Hardware System Verification (HSV)
Vertical Solutions Engineering (VSE)

DDR4
Palladium Memory Model
User Guide

Document Version: 3.8

Document Date: July 2018

Contents

General Information

The Cadence Memory Model Portfolio provides memory device models for the Cadence Palladium XP, Palladium XP II and Palladium Z1 series systems. Optimizing the acceleration and/or emulation flow on these platforms for MMP memory models may require information outside the scope of the MMP user guides and related MMP documentation.

1.1 Related Publications

For basic information regarding emulation and acceleration, please refer to the following documents:

For Palladium XP and Palladium XP II:

UXE User Guide
UXE Library Developer's Guide
UXE Known Problems and Solutions
UXE Command Reference Manual
Palladium XP Planning and Installation Guide
Palladium Target System Developer's Guide
What's New in UXE

For Palladium Z1:

VXE User Guide
VXE Library Developer's Guide
VXE Known Problems and Solutions
VXE Command Reference Manual
Palladium Z1 Planning and Installation Guide
Palladium Target System Developer's Guide
What's New in VXE

DDR4 Memory Model

1. Introduction

The Cadence Palladium DDR4 Model is based on the JEDEC specification JESD79-4B (June 2017).

The model is available in several configurations based on generic configurations from the JEDEC spec. As real devices become available from vendors those will be added to the catalog.

Different sizes from 2Gb up to 32Gb are available, please consult the memory model catalog for the current available list.

DIMM models are derived from their base part by expanding the data width to 64 or 72 bits rather than instantiating multiple base parts as in real devices. This minimizes the number of memory ports in order to improve emulation performance. Current DIMM models require only one clock input. Please use CK0 if more than one CK is available.

2. Model Release Levels

All models in the Memory Model Portfolio are graded with a release level. This release level informs users of the current maturity and status of the model. All families in the library are graded at one of these levels.

The different levels give an overall indication of the amount of testing, level of quality and feature availability in the model. For details on supported features check the User Guide for that particular model family.

There are three release levels for models in the MMP release.

Release Level		Model Status	Available in Release	Listed in Catalog	Requires Beta Agreement
Mainstream Release	MR	Fully released and available in the catalog for all customers to use.	Yes	Yes	No
Emerging Release	ER	Model has successfully completed Beta engagement(s). Most, but not all features have been tested. Documentation is available.	No	Yes	Yes
Initial Release	IR	Model has completed initial development and has been released to Beta customer(s). The model may have missing features, may not be fully tested, may not have documentation. Model may contain defects.	No	Yes	Yes

Access to Initial and Emerging Release versions of the models will require a Beta Agreement to be signed before the model can be delivered.

3. Configurations

3.1. DDR4 SDRAM Addressing

The following table lists the possible configurations. Not all configurations are available from all vendors. Please consult the appropriate vendor site for details on the parts they offer.

Memory Size	Data Width	X4	X8	X16
	Bank Groups	4	4	2
	Banks within a group	4	4	4
2Gb	Row Address	A[14:0]	A[13:0]	A[13:0]
	Column Address	A[9:0]	A[9:0]	A[9:0]
4Gb	Row Address	A[15:0]	A[14:0]	A[14:0]
	Column Address	A[9:0]	A[9:0]	A[9:0]
8Gb	Row Address	A[16:0]	A[15:0]	A[15:0]
	Column Address	A[9:0]	A[9:0]	A[9:0]
16Gb	Row Address	A[17:0]	A[16:0]	A[16:0]
	Column Address	A[9:0]	A[9:0]	A[9:0]

NOTE : Per specification, a portion of the SDRAM address bus, in other words a slice of the col/row address width, is multiplexed with non-address signals. For example, in A[17:0] the following functions are shared:
A[10] serves as the auto pre-charge bit.
A[12] is BC_n
A[14] is WE_n
A[15] is CAS_n
A[16] is RAS_n

A[13:0] inputs are used for ADDR. This allocation is reflected in wrappers and core model. Please see the Input/Output Functional Description in the specification for additional detail.

3.2. DDP Single Rank(SR) x16 from two x8

The Single Rank (SR) x16 DDP (Dual Die Package) is composed of two x8 devices connected in parallel.

Memory Size	Data Width	X8X2
	Bank Groups	4
	Banks within a group	4
8Gb	Row Address	A[14:0]
	Column Address	A[9:0]
16Gb	Row Address	A[15:0]
	Column Address	A[9:0]
32Gb	Row Address	A[16:0]
	Column Address	A[9:0]

4. Model Parameter Descriptions

The following table provides details on the user adjustable parameters for the Palladium DDR4 Memory Model. These parameters may be modified when instantiating the model.

User Adjustable Parameter	Default Value	Description
data_bits	16	Width of data bus DQ
addr_bits	14	Width of row address bus
bank_grp_width	2	Width of bank group bus BG
bank_addr_width	2	Width of bank address bus BA
row_addr_width	14	Width of row address bus
low_addr_bits	14	Width of ADDR bus
col_addr_width	10	Width of column address bus
byte_width	8	Width of one byte, or word for x4 models
tWLO_in_clocks	3	tWLO time in clocks
ddr4_3ds_flag	0	3DS core flag
rank_id	0	Rank id, used only in 3DS model

The following table provides some information about exposed localparams that are NOT user adjustable. On rare occasion the user may find one of these localparam needs adjusting for their configuration. If this case arises, please contact Cadence emulation or MMP support.

Localparam	Default Value	Description
num_bytes	2	data_bits / byte_width

total_addr_bits	28	BG width + BA width + row_addr_width + col_addr_width
ddr4_not_wide_flag	1	DDR4 data bit width is not larger than 16 flag

5. Verilog Macro Defines

The following table lists the Verilog macro defines related to the MMP DDR4 model.

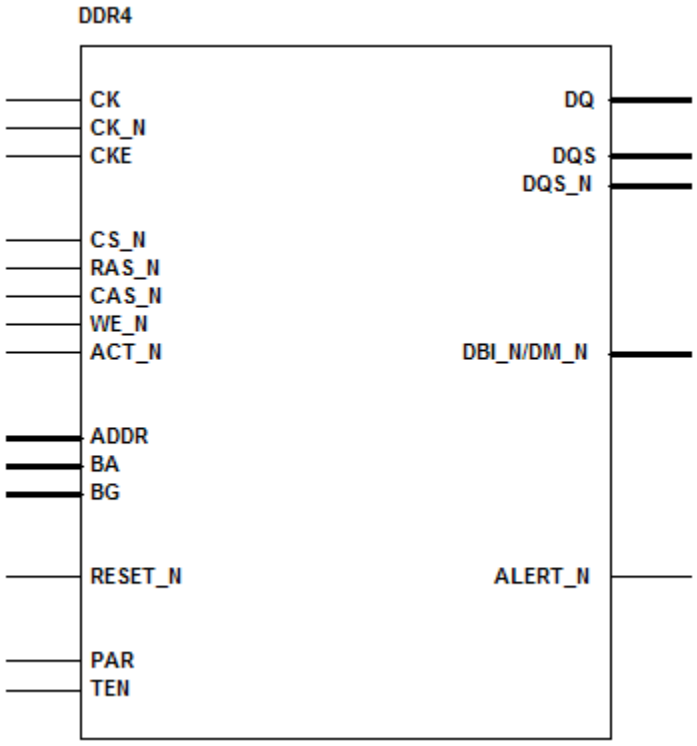
As described in the sections “IXCOM Compilation” and “Large Memory Support,“ each MMP DDR4 or DDR4 DIMM model that exceeds 30 bits of address width needs to incorporate the multiple core memory array generator (mmp_gen_mem.vp) into the memory build. In these cases, the file mmp_gen_mem.vp automatically defines Verilog macro MMP_LG_MEM_BITS with a value of ‘30’ by default. This default value may be changed within the file mmp_gen_mem.vp or overridden on the command with the define option. Users of DDR4 and DDR4 DIMM memories that are smaller than the 30 bit address width should NOT need to be aware of or modify this file or its Verilog macro value.

The Verilog macro DQ_MAPPING is defined by default in the ddr4_wide.vp file for all DDR4 DIMMS. The user should not need to be aware of or modify this Verilog macro.

`define Macro Purpose	**Optional Verilog `define**	**Default Value**
Large memory support	MMP_LG_MEM_BITS	30 Range 25 .. 30
Enables CRC mapping in DIMMs	DQ_MAPPING	Defined by default; no value needed
Enables Row,Bank,Col address mapping	MMP_RBC	Undefined by default; no value needed
Enables BA,BG address mapping	MMP_BA_BG	Undefined by default; no value needed

6. Model Block Diagram

The widths of the ADDR, BA, BG, DM_N, DQ, and DQS buses are dependent on the density of the part being used.



7. Address Mapping

The array of the DDR4 model is mapped into the internal memory of the Palladium system. This array is a single two dimensional array. The mapping of bank group, bank, row and column addresses to the internal model array is as follows:

ARRAY_ADDR = {BG, BA, ROW, COL}

This information is required if the memory needs to be preloaded with user data.

The array name in the model hierarchy is: memcore

If {row,bg,ba,col} addressing is needed instead of the default {bg,ba,row,col} addressing, add +define+MMP_RBC to the vlan invocation (IXCOM flow) or to the appropriate HDL-ICE synthesis (Classical flow) command. No value is required for MMP_RBC – only the compile phase define. This option is applicable when using the <model>.vp file.

If {ba,bg,row,col} addressing is needed instead of the default {bg,ba,row,col} addressing, add +define+MMP_BA_BG to the vlan invocation (IXCOM flow) or to the appropriate HDL-ICE synthesis (Classical flow) command. No value is required for MMP_BA_BG – only the compile phase define. This option is applicable when using the <model>.vp file.

8. Register Definitions

In the DDR4 there are seven Mode Registers. The Palladium DDR4 model implements all seven Mode Registers. However not all features are supported in the model.

8.1. Mode Register 0

Mode register 0 is implemented in the DDR4 model. The model supports the following parameter values. RFU must be programmed to 0 during MRS command, for all mode registers.

BG1	BG0, BA	17	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFU	MR Select	RFU	WR and RTP	CAS Latency	WR and RTP			DLL Reset	Test Mode	CAS Latency			Burst Type	CAS Latency	Burst Length	

BG0	BA[1:0]	MR Select
0	00	MR0
0	01	MR1
0	10	MR2
0	11	MR3
1	00	MR4
1	01	MR5
1	10	MR6
1	11	Reserved

BG1	Bit 17	RFU
0	0	Not Used

Bit 13	Bit 11	Bit 10	Bit 9	Write Recovery
X	X	X	X	Not supported

Bit 8	DLL Reset	
0	No	Not supported
1	Yes	Not supported

Bit 7	Mode	
0	Normal	Supported
1	Test	Not supported

Bit 12	Bit 6	Bit 5	Bit 4	Bit 2	CAS Latency
0	0	0	0	0	9
0	0	0	0	1	10
0	0	0	1	0	11
0	0	0	1	1	12
0	0	1	0	0	13
0	0	1	0	1	14
0	0	1	1	0	15

0	0	1	1	1	16
0	1	0	0	0	18
0	1	0	0	1	20
0	1	0	1	0	22
0	1	0	1	1	24
0	1	1	0	0	23
0	1	1	0	1	17
0	1	1	1	0	19
0	1	1	1	1	21
1	0	0	0	0	25 (only 3DS available)
1	0	0	0	1	26
1	0	0	1	0	27 (only 3DS available)
1	0	0	1	1	28
1	0	1	0	0	reserved for 29
1	0	1	0	1	30
1	0	1	1	0	reserved for 31
1	0	1	1	1	32
1	0	0	0	0	reserved

Bit 3	Burst Type	
0	Sequential	Supported
1	Interleaved	Supported

Bit 1	Bit 0	Burst Length
0	0	8 Fixed
0	1	BC4 or 8 On the Fly
1	0	BC4 Fixed
1	1	Reserved

8.2. Mode Register 1

Mode register 1 is implemented in the DDR4 model. The model supports the following parameter values.

BG1	BG0,BA	17,13	12	11	10:8	7	6:5	4	3	2:1	0
RFU	MR Select	RFU	QOFF	TDQS Enable	RTT-NOM	Write Leveling	RFU	Additive Latency		ODIC	DLL Enable

TDQS Enable, RTT_NOM, ODIC (Output Driver Impedance Control), and DLL Enable are not applicable to the DDR4 Palladium Model.

Bit 4	Bit 3	Additive Latency
0	0	0
0	1	CL-1
1	0	CL-2
1	1	Reserved

For DDR4 3DS model, bit[4:3]=11 (CL-3) is supported.

Bit 7	Write Leveling
0	Disabled
1	Enabled

Bit 12	Output Buffer
0	Enabled
1	Disabled

8.3. Mode Register 2

Mode register 2 is implemented in the DDR4 model. The model supports the following parameter values.

BG1	BG0,BA	17	13	12	11	10	9	8	7:6	5	4	3	2:0
RFU	MR Select	RFU	RFU	Write CRC	RTT_WR			RFU	ASR	CAS Write Latency			RFU

TRR (Target Row Refresh), RTT_WR and ASR (Array Self Refresh) are not applicable to the DDR4 Palladium Model.

Bit 5	Bit 4	Bit 3	CAS Write Latency
0	0	0	9
0	0	1	10
0	1	0	11
0	1	1	12
1	0	0	14
1	0	1	16
1	1	0	18
1	1	1	20

Bit 12	Write CRC
0	Disabled
1	Enabled

8.4. Mode Register 3

BG1	BG0,BA	17,13	12:11	10:9	8:6	5	4	3	2	1:0
RFU	MR Select	RFU	MPR Read Format	Write CMD Latency (CRC, DM)	Fine Gran-ularity Refresh Mode	Temp-erature sensor readout	Per DRAM Address-ability	Gear-down Mode	MPR Oper-ation	MPR Page Select

Mode register 3 is implemented in the DDR4 model. The model does not support bits MR3[10:3].

Bit 1	Bit 0	MPR Page Select
0	0	Page 0
0	1	Page 1
1	0	Page 2
1	1	Page 3

Bit 2	MPR Operation
0	Normal
1	MPR data

Bit 12	Bit 11	MPR Read Format
0	0	Serial
0	1	Parallel
1	0	Staggered
1	1	Reserved

8.5. Mode Register 4

BG1	BG0,BA	17	13	12	11	10	9	8:6	5
RFU	MR Select	RFU	PPR	Write Preamble	Read Preamble	Read Preamble Training	Self Refresh Abort	CS to C/A Latency Mode	sPPR

4	3	2	1	0
Internal Vref Monitor	Temp Controlled Refresh Mode	Temp Controlled Refresh Range	Maximum Power Down Mode	RFU

Mode register 4 is implemented in the DDR4 model. The model does not support PPR, Self Refresh Abort, Internal Vref Monitor, Temperature Controlled Refresh Mode and Range, and Maximum Power Down Mode.

Bit 8	Bit 7	Bit 6	CAL
0	0	0	Disable
0	0	1	3
0	1	0	4
0	1	1	5
1	0	0	6
1	0	1	8
1	1	0	Reserved
1	1	1	Reserved

Bit 10	Read Preamble Training Mode
0	Disabled
1	Enabled

Bit 11	Read Preamble
0	1 nCK
1	2 nCK

Bit 12	Write Preamble
0	1 nCK
1	2 nCK

8.6. Mode Register 5

BG1	BG0,BA	17,13	12	11	10	9	8:6	5	4	3	2:0
RFU	MR Select	RFU	Read DBI	Write DBI	Data Mask	CA Parity Persistent Error	RTT-PARK	*	C/A Parity Error Status	CRC Error Clear	C/A Parity Latency

*ODT Input Buffer for Power Down

RTT_PARK and ODT Input Buffer for Power Down are not applicable to the DDR4 Palladium Model.

Bit 2	Bit 1	Bit 0	C/A Parity Latency
0	0	0	Disable
0	0	1	4
0	1	0	5
0	1	1	6
1	0	0	8
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

Bit 3	CRC Error Clear
0	Clear
1	Error

Bit 10	C/A Parity Error Status
0	Clear
1	Error

Bit 10	Data Mask
0	Disabled
1	Enabled

Bit 11	Write DBI
0	Disabled
1	Enabled

Bit 12	Read DBI
0	Disabled

1	Enabled

Write operation: Either Data Mask or Write DBI can be enabled but both cannot be enabled at the same time.

8.7. Mode Register 6

Mode register 6 is implemented in the DDR4 model. The model does not support any contents but is required to be present as it is part of the initialization sequence.

9. Features

The following table shows a list of features and feature support for the DDR4 model:

FEATURE	SUPPORT	NOTE
COMMANDS		
Mode Register Set	Yes	
Refresh	Yes	
Self Refresh Entry	No	
Self Refresh Exit	No	
Single Bank Precharge	Yes	
Precharge all Banks	Yes	
Bank Activate	Yes	
Write (Fixed BL8 or BC4)	Yes	
Write (BC4, on the Fly)	Yes	
Write (BL8, on the Fly)	Yes	
Write with Auto Precharge (Fixed BL8 or BC4)	Yes	
Write with Auto Precharge (BC4, on the Fly)	Yes	
Write with Auto Precharge (BL8, on the Fly)	Yes	
Read (Fixed BL8 or BC4)	Yes	
Read (BC4, on the Fly)	Yes	
Read (BL8, on the Fly)	Yes	
Read with Auto Precharge (Fixed BL8 or BC4)	Yes	
Read with Auto Precharge (BC4, on the Fly)	Yes	
Read with Auto Precharge (BL8, on the Fly)	Yes	
No Operation	Yes	
Device Deselected	Yes	
Power Down Entry	No	
Power Down Exit	No	
ZQ calibration Long	No	ZQC command is required to complete initialization sequence
ZQ calibration Short	No	
SPECIAL FEATURES		
On Die Termination (ODT)	No	ODT related features are not supported
WR and RTP	No	MR0[11:9]
DLL Reset	No	MR0[8]
TM	No	MR0[7]
CAS Latency	Yes	MR0[6:4],MR0[2]
Qoff	Yes	MR1[12]
TDQS enable	No	MR1[11]
RTT_NOM	No	MR1[10:8]
Write Leveling Enable	Yes	MR1[7]
Additive Latency	Yes	MR1[4:3]

FEATURE	SUPPORT	NOTE
Output Driver Impedance Control	No	MR1[2:1]
DLL Enable	No	MR1[0]
Target Row Refresh (TRR)	No	MR2[13]
Write CRC	Yes	MR2[12]
RTT_WR	No	MR2[11:9]
TRR Mode	No	MR2[8],MR2[2:0]
Low Power Array Self Refresh	No	MR2[7:6]
MPR Write	Yes	Page 0 only per Jedec spec
MPR Read Format	Yes	MR3[12:11]
MPR serial read format	Yes	
MPR parallel read format	Yes	Page 0 only per Jedec spec
MPR stagger read format	Yes	Page 0 only per Jedec spec
Write Command Latency when CRC and DM are both enabled	No	MR3[10:9]
Fine Granularity Refresh Mode	No	MR3[8:6]
Temperature sensor readout	No	MR3[5]
Per DRAM Addressability	No	MR3[4]
Geardown Mode	No	MR3[3]
MPR Operation	Yes	MR3[2]
MPR page selection	Yes	MR3[1:0]
Post Package Repair (PPR)	No	MR4[13]
Write Preamble	Yes	MR4[12]
Read Preamble	Yes	MR4[11]
Read Preamble Training	Yes	MR4[10]
Self Refresh Abort	No	MR4[9]
CS to CMD/ADDR Latency	Yes	MR4[8:6]
Internal Vref Monitor	No	MR4[4]
Temperature Controlled Refresh Mode/Range	No	MR4[3:2]
Maximum Power Down Mode	No	MR4[1]
Read DBI	Yes	MR5[12]
Write DBI	Yes	MR5[11]
Data Mask	Yes	MR5[10]
CA Parity Persistent Error	Yes	MR5[9]
RTT_PARK	No	MR5[8:6]
ODT Input Buffer during Power Down mode	No	MR5[5]
C/A Parity Error Status	Yes	MR5[4]
CRC Error Clear	Yes	MR5[3]
C/A Parity Latency	Yes	MR5[2:0]
tCCD_L	No	MR6[12:10]
VrefDQ Training	No	MR6[7:0]
DIMM FEATURES		
SPD – Serial Presence Detect	Yes	
Address Mirroring	Yes	
DQ Mapping for CRC	Yes	

FEATURE	SUPPORT	NOTE
RCD – Registering Clock Driver	No	Address and Control signals are registered; however, other functions like "Output Inversion" are not supported.

10. Initialization Sequence

The DDR4 model requires that the memory controller follows the initialization sequence as documented in the specification. The sequence basically entails the following steps:

1. Assert RESET
2. De-assert RESET
3. Start clocks
4. Wait for CKE to asserted
5. Write to all seven Mode Registers
6. Issue ZQC command

Generally there is no ordering required for step 5 but all mode registers need to be written. The model requires that these steps are performed in the correct sequence in order to complete initialization. The model will not respond to any others commands unless this sequence is completed. If initialization sequence needs to be bypassed the init_done signal may be forced to high, and mode registers can be set by forcing signals MR0, MR1, MR2, MR3, MR4, MRS5, MR6.

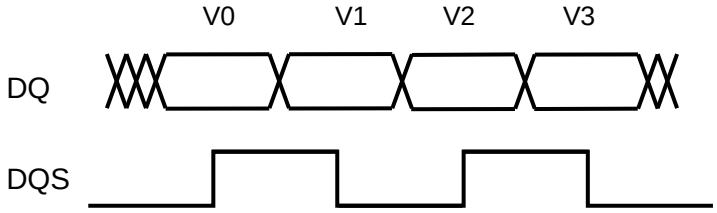
11. Limitations

Currently the DDR4 model does not support the following features, as well as others listed as unsupported in the Features section of this user guide:

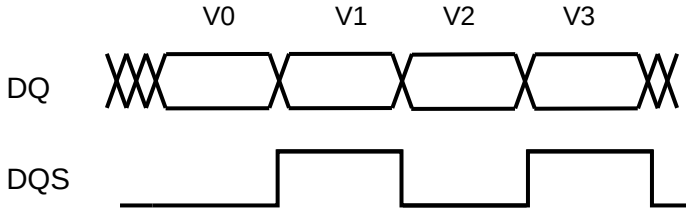
- MR3[3] Geardown Mode
- MR3[4] Per DRAM Addressability
- MR3[10:9] Write CMD Latency
- ODT related features
- RDIMM vendor specific components such as Texas Instruments CAB4A and temperature sensor
- Thermal Sensor operations
- Timing parameters

12. Handling DQS in Palladium Memory Models

For writes to a DDR memory, industry datasheets show each DQS edge centered within the corresponding valid period (v0, v1, v2, etc.) of DQ, as in the following diagram.

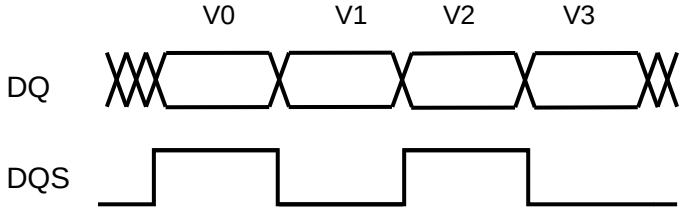


For DDR models provided by Cadence for Palladium, if the design drives DQ and DQS signals with the above timing, the DDR memory will behave correctly. However, to obtain this timing in Palladium, the fastest design clock must toggle twice as frequently as the DQS signal. If this faster clock is not needed for any other reason, the presence of the faster clock will usually cause an unnecessary 2X slowdown in emulation speed. To eliminate the need for a faster clock, you can have the design generate each DQS edge at the end of the corresponding DQ valid period (rather than the middle), as in the following diagram:



Note that the first DQS edge is at the *end* of first valid DQ, not at the beginning.

For reads from the DDR model, the DDR model will drive DQ and DQS with the first DQS edge at the *beginning* of the first valid data, not at the end:



The DDR model behaves this way to conform with industry datasheets for DDR memories. The design reading the data from the DDR model must delay the DQS signal, and use the delayed-DQS signal to sample the DQ. A delay of one Q_FDP0B should work fine, even in CAKE 1X mode. If you are using CAKE 1X mode and the DDR clock is the fastest design clock, the DQ signal will change twice per FCLK, and the Q_FDP0B delaying DQS will provide one-half FCLK delay, so that each delayed-DQS edge is at the end of the corresponding data valid period.

To delay the DQS signal, a commonly used approach is to create a special pad cell for DQS, that has a Q_FDP0B delay cell inserted on the path that leads from the DDR memory into the design.

The user may insert delays into pad cells (or elsewhere in the design) using the below code example which leverages ixc_pulse, an internal primitive that can be used to access FCLK and to create controlled delay, for IXCOM flow and leverages the Q_FDP0B primitive for delay generation in the Classic ICE flow. For more detailed information about ixc_pulse please reference the *UXE User Guide* section called *Generating Pulses*. There is no need for the user to define IXCOM_UXE for the Verilog macro; it is predefined for the user in IXCOM flow. Note that in UXE 13.1.0 and prior the equivalent pulse generating function was named axis_pulse.

```
// Flow independent delay cell
module pxp_fclk_delay (in, out_delay);
input in;
output out_delay;

reg out_delay;

`ifdef IXCOM_UXE
  wire VCC=1'b1;
  ixc_pulse #(1)(Fclk,VCC);
  always @(posedge Fclk)
    out_delay <= in;
`else
  Q_FDP0B fclk_dly (.D(in), .Q(out_delay));
`endif

endmodule
```

13. DIMMs

13.1. Configurations

The following table shows a list of available DIMM configurations.

Model Name	Memory Size	Data Width	Registered/ Unbuffered	Bank Group	Bank Address	Row Address	Column Address	Ranks
jedec_ddr4_2GB_72_rdimm	2GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[13:0]	A[9:0]	1
jedec_ddr4_2GB_72_udimm	2GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[13:0]	A[9:0]	1
jedec_ddr4_2GB_64_rdimm	2GB	x64	RDIMM	BG[1:0]	BA[1:0]	A[13:0]	A[9:0]	1
jedec_ddr4_2GB_64_udimm	2GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[13:0]	A[9:0]	1
jedec_ddr4_4GB_72_rdimm	4GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
jedec_ddr4_4GB_72_udimm	4GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
jedec_ddr4_4GB_64_rdimm	4GB	x64	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
jedec_ddr4_4GB_64_udimm	4GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
jedec_ddr4_8GB_72_rdimm	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
jedec_ddr4_8GB_72_udimm	8GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
jedec_ddr4_8GB_64_rdimm	8GB	x64	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
jedec_ddr4_8GB_64_udimm	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
jedec_ddr4_16GB_72_rdimm	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
jedec_ddr4_16GB_72_udimm	16GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
jedec_ddr4_16GB_64_rdimm	16GB	x64	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
jedec_ddr4_16GB_64_udimm	16GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
jedec_ddr4_16GB_2r_72_rdimm	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
jedec_ddr4_16GB_2r_72_udimm	16GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
jedec_ddr4_16GB_2r_64_rdimm	16GB	x64	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
jedec_ddr4_16GB_2r_64_udimm	16GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
jedec_ddr4_32GB_2r_72_rdimm	32GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A{9:0]	2
jedec_ddr4_32GB_2r_72_udimm	32GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[16:0]	A{9:0]	2
jedec_ddr4_32GB_2r_64_rdimm	32GB	x64	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A{9:0]	2
jedec_ddr4_32GB_2r_64_udimm	32GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[16:0]	A{9:0]	2
mta4atf25664az	2GB	x64	UDIMM	BG[0]	BA[1:0]	A[14:0]	A[9:0]	1
mta4atf51264az	4GB	x64	UDIMM	BG[0]	BA[1:0]	A[15:0]	A[9:0]	1
mta8atf51264az	4GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
mta9asf51272az	4GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
mta9asf51272pz	4GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
mt8atf1g64az	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
mta16atf1g64az	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
mta18adf1g72pz	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
mta18asf1g72pdz	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
mta18asf1g72pz	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
mta16atf2g64az	16GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
mta18asf2g72az	16GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
mta18adf2g72pz	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
mta18asf2g72pz	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
mta18asf2g72pdz	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
mta36asf2g72pz	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
mta36ads4g72pz	32GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	2
mta36asf4g72pz	32GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	2
m393a5143db0	4GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
m393a1g40db0	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
m393a1g40db1	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
m393a1g43db0	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m393a1g43db1	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m393a1g40eb1	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
m393a1g43eb1	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m393a1k43bb0	8GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
m393a2g40db0	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
m393a2g40db1	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2

Model Name	Memory Size	Data Width	Registered/ Unbuffered	Bank Group	Bank Address	Row Address	Column Address	Ranks
m393a2g40eb1	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
m393a2k40bb0	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
m393a2k40bb1	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	1
m393a2k43bb1	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
m393a4k40bb0	32GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	2
m393a4k40bb1	32GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	2
m393a8g40d40	64GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2 x 4H
m393a8k40d21	64GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	2 x 2H
m393aak40d41	128GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	2 x 4H
m392a2g40dm0	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
m392a2k43bb0	16GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
m392a4k40bm0	32GB	x72	RDIMM	BG[1:0]	BA[1:0]	A[16:0]	A[9:0]	2
m474a1g43db0	8GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m474a1g43db1	8GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m474a1g43eb1	8GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m474a2k43bb1	16GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
m471a5644eb0	2GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[13:0]	A[9:0]	1
m471a5143db0	4GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
m471a5143eb0	4GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
m471a5143eb1	4GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
m471a1g43db0	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m471a1g43eb1	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m471a1k43bb0	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
m471a1k43bb1	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
m471a2k43bb1	16GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
m391a5143eb1	4GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
m391a1g43db0	8GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m391a1g43db1	8GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m391a1g43eb1	8GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
m391a2k43bb1	16GB	x72	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2
M378a5644eb0	2GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[13:0]	A[9:0]	1
M378a5143db0	4GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
M378a5143eb1	4GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	1
M378a1g43db0	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
M378a1g43eb1	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[14:0]	A[9:0]	2
M378a1k43bb1	8GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	1
M378a2k43bb1	16GB	x64	UDIMM	BG[1:0]	BA[1:0]	A[15:0]	A[9:0]	2

There are a few parameters that can be set to adjust the configuration of a model. One limitation is the sum of row_addr_width, bank_grp_width, bank_addr_width, and col_addr_width should be less than 31 because of the 1G address limit in IXCOM. Please see Large Memory Support section if the sum exceeds 30.
Here is a list of parameters:

```
parameter   data_bits = 72;  // width of DQ bus
parameter   addr_bits = 15;  // row address width
parameter   bank_grp_width = 2;  // bank group width
parameter   bank_addr_width = 2;  // bank address width
parameter   row_addr_width = addr_bits;  // row address width
parameter   low_addr_bits = 14;  // mode register data width
parameter   col_addr_width = 10;  // column address width
parameter   byte_width = 8;  // byte_width = 4 if DQ width is 4; otherwise byte_width = 8
```

13.2. DQ Mapping for CRC

To support CRC in DIMMs, DQ mapping may be enabled by assigning an index to each nibble of DQ. The following table from JEDEC's RDIMM spec shows the possible index values a vendor could use. Please see JEDEC's spec for a more thorough description of this mapping requirement.

DQ Map Index (Hex)	Connector – bit within nibble				DQ Map Index (Hex)	Connector – bit within nibble			
	0	1	2	3		0	1	2	3
	SDRAM bit					SDRAM bit			
0x01	0	1	2	3	0x21	4	5	6	7
0x02	0	1	3	2	0x22	4	5	7	6
0x03	0	2	1	3	0x23	4	6	5	7
0x04	0	2	3	1	0x24	4	6	7	5
0x05	0	3	1	2	0x25	4	7	5	6
0x06	0	3	2	1	0x26	4	7	6	5
0x07	1	0	2	3	0x27	5	4	6	7
0x08	1	0	3	2	0x28	5	4	7	6
0x09	1	2	0	3	0x29	5	6	4	7
0x0A	1	2	3	0	0x2A	5	6	7	4
0x0B	1	3	0	2	0x2B	5	7	4	6
0x0C	1	3	2	0	0x2C	5	7	6	4
0x0D	2	0	1	3	0x2D	6	4	5	7
0x0E	2	0	3	1	0x2E	6	4	7	5
0x0F	2	1	0	3	0x2F	6	5	4	7
0x10	2	1	3	0	0x30	6	5	7	4
0x11	2	3	0	1	0x31	6	7	4	5
0x12	2	3	1	0	0x32	6	7	5	4
0x13	3	0	1	2	0x33	7	4	5	6
0x14	3	0	2	1	0x34	7	4	6	5
0x15	3	1	0	2	0x35	7	5	4	6
0x16	3	1	2	0	0x36	7	5	6	4
0x17	3	2	0	1	0x37	7	6	4	5
0x18	3	2	1	0	0x38	7	6	5	4

The index values are provided in spd byte number 60 to 77 or 3Ch to 4Dh. To enable this mapping in the Palladium DIMM model, a data file needs to be preloaded into an array in the spd eeprom.

Here is an example data file for this array:

```
@3c
2b
03
15
27
30
18
17
25
09
34
10
24
27
05
31
0a
12
2e
```

These values, along with other parameters, may also be preloaded into the SPD EEPROM memory at the following path:

<path.to.dimm.inst>.mem for single rank DIMMs.

<path.to.dimm.inst>.ddr4_rdimm_i0.mem for 2 or 4-rank RDIMMs.
<path.to.dimm.inst>.ddr4_rdimm_i1.mem for 2 or 4-rank RDIMMs.
<path.to.dimm.inst>.ddr4_rdimm_i2.mem for 4-rank RDIMMs.
<path.to.dimm.inst>.ddr4_rdimm_i3.mem for 4-rank RDIMMs.

<path.to.dimm.inst>.ddr4_udimm_i0.mem for 2 or 4-rank UDIMMs.
<path.to.dimm.inst>.ddr4_udimm_i1.mem for 2 or 4-rank UDIMMs.
<path.to.dimm.inst>.ddr4_udimm_i2.mem for 4-rank UDIMMs.
<path.to.dimm.inst>.ddr4_udimm_i3.mem for 4-rank UDIMMs.

<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i0.ddr4_rdimm_i0.mem for rank 0 of stack 0.
<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i0.ddr4_rdimm_i1.mem for rank 1 of stack 0.
<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i1.ddr4_rdimm_i0.mem for rank 0 of stack 1.
<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i1.ddr4_rdimm_i1.mem for rank 1 of stack 1.
<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i2.ddr4_rdimm_i0.mem for rank 0 of stack 2.
<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i2.ddr4_rdimm_i1.mem for rank 1 of stack 2.
<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i3.ddr4_rdimm_i0.mem for rank 0 of stack 3.
<path.to.3ds.dimm.inst>.ddr4_rdimm_2rank_i3.ddr4_rdimm_i1.mem for rank 1 of stack 3.

Rank mapping is defined by bits 7 and 6 of the DQ Map Index value as described in JEDEC's DDR4 SPD Contents Master Specification JC-45-2220.01 (ddr4_spd_spec_10_22_2012.pdf). Rank mapping defines the connectivity between bits in different package ranks (even and odd ranks).

Package Rank Map	
Bits 7:6	**Bit Order at SDRAM**
00	Even package ranks (0, 2, etc.) have the same mapping. Odd package ranks (1, 3, etc.) map SDRAM data bits relative to Package Rank 0 as follows: DQ0 → DQ1 DQ1 → DQ0 DQ2 → DQ3 DQ3 → DQ2 DQ4 → DQ5 DQ5 → DQ4 DQ6 → DQ7 DQ7 → DQ6 DQ8:DQ31 → TBD
01	Reserved
10	Reserved
11	Reserved

13.3. Address Mirroring

The DIMM models are compiled as single rank by default which does not have address mirroring enabled. If an odd rank or a multiple rank model is needed please request it from customer support. Or, the user may change the following line in the .vp file and recompile for an odd rank.

From: wire MF = 1'b0;
To: wire MF = 1'b1;

14. Synthesis & Compilation

The memory models are currently provided in one format: an encrypted RTL file(s) (*.vp) that targets use in either the IXCOM flow or in the ICE flow. The encrypted RTL (*.vp) file(s) must be synthesized along with other design code prior to acceleration / emulation.

Shown below are some simplified commands for compiling the base DDR4 models in IXCOM flow and ICE flow.

NOTE: The DDR4 family has several variations and configurations. Consult the discussion below in this section and the sections titled "**Base Model Filelist**" and "**DIMM Model Filelis**t" to know what files are necessary for building the selected DDR4 or DDR4DIMM part. Be sure to include all the needed files when compiling the model into a design. Alternatively, the user may include all the files listed in the filelist and let the synthesis tool discard the unneeded modules.

If {row,bg,ba,col} addressing is needed, add **+define+MMP_RBC** to the vlan invocation. If {ba,bg,row,col} addressing is needed, add **+define+MMP_BA_BG** to the vlan invocation.

IXCOM example commands:

```
vlan      -64bit +sv tb.v \
          <model_name>.vp \
          -incdir ../../../utils/cdn_mmp_utils/sv \
          ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
          ${UXE_HOME}/etc/ixcom/IXCclkgen.sv \
          -vlog_ext .vp \
          -v ${UXE_HOME}/etc/ixcom/ixcom_hdlice_ref.vp

ixcom –ua +dut+<model_name> -top tb
```

ICE flow synthesis commands:

vavlog ../src/<model_name>.vp

vaelab –keepRtlSymbol –keepAllFlipFlop –outputVlog <model_name>.vgp <model_name>

NOTE: It is common for Palladium flows to require –keepallFlipFlop since it removes optimizations that are in place by default. For example, without –keepAllFlipFlop, HDL-ICE can remove flops with constant inputs and merge equivalent FF. The picture above is modified a bit when ICE ATB mode (–atb) is used since then a constant input FF is only optimized out when there is no initial value for it or the initial value is the same as the constant input value.

It is also common for Palladium flows to require –keepRtlSymbol. This option enables the HDL Compiler to keep original VHDL RTL symbols, such as ".", whenever possible. In

other words, it maps VHDL RTL signal name a.b to the netlist entry, \a.b. Without this modifier, the signal name would otherwise be converted to a_b in the netlist.

If the recommended compile script includes the aforementioned options, the user must include them to avoid affecting functionality of the design.

For large memory models that exceed 30 bits of address width, include the files mmp_gen_mem.vp and mmp_submem.vp in the build as shown below. These files will incorporate the multiple core memory array generator into the build and the Verilog macro MMP_LG_MEM_BITS will automatically be defined. See the section on Large Memory Support for more details.

```
vlan      -64bit +sv tb.v \
          <model_name>.vp \
          mmp_gen_mem.vp mmp_submem.vp \
          ${UXE_HOME}/etc/ixcom/IXCclkgen.sv \
          -incdir ../../../utils/cdn_mmp_utils/sv \
          ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
          -vlog_ext .vp \
          -v ${UXE_HOME}/etc/ixcom/ixcom_hdlice_ref.vp

ixcom –ua +dut+<model_name> -top tb

xeDebug -64 --ncsim \
 -sv_lib ../../../utils/cdn_mmp_utils/lib/64bit/libMMP_utils.so -- \
 hw.i
```

Note: Files mmp_gen_mem.vp and mmp_submem.vp are located in the sdram/common sub-directory under the MMP installation.

Below are shown some simple commands for compiling the standard DDR4 DIMM models in the IXCOM flow. Please notice that a few additional input files are needed.

```
vlan      -64bit +sv tb.v \
          <dimm_model_name>.vp \
          ddr4_wide.vp \
          crc_map_read.vp \
          crc_map_write.vp \
          -incdir ../../../utils/cdn_mmp_utils/sv \
          ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
          ${UXE_HOME}/etc/ixcom/IXCclkgen.sv \
          -vlog_ext .vp \
          -v ${UXE_HOME}/etc/ixcom/ixcom_hdlice_ref.vp

ixcom –ua +dut+<dimm_model_name> -top tb

xeDebug -64 --ncsim \
-sv_lib ../../../utils/cdn_mmp_utils/lib/64bit/libMMP_utils.so -- \
hw.i
```

For large memory DIMM models that exceed 30 bits of address width, include the files mmp_gen_mem.vp, mmp_submem.vp, and ddr4_wide_lg_mem.vp in the vlan step of the build as shown below. These files will incorporate the multiple core memory array generator into the build and the Verilog macro MMP_LG_MEM_BITS will automatically be defined. See the section on Large Memory Support for more details.

```
vlan      -64bit +sv tb.v \
          <dimm_model_name>.vp \
          ddr4_wide_lg_mem.vp \
          crc_map_read.vp \
          crc_map_write.vp \
          mmp_gen_mem.vp mmp_submem.vp \
          -incdir ../../../utils/cdn_mmp_utils/sv \
          ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
          ${UXE_HOME}/etc/ixcom/IXCclkgen.sv \
          -vlog_ext .vp \
          -v ${UXE_HOME}/etc/ixcom/ixcom_hdlice_ref.vp
```

For 3DS RDIMM models that have 2 ranks per stack please add ddr4_rdimm_2rank.vp and ddr4_rdimm.vp to the compilation step as follows:

```
vlan      -64bit +sv tb.v \
          <dimm_model_name>.vp \
          ddr4_rdimm_2rank.vp ddr4_rdimm.vp \
          ddr4_wide.vp \
          crc_map_read.vp \
          crc_map_write.vp \
          -incdir ../../../utils/cdn_mmp_utils/sv \
          ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
          ${UXE_HOME}/etc/ixcom/IXCclkgen.sv \
          -vlog_ext .vp \
          -v ${UXE_HOME}/etc/ixcom/ixcom_hdlice_ref.vp
```

The content of the hw.i file used in the example runtime command shown above may have commands such as these shown below:

```
debug .
host .
xc xt0 zt0 run
run
exit
```

The above examples are intended to show the difference between compiling DDR4 models and DDR4 DIMM models. Please see the UXE or VXE user guide for more details on the IXCOM flow.

14.1 Base Model File List

<model_name>.vp – DDR4 model
mmp_gen_mem.vp – generate block to generate multiple arrays*
mmp_submem.vp – core memory array*

14.2 DIMM Model File List

<dim_model_name>.vp – DIMM model wrapper
ddr4_wide.vp – ddr4 model with expanded bus width
ddr4_wide_64.vp—ddr4 model with expanded bus width + 64 bit
ddr4_wide_lg_mem.vp – ddr4 model with expanded bus and large memory support*
crc_map_read.vp – output DQ mapper
crc_map_write.vp – input DQ mapper
mmp_gen_mem.vp – generate block to generate multiple arrays*
mmp_submem.vp – core memory array*
ddr4_rdimm_2rank.vp – stack wrapper that instantiates multiple ranks
ddr4_rdimm.vp – rdimm wrapper that instantiates ddr4_wide
ddr4_rdimm_64.vp – 64-bit model
ddr4_udimm.vp -- udimm wrapper that instantiates ddr4_wide
ddr4_udimm_64.vp — 64 bit model

*Please see Large Memory Support section for more details. Files mmp_gen_mem.vp and mmp_submem.vp are located in the sdram/common sub-directory under the MMP installation.

15. SWI Smart Memory Interface

This DDR4 core model supports a fixed, Verilog macro controlled interface, or “hooks”, to one of the Smart Memory components in Cadence’s Software Integrator (SWI) product. Software Integrator (SWI) is a support library that is part of Cadence’s Hybrid Solution--a multi-tool system level solution that runs in a hybrid PXP + IES simulation mode and targets the increasing number of SoC performance conscious projects requiring the booting of commercial operating systems and the running of complex software-driven tests against an accurate model of the SoC prior to tapeout.

For additional details about the SWI product at large please consult the SWI product documentation which includes a user guide. This documentation can be accessed via support.cadence.com and is located on the Product Pages/Product Manuals link where SWI 13.1 is located with other Functional Verification products.

The user of the SWI solution who is integrating this core model to the corresponding SWI Smart Memory component side should define the Verilog macro “MMP_SM” to enable the Smart Memory interface. This will enable the portion of the interface that resides in the MMP model core, thus completing access to the implemented SWI Smart Memory functionality.

Table 1: SWI Smart Memory Verilog Define

`define Macro purpose	Possible `define Values
Set the Smart Memory interface to compile “in” as part of the memory model	MMP_SM

The SWI Smart Memory interface includes the signals shown in Table 2: SWI Smart Memory Interface Signals. It is outside the MMP scope to treat the integration of the MMP model into a hybrid solution. For additional details, please consult the SWI documentation and other Hybrid Solution documentation.

Table 2: SWI Smart Memory Interface Signals

NAME	DECLARATION	DESCRIPTION
sm_raddr	output [(total_addr_bits-1):0] sm_raddr	Smart Memory read address
sm_re	output sm_re	Smart Memory read enable
sm_raddr_en	output sm_raddr_en	Smart Memory read address enable
sm_dout	input [data_bits-1:0] sm_dout	Smart Memory read data
sm_waddr	output [(total_addr_bits-1):0] sm_waddr	Smart Memory write address
sm_we	output sm_we	Smart Memory write enable
sm_waddr_en	output sm_waddr_en	Smart Memory write address enable
sm_din	output [data_bits-1:0] sm_din	Smart Memory write data
sm_bw	output [data_bits-1:0] sm_bw	Smart Memory data mask
verbosity	input [3:0] verbosity	Smart Memory debug info display level control

The Smart Memory interface includes a user adjustable parameter that passes user data from wrapper layers that are external to this MMP model into MMP model. The single 64-bit parameter is subdivided by field to accommodate data as shown in Table 3: SWI Smart Memory User Adjustable Parameters below. This parameter defaults to a value of '0' and is managed by the SWI product Smart Memory component. For additional details, please consult the SWI documentation and other Hybrid Solution documentation.

Table 3: SWI Smart Memory User Adjustable Parameters

PARAMETER	DESCRIPTION
parameter [63:0] SMConfigSpecific_UserData	Smart Memory User Data Passing
FIELD	**DESCRIPTION**
[2:0]	Used for specifying device/channel/subpart
[63:61]	Extension value of total_addr_bits

The Smart Memory interface includes non-user adjustable localparams and parameters as shown in Table 4: SWI Smart Memory Non-User Adjustable Parameters below. Note that a localparam of the same name (total_addr_bits) but of different size is part of the standard MMP core model.

Table 4: SWI Smart Memory Non-User Adjustable Parameters

LOCALPARAM	VALUE	DESCRIPTION
total_addr_bits	bank_addr_width+row_addr_width +col_addr_width + SMConfigSpecific_UserData[63:61]	Memory capacity width

The SWI Smart Memory interface has dependencies on the inclusion of external file(s). For additional details about purpose and content of any Smart Memory related external files, please consult the SWI documentation and other Hybrid Solution documentation.

```
`ifdef MMP_SM
    `include "cdn_sm_mapDRBCToLinAdr.vh"
`endif
```

16. MMP and ECC (Error Correcting Code)

MMP models do not support Error Correcting Code (ECC) functionality. ECC functions, if they are present in a memory device, are typically found in the NAND and DDRx families. The MMP product does not have any plans to provide such functions in the models. MMP models are provided as system level emulation models and not as verification IP. The below sections discuss work-arounds that enable the user to deal with some ECC scenarios. Note that ECC means different things to different device families.

16.1. DDRx and ECC (Error Correcting Code)

Error Correction Code (ECC) allows single bit errors to be corrected and other bit errors to be detected, thus improving high frequency operation reliability and data accuracy.
While the MMP DDRx models do not include any ECC functionality or handling, the user can arrange to “support” ECC on the DDR model first by using a model with a 72bit datapath and, additionally, by artificially injecting errors using some external mechanism, if such is needed. To word this differently, MMP models can often be found that interface with the memory controller data path, thus satisfying connectivity requirements. This arrangement might support a user who needs to test ECC related error conditions in emulation. The following paragraphs provide some details for consideration.

For DDRx models, ECC is performed by the controller. The controller calculates an ECC value and writes it to the upper 8 bits of a 72bit DIMM. When the controller writes data or when it reads data, it re-calculates the ECC based on current read data then checks the resulting value against the read ECC stored in the upper 8 bits. If the re-calculated value is equivalent to the stored value, then there is no error.

So, to support ECC in an MMP model during acceleration or emulation, the user first needs a 72bit data path to provide the added ports and data path for handling the upper 8 bits of ECC above and beyond the standard 64bit data path. In other words, a normal DDR memory has a 64bit data bus where the ECC memory will have a 72bit data bus and 9 bits for DQS and DM. For most scenarios, ones where the user will not enable ECC function in their controller and will not inject errors, this operation is compatible for use with MMP models. The expanded ports and data path constitutes all of the support that MMP models can provide toward ECC handling.

There are two potential paths for achieving the 72bit data bus. The first is to select and use a 72bit DIMM. Please review the DIMM pages of the MMP catalogue for an appropriate 72bit DIMM. If an appropriate 72bit part cannot be found, the user may contact Cadence support to initiate a request for a 72bit data path version of the model of interest. A second approach to consider for some models is to expand the data path of a smaller width data bus to 72bits. The safest way to expand the data path to 72bits is to modify the data width parameter to 72 in the standard 64bit model. The user can set the parameter *data_bits* to 72 when instantiating the model. Not all models, however, have a *data_bits* parameter available for configuration.

Next, the user considers the error injection aspect. There is no capability in MMP models for error injection. MMP models are provided as system level emulation models and not as verification IP. For specifically testing error conditions, the user needs to work around the gap. One alternative that the user might consider is to corrupt memory data in order to mimic data corruption. In this scenario, the user can execute the xeDebug memory command to write some incorrect data to the array. For example, the following command will corrupt one location in an array:

```
memory -setvalue ddr3_inst -value 0x01234567 -start 0x10000100 -end 0x10000100
```

Another alternative is to consider using MMAV/Denali simulation models in IUS.

17. Large Memory Support

MMP model capacity has been limited to 1G words due to the current 30 bit address width limitation in IXCOM. To work around this constraint, a multiple core memory array generator (mmp_gen_mem.vp) has been incorporated into these large sized memory models that splits larger core memory arrays into multiple 1G arrays. In these cases, the file mmp_gen_mem.vp automatically defines Verilog macro MMP_LG_MEM_BITS with a value of '30' by default.

17.1 Preloading Multiple Arrays

With a single array of 30 bits or less the address mapping is simply: array_addr = {BG,BA,ROW,COL}. In this case the hierarchical memory array name appears thusly:

```
tb_top.rtl_module.ddr4_wide_i0.memcore
```

In scenarios with array address space that is greater than 30 bits, there are multiple arrays which are mapped using distinctly different hierarchical naming. The hierarchical path for the array names associated with each core memory array of the large memory are reported as output to the "memory –list" command or can be viewed in the dbFiles/xcva_top_et5mpart file. For example, with a 32 bit address the user will see 4 arrays with naming as follows:

```
tb_top.rtl_module.ddr4_wide_i0.mem1.\multiple.array_0_.u1 .memcore addresses 0  ..  1G - 1
tb_top.rtl_module.ddr4_wide_i0.mem1.\multiple.array_1_.u1 .memcore addresses 1G .. 2G - 1
tb_top.rtl_module.ddr4_wide_i0.mem1.\multiple.array_2_.u1 .memcore addresses 2G .. 3G - 1
tb_top.rtl_module.ddr4_wide_i0.mem1.\multiple.array_3_.u1 .memcore addresses 3G .. 4G - 1
```

Multiple data files for preloading each separate memory array are also required. In the example above, there will be four data files needed to preload the entire large memory.

Likewise, multiple memory –load commands are needed to preload the large memory of this example. An xeDebug preload example for the case above will look as follows:

```
memory -load %readmemh
tb_top.rtl_module_w.rtl_module.ddr4_wide_i0.mem1.multiple.array_0_.u1.memcore -file mem0.dat

memory -load %readmemh
tb_top.rtl_module_w.rtl_module.ddr4_wide_i0.mem1.multiple.array_1_.u1.memcore -file mem1.dat

memory -load %readmemh
tb_top.rtl_module_w.rtl_module.ddr4_wide_i0.mem1.multiple.array_2_.u1.memcore -file mem2.dat

memory -load %readmemh
tb_top.rtl_module_w.rtl_module.ddr4_wide_i0.mem1.multiple.array_3_.u1.memcore -file mem3.dat
```

Another example, 2 ranks with 31 bit address the user will see 4 arrays with naming as follows:

```
tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i0.ddr4_wide_i0.mem1.\multiple.array_0_.u1 .memcore addresses 0  ..  1G - 1
tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i0.ddr4_wide_i0.mem1.\multiple.array_1_.u1 .memcore addresses 1G .. 2G - 1
```

tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i1.ddr4_wide_i0.mem1.\multiple.array_0_.u1 .memcore addresses 0 .. 1G - 1
tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i1.ddr4_wide_i0.mem1.\multiple.array_1_.u1 .memcore addresses 1G .. 2G - 1

An xeDebug preload example for the case above will look as follows:

```
memory -load %readmemh
tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i0.ddr4_wide_i0.mem1.multiple.array_0_.u1.memcore -file r0_mem0.dat

memory -load %readmemh
tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i0.ddr4_wide_i0.mem1.multiple.array_1_.u1.memcore -file r0_mem1.dat

memory -load %readmemh
tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i1.ddr4_wide_i0.mem1.multiple.array_0_.u1.memcore -file r1_mem0.dat

memory -load %readmemh
tb_jedec_ddr4_32GB_2r_64_udimm.mmp0.ddr4_udimm_64_i1.ddr4_wide_i0.mem1.multiple.array_1_.u1.memcore -file r1_mem1.dat
```

17.2 Models with Multiple Arrays

Not every DDR4 model exceeds the 30 bit IXCOM memory limit. Below is a list of models with their associated address width, an indication of whether multiple arrays are required, and the number of arrays in the core memory.

Model Name	BG	BA	ROW	COL	Total bits	Multi Array
jedec_ddr4_2gb_16	1	2	14	10	27	No
jedec_ddr4_4gb_16	1	2	15	10	28	No
jedec_ddr4_8gb_16	1	2	16	10	29	No
jedec_ddr4_16gb_16	1	2	17	10	30	No
jedec_ddr4_2gb_8	2	2	14	10	28	No
jedec_ddr4_4gb_8	2	2	15	10	29	No
jedec_ddr4_8gb_8	2	2	16	10	30	No
jedec_ddr4_16gb_8	2	2	17	10	31	Yes, 2
jedec_ddr4_2gb_4	2	2	15	10	29	No
jedec_ddr4_4gb_4	2	2	16	10	30	No
jedec_ddr4_8gb_4	2	2	17	10	31	Yes, 2
jedec_ddr4_16gb_4	2	2	18	10	32	Yes, 4
jedec_ddr4_ddp_8gb_16	2	2	15	10	29	No
jedec_ddr4_ddp_16gb_16	2	2	16	10	30	No
jedec_ddr4_ddp_32gb_16	2	2	17	10	31	Yes, 2
mt40a1g4	2	2	16	10	30	No
mt40a512m8	2	2	15	10	29	No
mt40a256m16	1	2	15	10	28	No
mt40a2g4	2	2	17	10	31	Yes, 2
mt40a1g8	2	2	16	10	30	No

Model Name	BG	BA	ROW	COL	Total bits	Multi Array
mt40a512m16	1	2	16	10	29	No
k4a4g045wd	2	2	16	10	30	No
k4a4g085wd	2	2	15	10	29	No
k4a4g165wd	1	2	15	10	28	No
k4a8g045wb	2	2	17	10	31	Yes, 2
k4a8g085wb	2	2	16	10	30	No
k4a8g165wb	1	2	16	10	29	No
k4a4g045we	2	2	16	10	30	No
k4a4g085we	2	2	15	10	29	No
k4a4g165we	2	2	14	10	28	No

For DIMMs, the following table shows the same information per rank.

Model Name	BG	BA	ROW	COL	Total bits	Arrays
jedec_ddr4_2GB_72_rdimm	2	2	14	10	28	1
jedec_ddr4_2GB_72_udimm	2	2	14	10	28	1
jedec_ddr4_2GB_64_rdimm	2	2	14	10	28	1
jedec_ddr4_2GB_64_udimm	2	2	14	10	28	1
jedec_ddr4_4GB_72_rdimm	2	2	15	10	29	1
jedec_ddr4_4GB_72_udimm	2	2	15	10	29	1
jedec_ddr4_4GB_64_rdimm	2	2	15	10	29	1
jedec_ddr4_4GB_64_udimm	2	2	15	10	29	1
jedec_ddr4_8GB_72_rdimm	2	2	16	10	30	1
jedec_ddr4_8GB_72_udimm	2	2	16	10	30	1
jedec_ddr4_8GB_64_rdimm	2	2	16	10	30	1
jedec_ddr4_8GB_64_udimm	2	2	16	10	30	1
jedec_ddr4_16GB_72_rdimm	2	2	17	10	31	***2***
jedec_ddr4_16GB_72_udimm	2	2	17	10	31	***2***
jedec_ddr4_16GB_64_rdimm	2	2	17	10	31	***2***
jedec_ddr4_16GB_64_udimm	2	2	17	10	31	***2***
jedec_ddr4_16GB_2r_72_rdimm	2	2	16	10	30	1
jedec_ddr4_16GB_2r_72_udimm	2	2	16	10	30	1
jedec_ddr4_16GB_2r_64_rdimm	2	2	16	10	30	1
jedec_ddr4_16GB_2r_64_udimm	2	2	16	10	30	1
jedec_ddr4_32GB_2r_72_rdimm	2	2	17	10	31	***2***
jedec_ddr4_32GB_2r_72_udimm	2	2	17	10	31	***2***
jedec_ddr4_32GB_2r_64_rdimm	2	2	17	10	31	***2***
jedec_ddr4_32GB_2r_64_udimm	2	2	17	10	31	***2***
mta4atf25664az	1	2	15	10	28	1
mta4atff51264az	1	2	16	10	29	1
mta8atf51264az	2	2	15	10	29	1
mta9asf51272az	2	2	15	10	29	1
mta9asf51272pz	2	2	15	10	29	1
mta8atf1g64az	2	2	16	10	30	1
mta16atf1g64az	2	2	15	10	29	1
mta18adf1g72pz	2	2	16	10	30	1
mta18asf1g74pdz	2	2	15	10	29	1

Model Name	BG	BA	ROW	COL	Total bits	Arrays
mta18asf1g72pz	2	2	16	10	30	1
mta16atf2g64az	2	2	16	10	30	1
mta18asf2g72az	2	2	16	10	30	1
mta18adf2g72pz	2	2	17	10	31	***2***
mta18asf2g72pz	2	2	17	10	31	***2***
mta18asf2g72pdz	2	2	16	10	30	1
mta36asf2g72pz	2	2	16	10	30	1
mta36ads4g72pz	2	2	17	10	31	***2***
mta36asf4g72pz	2	2	17	10	31	***2***
m393a5143db0	2	2	15	10	29	1
m393a1g40db0	2	2	16	10	30	1
m393a1g40db1	2	2	16	10	30	1
m393a1g43db0	2	2	15	10	29	1
m393a1g43db1	2	2	15	10	29	1
m393a1g40eb1	2	2	16	10	30	1
m393a1g43eb1	2	2	15	10	29	1
m393a1k43bb0	2	2	16	10	30	1
m393a2g40db0	2	2	16	10	30	1
m393a2g40db1	2	2	16	10	30	1
m393a2g40eb1	2	2	16	10	30	1
m393a2k40bb0	2	2	17	10	31	***2***
m393a2k40bb1	2	2	17	10	31	***2***
m393a2k43bb1	2	2	16	10	30	1
m393a4k40bb0	2	2	17	10	31	***2***
m393a4k40bb1	2	2	17	10	31	***2***
m393a8g40d40	2	2	16	10	30	1
m393a8k40d21	2	2	17	10	31	***2***
m393aak40d41	2	2	17	10	31	***2***
m392a2g40dm0	2	2	16	10	30	1
m392a2k43bb0	2	2	16	10	30	1
m392a4k40bm0	2	2	17	10	31	***2***
m474a1g43db0	2	2	15	10	29	1
m474a1g43db1	2	2	15	10	29	1
m474a1g43eb1	2	2	15	10	29	1
m474a2k43bb1	2	2	16	10	30	1
m471a5644eb0	2	2	14	10	28	1
m471a5143db0	2	2	15	10	29	1
m471a5143eb0	2	2	15	10	29	1
m471a5143eb1	2	2	15	10	29	1
m471a1g43db0	2	2	15	10	29	1
m471a1g43eb1	2	2	15	10	29	1

Model Name	BG	BA	ROW	COL	Total bits	Arrays
m471a1k43bb0	2	2	16	10	30	1
m471a1k43bb1	2	2	16	10	30	1
m471a2k43bb1	2	2	16	10	30	1
m391a5143eb1	2	2	15	10	29	1
m391a1g43db0	2	2	15	10	29	1
m391a1g43db1	2	2	15	10	29	1
m391a1g43eb1	2	2	15	10	29	1
m391a2k43bb1	2	2	16	10	30	1
M378a5644eb0	2	2	14	10	28	1
M378a5143db0	2	2	15	10	29	1
M378a5143eb1	2	2	15	10	29	1
M378a1g43db0	2	2	15	10	29	1
M378a1g43eb1	2	2	15	10	29	1
M378a1k43bb1	2	2	16	10	30	1
M378a2k43bb1	2	2	16	10	30	1

DIMMs that need multiple arrays per rank use the memory array generator. To locate the hierarchical naming to use in preloading and other memory references, the user can examine the output from the “memory –list” command or review the dbFiles/xcva_top_et5mpart file.

The user also needs to compile with files mmp_gen_mem.vp, mmp_submem.vp, and ddr4_wide_lg_mem.vp instead of ddr4_wide.vp. Please see the section named “IXCOM Compilation” for an example. Below is an example for the vlan step using mmp_gen_mem.vp, mmp_submem.vp, and ddr4_wide_lg_mem.vp:

```
vlan    -64bit +sv tb.v \
        <dimm_model_name>.vp \
        ddr4_wide_lg_mem.vp \
        crc_map_read.vp \
        crc_map_write.vp \
        mmp_gen_mem.vp mmp_submem.vp \
        ${UXE_HOME}/etc/ixcom/IXCclkgen.sv \
        -vlog_ext .vp \
        -v ${UXE_HOME}/etc/ixcom/ixcom_hdlice_ref.vp
```

Note: Files mmp_gen_mem.vp and mmp_submem.vp are located in the sdram/common sub-directory under the MMP installation.

17.3 IUS Compilation

In IUS the address width limit is less than 30 bits. Even when the Verilog sparse directive is used in the model sometimes large data width may cause an error. If this data width error occurs, the user may override the default array generator bit setting to a smaller number to work around this issue. The macro named MMP_LG_MEM_BITS is set to 30 by default. It may be changed within the file mmp_gen_mem.vp or overridden on the command with the define

option. The user should note that more core memory arrays will be generated when the MMP_LG_MEM_BITS bit setting is reduced, which may then require additional data files for preloading the memory.

18. Debugging

This model has several debugging options, techniques and tips that may assist the user may use in isolating a problem.

- For issues that may not be DDR4 specific please review the *Memory Model Portfolio FAQ for All Models User Guide.*

- **Debug signals:** The following signals can be monitored to help detect incorrect command sequence:
  - err_init_cmd: Commands other than MRS, ZQC, NOP issued during init
  - err_ba_mrs: BG[1] is not '0' during MRS
  - err_mrs: MRS issued without all the banks precharged
  - err_ref: REF issued without all the banks precharged
  - err_act: ACT issued to a bank not precharged
  - err_zqc: ZQC issued without all the banks precharged
  - err_wr/err_rd: WRITE/READ issued to a bank not active
  - bad_rd/bad_wr_0: WRITE/READ doesn't satisfy tCCD = 4 cycles
  - bad_wr_1: Read to write violation

- **Golden waveform:** A package with a reference waveform is available which shows the following command sequence(s):

  (1) Initialization sequence:
  - CKE should be high
  - CK should be toggling
  - RESET_N transistions from low to high
  - MRS command (cmd=0) to write all seven Mode Registers followed by a ZQC command to complete initialization (init_done=1)

  (2) Additional commands in sequence include:
  - Various Write (cmd=4) and Read (cmd=5) sequences with Burst Length changing between 8 and 4

- **Debug Display:** This MMP memory model has available a built-in debug methodology called MMP Debug Display that is based on the Verilog system task $display. Please see the *Palladium Memory Model Debug Display User Guide* in the release docs directory for additional information.

- **Manual Configuring of this MMP Model Family**

  This MMP model supports manual configuration by accompanying the model mode register or configuration register declarations with synthesis directives, such as keep_net

directives, that instruct the compiler to ensure that the relevant nets remain available for runtime forcing. For a general description of this support please see the user guide in the MMP release with path and filename *docs/MMP_FAQ_for_All_Models.pdf*.

While MMP strongly recommends following protocol-based commands to configure MMP models, MMP recognizes that the design test environment may desire to trade off the risks inherent in streamlining or circumventing the initialization sequence part of the protocol in order to better support some testing environments.

The following table lists the internal register path and naming along with the specification or datasheet naming for model mode registers or configuration registers that are accompanied by keep_net synthesis directives in support of such manual configuration. ONLY writeable configuration registers or fields are supported thusly. Please read the relevant datasheet for details about individual register behavior and mapping to fields.

Table: Writeable Mode Register / Configuration Register Info

Hierarchical RTL Naming for Writeable Configuration Related Registers & Signals	Specification or Vendor Datasheet Naming for Configuration Related Registers	Access
<model_name>.MR0	MR0	W
<model_name>.MR1	MR1	W
<model_name>.MR2	MR2	W
<model_name>.MR3	MR3	W
<model_name>.MR4	MR4	W
<model_name>.MRS5	MR5	W
<model_name>.MR6	MR6	W
<model_name>.init_done	[Not Applicable]	1'b1 indicates initialization is complete

For RDIMM models the path to MR0 is <model_name>.ddr4_wide_i0.MR0.
For multi-rank RDIMM models the path to MR0 are <model_name>.ddr4_rdimm_2rank_i0.ddr4_wide_i0.MR0 and <model_name>.ddr4_rdimm_2rank_i1.ddr4_wide_i0.MR0.
To locate the hierarchical path to module instance, the user can examine the output from the "memory –list" command or review the dbFiles/xcva_top_et5mpart file.

19. Revision History

The following table shows the revision history for this document.

Date	Version	Revision
July 2012	1.0	Initial Release
July 2013	1.1	Added DIMM and IXCOM Compilation sections
October 2013	1.2	Removed Beta watermark
December 2013	1.3	Added JEDEC 2 rank DIMMs and Micron DIMMs, rank mapping, model size parameters, and MMP_RBC define
May 2014	1.4	Removed m34e02.vp from the ixcom compilation example
June 2014	1.5	Corrected a typo in DIMM configurations table. Added a note on {row,bank,col} address mapping.
July 2014	1.6	Repaired doc property title. Updated legal.
September 2014	1.7	Remove version from UG file name. Update UXE / IXE documentation reference titles. Updated Configuration section with info about multiplexed bits with non-address functions. Added paragraph about flow independent delay cell in “Handling DQS …” section. Added note about two formats in IXCOM compile section.
October 2014	1.8	Added CL 17, 19, 21 and CWL 20.
November 2014	1.9	Remove emulation capacity info.
January 2015	1.10	Added MMP and ECC section. Update related publications list.
March 2015	1.11	Removed 16GB per rank models due to IXCOM’s 30 bit address width limitation.
March 2015	1.12	Added Write Leveling and MPR support for page 0.
April 2015	1.13	Added parameter description and tables
July 2015	1.14	Added large memory support. Added back the 16GB per rank models.
July 2015	1.15	Update Cadence naming on front page
August 2015	1.16	Replaced gen_mem.vp with mmp_gen_mem.vp and added mmp_submem.vp.
September 2015	1.17	Added MMP_BA_BG macro for {ba,bg,row,col} address mapping. Removed references to <model_name>.vgp as an input file.
December 2015	1.18	Replaced Commands section with Features table and extended MPR support to pages 1, 2, 3. Added support for reading out CRC and CA parity status error log and Read Preamble Training.
January 2016	2.0	Update for Palladium-Z1 and VXE
February 2016	2.1	Added Samsung models.

Date	Version	Revision
March 2016	2.2	Added Micron models.
April 2016	2.3	Added SWI Smart Memory Interface
May 2016	2.4	Modified Synthesis and Compile section and filelist sections.
July 2016	2.5	Remove hyphen in Palladium naming
August 2016	2.6	Add DIMM features, limitations, debug signals
September 2016	2.7	Clarified BETA status of SWI interface in DDR4 models
November 2016	2.8	Remove BETA status of SWI infterface
November 2016	2.9	Bypass initialization sequence by forcing init_done and mode register signals
December 2016	3.0	Add “Manual Configuring of DDRx/LPDDRx” section
May 2017	3.1	Move mmp_gen_mem.vp and mmp_submem.vp files to sdram/common sub-directory under MMP installation
June 2017	3.2	Add note for MR1 supporting CL-3 for 3DS models
August 2017	3.3	Update for new SM interface
September 2017	3.4	Update Introduction section with spec version and DIMM model description
October 2017	3.5	Change “bank” to “bg,ba” in Address Mapping section
January 2018	3.6	Modify header and footer
June 2018	3.7	Update to JESD79-4B (June 2017). Update Manual Configuring description in Debugging section.
July 2018	3.8	Update for new utility library