Direct Rambus™ Memory Controller (RMC2)

Overview

RMC2 is referred to as a "constraint-based" memory controller. That is, it explicitly models all the logical and physical constraints on the operation of a Rambus memory system.

To simplify the RMC2 design, *logical* constraints (e.g. a bank can't be activated unless all neighbor banks are precharged) are considered separately from *timing* (e.g. a bank can't be activated until t_{RP} after it is precharged) and *retire* (e.g. following a write, that word can't be read until the write data has been transferred to the DRAM core) constraints.

Logical constraints are tracked by the Protocol Module (PM), which receives transaction requests from the Bus Interface Unit, and requests all the Row and Column packets necessary to implement the requested transaction, in the correct logical order. Timing and retire constraints are tracked by the Constraint Module (CM), which receives the packet requests from the PM and outputs the formatted packets to the RAC when all timing and retire constraints are satisfied.

Within the PM there is a separate Service Protocol Unit (SPU) for each outstanding transaction and within the CM there is a separate Constraint Timer for each packet whose timing constraints have not all been satisfied .

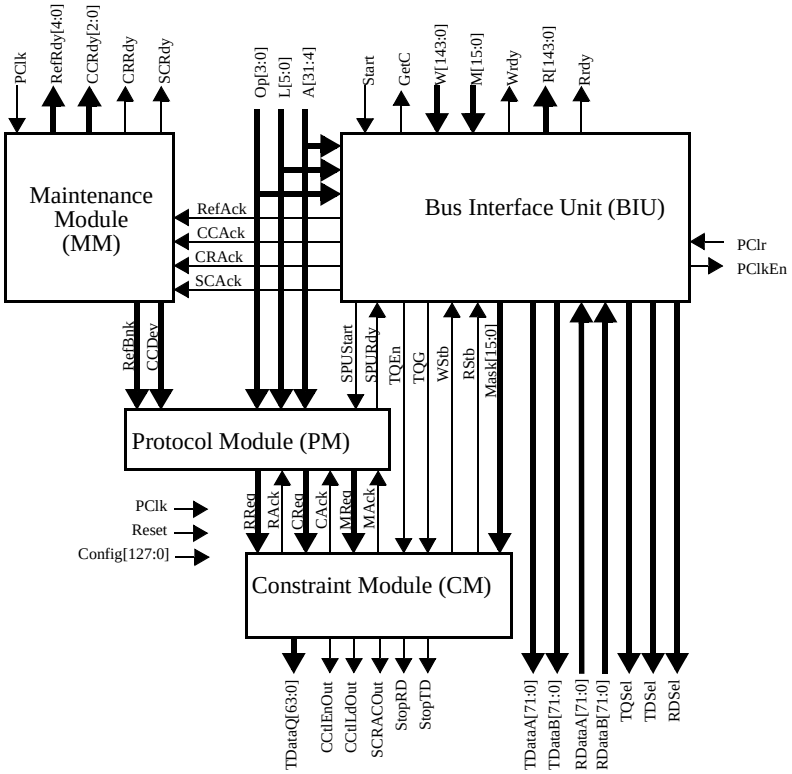

Figure 1: RMC2 Block Diagram

Features

- RTL coding style
- Closed and open page policy support
- Optimized support for 16 byte transfers
- Optimized support for variable burst length requests
- Handles all RDRAM protocol, housekeeping functions
- Synchronous operation to system clock frequencies to 200 Mhz
- Interfaces directly to Rambus ASIC I/O Cell (RAC)

Related Documentation

Data sheets for the Rambus memory system components, including the Rambus DRAMs, RIMM Module and Rambus ASIC Cell (RAC) are available on the Rambus web site at http://www.rambus.com.

General Description

Some of the features of the RMC2 are:

1. It generally provides the optimal Channel bandwidth for any sequence of transaction requests possible without re-ordering packets between transactions. Transactions are serviced in-order.

2. A new transaction can be accepted and a new transaction started on the Channel every ASIC clock (PClk).

3. Up to seven transactions may be outstanding at a time (default). This provides optimal bandwidth for closed-page operation and is derived from $t_{RC}/t_{PACKET} = 7$ for -40 or -45 RDRAM spec and represents the largest number of closed-page transactions that can be outstanding on the Channel. Each outstanding transaction requires one SPU. The number of outstanding transactions allowed is a parameter than may be modified for each application to minimize the amount of logic; for example the number of outstanding transactions may be raised to 8 for -50 RDRAMs (where $t_{RC}/t_{PACKET} = 8$), or the number of outstanding transactions may be lowered if the ASIC can't generate that many outstanding memory transactions.

4. The RMC2 will not close a page if another outstanding transaction is addressing the same page, even if closed-page policy is selected. For example, a read access to a closed-page bank requires an ACT–RD–[RD–...]–PREX sequence. But if during the transaction another transaction to the same page is received (i.e. a page hit), the SPU for the first transaction will skip the PREX packet and the SPU for the second transaction will skip the ACT packet.

5. t_{RCD} and t_{CAC} timings are adjustable with t_{CYCLE} granularity.

6. RMC2 initially supports closed-page policy, open-page mode will be added.

7. RDRAM and RAC Standby modes are supported. In closed-page mode RDRAMs will be relaxed (placed into Standby mode), but in open-page mode RDRAMs will be kept in Attention mode to reduce CAS latency.

8. RDRAM Nap and Powerdown packets are supported.

9. It supports three RDRAM regions (all RDRAMs in a region have the same number of banks, number of rows/bank, page size, bank type (doubled or independent) and split core or not). RMC2 only supported two regions, but RMC2 supports three because the Channel supports up to three RIMMs so it is reasonable to support three regions

10. Wherever possible, features may be stripped out of the RTL code using Verilog 'define' directives and the associated logic will not be synthesized.

Port Descriptions.

Table 1: Host Interface

Port	Type	Description
PClk	input	System clock.
PClr[3:0]	input	From Gearblk, indicates the phase of SynClk with respect to PClk. This is a one-hot vector, and 4'b0001 indicates the phase where PClk and SynClk rising edges occur simultaneously.
PClkEn	output	To Init block, disables one in four PClks (4:3 gear ratio), one in three PClks (3:2 gear ratio) or one in two PClks (2:1 gear ratio). Init block is clocked by PClk, qualifying PClk with PClkEn enables Init to count SynClk cycles.
Reset	input	System reset; puts RMC2 in a known initial state; synchronous to PClk.
Start	input	This port is asserted to request a transaction; it is de-asserted synchronously when GetC is asserted and there are no more transactions to request.
GetC	output	This port is asserted in response to Start to accept a transaction. If CfgSyncGetC = 0, GetC may be asserted in the same PClk cycle as Start is activated, if CfgSyncGetC = 1, GetC is asserted synchronously in the next PClk cycle.

Table 1: Host Interface

Port	Type	Description
Op[3:0]	input	4'b1100 = RDL: Read (linear address order, 1 to 64 dualocts) 4'b1000 = RDP: Read (Pentium address order, 2 dualocts only) 4'b1101 = WRL: Write (linear address order, 1 to 64 dualocts) 4'b1001 = WRP: Write (Pentium address order, 2 dualocts only) 4'b0010 = REF: RDRAM Refresh 4'b0110 = SCE: RDRAM TCEN 4'b1010 = CC: RDRAM current calibrate 4'b1110 = SCC: RDRAM TCAL and RAC slew rate control 4'b0011 = NAP: RDRAM Nap. A[7:4] is DevID; A[8] = 1 for broadcast. 4'b0111 = PDN: RDRAM Powerdown. A[7:4] is DevID; A[8] = 1 for broadcast 4'b1011 = CR: RAC current calibrate 4'b1111 = NAPC: RDRAM Conditional Nap
L[5:0]	input	Transaction length, in dualocts; valid when Start is asserted. Transactions may not cross RDRAM page boundaries, so some combinations of L and A aren't allowed.
A[31:4]	input	Address of the first dualoct to be transferred; valid when Start is asserted. Transactions may not cross RDRAM page boundaries, so some combinations of L and A aren't allowed.
M[15:0]	input	Write mask; valid in the same cycle in which W is valid.
W[143:0]	input	Write data (one dualoct); it is sampled by RMD.d2 on PClk cycles when Wrdy is asserted; W and M must be valid during these PClk cycles. The next dualoct must be provided the PClk cycle after Wrdy is asserted, until the last dualoct of write data has been transferred. If 'define RDRAM_18_BIT, W is 128 bits.
Wrdy	output	Write ready; asserted for one PClk cycle to acknowledge write data.
R[143:0]	output	Read data; valid on PClk cycles when Rrdy is asserted. If 'define RDRAM_18_BIT, R is 128 bits.
Rrdy	output	Read ready; asserted for one PClk cycle to indicate read data is valid.
RefRdy[4:0]	output	RefRdy is decremented each time the refresh interval timer elapses, it is incremented each time a refresh is performed; range is +15 to -16. The ASIC uses RefRdy to schedule refresh operations.
CCRdy[2:0]	output	CCRdy is decremented each time the Current Control (CC) interval timer elapses, it is incremented each time a RDRAM CC is performed; range is +3 to -4. The ASIC uses CCRdy to schedule CC operations.
CRRdy	output	CRRdy is decremented each time the CC interval timer elapses CfgNDev times, and is incremented each time a RAC CC operation is performed, where CfgNDev is usually programmed with the number of RDRAM devices.
SCRdy	output	SCRdy is decremented each time the CC interval timer elapses CfgCCPerSRC times, and is incremented each time an RDRAM and RAC Slew Rate Control (SRC) operation is performed, where CfgCCPerSRC is the programmed number of RDRAM CC operations per SRC.

RAC Interface.

Table 2: RAC Interface

Port	Type	Description
RDataA[71:0]	input	RAC receive (read) data. RDataA corresponds to R[7:0][8:0] on the Application Interface. If not ‘define RDRAM_18_BIT, RDataA[71, 62, 53, 44, 35, 26, 17, 8] are not used and the associated logic isn’t synthesized.
RDataB[71:0]	input	RAC receive (read) data. RDataB corresponds to R[15:8][8:0] on the Application Interface. If not ‘define RDRAM_18_BIT, RDataB[71, 62, 53, 44, 35, 26, 17, 8] are not used and the associated logic isn’t synthesized.
TDataA[71:0]	output	RAC transmit (write) data. TDataA corresponds to W[7:0][8:0] on the Application Interface. If not ‘define RDRAM_18_BIT, TDataA[71, 62, 53, 44, 35, 26, 17, 8] are not used and the associated logic isn’t synthesized.
TDataB[71:0]	output	RAC transmit (write) data. TDataB corresponds to W[15:8][8:0] on the Application Interface. If not ‘define RDRAM_18_BIT, TDataA[71, 62, 53, 44, 35, 26, 17, 8] are not used and the associated logic isn’t synthesized.

TDataQ[63:0]	output	RAC request (i.e. ROW and COL packet) bus. ROW and COL packets use separate bits of the TDataQ bus, ROW and COL bits can be viewed separately as shown:
		`for (i = 7; i >= 0; i = i-1)`
		`  for (j = 2; j>= 0; j= j-1)`
		`  assign ROW[3*i + j] = TData[8*i + j+5];`
		
		`for (i = 7; i >= 0; i = i-1)`
		`  for (j = 4; j>= 0; j= j-1)`
		`  assign COL[5*i + j] = TData[8*i + j]`

TQSel	output	RAC timing select for TDataQ bus.
TDSel	output	RAC timing select for TDataA and TDataB.
RDSel	output	RAC timing select for RDataA and RDataB.
StopRD	output	Stops RAC clock for RDataA and RDataB.
StopTD	output	Stops RAC clock for TDataA and TDataB.
CCtlEnOut	output	Used for automatic current calibration of RAC.
CCtlLdOut	output	Used for automatic current calibration of RAC.
SCRACOut	output	Used for automatic slew rate control of RAC.

Configuration Port

The Configuration Port configures RMC2 for the intended operating mode. These bits are static during normal operation; they may be hardwired or driven by software registers, in the latter case values may be determined during the initialization process (described below). Information from the SPD ROM (if RIMMs are used) or the RDRAM registers may be used to determine the correct values.

To eliminate synchronization delays, PClk frequency is an integer ratio of SynClk frequency and the clocks are phase-locked by the ASIC. Four PClk:SynClk “gear ratios” are supported: 1:1, 4:3, 3:2 and 2:1, indicated by the CfgGear field of the Config bus. For example, in 4:3 gear ratio there are four PClk cycles per three SynClk cycles, so one of every four PClk cycles is skipped. In 3:2 one of every three PClk cycles is skipped, in 2:1 one of every two PClk cycles is skipped. CfgGear selects the gear ratio.

CfgGear = 000 and 001 both select 1:1 gear ratio, but 000 is a “low latency” mode where RMC2 inputs generate ROW packets at RMC2 outputs in the same PClk cycle (i.e. it is an asynchronous path); this may only be used

for closed page mode and even then it may not be possible to meet timing constraints for a given process. CfgGear = 001 adds one register in the path from RMC2 inputs to ROW packet outputs.

If CfgSyncGetC = 1, GetC (the handshaking signal to acknowledge transaction requests) is asserted synchronously to PClk in response to Start being asserted; the ASIC must de-assert Start by the next PClk edge. Alternatively, if CfgSyncGetC = 0, GetC is asserted asynchronously in response to Start being asserted, and GetC may be sampled synchronous to PClk by the ASIC; in this mode, Start must be valid earlier in the PClk cycle, so most applications use CfgSyncGetC = 1.

Besides selecting open-page vs. closed-page operation, CfgOpen = 1 also selects the use of ROW packets (PRER) to precharge banks. CfgOpen = 0 selects auto-precharge (WRA) for writes and XOP packets (PREX) for reads to precharge banks, to reduce bandwidth on the ROW bus. These are beleived to be the optimal precharge mechanisms for each mode.

In open-page mode, CfgRelax = 1 relaxes each RDRAM (i.e. puts it in Standby power mode) when its last open bank is closed; if CfgRelax = 0 the RDRAM is left in Attention power mode at all times.

CfgRelax has no effect in closed-page mode, when the RDRAM is always relaxed, unless CfgTrcd = 5, in whch case the RDRAM is left in Attention at all times because $t_{RCD} < t_{FRM}$, which means the RDRAM can't frame a RD or WR packet t_{RCD} after an ACT packet. This means one SynClk would be added to all accesses when CfgTrcd = 5, which is not acceptable. CfgRelax is not implemented in the current RMC2.

CfgBackToBackRD = 1 prevents two RD packets being issued to *different* RDRAMs without at least t_{PACKET} time between them. Some three-RIMM systems suffer reduced voltage margin for back-to-back RD packets if the first RD is to an RDRAM near the controller and the second is to the RDRAM at the end of the Channel. This is a simple mechanism to prevent this situation. Generally, CfgBackToBackRD = 0.

CfgTrcd, CfgTcac, CfgTrp and CfgToffp have t_{CYCLE} granularity, while CfgTrasSyn, CfgTrasrefSyn and CfgTrpSyn have SynClk (4 t_{CYCLE}) granularity. CfgTrasrefSyn and CfgTrprefSyn are similar to CfgTrasSyn and CfgTrp, respectively, but apply to refresh transactions. For all current RDRAMs, CfgTrasrefSyn = CfgTrasSyn and CfgTrprefSyn = ceiling(CfgTrp / 4), but future RDRAMs may incorporate multibank refresh where a single refresh transaction refreshes two or four banks, in which case CfgTrasrefSyn and CfgTrprefSyn may be increased relative to CfgTras and CfgTrp.

There are three RDRAM "regions" called U, V and W. Within each region all RDRAMs must have the same numbers of banks, rows and columns, and core organization (dependent banks and split core). Each of these paramaters may vary between regions. Regions correspond to RIMMs, which is why three regions are supported. CfgNDev, CfgDepBnk, CfgSpCore, Cfg2KBPage, CfgNRowBits and CfgNBnkBits configuration fields are replicated for each of the three regions.

Table 3: Configuration Port

Bits	Name	Description
2:0	CfgGear[2:0]	3'b000: 1:1 gear ratio (low latency) 3'b001: 1:1 gear ratio (normal) 3'b010: 4:3 gear ratio 3'b011: 3:2 gear ratio 3'b101: 2:1 gear ratio. Other options are reserved.
3	CfgSyncGetC	1'b1: GetC activated PClk cycle after Start is asserted. 1'b0: GetC activated asynchronously in same cycle as Start is asserted.
4	CfgOpen	1'b1: Open-page mode; keep banks open following transactions. 1'b0: Closed-page mode; precharge after each transaction unless another transaction is already requested to same page and there are no intervening transactions to different pages in same bank.
5	CfgRelax	1'b1: RDRAM is relaxed when its last open bank is closed. 1'b0: RDRAMs are never relaxed, stay at attention always.
6	CfgBackToBackRD	1'b1: Back-to-back RD packets may be issued to different RDRAMs 1'b0: There must be at least one SynClk between RD packets to different RDRAMs.

Table 3: Configuration Port

Bits	Name	Description
10:7	CfgTrcd[3:0]	4'b0101: t_{RCD} = 5 t_{CYCLE} 4'b0111: t_{RCD} = 7 t_{CYCLE} 4'b1001: t_{RCD} = 9 t_{CYCLE} 4'b1011: t_{RCD} = 11 t_{CYCLE} Other values are reserved.
14:11	CfgTcac[3:0]	4'b1000: t_{CAC} = 8 t_{CYCLE} 4'b1001: t_{CAC} = 9 t_{CYCLE} 4'b1010: t_{CAC} = 10 t_{CYCLE} 4'b1011: t_{CAC} = 11 t_{CYCLE} 4'b1100: t_{CAC} = 12 t_{CYCLE} Other values are reserved.
17:15	CfgTrasSyn[2:0]	3'b100: t_{RAS} = 16 t_{CYCLE} 3'b101: t_{RAS} = 20 t_{CYCLE} 3'b110: t_{RAS} = 24 t_{CYCLE} 3'b111: t_{RAS} = 28 t_{CYCLE} Other values are reserved.
21:18	CfgTrp[3:0]	4'b0110: t_{RP} = 6 t_{CYCLE} 4'b1000: t_{RP} = 8 t_{CYCLE} 4'b1010: t_{RP} = 10 t_{CYCLE} 4'b1100: t_{RP} = 12 t_{CYCLE} Other values are reserved.
24:22	CfgToffp[2:0]	3'b100: t_{OFFP} = 4 t_{CYCLE} 3'b101: t_{OFFP} = 5 t_{CYCLE} 3'b110: t_{OFFP} = 6 t_{CYCLE} 3'b111: t_{OFFP} = 7 t_{CYCLE} Other values are reserved
28:25	CfgTrasrefSyn[3:0]	4'b0100: t_{RASREF} = 16 * t_{CYCLE} 4'b0101: t_{RASREF} = 20 * t_{CYCLE} 4'b0110: t_{RASREF} = 24 * t_{CYCLE} ... 4'b1111: t_{RASREF} = 60* t_{CYCLE} t_{RASREF} is the interval between REFA and a subsequent REFP packet to the same bank.
31:29	CfgTrprefSyn[2:0]	3'b010: t_{RPREF} = 8 t_{CYCLE} 3'b011: t_{RPREF} = 12 t_{CYCLE} 3'b100: t_{RPREF} = 16 t_{CYCLE} ... 3'b111: t_{RPREF} = 28 t_{CYCLE} t_{RPREF} is the interval between REFP and subsequent REFA or ACT packets to the same bank.
36:32	CfgMaxDevID[4:0]	Largest Device ID for any RDRAM; Device IDs must be contiguous starting with 0 so CfgMaxDevID equals the number of RDRAMs, minus 1. Tells RMC2 how many RDRAMs to perform current calibration on. Every CfgMaxDevID + 1 RDRAM CC transactions, CRRdy is decremented to request a RAC CC cycle.
47:37	CfgRefCnt[10:0]	RefCnt resets to 0 and increments every PClk cycle it reaches CfgRefCnt, then it returns to 0 and RefRdy is decremented.

Table 3: Configuration Port

Bits	Name	Description
50:48	CfgNRefBnkBits[2:0]	3'b010: 4 refresh banks 3'b011: 8 refresh banks 3'b100: 16 refresh banks 3'b101: 32 refresh banks CfgNRefBnkBits should be set to the least of RefreshBankBits for any of Region U, V, or W.
57:51	CfgCCCnt[6:0]	CCIntCnt[19:0] resets to 0 and increments every PClk cycle. When CCIntCnt[19:13] == CfgCCCnt, it retunrs to 0 and CCRdy is decremented.
63:58	CfgCCPerSRC[5:0]	Every CfgCCPerSRC CC transactions, SCRdy is decremented to request RAC and RDRAM slew rate control. This field adjusts the rate of SRC requests relative to CC requests, usually to adjust for the number of devices. If CfgCCPerSRC is programmed to CfgMaxDevID + 1, each device has SRC and CC transactions at the same rate.
79:64	reserved	Must be set to zero.
85:80	CfgNDevU[5:0]	Number of devices in region U (0–32)
86	CfgDepBnkU	1'b1: dependent (doubled) banks in region U 1'b0: independent banks in region U
87	CfgSpCoreU	1'b1: split core in region U 1'b0: no split core in region U
88	Cfg2KBPageU	1'b1: 2KB (128 dualoct) page size in region U 1'b0: 1KB (64 dualoct) page size in region U
92:89	CfgNRowBitsU[3:0]	4'b1001: 512 rows per bank in region U 4'b1010: 1024 rows per bank in region U 4'b1011: 2048 rows per bank in region U 4'b1100: 4096 rows per bank in region U 4'b1101: 8192 rows per bank in region U 4'b1110: 16,384 rows per bank in region U Other values are reserved.
95:93	CfgNBnkBitsU[2:0]	3'b010: 4 banks per device in region U 3'b011: 8 banks per device in region U 3'b100: 16 banks per device in region U 3'b101: 32 banks per device in region U Other values are reserved.
101:96	CfgNDevV[5:0]	Number of devices in region V (0–32).
102	CfgDepBnkV	1'b1: dependent (doubled) banks in region V 1'b0: independent banks in region V
103	CfgSpCoreV	1'b1: split core in region V 1'b0: no split core in region V
104	Cfg2KBPageV	1'b1: 2KB (128 dualoct) page size in region V 1'b0: 1KB (64 dualoct) page size in region V

Table 3: Configuration Port

Bits	Name	Description
108:105	CfgNRowBitsV[3:0]	4'b1001: 512 rows per bank in region V 4'b1010: 1024 rows per bank in region V 4'b1011: 2048 rows per bank in region V 4'b1100: 4096 rows per bank in region V 4'b1101: 8192 rows per bank in region V 4'b1110: 16,384 rows per bank in region V Other values are reserved.
111:109	CfgNBnkBitsV[2:0]	3'b010: 4 banks per device in region V 3'b011: 8 banks per device in region V 3'b100: 16 banks per device in region V 3'b101: 32 banks per device in region V Other values are reserved.
117:112	CfgNDevW[5:0]	Number of devices in region W (0–32)
118	CfgDepBnkW	1'b1: dependent (doubled) banks in region W 1'b0: independent banks in region W
119	CfgSpCoreW	1'b1: split core in region W 1'b0: no split core in region W
120	Cfg2KBPageW	1'b1: 2KB (128 dualoct) page size in region W 1'b0: 1KB (64 dualoct) page size in region W
124:121	CfgNRowBitsW[3:0]	4'b1001: 512 rows per bank in region W 4'b1010: 1024 rows per bank in region W 4'b1011: 2048 rows per bank in region W 4'b1100: 4096 rows per bank in region W 4'b1101: 8192 rows per bank in region W 4'b1110: 16,384 rows per bank in region W Other values are reserved
127:125	CfgNBnkBitsW[2:0]	2'b010: 4 banks per device in region W 2'b011: 8 banks per device in region W 2'b100: 16 banks per device in region W 2'b101: 32 banks per device in region W Other values are reserved.

Table 4 shows configurations for all DRAM bin specs (values are decimal).

Table 4: Configuration Port settings for 64/72M RDRAM Bin Specs

Bits	Name	-40-800	-45-800	-50-800	-45-711	-50-711	-45-600	-53-600
10:7	CfgTrcd	7	9	11	7	9	5	7
17:15	CfgTrasSyn	5	5	6	5	5	4	5
21:18	CfgTrp	8	8	10	8	8	6	8
24:22	CfgToffpSyn	4	4	4	4	4	4	4
28:25	CfgTrasrefSyn	5	5	6	5	5	4	5
31:29	CfgTrprefSyn	2	2	3	2	2	2	2

Timings

All timings are function of chip layout and process. Values below are only approximate.

Table 5: Timing Conditions

Symbol	Description	Min	Max	Units	Figure
t_R, t_F	Rise and fall times for all inputs	—	200	ps	3-1
t_{PCYCLE}	PClk cycle time	5.0	—	ns	3-1
t_{ERRDC}	PClk duty cycle error (compared to 50%)	—	250	ps	3-1
t_{ERRP}	Instantaneous PClk / SynClk phase error	—	500	ps	3-1
t_1	Reset setup time to rising edge of PClk		—	ps	none
t_2	Reset hold time from rising edge of PClk		—	ps	none
t_3	A[31:4], Op[3:0], L[5:0] setup time to rising edge of PClk		—	ps	3-2, 3-3
t_4	A[31:4], Op[3:0], L[5:0] hold time from falling edge of PClk		—	ps	3-2, 3-3
t_5	Start setup time to rising edge of PClk		—	ps	3-2, 3-3
t_6	Start hold time from rising edge of PClk		—	ps	3-2, 3-3
t_7	W[15:0][8:0], M[15:0] setup time to rising edge of PClk		—	ps	3-2
t_8	W[15:0][8:0], M[15:0] hold time from rising edge of PClk		—	ps	3-2
t_9	RDataA, RDataB setup time to either edge of PClk	250	—	ps	3-3
t_{10}	RDataA, RDataB hold time from either edge of PClk	100	—	ps	3-3

Table 6: Timing Characteristics

Symbol	Description	Min	Max	Units	Figure
t_{11}	GetC valid from PClk (CfgSyncGetC = 1)			ps	3-2, 3-3
t_{12}	GetC valid from Start (CfgSyncGetC = 0)			ps	3-2, 3-3
t_{13}	Rrdy, Wrdy, RefRdy, CCRdy, CRRdy, SCRdy valid from PClk			ps	3-2, 3-3
t_{14}	R[15:0][8:0] valid from PClk			ps	3-3
t_{15}	TDataQ (Row bits) valid from rising edge of PClk			ps	3-2, 3-3
t_{16}	TDataQ (Row bits) valid from falling edge of PClk			ps	none
t_{19}	TDataA, TDataB valid from rising edge of PClk			ps	none
t_{20}	TDataA, TDataB valid from falling edge of PClk			ps	3-2
t_{RD}	Read latency from Start active to first read data valid	see Table		t_{PCYCLE}	3-3

Read latency is a function of Pclk / SynClk frequency ratio (CfgGear), RDRAM t_{RCD}, the state of the addressed RDRAM bank, as well as any other pending transactions that may delay issuing ROW or COL packets for the read transaction. tRD values assume the addressed bank is precharged and there are no timing constraints due to any outstanding transactions; in this case the transaction consists of an ACT and one or more RD packets.

For CfgGear other than 3'b001, the read data latency depends on the phase of PClk compared to SynClk when Start is issued, and there will be two possible minimum values, and an "average" assuming Start is asserted at a random PClk phase.

Table 7: Read Latency Values

CfgGear	t_{RCD}	Min	Average	Max	Units
3'b001 (1:1)	7		8		PClk
	9		8		PClk
	11		9		PClk
3'b010 (4:3)	7	10	10.5	11	PClk
	9	10	10.75	11	PClk
	11	11	11.75	12	PClk
3'b011 (3:2)	7	11	10.7	12	PClk
	9	12	12.3	13	PClk
	11	12	12.7	13	PClk
3'b10 (2:1)	7	15	15.5	16	PClk
	9	16	16.5	17	PClk
	11	17	17.5	18	PClk

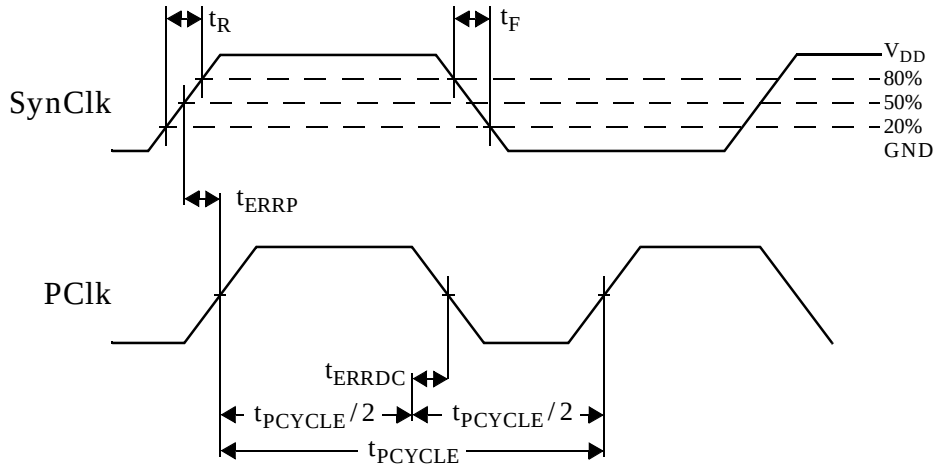

Figure 2: SynClk and PClk waveform

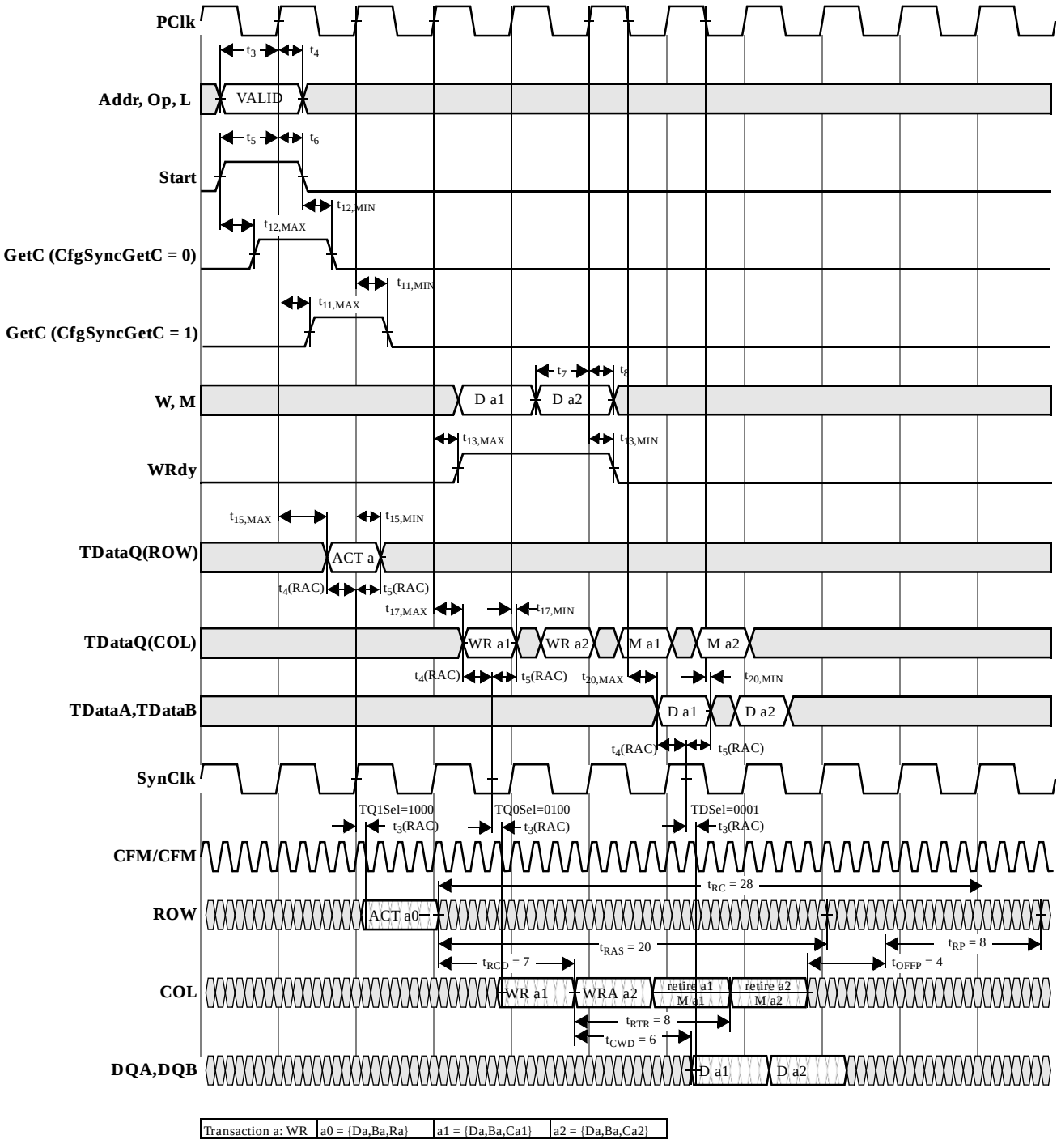

Figure 3: Write System Timing (CfgGear=3'b001, t_{RCD} = 7, Closed Page Mode)

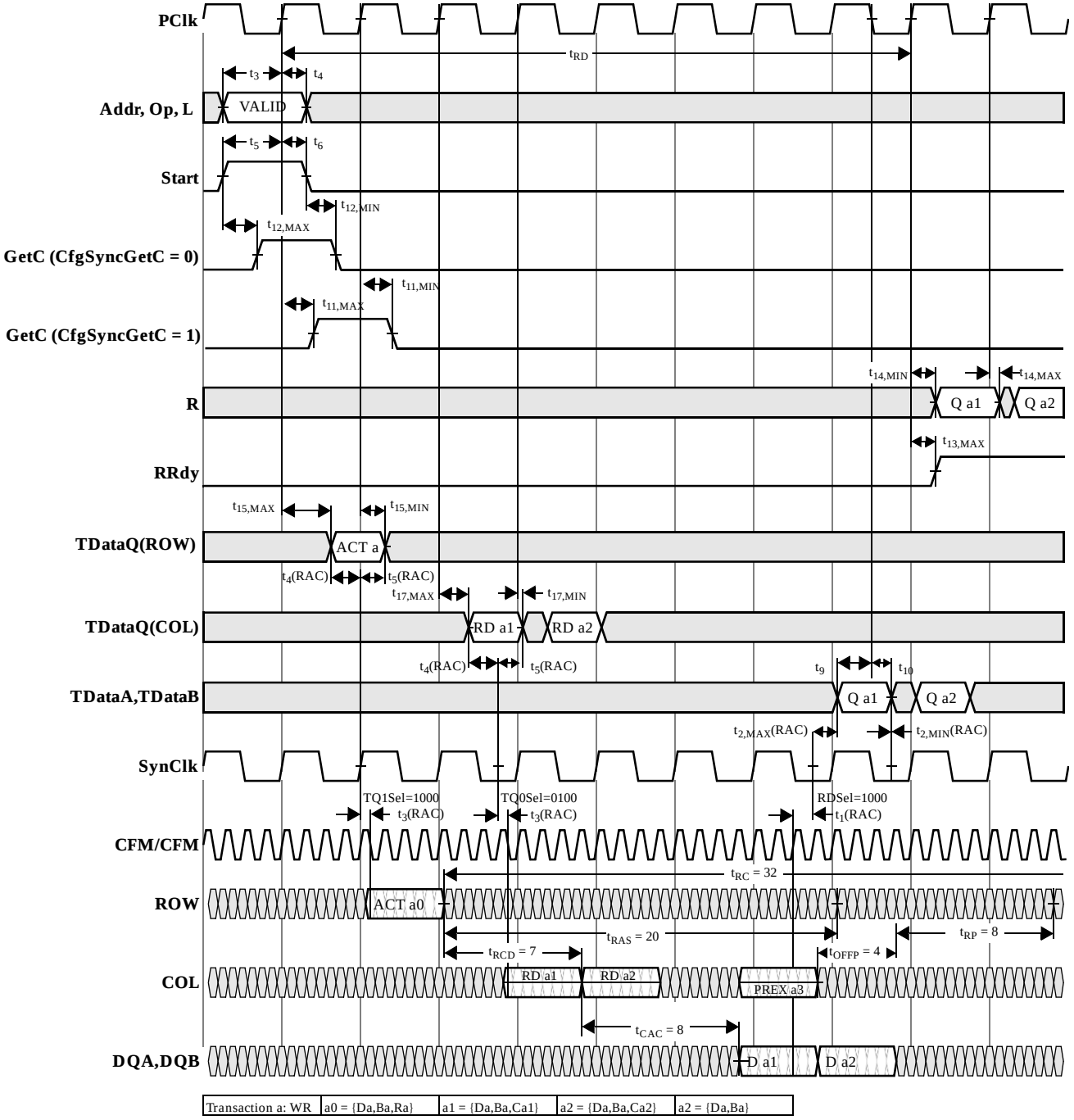

Figure 4: Read System Timing (CfgGear=3'b001, t_{RCD} = 7, Closed Page Mode)

Simulating and Synthesizing RMC2

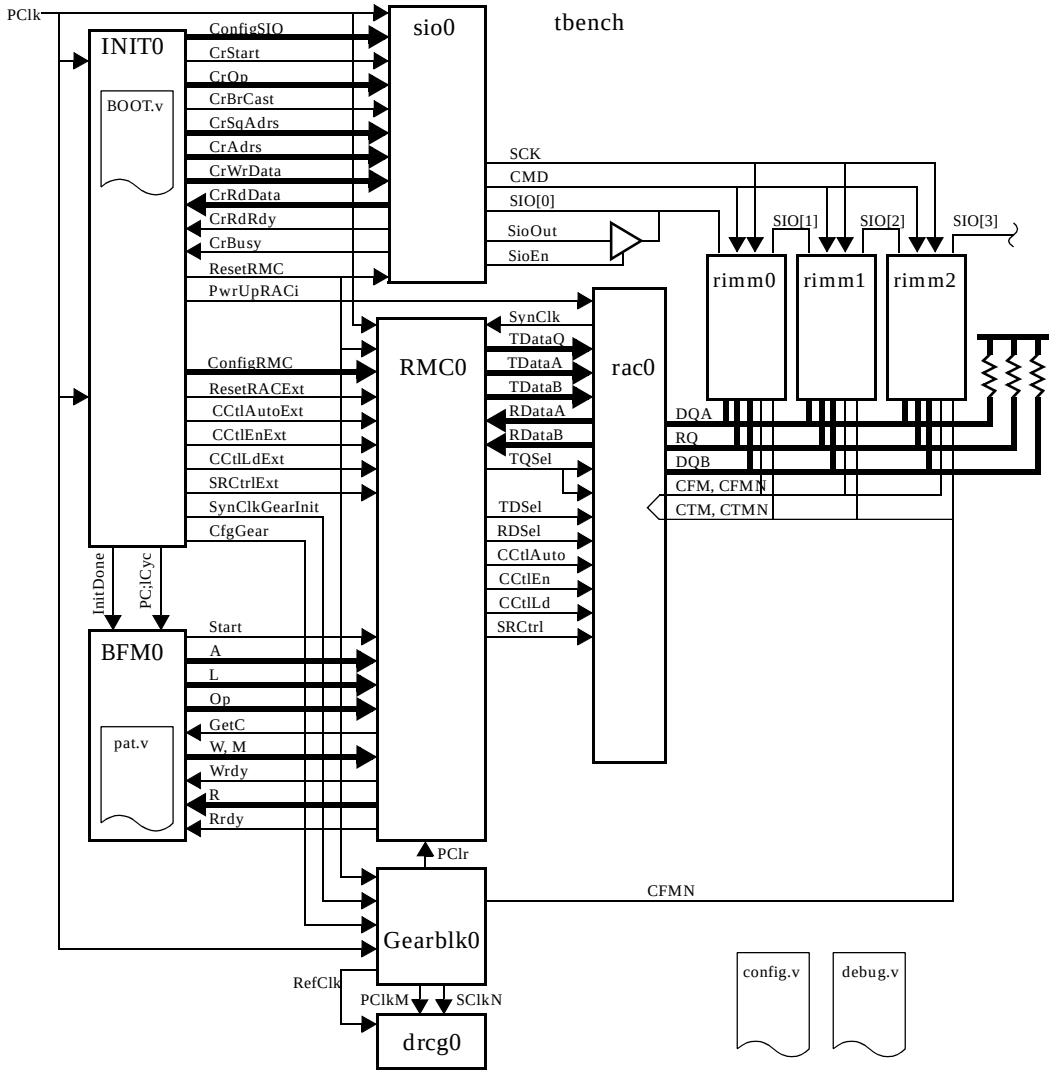

Figure 5: Testbench Organization

Testbench organization

Figure 5 shows a simplified view of the RMC2 testbench (tbench). There are 10 modules instantiated at the top level, and four included files, described below:

At the start of a simulation, INIT0 initializes the RDRAMs, RAC, and RMC0. sio0 controls the RDRAM SIO bus to allow read and writing RDRAM registers and exiting Powerdown and Nap modes. The Bus Functional Model (BFM0) controls the host interface to RMC0; the included ‘pat.v’ file is a list of read, write, and maintenance operations to be performed. RMC0, rac0 and Direct Rambus Clock Generator (drcg0) modules are self-explanatory. rac0 is connected from one to three RIMMs.

The gear module (Gearblk0) receives the host clock (PClk) and Rambus clock divided by four (SynClk); these must be harmonically related in the ratios of 1:1, 4:3, 3:2 or 2:1. It generates three outputs: PClr tells RMC0 the phase of SynClk with respect to PClk; PClkM and SClkN are PClk and SynClk divided by down to generate a common base frequency. If PClk and SynClk are phase-locked, PClkM and SClkN edges occur simultaneously. In a real system the DRCG uses the phase delay between PClkM and SClkN to adjust the delay of CTM and CTMN to acheive and maintain phase lock.

The three RIMM modules (rimm0, rimm1 and rimm2) may be configured to model any number and type of RDRAMs. Note the Rambus channel is bussed, not routed through the RIMM modules like a real RIMM. Channel transport delay (t_{TR}) is not modeled, but there are parameters in the RIMM to delay the data, simulating different t_{TR} values.

The channel is terminated by Verilog pullup primitives to model termination resistors. These are necessary because the RAC uses open-collector drivers, so the RAC model only drives the channel to '0' or 'z' states; the pullups are required to achieve '0' and '1' states on the Channel.

There are four included files at the top level:

BOOT.v models the operations necessary to make the testbench functional and in a known state. For reference, the sequence is broken into the following parts:

```
// Stage 1 - Start Clocks
//    Determine maximum Channel frequency
//    Set PClk and RefClk frequencies and DRCG Mult0 and
Mult1 inputs
//    Configure RMC gear ratio
//    Begin DRCG Lock period
//
//  Stage 2 - RAC Initialization
//    Assert Init1 sequence
//
//  Stage 3 - RDRAM Register Initialization
//  3.1  Initial quiet period on CMD
//  3.2  SIO Reset
//  3.3  Write TEST77 register
//  3.4  Write TCYCLE register
//  3.5  Write SDEVID register
//  3.6  Write DEVID register
//  3.7  Write PDNX and PDNXA registers
//  3.8  Write NAPX register
//  3.9  Write TPARM register
//  3.10 Write TCDLY1 register
//  3.11 Write TFRM register
//  3.12 SETR/CLRR
//  3.13 Write CCA and CCB registers
//  3.14 Powerdown Exit
//  3.15 SETF
//
//  Stage 4 - RMC Configuration
//  4.1 Initial value for read data offset
//  4.2 Configure cycles for row and column operations
//  4.3 Set Refresh Interval
//  4.4 Set Current Control Interval
//  4.5 Set Slew Rate Control Interval
//  4.6 Configure RMC for Bank/Row/ColBits and ByteWidth
per region
//  4.7 Configure RMC for Address Mapping
//  4.8 Configure the RMC to issue Relax commands
//  4.9 Any other steps necessary to configure memory con-
troller.
//
//  Stage 5 - RDRAM Slew Rate and Current Control
//    Assert Init2 sequence
//
//  Stage 6 - RDRAM Core Initialization and Read Leveliza-
tion
//  6.1 Core Initialization
//  6.2 Levelize Read Domains
//
//  Stage 7 - RDRAM Remaining Registers
//  7.1 Set Self Refresh for Powerdown and Nap
```

config.v is the RMC and RIMM configuration file. It sets the gear ratio, number and type of RIMMs, and all RMC configuration bits.

debug.v contains all code for debuging the RMC. This includes $dump or $vtDump, $dumpon, $dumpoff, $dumpvars. It is good to include a "*#long_delay* $finish" statement so the simulation won't run forever in the event of an endless loop. The Rambus Channel monitor module, MonitorD, should be instantiated here if you want graphical output of Channel activity. Also, it is useful to define signals for debug here.

The test pattern, pat.v, is included in module BFM0. This defines all the read, write and maintenance operations to be requested in the simulation. pat.v is orthogonal to config.v, any pattern can be run using any configuration (see section)

File structure

The file structure for the RMC2 project is:

```
$PROJECT/         top-level directory
  bin/            setup file, scripts, etc.
  doc/            documentation
    frame/        FrameMaker doc's (including this spec)
    pdf/          PDF files for release
    ppt/          Powerpoint slides
    text/         text documentation files
    xls/          Excel spreadsheets
  models/         model (simulation) directories
    $MODEL/       your model directory; files in this
                  directory aren't under RCS or included
                  in releases (all subdirectories are).
                  Place simulation logs, etc. here.
      tbench/     all Verilog testbench files
        rmc/      all Verilog RMC2 files
          BIU.v   Bus Interface Unit; buffers read and
                  write data
          CM.v    Contraint Module; check timing con-
                  straints on packets
          MM.v    Maintenance Module; tells when to do
                  refresh, current calibration and slew
                  rate control, track addresses
          PC.v    Page Cache; tracks bank state in open-
                  page systems
          PM.v    Protocol Module; assigns transactions to
                  SPUs in a round-robin manner; selects
                  packet request from SPUs
          RMC.v   Top-level RMC2 module
          SPU.v   Service Protocol Unit; converts a single
                  transaction into a logically correct
                  sequence of Rambus packets
```

```
        pat/     test pattern files
        config/  configuration files
        debug/   debug (instrumentation) files
      results/   simulation results
        PASS     list of all passing simulation
        FAIL     list of all failing simulations
        sim/     directories for log & dump files from
                 each failed sim
  releases/      RCS directories and archives of past
                 releases
    vYYMMDD.Z    archived releasees, where YYYYMMDD is
                 release date
    tbench/RCS/  RCS directory for tbench files
    rmc/RCS/     RCS directory for RMC2 files
    pat/RCS/     RCS directory for test pattern files
    config/RCS/  RCS directory for configuration files
    debug/RCS/   RCS directory for debug (instrumentation)
                 files
```

Scripts

Setup

Add the following to your .cshrc file prior to running setup.

```
setenv PROJECT /proj/rmc/RMC.d2
```

Also, make sure $PROJECT/bin is in your search path.

$PROJECT/bin/setup should be run before performing simulations, making testbench releases, etc.; please run it from your .cshrc file. This script adds $PROJECT/bin/ to the user's $PATH, and creates several aliases, including:

```
alias mkmod 'setenv MODEL \!*; makemodel; pushd
$PROJECT/models/$MODEL'


alias pbin    'pushd $PROJECT/bin'
alias pdoc    'pushd $PROJECT/doc'
alias pmod    'pushd $PROJECT/models/$MODEL'
alias ptbench 'pushd $PROJECT/models/$MOD-
EL/tbench'
alias ppat    'pushd $PROJECT/models/$MOD-
EL/tbench/pat'
alias pconfig 'pushd $PROJECT/models/$MOD-
EL/tbench/config'
alias prel    'pushd $PROJECT/releases'
alias psyn    'pushd $PROJECT/syn'
```

Creating a model (simulation) directory

The 'mkmod *model*' alias sets the environment variable $MODEL to *model*, executes the makemodel script to create the *model* directory and all subdirectories, make soft links to all the RCS directories, and check out read-only copies of all the files from RCS. It then changes tothe $PROJECT/models/$MODEL directory; if that directory already exists, mkmod informs the user and just pushes to that directory.

Simulation

Simulations are run using the runpat script:

```
runpat [pattern_file [config_file [debug_file
[simulator_options]]]]
```

runpat creates the following three soft links:

$MODEL/tbench/pat.v -> *pattern_file*.v
$MODEL/tbench/config.v -> *config_file*.v
$MODEL/tbench/debug.v -> *debug_file*.v

Arguments have a fixed order: if there is only one argument it is interpreted as *pattern_file*. Two arguments are interpreted as *pattern_file* and *config_file*, respectively. Three arguments are interpreted as *pattern_file*, *config_file* and *debug_file*, respectively. With no parameters, runpat re-runs the previous simulation. If *debug_file*, *config_file* or *pattern_file* are not specified, runpat uses the existing links. If any of *pattern_file*, *config_file* or *debug_file* are not specified and there is no existing link, or the target of the link does not exist, runpat issues an error message.

"runpat -h" displays usage information.

Pattern, configuration and debug files are stored in separate subdirectories to avoid clutter; they are maintained under RCS so they can be shared. Pattern and configuration files are included in design relases, but not debug files because these are site-specific (e.g. if the customer doesn't use Undertow, a $vtDump command will cause a compile error).

Verilog Compiler Directives

Verilog 'define directives are used to configure the RMC, RAC, RDRAMs and testbench for simuation and testing. For simulation, these are all contained in the file *config.v* (this is actually a soft link), and they effect the whole design. For Synopsys, 'defines in one file do not have any effect in another file; the same definitions must be made in the synthesis script using the Synopsys command

analyze -f verilog -d { *defines_for_that_file* } *filename*

Be sure to set Synopsys variable hdlin_enable_vpp true to enable interpretation of 'indef... 'else... 'endif constructs.

The following table lists user 'defines in *config.v*, in what files the 'define is used, and what the 'define does.

Table 8: User ‘defines in config.v

Name	Values	Used in	Description
BackdoorInit	none	BOOT.v, BOOT_tasks.v DRCG.v, tbench.v	If defined, the normal sequence register writes to configure RDRAMs is bypassed and RDRAM model parameters are set directly. This saves considerable simulation time.
BFM_144_BIT	none	BFM.v, tbench.v	If defined, 144 bits are read and written by the BFM (9 bit bytes); otherwise 128 bits are read and written. BFM_144_BITS should not be defined unless RDRAM_18_BIT is also defined. If RDRAM_18_BIT is defined and BFM_144_BIT is not, the testbench generates odd parity on the ninth bit of each write data byte and checks parity on the ninth bit of each read data byte (if parity doesn’t match it asserts PERR).
CfgOpen	none	BOOT.v, BOOT_tasks.v tbench.v	Sets the value of the RMC CfgOpen field Selects open-page (1) or closed-page (0) operation. OPEN_PAGE and / or CLOSED_PAGE must also be defined to include the logic in SPU to support open- or closed-page operation.
CfgRelax	none	BOOT_tasks.v tbench.v	Sets the value of the RMC CfgRelax field. If 1, the RDRAMs are Relaxed (put into Standby mode) after transactions when in closed-page mode (CfgRelax is don’t care in open-page mode); if 0, the RDRAMs are always left in Attention mode.
CfgSyncGetC	none	BFM.v BOOT.v, BOOT_tasks.v tbench.v	Sets the value of the RMC CfgSyncGetC field (1 or 0). If 1 GetC is driven synchronously to PClk and sampled asynchronously by the BFM; if 0 GetC is driven asynchronously by the RMC and sample synchronously by the BFM.
CLOSED_PAGE	none	SPU.v	If defined, code is included in SPU to support closed-page operation. This is not exclusive of OPEN_PAGE, both may be defined to include logic for both modes. CfgOpen defines which mode is selected.
CTMCyc800 Gear800	2.500*‘ns2v1	BOOT.v INIT.v	Defines 800 MHz Channel operation. Both must be defined. One one pair should be defined of{CTMCyc800, Gear800}; {CTMCyc711, Gear711};{CTMCyc600, Gear600}.
CTMCyc711 Gear 711	2.813*‘ns2v1	BOOT.v INIT.v	Defines 711 MHz Channel operation. Both must be defined. One one pair should be defined of {CTMCyc800, Gear800}; {CTMCyc711, Gear711};{CTMCyc600, Gear600}.
CTMCyc600 Gear 600	3.336*‘ns2v1	BOOT.v INIT.v	Defines 600 MHz Channel operation. Both must be defined. One one pair should be defined of {CTMCyc800, Gear800}; {CTMCyc711, Gear711};{CTMCyc600, Gear600}.
false	0	BFM.v BOOT.v, BOOT_tasks.v MonitorD.v RACD.v RIMM.v SPDparams.v tbench.v	Defines alias for logic 0.

Table 8: User ‘defines in config.v

Name	Values	Used in	Description
FAST_SCK	none	BOOT.v INIT.v SIO.v tbench.v	If defined, selects ~50 MHz frequency for SCK; else selects ~1 MHz. 1 MHz is the maximum frequency for RDRAM register operations, but the higher frequency speeds simulation when BackdoorInit is not defined. It’s used in config.v to select nine other ‘defines, which are used in INIT.v to determine the CTM / SCK frequency ratio.
OPEN_PAGE	none	PM.v SPU.v	If defined, the PC module is instantiated in PM and code is included in SPU to support open-page operation. This is not exclusive of CLOSED_PAGE, both may be in defined to include logic for both modes. CfgOpen defines which mode is selected.
rac_netlist_model	none	tbench.v	If defined, useRAC netlist model (rac_syn.v) instead of behavioral model (rac.v). The behavioral model simulates faster, but at this time the netlist model is the most accurate representation of the actual RAC.
rac_062	none	tbench.v	If defined, use RAC revision 0.6.2 model (RACD_062.v) instead of RAC behavioral model (rac.v). Some ASIC vendors use revision 0.6.2-compatible RAC cells.
RDRAM_18_BIT	none	BFM.v	If defined, 18-bit-wide RDRAM models are used and 144-bit datapaths are included in the RMC2 code; otherwise 16-bit-wide RDRAM models and 128-bit datapaths in the RMC2 are used. If RDRAM_18_BIT is defined, BFM_144_BIT may also be defined.
REF_TEST	none		RDRAM refresh is performed by a bfm_Op(‘REF, 0, *bank*) call in *pattern.v.* Usually the refresh bank address is provided by the Maintenance Module (MM), but if REF_TEST is defined the bank address comes from the *bank* field in the bfm_Op() call.
rimm0, rimm1, rimm2	RIMM0, RIMM1, RIMM2	BOOT.v, BOOT_tasks. v tbench.v	Define which RIMMs are present and their names. rimm0 must be defined; for two RIMMs rimm0 and rimm1 must be defined; for three RIMMs rimm0, rimm1 and rimm2 must be defined. The defined strings are the names of the RIMMs
rimm0_NumDevs, rimm1_NumDevs, rimm2_NumDevs	1–16	tbench.v	Defines number of RDRAM devices on RIMM0, RIMM1 and RIMM2, respectively.
rimm0_CoreOrg rimm1_CoreOrg rimm2_CoreOrg	“r64MD” “r72MD” “r128MD” “r144MD” “r256MDc” “r288MDc” “r256MDr” “r288MDr” “user”	tbench.v	RDRAM core organization for RIMM0, RIMM1 and RIMM2, respectively. For 256M or 288M parts, “c” or “r” mean “2KB page size, 1024 rows” or “1KB page size, 2048 rows”, respectively. RDRAM architectural parameters are derived from these defines; for “user” the default parameters are: DoubledBanks=1, Split Core=0, BankBits=4, RowBits=9, ColBits=6, RefBankBits=4, RefRowBits=10 and DataWidth=16 but these may be overridden by defparams.
rimm0_Option rimm1_Option rimm2_Option	“40-800” “45-800” “50-800” “45-600” “50-600” “user”	tbench.v	RDRAM option (speed bin) for RIMM0, RIMM1 and RIMM2, respectively. RDRAM AC parameters are derived from these defines; for “user” the defaults are fIMIN=300, fIMAX=400, fRAS=400, tRPr_MIN=8, tRASr_MIN=20, tRCDr_MIN=10, tRRr_MIN=8, tPPr_MIN=8, tCYCLEA_MIN=19, tCYCLEA_MAX=27, tCDLYA_MIN=5 and tCDLYA_MAX=9; these may be overridden by defparams.

Table 8: User ‘defines in config.v

Name	Values	Used in	Description
rimm0_ChPropDly rimm1_ChPropDly rimm2_ChPropDly	0-7	tbench.v	The testbench does not model propagation delay of signals on the Rambus channel. The RDRAM model includes a parameter, ChPropDly, that emulates the RDRAM being in different read domains. These define ChPropDly for RIMM0, RIMM1 and RIMM2, respectively.
SPD	none	BOOT.v, BOOT_tasks.v DRCG.v MonitorD.v RIMM.v SPDparams.v tbench.v	Number and type of RDRAMs are taken from SPD ROM, otherwise these must be specified directly
SPU6, SPU5, SPU4, SPU3, SPU2, SPU1, SPU0	none	PM.v SPU.v	Defines the number of SPUs - 1 (i.e. SPU6 -> seven SPUs). At most one of these should be defined, if none are defined default is one SPU. This determines how many transactions may be in progress at once, and allows optimizing performance vs. gate count.
true	1	BFM.v BOOT.v, BOOT_tasks.v INIT.v MonitorD.v RACD.v RIMM.v SPDparams.v tbench.v	Defines alias for logic 1.

Reporting results

Results from runpat are reported in the $MODEL/results directory.

runpat scans the Verilog and BFM log files and RIMM message files for the words “error” or “warning” (case insensitive). If it finds any, or if Verilog exits with a non-zero exit code, it considers the simulation to have failed and informs the user.

For each simulation that fails, one line is concatenated to the file FAIL; the line contains (separated by spaces) “*pattern_file config_file fail_time*” where fail_time is returned by

set fail_time = `date`

For each simulation that passes, a similar line is concatenated to the file PASS.

This provides an easy way to check which simulations passed and failed, and when.

For each simulation that fails, a new subdirectory is created named (separated by a space) *pattern_file config_file*. The verilog log file, BFM log file, RIMM message files, and (if they exist) Channel tiling and dump files are copied to this directory. This way results of failing simulations can be examined later. If a subdirectory by the same name already exists, it is overwritten.