activation, XDR DRAMs will need to provide some additional support in the row decoders of the XDR DRAM. The row decoders will need to receive the SR bits during an Activate operation and drive the appropriate sub-wordline.

Datapath routing logic in the Rambus interface routes data to/from the appropriate core data lane as a function of the programmed width and SC bits provided.

Memory Controller Sub-Page Considerations

The memory controller must generally ensure that the sub-page activated by the choice of SR bits corresponds to the sub-page accessed by the choice of SC bits in subsequent column operations. As an example, for a x16/x8 device which supports half-page activation in x8 mode, SR[1] should always be equal to SC[3].

The memory controller designer will need to understand whether the addition of more devices increases system page size or whether it remains the same due to the use of sub-page activation. This information will be available from the Serial Presence Detect (SPD) ROM in module-based systems.

Netlist EX2009-p.16
Samsung v Netlist
IPR2023-00454

ECC/Chipkill Support

XDR architecture supports reliability features such as ECC and Chipkill through the use of a ninth device per group of eight devices. For a x36 data interface, supported configurations include 9d x4, 18d x2, and 36d x1. For configurations requiring fewer than nine devices, five devices are also supported with 90% of the memory capacity utilized (5d x8).

XDR DRAM ECC/Chipkill Considerations

There are no special XDR DRAM considerations to support ECC/Chipkill. However, ECC modules may need to accommodate nine devices per side which will set a maximum limit on the horizontal XDR DRAM package dimension.

Memory Controller ECC/Chipkill Considerations

Memory Controllers and XIOs which support ECC and/or Chipkill need to support x36 or x72 data interfaces and integrate appropriate ECC/Chipkill syndrome generation, error checking, and error correction capability. There will generally be some latency and area impact to integrate this functionality.

Module Upgradability (DPP)

As described above, the XDR architecture supports a Dynamic Point-to-Point (DPP) topology to allow upgradable module solutions while preserving point-to-point data connections.

XDR DRAM DPP Considerations

From the XDR DRAM device-level perspective, the key feature that is required for DPP support is support for a minimum of two width options: full-width and half-width, selectable via a register setting. The full-width mode will be used for base module configurations and the half-width mode will be used for upgraded module configurations. In the event that the devices used on the two modules support different default interface timing parameters (for example, tCAC and tCWD), it is recommended that the memory device support programming options to set the two devices to a common configuration. All other aspects of DPP support are handled at the module or system level — not the device level.

Memory Controller DPP Considerations

As described in the section above on Addressing Considerations for DPP Systems, the memory controller will need to be configured to pair up equivalent capacity regions of the two modules and handle addressing of any un-paired region.

Write Masking

Byte-level write masking is an optional feature of the XDR architecture. The write masking method used allows write masking with no additional interface signals and provides support across all possible device widths.

The decision to implement write masking at the XDR DRAM level should be viewed as a joint decision between Rambus memory partners, logic customers, and Rambus. Considerations for implementing write masking in both the XDR DRAM and the memory controller are provided in the sections below.

XDR DRAM Write Masking Considerations

XDR DRAMs provide support for byte-level write masking via an 8-bit mask key which is embedded in the COLM packet. Data is organized so that all bits of a byte are transmitted across a single DQ link. During a masked write (WRM) command, the mask key is sampled and propagated to the XDR DRAM datapath, where it is compared against incoming data bytes. Write-enable signals for data bytes which match the mask key are suppressed, to prevent the column I/O drivers from overwriting the data which is present at that byte location.

XDR DRAM cores must generally support some level of write masking for programmable-width support. Therefore, the incorporation of byte-level write masking has minimal impact upon the core design for programmable-width devices.

Memory Controller Write Masking Considerations

To support byte-level write masking, memory controller customers will need to integrate a mask generation and insertion block called mask_gen. The mask_gen block is a datapath-only block which receives a write data packet along with corresponding write enables, generates the 8-bit mask key, and inserts the mask key for masked bytes. Rambus provides a synthesizable RTL reference design for the mask_gen block. Throughput for this block is one data packet per PCLK cycle, with a nominal latency of two PCLK cycles (equivalent to one data packet time). Actual latency may vary depending upon the data packet size and target process technology. The mask_gen block for a design featuring 64-byte access granularity with a 2-cycle latency is approximately 5K gates.

Rambus recommends that logic customers who support byte-level write masking also provide support for Read-Modify-Write (RMW) to achieve an optimal forward compatibility strategy with future XDR DRAM devices. The RD and WR commands provided in the protocol are sufficient for this purpose.

Netlist EX2009-p. 17
Samsung v Netlist
IPR2023-00454

W/R Turnaround Improvement

In any DRAM which provides a single port for driving write data and sensing read data, a resource conflict will exist within the device which introduces wait states (that is, a bubble) on Write-to-Read (W/R) turnarounds. This bubble decreases the efficiency of the memory system and can increase read latency. The XDR architecture provides two options for minimizing the size of a Write-to-Read (W/R) turnaround bubble on the DQ interface: Write Buffer and Early Read. The decision as to which of these features to implement should be viewed as a joint decision between Rambus memory partners, logic customers, and Rambus.

XDR DRAM Considerations for Write Buffer

The Write Buffer option includes support for a 4-deep Write Buffer within the XDR DRAM. The size of each entry of the Write Buffer corresponds to the width of the core datapath interface. For a x16 device with a Burst Length (BL) of 16, each Write Buffer entry is 256 bits, resulting in a total Write Buffer size of 1024 bits. The Write Buffer allows write data to be written to a circular queue during a W/R turnaround, so that read data may be returned immediately.

Pointer management of the Write Buffer is handled by the Rambus interface logic. As long as the correct protocol is followed, the pointer management should be transparent to the memory controller designer. Please refer to the XDR DRAM datasheet for complete details on the protocol.

For compatibility reasons, Rambus requires that any XDR DRAM designs implementing a Write Buffer also provide a mode bit to disable this feature. Therefore, any XDR DRAM designs implementing a Write Buffer will support two modes: Write Buffer Enabled and Write Buffer Disabled.

Memory Controller Considerations for Write Buffer

The primary objective of the memory controller in using the Write Buffer is to ensure that write data has been loaded to the Write Buffer (called a Transport step) prior to the time that the data must be sourced from the Write Buffer for delivery to the core (called a Retire step). A Preload command (PLU/PLM) can be issued as part of a COLX packet to perform the Transport step with no associated Retire step. A Write (WR/WRM) command first performs a Retire of the next data entry in the Write Buffer, and then performs a Transport of the incoming data to the next available Write Buffer location (generally the entry vacated during the Retire step). The number of Preload commands issued determines the maximum number of back-to-back writes which can be issued. When four Preloads are issued, the Write Buffer is full, and a continuous series of Write commands can be sustained, as long as valid data is provided during the data phase of the Write command (a Transport step for a future write).

While it is possible to flush the Write Buffer contents to the core, this is generally not recommended. The recommended usage model is to keep the Write Buffer full and for the Memory Controller to keep a copy of the contents of the Write Buffer available in a Local Write Buffer (LWB) until the associated Write command is issued. In the event that a read cycle hits the same address as the data stored in the LWB, data should be forwarded from the LWB.

In the event that it is desired to flush the Write Buffer, a series of four Write commands with dummy data can be issued. This will not change the relative positions of the Transport and Retire pointers within the XDR DRAM (assuming the buffer was previously full). However, before a series of Writes can resume, four Preloads should be issued to fill up the Write Buffer again.

XDR DRAM Considerations for Early Read

Early Read is an alternative to the Write Buffer feature which can also be used to reduce the size of the bubble on a W/R turnaround. Early Read support in the XDR DRAM requires the core to be divided into two independent bank sets. One bank set of the core can be written while the other bank set is read, allowing a brief period of overlap of the write and read column cycles on a W/R turnaround. In a R/W turnaround, there will be no overlap. In a core design in which even banks are allocated to the top half of the core and odd banks are allocated to the bottom half of the core, an Early Read can be issued to the bottom half of the core as long as the previous series of Writes were directed to the top half of the core (and vice versa).

Early Read essentially allows the Read command to be issued earlier than would otherwise be possible, because it does not have to wait for the write data to be propagated through to the core. As long as there is a bank switch between independent bank sets (for example, odd-to-even or even-to-odd), the Read command can be issued at the earliest possible time to reduce the W/R turnaround bubble.

The primary constraint for the XDR DRAM is that independent sets of column decoders and column I/O drivers/sense latches be provided between the two bank sets.

Netlist EX2009-p.18
Samsung v Netlist
IPR2023-00454

Memory Controller Considerations for Early Read

Memory Controller support for Early Read involves identifying the independent bank sets from information available in the system BIOS or SPD ROM, scheduling transactions such as to guarantee that Reads are directed to an independent bank set from the preceding Writes on a W/R turnaround, and scheduling the Read commands earlier than usual in the case of a switch between opposite bank sets.

Certain core timing constraints will also differ depending upon whether an XDR DRAM implements a Write Buffer or Early Read support. In general, a Write Buffer can help to reduce the t_{WRP} and $t_{RC\text{-}W}$ parameters and makes request scheduling very similar between Reads and Writes, but Early Read has fewer dependencies that need to be tracked for scheduling.

Recommendations

Rambus generally recommends Early Read support. With proper optimization, the performance difference between Early Read and Write Buffer implementations can be very small. A disable option should also be supported for an optimal forward and backward compatibility strategy.

Turnaround Bubbles

Table 9 shows turnaround bubble size versus. turnaround direction and bubble location assuming an Early Read option. The bubbles shown in this table are measured at the DQ pins. This table provides general equations for determining both actual and minimum bubble sizes and the value of the minimum bubble size supported for first generation XDR components. Initialization software must ensure that the timing parameters $t_{\Delta RW}$ and $t_{\Delta WR\text{-}D}$ are programmed to appropriate values to meet the minimum bubble sizes based upon the rules specified in Table 14.

The R-W turnaround bubble size at the XDR DRAM will be a function of the round trip flight time of the longest DQ link ($t_{PD,MAX}$) rounded up to the nearest integer cycle count ($t_{PD,CYC}$). This value is typically obtained from the system ROM.

The following equations are used to set $t_{\Delta RW}$:

$t_{PD,MAX} = MAX(t_{PD,DQi})$ for all i

$t_{PD,CYC} = ROUNDUP(t_{PD,MAX} / t_{CYC})$

$t_{\Delta RW} \geq (t_{CAC} - t_{CWD}) + t_{CC} + t_{RW\text{-}BUB,XDRDRAM}$

$t_{RW\text{-}BUB,XDRDRAM} \geq t_{RW\text{-}BUB,XDRDRAM,MIN} + (t_{PD,CYC} - t_{PD,CYC,MIN})$

where:

$t_{PD,DQi}$ = Round-trip flight time of DQ link i

$t_{PD,MAX}$ = Round-trip flight time of longest DQ link

t_{CYC} = CFM cycle time

$t_{\Delta RW}$ = R-W command delay required for system

For first-generation XDR components, the minimum R-W bubble at the XDR DRAM will be three CFM cycles, while the minimum R-W bubble at the XIO will be two CFM cycles. Also note that bubbles at the XDR DRAM will be an integer number of CFM cycles, while bubbles at the XIO will generally be a fractional number of cycles which varies at each DQ pin.

Table 9 **System Turnaround Bubble Sizes (Early Read Option)**

Direction	Bank Set	Parameter	General Equation / Constraint	Min Bubble (1st Gen)	Units
R-W		$t_{RW\text{-}BUB,XDRDRAM}$	$= t_{\Delta RW} + (t_{CWD} - t_{CAC}) - t_{CC}$ $\geq t_{RW\text{-}BUB,XDRDRAM,MIN}$[a] $+ (t_{PD,CYC} - t_{PD,CYC,MIN})$[b]	$t_{PD,CYC}$[b] + 2	CFM Cycles
		$t_{RW\text{-}BUB,XIO,DQi}$	$= t_{RW\text{-}BUB,XDRDRAM} - t_{PD,DQi}$ $\geq t_{RW\text{-}BUB,XIO,MIN}$[c]	2	CFM Cycles
W-R	Same	$t_{WR\text{-}BUB,XDRDRAM}$	$= t_{\Delta WR} + (t_{CAC} - t_{CWD}) - t_{CC}$	9-10	CFM Cycles
	Different	$t_{WR\text{-}BUB,XDRDRAM}$	$= t_{\Delta WR\text{-}D} + (t_{CAC} - t_{CWD}) - t_{CC}$ $\geq t_{WR\text{-}BUB,XDRDRAM,MIN}$[a]	3	CFM Cycles
		$t_{WR\text{-}BUB,XIO,DQi}$	$= t_{WR\text{-}BUB,XDRDRAM} + t_{PD,DQi}$	3	CFM Cycles

a. XDR DRAM parameter (3 CFM cycles for first-generation XDR DRAM). Please refer to XDR DRAM component datasheet.

b. $t_{PD,CYC}$ is a platform-dependent parameter. $t_{PD,CYC,MIN}$ = 1 CFM cycle.

c. XIO parameter (2 CFM cycles for first-generation XIO). Please refer to XIO component datasheet.

Netlist EX2009-p. 19
Samsung v Netlist
IPR2023-00454

Refresh Support

The XDR architecture provides refresh support via Single-Bank Refresh during XDR DRAM Active power state and via Self-Refresh during XDR DRAM Powerdown state. The architecture also supports a feature called Simultaneous Refresh which is planned for future devices. Details of Simultaneous Refresh operations are described later in this section, while details of Self-Refresh operations will be provided in a future draft of this document.

Table 10 summarizes the architectural range of features and parameters related to refresh as well as recommended design and verification ranges for initial product implementations.

Please refer to the XDR DRAM component datasheet for full details on refresh transactions and constraints supported by a particular device.

During normal system operation, the key constraint which must be met for the memory system is that each row of each bank must be refreshed once in every $t_{REF,MAX}$ interval. The memory controller can choose whether to interleave refresh operations with normal transactions depending upon the aggressiveness of the design and the target efficiency.

Table 10 **Architectural Range for Refresh**

Feature / Parameter	Architectural Range	Recommended Initial Range			
		XDR DRAMs		Memory Controllers	
		Design	Verification Priority	Design	Verification Priority
Refresh Mechanisms	- Self-Refresh - Single-Bank Refresh - Simultaneous Refresh	- Self-Refresh - Single-Bank Refresh	- Self-Refresh - Single-Bank Refresh	- Self-Refresh - Single-Bank Refresh	- Self-Refresh - Single-Bank Refresh
$t_{REF,MAX}$	8, 16, 32 ms	16, 32 ms (vendor-dependent)	16, 32 ms (vendor-dependent)	8, 16, 32 ms	16, 32ms
Full-Page (FPA) / Sub-Page (SPA) Refresh Activation	- FPA, SPA	FPA	FPA	FPA, SPA	FPA

Impact of Sub-Page Activation on Refresh

In sub-page activation mode, a ROWA operation will only activate a portion of a page depending upon which Sub-Row (SR) bits are provided. The XDR architecture supports both Full-Page Activation (FPA) and Sub-Page Activation (SPA) modes for refresh operations such as REFA, REFI, or REFP.

Table 10 summarizes Rambus recommendations for initial products with respect to FPA/SPA support for refresh. When SPA mode is enabled for refresh, there are effectively two or four times as many rows in the system which need to be refreshed. This will decrease average refresh power (and noise) per refresh command, but will require two to four times as many total refresh commands per $t_{REF,MAX}$ interval, increasing refresh overhead. Generally speaking, FPA should be used for refresh unless peak refresh current limits are exceeded, in which case, memory controllers should provide the ability to perform SPA during refresh. Rambus is in the process of determining peak refresh current limits for various memory subsystems.

Netlist EX2009-p.20
Samsung v Netlist
IPR2023-00454

Simultaneous Refresh Operations

The XDR DRAM component datasheet describes the use of Single-Bank Refresh. However, the timing diagram in Figure 7 illustrates another way that a memory controller might handle refresh maintenance. In this example, there is a continuous background of interleaved read transactions. These read transactions are directed to the banks so that there are no bank conflicts. It is also assumed that a closed page policy is being pursued by the controller; each transaction will open a bank, perform a read transfer, and close the bank again. This example also assumes that the $t_{RR\text{-}D}$ and $t_{PP\text{-}D}$ minimum parameters are both zero. The timing will be modified if the minimum values of the two parameters are equal to one or two t_{CYCLE}, and simultaneous refresh will not be possible if the minimum values of either of the two parameters is equal to four t_{CYCLE} or more. This transaction profile will utilize every available timing slot on the DQ and RQ buses. However, there is a refresh solution for this heavily utilized memory system which does not degrade performance. The ROWP packet permits a precharge command and a refresh activate or refresh precharge command to be simultaneously issued. Also, a refresh transaction does not require any timing slots on the DQ bus. As a result, there is an RQ slot during each ROWP packet (one out of every four RQ packets) that is available for refresh. This represents far more control bandwidth than is needed for refresh, so it is not necessary to interleave successive refresh transactions (although the protocol would permit this, if needed).

The controller can place a REFA command in the ROWP packet that is cycled on cycle T_2. A time t_{RAS} later, the controller places a REFP command in the ROWP packet that is cycled on cycle T_{14}. A time t_{RP} later, the controller places a REFA command in the ROWP packet that is cycled on cycle T_{18}.

The result is that a refresh transaction will be scheduled at the same time that four interleaved read transactions are also taking place. The one constraint is that the bank that is refreshed (bank Bp in the example) must be different from all the banks involved in the interleaved read transactions (Bx, By, Bz, Ba, Bb, Bc, and Bd) with which its t_{RC} interval overlaps.

Note that the REFA and REFP commands are using the delay mechanism in the ROWP packets (circled in gray). The delay values selected are two cycles in this example, the same as the delay of the PRE commands in the same ROWP packets. The reason for choosing this value is so that the REFA commands are executed in the same cycle as a REFA command for an interleaved read transaction. Likewise, the REFP commands are executed in the same cycle as a PRE command for an interleaved read transaction.

By aligning a refresh transaction with one of the interleaved read transactions, the refresh transaction will overlap seven read transactions and not eight. This will ensure that even in a worst-case interleaving situation like this one, there will always be a bank available to refresh.

A real controller managing a real request stream will introduce delays in the request stream to account for read-write turnaround, for write-read turnaround and for bank conflicts. The controller may also be dealing with a page policy in which some pages are left open and some are left closed. As a result, a simultaneous refresh transaction will probably overlap fewer than seven other transactions, as in this worst-case interleaved read example.

If a controller is to take full advantage of the zero-overhead benefit of simultaneous refresh, it must incorporate two hardware structures. First, the controller will need a request queue which not only keeps track of the last three transactions issued, but also has the next four transactions queued up in a FIFO (first-in-first-out) structure. Note that the transaction depths of three and four assume a closed page policy; an open page policy or a mixed page policy would have some differences.

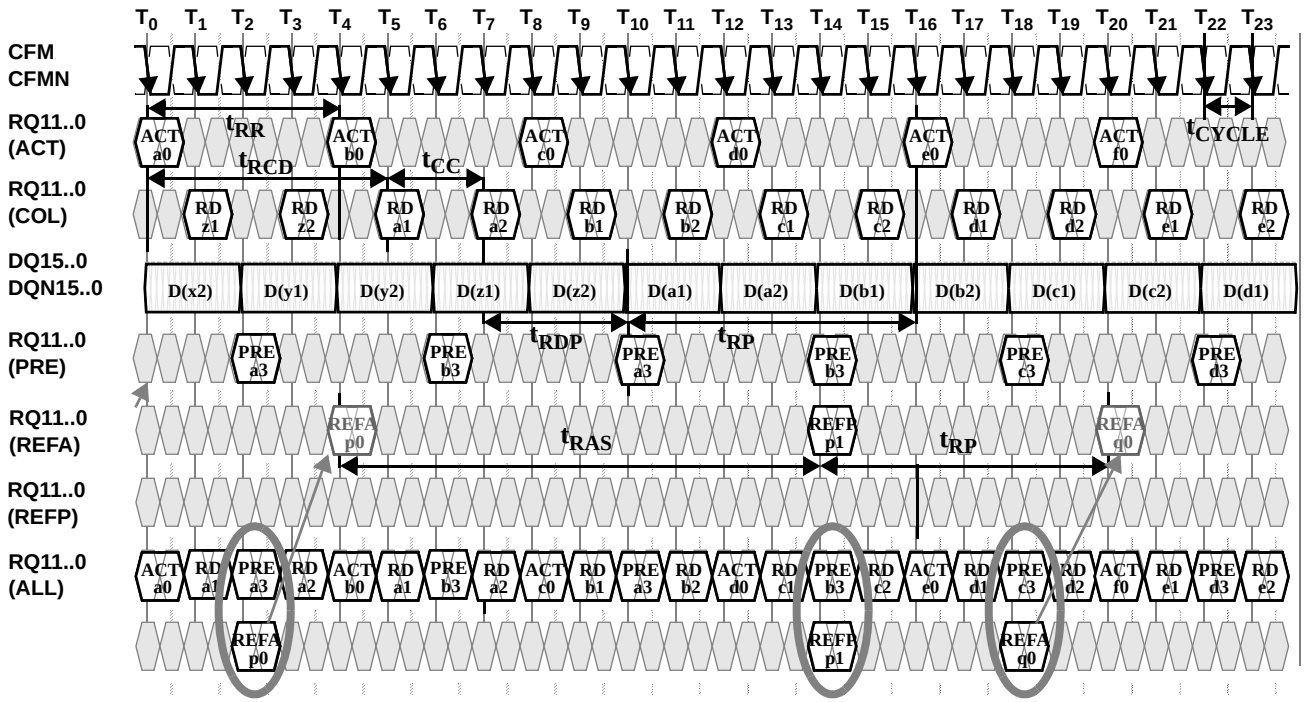
When the controller knows what banks the last three and the next four transactions will use, it will always be able to schedule a simultaneous refresh transaction to an unused bank. This simultaneous refresh transaction requires no overhead, since it will use the available ROP and RA fields of the ROWP packets that are being issued with the transactions.

The controller will require a second structure to track which banks have been refreshed with the current row value in the REFL/REFM/REFH registers (also called REFr). Because a simultaneous refresh transaction must be issued once in every 100 read/write transactions (approximately), there will be many opportunities to refresh all the banks before updating the current row value. As a last resort, the controller can delay read/write transactions to refresh a bank that is being heavily utilized.

The ROP and RA fields of the ROWP packet include commands to load the REFL/REFM/REFH registers with arbitrary values. This would permit a controller to refresh many rows in one set of banks while read and write transactions are issued to the remaining banks. This would require additional tracking structures in the controller to remember which rows of which banks have been refreshed.

Netlist EX2009-p. 21
Samsung v Netlist
IPR2023-00454

Figure 7 Simultaneous Refresh Transactions



Bank Restrictions

Bx # Bu.Bv.Bw, By,Bz,Ba	Transaction x: RD	x0 = {Bx,Rx}	x1 = {Bx,Cx1}	x2 = {Bx,Cx2}	x3 = {Bx}
By # Bv.Bw,Bx,,Bz,Ba,Bb	Transaction y: RD	y0 = {By,Ry}	y1 = {By,Cy1}	y2 = {By,Cy2}	y3 = {By}
Bz # Bw,Bx, By,Ba,Bb,Bc	Transaction z: RD	z0 = {Bz,Rz}	z1 = {Bz,Cz1}	z2 = {Bz,Cz2}	z3 = {Bz}
Ba # Bx, By,Bz,Bb,Bc,Bd	Transaction a: RD	a0 = {Ba,Ra}	a1 = {Ba,Ca1}	a2 = {Ba,Ca2}	a3 = {Ba}
Bb # By,Bz,Ba,Bc,Bd,Be	Transaction b: RD	b0 = {Bb,Rb}	b1 = {Bb,Cb1}	b2 = {Bb,Cb2}	b3 = {Bb}
Bc # Bz,Ba,Bb,Bd,Be,Bf	Transaction c: RD	c0 = {Bc,Rc}	c1 = {Bc,Cc1}	c2 = {Bc,Cc2}	c3 = {Bc}
Bd # Ba,Bb,Bc,Be,Bf ,Bg	Transaction d: RD	d0 = {Bd,Rd}	d1 = {Bd,Cd1}	a2 = {Bd,Cd2}	d3 = {Bd}
Be # Bb,Bc,Bd,Bf ,Bg.Bh	Transaction e: RD	e0 = {Be,Re}	e1 = {Be,Ce1}	e2 = {Be,Ce2}	e3 = {Be}
Bf # Bc,Bd,Be,Bg.Bh,Bi	Transaction f: RD	f0 = {Bf,Rf}	f1 = {Bf,Cf1}	f2 = {Bf,Cf2}	f3 = {Bf}

Bank Restrictions

Bp # Bx, By,Bz,Ba,Bb,Bc,Bd	Transaction p: REF	p0 = {Bp,REFR}	p1 = {Bp}
Bq # Bc,Bd,Be,Bf,Bg.Bh,Bi	Transaction q: REF	q0 = {Bq,REFR+1}	q1 = {Bq}

Calibration Support

XDR architecture supports a variety of calibration operations to achieve maximum performance. Please consult the XIO and XDR DRAM component datasheets for specific details on each type of calibration operation. Calibration operations are divided into the following three classes:

- Initial Calibration operations (I)
- Periodic Calibration operations (P)
- Continuous Calibration operations (C)

Initial Calibration operations are performed once during system initialization. Periodic Calibration operations are performed at some specified period after initialization and require basic controller support for maintenance. Continuous Calibration operations occur continuously and do not require controller support for maintenance. Initial Calibration operations may take more time to converge on the optimal target value than Periodic Calibration operations, which generally increment or decrement the value being calibrated.

The calibration operation types, classes, and targets are defined in Table 11 below:

Table 11 **XDR Calibration Operations**

Operation Type	Name	XIO		XDR DRAM	
		RQ	DQ	RQ	DQ
Timing Calibration	TCAL		I, P		
Current Calibration	CCAL	I, P	C		I, P
Impedance Calibration	ZCAL		I, P		I, P

The XDR DRAM and XIO Periodic Calibration operations generally require some coordination between memory requests and XIO register operations. Please refer to the XDR DRAM and XIO datasheets for details on these operations.

Netlist EX2009-p. 23
Samsung v Netlist
IPR2023-00454

Timing Parameters

Frequency Scalability

The XDR architecture supports frequency scalability as well as feature-scalability to improve performance over time. Table 12 provides an overview of the frequency scalability plan for the XDR architecture and recommended initial ranges for product design and highest verification priority. This information is shown graphically in Figure 8.

The introductory speed grades of XDR DRAM support a 2.4-4.0 Gbps DQ bit rate and a 600-1000Mbps RQ bit rate, using a 4:1 frequency ratio between DQ and RQ links. Up to 36 devices can be supported per RQ channel.

This ratio is maintained until the RQ channel becomes a bottleneck to further performance scalability. Above a 4.8 Gbps DQ bit rate, an 8:1 frequency ratio between DQ and RQ links will be used, allowing the RQ bit rate to drop down to 600 Mbps, providing further headroom for scaling the RQ channel, while maintaining support for 36 devices per RQ channel.

Figure 8 **XDR Frequency Roadmap**

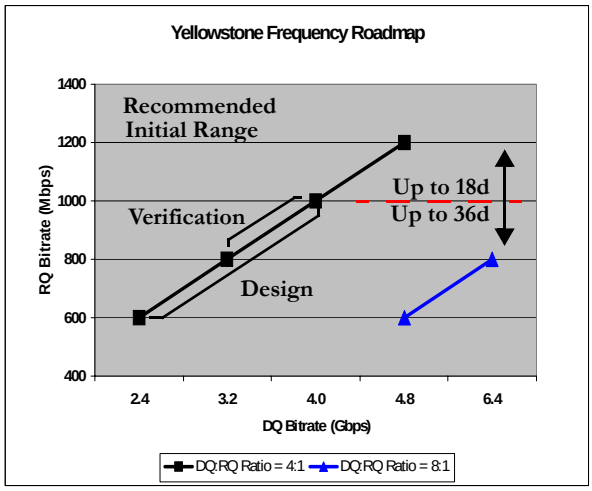


Table 12 **Frequency Scalability for XDR Architecture**

Parameter	Architectural Range	Recommended Initial Range	
		Design	Verification Priority
RQ Channel Bit Rate	600 - 1200 Mbps	600 - 1000 Mbps	800 - 1000 Mbps
DQ Channel Bit Rate	2.4 - 6.4 Gbps	2.4 - 4.0 Gbps	3.2 - 4.0 Gbps
DQ : RQ Frequency Ratio	4:1 for 2.4 - 4.8 Gbps (BL=16) 8:1 for 4.8 - 6.4 Gbps (BL=32)	4:1	4:1
RQ Channel Capacity	600 - 1000 Mbps: up to 36d 1000 - 1200 Mbps: up to 18d	600 - 1000 Mbps: up to 36d	800 - 1000 Mbps: up to 36d

Netlist EX2009-p.24
Samsung v Netlist
IPR2023-00454

XDR DRAM Timing Parameters

Table 13 summarizes the timing parameters that characterize the core logic of the XDR DRAM. These timing parameters will vary as a function of the component's speed bin.

Rules for Setting MC Timing Parameters

Table 14 lists a number of Rules and Guidelines to use when programming memory controller timing parameters. Rules define constraints which must be met to ensure reliable system operation. Guidelines define constraints which, if guaranteed to be met for all supported system configurations, can help to simplify the memory controller design.

XDR DRAMs support Dynamic Request Scheduling to allow memory controllers to use aggressive collision avoidance mechanisms. However, it is possible to simplify the collision avoidance mechanisms used by the memory controller if certain constraints (that is, Guidelines) are met by the choice of timing parameters. More specifically, if these Guidelines are guaranteed to be met for all supported system configurations, the memory controller does not need to use the DELA or DELC functions associated with the ROWA and COL packets. Support for dynamic delay control of precharge and refresh operations in ROWP packets (0-3 cycles) is recommended for all designs.

XIO Interface Timing Parameters

Table 15 summarizes the timing parameters that characterize the Memory Controller-to-XIO and XIO-to-Channel interfaces. These timing parameters will vary as a function of the XDR DRAM speed bin and system-level skew/latency characteristics.

Netlist EX2009-p. 25
Samsung v Netlist
IPR2023-00454

This page intentionally left blank.

Table 13 **XDR DRAM Timing Parameters**

Symbol	Parameter and Other Conditions	Architectural Range		Recommended Initial Range for MC				Units
				Design		Verification		
		Min	Max	Min	Max	Min	Max	
t_{RC}	Row-cycle time: interval between successive ROWA-ACT or ROWP-REFA or ROWP-REFI activate commands to the same bank.	8-40	-	8-40	-	16-24 (even)	-	t_{CYCLE}
t_{RAS}	Row-asserted time: interval between a ROWA-ACT or ROWP-REFA or ROWP-REFI activate command and a ROWP-PRE or ROWP-REFP precharge command to the same bank.	6-32	64 us	6-32	64 us	10-18	64 us	t_{CYCLE}
t_{RP}	Row-precharge time: interval between a ROWP-PRE or ROWP-REFP precharge command and a ROWA-ACT or ROWP-REFA or ROWP-REFI activate command to the same bank.	2-10	-	2-10	-	6-8	-	t_{CYCLE}
t_{PP}, $t_{PP\text{-}D}$[a]	Precharge-to-precharge time: interval between successive ROWP-PRE or ROWP-REFP commands to same/different bank sets.	4 1-4	- -	4 1-4	- -	4 1	- -	t_{CYCLE}
t_{RR}, $t_{RR\text{-}D}$[b]	Row-to-row time: interval between ROWA-ACT or ROWP-REFA or ROWP-REFI commands to same/different bank sets.	4 2-4	- -	4 2-4	- -	4 4	- -	t_{CYCLE}
$t_{RCD\text{-}R}$	Row-to-column delay (reads): interval between a ROWA-ACT activate command and either a COL-RD read command or COL-WR or COL-WRM write command to the same bank.	3-9	-	3-9	-	5,7	-	t_{CYCLE}
$t_{RCD\text{-}W}$	Row-to-column delay (writes): interval between a ROWA-ACT activate command and either a COL-RD read command or COL-WR or COL-WRM write command to the same bank.	1-9	-	1-9	-	1,5,7[c]	-	t_{CYCLE}
t_{CAC}	Column access delay: interval from COL-RD command to Q read data	4-12	-	4-12	-	6-8	-	t_{CYCLE}
t_{CWD}	Column write delay: interval from a COL-WR or COLM-WRM write command to D write data.	3-11	-	3-11	-	3-5	-	t_{CYCLE}
t_{CC}	Column-to-column time: interval between successive COL-RD commands, or between successive COL-WR or COLM-WRM commands.	2, 4	-	2, 4	-	2, 4	-	t_{CYCLE}
$t_{RW\text{-}BUB,\ XDRDRAM}$	Read-to-write bubble time for XDR DRAM. Interval between the last bit of a read data packet and the first bit of a write data packet.	2-4	-	2-4	-	3	-	t_{CYCLE}
$t_{WR\text{-}BUB,\ XDRDRAM}$	Write-to-read bubble time for XDR DRAM. Interval between the last bit of a write data packet and the first bit of a read data packet.	2-4	-	2-4	-	3	-	t_{CYCLE}
$t_{\Delta RW}$[d]	Read-to-write time: interval between a COL-RD read command and a COL-WR or COLM-WRM write command (factoring in DQ channel propagation delay).	4-16	-	4-16	-	7-12	-	t_{CYCLE}
$t_{\Delta WR}$ $t_{\Delta WR\text{-}D}$[e]	Write-to-read time: interval between a COL-WR or COLM-WRM write command and a COL-RD command to same/different banksets.	6-16 2-6	-	6-16 2-6	-	9-10 2,3,6	-	t_{CYCLE}
t_{RDP}	Read-to-precharge time: interval between a COL-RD read command and a ROWP-PRE precharge command to the same bank.	1-8	-	1-8[f] 3-8[g]	-	3,4,8	-	t_{CYCLE}
t_{WRP}	Write-to-precharge time: interval between a COL-WR or COLM-WRM write command and a ROWP-PRE command to the same bank.	6-16	-	6-16	-	10-12	-	t_{CYCLE}
t_{DP}	Last Data-to-precharge time: interval between the beginning of the last write data packet and a ROWP-PRE precharge command	3-13	-	3-13	-	7-9	-	t_{CYCLE}
t_{DR}	Last Data-to-read time: interval between the beginning of the last write data packet and a COL-RD read command	3-13	-	3-13	-	6-7	-	t_{CYCLE}

a. $t_{PP\text{-}D}$ is the t_{PP} parameter for precharges to different bank sets. See XDR DRAM datasheet for further details.

b. $t_{RR\text{-}D}$ is the t_{RR} parameter for activates to different bank sets. See XDR DRAM datasheet for further details.

c. Support for tRCD-W = 1 is provided to help in reducing tRC for writes to minimize bank resource contention. When using Early Read, Rambus recommends that tRCD-W match tRCD-R for improved tiling of requests. The choice of tRCD-W represents a trade-off between improved channel efficiency during turnarounds and reduced bank interference.

d. See "Turnaround Bubbles" on page 19. Table 14 provides rules and guidelines for setting this parameter. Also note that BL=32 devices reduce the R-W bubble at the XDR DRAM by two cycles, which requires $t_{\Delta RW}$ to be increased.

e. $t_{\Delta WR\text{-}D}$ is the $t_{\Delta WR}$ parameter for write-read accesses to different bank sets. See XDR DRAM datasheet for further details.

f. Applies to t_{CC} = two devices.

g. Applies to t_{CC} = four devices.

Netlist EX2009-p.28
Samsung v Netlist
IPR2023-00454

Table 14 **Rules and Guidelines for Setting MC Timing Parameters**

Rule / Guideline		Goal	Recommended Equation (Constraint)	Also See	Recommended Action / Comment
Rules (Required)					
R1		Staggered column cycle between opposite bank sets within XDR DRAM during Early Read W-R turn-around	$t_{CWD} - t_{\Delta WR\text{-}D}$ = odd	R2b,G2b	- Because Guideline #G2b recommends setting $t_{\Delta WR\text{-}D}$ to an even value, t_{CWD} should generally be set to an odd value. - All XDR DRAMs should support at least two programmable settings for t_{CWD}, at least one of which should be odd.
R2	a	Guarantee minimum R-W bubble is achieved at XDR DRAM and XIO	$t_{\Delta RW} \geq (t_{CAC} - t_{CWD}) + t_{CC} + (t_{PD,CYC} - t_{PD,CYC,MIN}) + t_{RW\text{-}BUB,XDRDRAM,MIN}$	G3a, G4a	See Table 9. Increase $t_{\Delta RW}$ as (t_{CAC} - t_{CWD}) and/or $t_{PD,CYC}$ increases.
	b	Guarantee minimum W-R bubble is achieved at XDR DRAM and XIO for Early-Read	$t_{\Delta WR\text{-}D} \geq (t_{CWD} - t_{CAC}) + t_{CC} + t_{WR\text{-}BUB,XDRDRAM,MIN}$	R1, G2b	See Table 9. Increase $t_{\Delta WR\text{-}D}$ as (t_{CWD} - t_{CAC}) increases.
R3	a	Guarantee that all data has been written to XDR DRAM core before performing precharge	$t_{WRP} \geq t_{CWD} + t_{DP,MIN}$[a]		Increase t_{WRP} as t_{CWD} or t_{CC} increases.
	b	Guarantee that all data has been written to XDR DRAM core before performing read (non-Early Read)	$t_{\Delta WR} \geq t_{CWD} + t_{DR,MIN}$[a]	G3b, G4b, G5	Increase $t_{\Delta WR}$ as t_{CWD} or t_{CC} increases.
Guidelines (Optional)					
G1	a	Efficient tiling of ROW and COL packets	$t_{RCD\text{-}R}$ = odd $t_{RCD\text{-}W}$ = odd	G2a, G3b	Program an odd number of cycles in the MC which is >= to the minimum value specified in the XDR DRAM datasheet
	b		$t_{RR} = t_{PP} = 4 * n$ (n = 1, 2, 3, ...)		Min value fixed at 4 for architecture, but MC can use a greater value.
G2	a	Efficient tiling during Early Read W-R turnaround	$t_{RCD\text{-}W} = t_{RCD\text{-}R}$	G1a, G3b	
	b		$t_{\Delta WR\text{-}D} = t_{CC}$ or (t_{CC}+4)	R1, R2b	
G3	a	Avoid collision between ROWA(wr) and COL-RD on a R-W turnaround	$t_{\Delta RW} > t_{RCD\text{-}W}$	R2a, G4a	Either: - Increase $t_{\Delta RW}$ to meet constraint - Dynamically adjust $t_{RCD\text{-}W}$ for the first write command to meet constraint
	b	Avoid collision between ROWA(rd) and COL-WR or COLM-WR on a W-R turnaround (non-Early-Read)	$t_{\Delta WR} > t_{RCD\text{-}R}$	R3b, G4b	Increase $t_{\Delta WR}$ to meet constraint[b]
G4	a	Guarantee that t_{PP} is met on a R-W turnaround	$t_{\Delta RW} \geq (t_{RDP} - t_{WRP}) + t_{PP} + (t_{CC} - 4)$	R2a, G3a	Increase $t_{\Delta RW}$ to meet constraint[b]
	b	Guarantee that t_{PP} is met on a W-R turnaround (non-Early-Read)	$t_{\Delta WR} \geq (t_{WRP} - t_{RDP}) + t_{PP} + (t_{CC} - 4)$	R3b, G3b, G5	Increase $t_{\Delta WR}$ to meet constraint
G5		Avoid TDATA conflict on W-R turnaround when checking RDATA	$t_{\Delta WR} \geq t_{QTD} - t_{QRD} + t_{ERD} + t_{CC}$	R3b, G3b, G4b	Increase $t_{\Delta WR}$ to meet constraint

a. This parameter will increase by two cycles for tCC=4 devices.

b. Not likely to ever be a problem.

Netlist EX2009-p. 29
Samsung v Netlist
IPR2023-00454

Table 15 **XIO Interface Timing Parameters**

Symbol	Parameter and Other Conditions	Architectural Range		Recommended Initial Range				Units
				Design		Verification		
		Min	Max	Min	Max	Min	Max	
Request/Data Interface								
t_{QTD}[a]	QDATA-to-TDATA Offset. Timing offset from COL-WR or COL-WRM command sampled on QDATA to first corresponding TDATA sample.	2	11	2	11	2	5	t_{CYCLE}
t_{QRD}[a]	QDATA-to-RDATA Offset. Timing offset from COL-RD command sampled on QDATA to first corresponding RDATA sample at MC.	6	20	6	20	11	14	t_{CYCLE}
t_{ERD}[b]	Expected DATA to RDATA Offset. Timing offset from Expected data driven on TDATA to associated RDATA sample to be checked.	3	8	3	5	3	5	t_{CYCLE}
t_{TET}	Transmit-Enable-to-TDATA Offset. Timing offset from TX_ENA sampled active to first valid TDATA sample.	2	--	2	--	2	--	t_{CYCLE}
Maintenance Interface (Initial/Periodic Calibration)								
t_{PEM}	PAT_ENA-to-PAT_MRKR. Timing offset from PAT_ENA sampled active to PAT_MRKR sampled active.	2	--	2	--	2	--	t_{CYCLE}
t_{QPM}	QDATA-to-PAT_MRKR. Timing offset from last command sampled on QDATA (generally a precharge) to PAT_MRKR sampled active.	2	--	2	--	2	--	t_{CYCLE}
t_{RPM}	RDATA-to-PAT_MRKR. Timing offset from last valid RDATA sampled by memory controller to PAT_MRKR sampled active by XIO.	2	--	2	--	2	--	t_{CYCLE}
t_{PMA}	PAT_MRKR-to-Activate. Timing offset from PAT_MRKR sampled active to ROWA-ACT command sampled on QDATA.	2	--	2	--	2	--	t_{CYCLE}
t_{PMT}	PAT_MRKR-to-TDATA. Timing offset from PAT_MRKR sampled active to first valid TDATA sample.	12	--	12	--	12	--	t_{CYCLE}
t_{PM0}	PAT_MRKR asserted time for PAT_TYPE 0. Number of cycles that PAT_MRKR should be sampled active for PAT_TYPE[2:0] == 000.	8	8	8	8	8	8	t_{CYCLE}
$t_{PM\text{-}CALC}$ $t_{PM\text{-}CALZ}$	PAT_MRKR-to-CALC/CALZ. Timing offset from PAT_MRKR sampled active to COLX-CALC or COLX-CALZ command sampled on QDATA.	8-32	--	14	--	14	--	t_{CYCLE}
t_{CALCE}[c] t_{CALZE}	CALC/Z-to-CALE. Timing offset between COLX-CALC/CALZ command sampled and COLX-CALE command sampled on QDATA.	8-32	--	30	--	30	--	t_{CYCLE}
t_{PDM}	Pattern Done-to-PAT_MRKR Inactive. Timing offset between last command on QDATA or last RDATA to PAT_MRKR sampled inactive.	2	--	2	--	2	--	t_{CYCLE}
t_{PME}	PAT_MRKR Inactive-to-PAT_ENA Inactive. Timing offset between PAT_MRKR sampled inactive and PAT_ENA sampled inactive.	2	--	2	--	2	--	t_{CYCLE}
t_{PEDA}	PAT_ENA Deasserted-to-Activate. Timing offset between PAT_ENA sampled inactive by MC and ROWA-ACT command sampled on QDATA.	2	--	2	--	2	--	t_{CYCLE}
t_{PER}	PAT_ENA Recovery Time. PAT_ENA inactive to PAT_ENA active.	2	--	2	--	2	--	t_{CYCLE}
t_{PCAL}[c]	Time interval between Periodic Calibration operations from PAT_MRKR sampled active to PAT_MRKR sampled active for subsequent operation.	0.125	12.5	0.125	12.5	0.125	12.5	ms
$t_{PCAL\text{-}ALL}$	Time interval over which all types of Periodic Calibration operations should be issued	1	100	1	100	1	100	ms

Netlist EX2009-p.30
Samsung v Netlist
IPR2023-00454

Table 15 **XIO Interface Timing Parameters (Continued)**

Symbol	Parameter and Other Conditions	Architectural Range		Recommended Initial Range				Units
				Design		Verification		
		Min	Max	Min	Max	Min	Max	
$t_{PCAL-OP}$	Duration of a single Periodic Calibration operation from PAT_MRKR sampled active to PAT_ENA sampled inactive.	--	64	--	64	--	64	t_{CYCLE}
Register Interface								
t_{REG_RESET}	REG_RESETN asserted time from startup of REG_CLK.	200	--	200	--	200	--	t_{CYCLE}
t_{RESET_CMD}	REG_RESETN deasserted-to-first register command. Time from REG_RESETN sampled inactive (=1) to REG_CYC sampled active for subsequent register transaction.	4	--	4	--	4	--	t_{CYCLE}
t_{REG_CYCLE}	Register Cycle Time. Time from REG_CYC sampled active for first transaction to REG_CYC sampled active for second transaction. Note that REG_CYC should only be asserted for one cycle per transaction.	4	--	4	--	4	--	t_{CYCLE}
t_{REG_ACC}	Register Read Data Access Time from REG_CYC sampled active.	4	4	4	4	4	4	t_{CYCLE}
t_{REG-PM}	Register Transaction-to-Pattern Marker. Time from REG_CYC sampled active for a register transaction to PAT_MRKR sampled active.	2	--	2	--	2	--	t_{CYCLE}
t_{PE-REG}	Pattern Enable deasserted-to-Register Transaction. Time from PAT_ENA sampled inactive to REG_CYC sampled active for a register transaction.	4	--	4	--	4	--	t_{CYCLE}
Channel Interface								
$t_{PD,CYC}$	Round Trip Flight Time of Longest DQ Link in Cycles. This is the round trip flight time of the longest DQ link in the system rounded up to the nearest integer cycle boundary (See "Turnaround Bubbles" on page 19.).	1	3	1	3	1	3	t_{CYCLE}
$t_{RW-BUB,XIO}$	Read-to-write bubble time for XIO. Interval between the last bit received for a read data packet and the first bit transmitted for a write data packet at the DQ pins of the XIO.	2-3	-	2-3	-	2	-	t_{CYCLE}
$t_{WR-BUB,XIO}$	Write-to-read bubble time for XIO. Interval between the last bit transmitted for a write data packet and the first bit received for a read data packet at the DQ pins of the XIO.	2-4	-	2-4	-	3	-	t_{CYCLE}

a. This parameter varies across platforms, memory configurations, and memory types, but is a constant for any given platform/configuration/type. This parameter is specified relative to a COL packet with the DELC field set to 0.

b. This parameter will be fixed for a given XIO design. By supporting a range for this parameter, the memory controller may be compatible with a variety of XIOs.

c. The XIO will issue Periodic Calibration operations using a programmable timer with 1µs granularity.

Netlist EX2009-p. 31
Samsung v Netlist
IPR2023-00454

NOTICE

© Copyright 2002-2003 Rambus Inc.
4440 El Camino Real
Los Altos, CA 94022 U.S.A.
All rights reserved.

Copyright in this document is owned by Rambus Inc. The use of this documentation is governed by an agreement containing restrictions on use, access, and disclosure.

Rambus Inc. reserves the right to make changes to this documentation without obligation to notify any person or organization.

No part of this document may be photocopied, reproduced, transmitted, transcribed, stored in a retrieval system or translated to another language, in any form or by any means, electronic, mechanical, magnetic, optical or otherwise, or disclosed to third parties without the prior written consent of Rambus Inc.

Rambus, RaSer, FlexPhase, SO-RIMM, RIMM, RDRAM, XDR, and the Rambus Logo:

are trademarks or registered trademarks of Rambus Inc. in the United States and other countries. Rambus may also have trademark rights in other terms used herein. In Japan, Rambus is a licensee of the XDR mark.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE DOCUMENT. RAMBUS INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE TECHNOLOGY DESCRIBED IN THIS DOCUMENT AT ANY TIME.

Document DL-0161 Version 0.8
Last Modified December 09, 2005

Rambus Inc.
4440 El Camino Real
Los Altos, California 94022 USA
Telephone: 650-947-5000
Fax: 650-947-5001

http://www.rambus.com

written by Rich Perego

Netlist EX2009-p. 33
Samsung v Netlist
IPR2023-00454