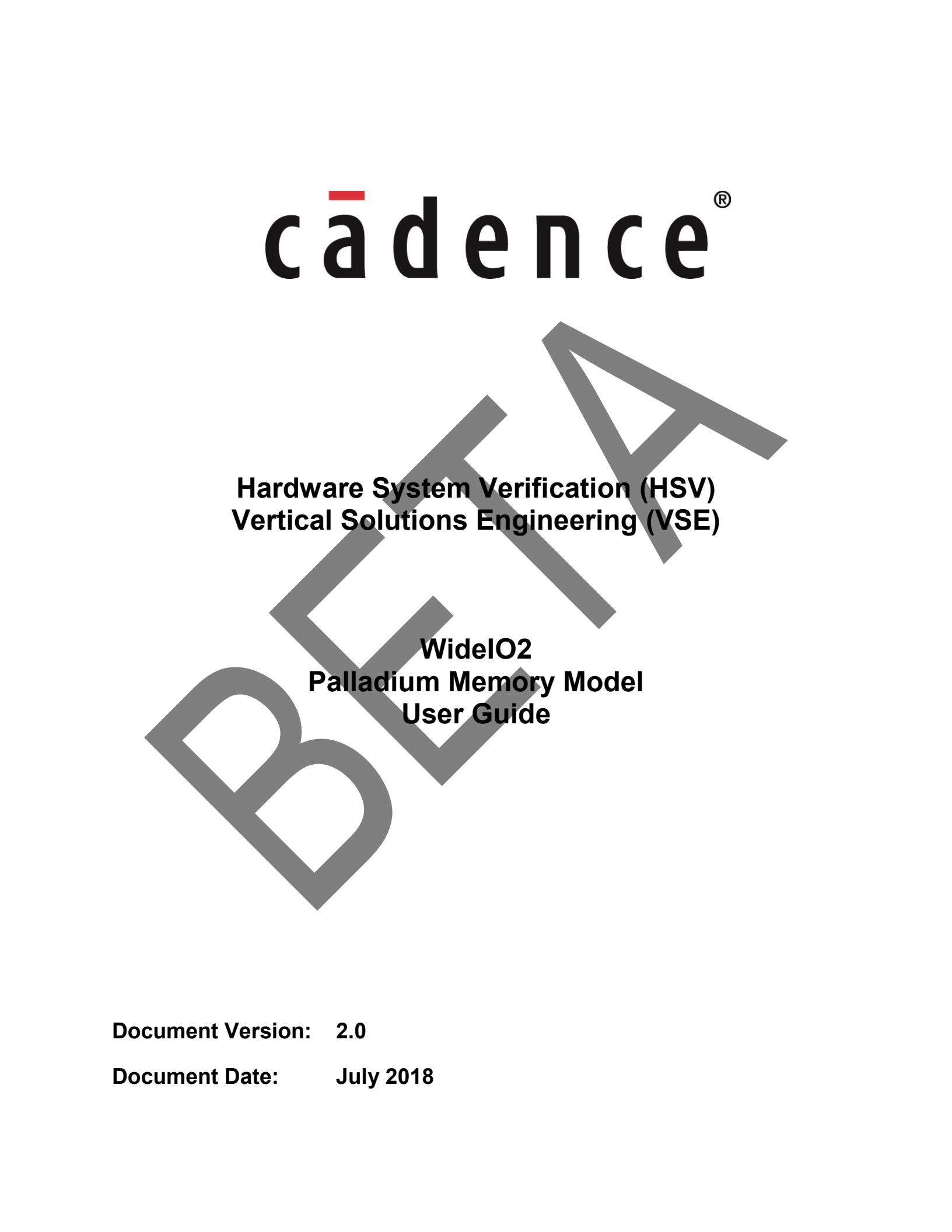


Hardware System Verification (HSV)
Vertical Solutions Engineering (VSE)

WideIO2
Palladium Memory Model
User Guide

Document Version: 2.0

Document Date: July 2018

Contents

1. 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

2. WideIO2 Memory Model

1. Introduction

The Cadence Palladium WideIO2 Model is based on early versions of vendor specifications and JEDEC Standard JESD229-2.

Different sizes from 8Gb up to 32Gb are available, please consult the memory model catalog for the current available list.

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 and may not have documentation. Model may contain defects.	No	Yes	Yes

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

3. Features

- Support WideIO2 majority of commands and functions
- Support 4x64die and 8x64 die topologies
- Support Mask Write and DBIac functions
- Support Burst Length 4, 8 and BL on the fly function.
- Support Pre-amble and Post-amble features
- Support Multiplexed Command Address
- Support TRR function

The following features are NOT supported in the Palladium WideIO2 model.

- SRE and SRX commands are accepted for WideIO2 model, but have no affect on the core memory
- PPR function is not supported
- PASR Bank and PASR Segment functions are not supported
- Scan chain function is not supported
- GPIO Mode test function is not supported

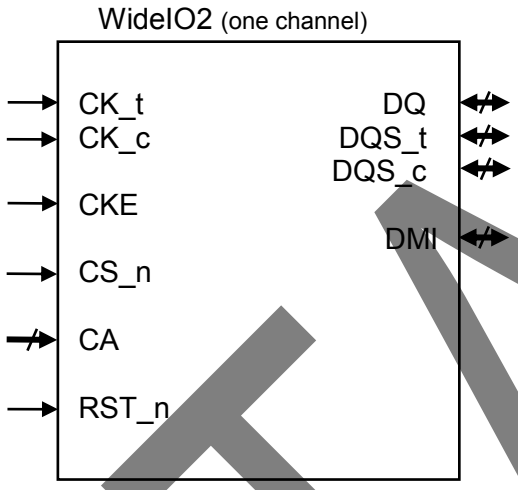
4. Configurations

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.

Topology	Parameter	Density				
4x64Die		8Gb	12Gb	16Gb	24Gb	32Gb
	Row Address	RA[12:0]	RA[13:0]	RA[13:0]	RA[14:0]	RA[14:0]
	Column Address	CA[8:0]	CA[8:0]	CA[8:0]	CA[8:0]	CA[8:0]
	Banks Address	BA[2:0]	BA[2:0]	BA[2:0]	BA[2:0]	BA[2:0]
8x64Die		8Gb	12Gb	16Gb	24Gb	32Gb
	Row Address	RA[13:0]	RA[14:0]	RA[14:0]	RA[15:0]	RA[15:0]
	Column Address	CA[7:0]	CA[7:0]	CA[7:0]	CA[7:0]	CA[7:0]
	Banks Address	BA[1:0]	BA[1:0]	BA[1:0]	BA[1:0]	BA[1:0]

5. Model Block Diagram

The width of CA, DQ, DQS and DMI buses are dependent on the density of the part being used.



6. I/O Signal Description

NAME	TYPE	DESCRIPTION
CK_0A_t CK_0A_c CK_1A_t CK_1A_c CK_0B_t CK_0B_c CK_1B_t CK_1B_c CK_0C_t CK_0C_c CK_1C_t CK_1C_c CK_0D_t CK_0D_c CK_1D_t CK_1D_c	Input	Clock: CK_t and CK_c are complementary clock inputs to each channel. All command signals are sampled on the positive edges of CK_t and CK_c providing a double rate command bus.
CKE_A[3:0] CKE_B[3:0] CKE_C[3:0] CKE_D[3:0]	Input	Clock Enable: CKE HIGH activates and CKE LOW deactivates the internal clock circuits, input buffers, and output drivers. Power-saving modes are entered and exited via CKE transitions. CKE is part of the command code.
CS_A_n[3:0] CS_B_n[3:0] CS_C_n[3:0] CS_D_n[3:0]	Input	Chip Select: CS_n is part of the command code. Each CS_n addresses a single rank on each of the channels.

CA_0A[10:0] CA_1A[10:0] CA_0B[10:0] CA_1B[10:0] CA_0C[10:0] CA_1C[10:0] CA_0D[10:0] CA_1D[10:0]	Input	Command/Address Inputs: There are 11 command/address signals per channel and a command packet is 2UI in length giving a payload of 22 bits of information. The CA bus will be clocked by CK_t/CK_c clocks.
DQ_0A[63:0] DQ_1A[63:0] DQ_0B[63:0] DQ_1B[63:0] DQ_0C[63:0] DQ_1C[63:0] DQ_0D[63:0] DQ_1D[63:0]	I/O	Data Inputs/Outputs: Bi-directional data bus. 64 DQs per channel.
DQS_0A_t[3:0] DQS_0A_c[3:0] DQS_1A_t[3:0] DQS_1A_c[3:0] DQS_0B_t[3:0] DQS_0B_c[3:0] DQS_1B_t[3:0] DQS_1B_c[3:0] DQS_0C_t[3:0] DQS_0C_c[3:0] DQS_1C_t[3:0] DQS_1C_c[3:0] DQS_0D_t[3:0] DQS_0D_c[3:0] DQS_1D_t[3:0] DQS_1D_c[3:0]	I/O	Data Strobe: The data strobe is bidirectional (used for read and write data) and complimentary (DQS_t and DQS_c). It is output with read data and input with write data. DQS_t and DQS_c are edge-aligned to read data and centered with write data. Each DQS pair strobes 16 DQ I/Os.
RST_n[3:0]	Input	RESET: Unidirectional reset inputs. These are per slice reset signals.
DMI_0A[7:0] DMI_1A[7:0] DMI_0B[7:0] DMI_1B[7:0] DMI_0C[7:0] DMI_1C[7:0] DMI_0D[7:0] DMI_1D[7:0]	I/O	Data Mask and Data Bus Inversion: DMI is a bidirectional signal sampled on the rising edges of DQS_t and DQS_c.

The table below provides the CKE and CS signal mapping for different configurations.

Configuration	4x64	8x64
1 High P2P	CKE[0], CS[0] for Ch. 0	CKE[0], CS[0] for Ch. 0 CKE[1], CS[1] for Ch. 1
2 High P2P	CKE[0], CS[0] for Ch. 0 CKE[1], CS[1] for Ch. 1	NA
2 High P22P	NA	CKE[0], CS[0] for Ch. 0 of Rank 0 CKE[1], CS[1] for Ch. 1 of Rank 0 CKE[2], CS[2] for Ch. 0 of Rank 1 CKE[3], CS[3] for Ch. 1 of Rank 1
4 High P22P	CKE[0], CS[0] for Ch. 0 of Rank 0 CKE[1], CS[1] for Ch. 1 of Rank 0 CKE[2], CS[2] for Ch. 0 of Rank 1 CKE[3], CS[3] for Ch. 1 of Rank 1	NA

7. Address mapping

The array of the WideIO2 model is mapped into the internal memory of the Palladium system. This array is a single two dimensional array. The mapping of bank, row and column addresses to the internal model array is as follows:

ARRAY_ADDR = (BA*rows + ROW)*cols + COL;

Where ‘rows’ is the maximum number of rows per bank and ‘cols’ is the maximum number of columns per row.

This information is required if the memory needs to be preloaded with user data.
The address format can be configured to {ROW, BA, COL} by setting the parameter “addr_format” to ‘1’.

The array name in the model hierarchy is: memcore

8. Register Definitions

In the WideIO2 specification there are up to 16 registers defined. The WideIO2 Palladium model implements the following registers.

MR#	MA <7:0>	Function	Access	OP7	OP6	OP5	OP4	OP3	OP2	OP1	OP0
0	00_H	RFU	RFU	RFU							
1	01_H	Device Feature 1	W	nWR			RFU	Programmable Postamble	BL		
2	02_H	Device Feature 2	W	SRA	PPRE	WL			RL & nRTP		
3	03_H	I/O Config-1	W	DBI WE	DBI RE	Thermal offset		DME	DS		
4	04_H	Refresh Rate	R	TUF	RFU				Refresh Rate		
5	05_H	Basic Config-1	R	Manufacturer ID							
6	06_H	Basic Config-2	R	Revision ID-1							
7	07_H	Basic Config-3	R	Revision ID-2							
8	08_H	Basic Config-4	R	RFU	I/O width / Ch		Density / die		Channel / die		
9	09_H	Test Mode	W	Test Mode							
10	$0A_H$	RFU	RFU	RFU							
11	$0B_H$	PASR-1	W	Bank Masking							
12	$0C_H$	PASR-2	W	Segment Masking							
13	$0D_H$	PPR Resources	R	Post Package Repair Resources							
14	$0E_H$	TTR-1	W	TRR Mode	TRR Mode - Bank			RFU			
15	$0F_H$	TTR-2	R	RFU				Mac Cap.	Mac Value		

8.1. MR0 Device Info Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFU							

8.2. MR1 Device Feature 1 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
nWR			RFU	Programmable Postamble	BL		

Bit 7	Bit 6	Bit 5	nWR
0	1	0	6
0	1	1	7
1	0	0	8 (default)
1	0	1	10
1	1	0	11
All others			Reserved

Bit 3	Post-amble Length
0	Standard Postamble

1	Extended Postamble

Bit2	Bit 1	Bit 0	Burst Length
0	1	0	BL4 (default)
0	1	1	BL8
1	1	1	BL4 or BL8 (on the fly enabled)
All others			Reserved

8.3. MR2 Device Feature 2 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SRA	PPRE	WL			RL & nRTP		

Frequency Ranges for RL, WL, and nWR, nRTP settings

Read Latency		Write Latency	nWR	nRTP
No DBI	w/ DBI			
5	6	3	6	2
6	7	4	7	3
7	8	5	8	3
8	9	6	10	4
9	10	7	11	4

SRA and PPRE are not supported in WideIO2 model.

8.4. MR3 IO Config Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBI WE	DBI RE	Thermal Offset		DME	DS		

Only the DBI-WE, DBI-RE and DME bits are supported in WideIO2 model.

8.5. MR4 SDRAM Refresh Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TUF	RFU				Refresh Rate		

Always returns a value of 0.

8.6. MR5 Basic Config 1 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Manufacturer ID							

See JEDEC-TBD WIO2 Manufacturer ID encodings. The model currently return 0.

8.7. MR6 Basic Config 2 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Revision ID 1							

Always returns a value of 0.

8.8. MR7 Basic Config 3 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Revision ID 2							

Always returns a value of 0.

8.9. MR8 Basic Config 4 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFU	I/O Width		Die Density			Channel	

Bit 1	Bit 0	Type
0	1	4 channel
1	0	8 channel
All Others		Reserved

Bit 4	Bit 3	Bit 2	Density
0	0	1	8Gb
0	1	0	12Gb
0	1	1	16Gb
1	0	0	24Gb
1	0	1	32Gb
All others			Reserved

Bit 6	Bit 5	IO Width
0	0	X 64
All others		Reserved

8.10. MR9 Test Mode Register

This register is implemented in the model but its content has no effect on the function of the model.

8.11. MR10 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFU							

8.12.MR11 PASR-1 Register

This register is implemented in the model but its content has no effect on the function of the model.

8.13.MR12 PASR-2 Register

This register is implemented in the model but its content has no effect on the function of the model.

8.14.MR13 PPR Resources Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PPR Resources							

Always returns a value of 0.

8.15.MR14 TRR-1 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TRR Mode	TRR Mode - Bank			RFU			

8.16.MR15 TRR-2 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFU				Mac Cap.	Mac Value		

Always returns a value of 0.

9. Commands

The WideIO2 model accepts the following commands:

- Deselect
- NOP
- Activate
- Precharge (Per Bank, All Bank)
- Refresh (Per Bank, All Bank)
- Read
- Write
- Mask Write
- Mode Register Write
- Mode Register Read

The following table shows the command encoding for WideIO2.

Function	Symbol	CKE		CS_n	Rising Edge of Clock	DDR CA Row Pins (11)										
		CK_t(n-1)	CK_t(n)			CA[0]	CA[1]	CA[2]	CA[3]	CA[4]	CA[5]	CA[6]	CA[7]	CA[8]	CA[9]	CA[10]
Deselect	DES	H	H	H	CK_t	V	V	V	V	V	V	V	V	V	V	V
				V	CK_c	V	V	V	V	V	V	V	V	V	V	V
NOP	NOP	H	H	L	CK_t	H	H	H	V	V	V	V	V	V	V	V
				V	CK_c	V	V	V	V	V	V	V	V	V	V	V
Activate	ACT	H	H	L	CK_t	L	H	R11	R12	R13	R14	R15	BA0	BA1	BA2	V
				V	CK_c	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Precharge(per Bank, all)	PRE	H	H	L	CK_t	H	H	L	H	AB	V	V	BA0	BA1	BA2	V
				V	CK_c	V	V	V	V	V	V	V	V	V	V	V
Refresh(per Bank, all)	REFA	H	H	L	CK_t	L	L	H	V	AB	V	V	BA0	BA1	BA2	V
				V	CK_c	V	V	V	V	V	V	V	V	V	V	V
Power Down Entry	PDE	H	L	H	CK_t	V	V	V	V	V	V	V	V	V	V	V
				V	CK_c	V	V	V	V	V	V	V	V	V	V	V
Self Refresh Entry	SRE	H	L	L	CK_t	L	L	H	V	V	V	V	V	V	V	V
				V	CK_c	V	V	V	V	V	V	V	V	V	V	V
Power Down & Self Refresh Exit	PDX/SRX	L	H	H	CK_t	X	X	X	X	X	X	X	X	X	X	X
				V	CK_c	X	X	X	X	X	X	X	X	X	X	X
Read	RD	H	H	L	CK_t	H	L	H	V	BL	C1	C2	BA0	BA1	BA2	V
				V	CK_c	AP	C3	C4	C5	C6	C7	C8	V	V	V	V
Write	WR	H	H	L	CK_t	H	L	L	L	BL	C1	C2	BA0	BA1	BA2	V
				V	CK_c	AP	C3	C4	C5	C6	C7	C8	V	V	V	V
Masked Write	MWR	H	H	L	CK_t	H	L	L	H	L	C1	C2	BA0	BA1	BA2	V
				V	CK_c	AP	C3	C4	C5	C6	C7	C8	V	V	V	V
Mode Register Write	MRW	H	H	L	CK_t	L	L	L	L	MA0	MA1	MA2	MA3	V	V	V
				V	CK_c	V	V	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7	V
Mode Register Read	RD	H	H	L	CK_t	L	L	L	H	MA0	MA1	MA2	MA3	V	V	V
				V	CK_c	V	V	V	V	V	V	V	V	V	V	V

10. DBI and DM function

The WideIO2 Model supports the Data Mask (DM) function for the Write operation and the Data Bus Inversion (DBIac) function for Write and Read operations. WideIO2 supports DM and DBIac function with a granularity at byte level. MR3 OP7 and OP6 define DBI-WR and DBI-RD enable bits, respectively, and MR3 OP3 defines DM enable bit. There are eight possible combinations in the WideIO2 model for DM and DBIac functions. The Mask Write command only supports BL4.

The below table describes the functional behavior for all combinations.

DM function	Write DBIdc Function	Read DBIdc Function	DMI Signal during Write Command	DMI Signal during Masked Write Command	DMI Signal During Read Command
Disable	Disable	Disable	Note:1	Note:1,3	Note:2
Disable	Enable	Disable	Note:4	Note:3	Note:2
Disable	Disable	Enable	Note:1	Note:3	Note:5
Disable	Enable	Enable	Note:4	Note:3	Note:5
Enable	Disable	Disable	Note:6	Note:7	Note:2
Enable	Enable	Disable	Note:4	Note:8	Note:2
Enable	Disable	Enable	Note:6	Note:7	Note:5
Enable	Enable	Enable	Note:4	Note:8	Note:5

Notes:
1) DMI input signal is a don't care. DMI input receivers are turned OFF.
2) DMI output drivers are turned OFF.
3) Masked Write Command is not allowed and is considered an illegal command as DM function is disabled.
4) DMI signal is treated as DBI signal and it indicates whether DRAM needs to invert the Write data received on DQs within a byte. The WIO2 device inverts Write data received on the DQ inputs in case DMI was sampled HIGH, or leaves the Write data non-inverted in case DMI was sampled LOW.
5) The WIO2 device inverts Read data on its DQ outputs associated within a byte and drives DMI signal HIGH when the number of ' 1' data bits within a given byte lane is greater than four; otherwise the DRAM does not invert the read data and drives DMI signal LOW.
6) The WIO2 device does not perform any mask operation when it receives Write command. During the Write burst associated with Write command, DMI signal is a don't care and ignored by DRAM.
7) The WIO2 device requires an explicit Masked Write command for all masked write operations. DMI signal is treated as DM signal and it indicates which bit time within the burst is to be masked. When DMI signal is HIGH, the WIO2 device masks that bit time across all DQs associated within a byte. All DQ input signals within a byte are don't care (either HIGH or LOW) when DMI signal is HIGH. When DMI signal is LOW, the WIO2 device does not perform mask operation and data received on DQ input is written to the array.
8) The WIO2 device requires an explicit Masked Write command for all masked write operations. The WIO2 device masks the Write data received on the DQ inputs in case DMI is sampled High, DQ[3:0] transition, and DQ[7:4] do not transition(previous DQ[7:0] XOR DQ[7:0] = 0x0F). Otherwise the WIO2 device inverts write data received on the DQ inputs in case DMI is sampled HIGH, or leaves the Write data non-inverted in case DMI is sampled LOW.

The WIO2 device resets DBI value to Low (all eight DQs and DMI signals within a byte group) whenever any of the following events occur:

a. RST_n signal de-assertion
b. The WIO2 device registers Mode Register Set command (MRS)
c. The WIO2 device registers Masked Write command after Read command
d. The WIO2 device registers back to back Masked Write commands with a gap (> tCCDmin)
e. The WIO2 device registers back to back Write followed by Masked Write command with a gap(> tCCDmin)
f. The WIO2 device registers back to back Read commands with a gap (>tCCDmin)

11. Initialization Sequence

The WideIO2 model requires that the memory controller follows the initialization sequence as documented in the specification. The sequence basically entails the following steps:

1. RST_n asserted.
2. RST_n de-asserted and then CKE goes high.
3. At least one DES or NOP command after CKE goes active.
4. (Optional) Issue MRR or MRW to load registers with all application settings
5. Initialization done

The model requires that these steps be performed in the correct sequence in order to complete initialization. The model will not respond to any other commands unless this sequence is completed correctly.

12. Compile and Emulation

The model is provided as a protected RTL file(s) (*.vp). The file(s) need to be synthesized prior to the back-end Palladium compile. An example of the command for compilation (including synthesis) and run of this model in the IXCOM flow is shown below.

```
ixcom -64bit +sv -ua +dut+jedec_wio2_8Gb_1h_8ch_8x64die \
    ./wideio2_pd.vp \
    ./jedec_wio2_8Gb_1h_8ch_8x64die.v \
    -incdir ../../../utils/cdn_mmp_utils/sv \
    ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
    ……

xeDebug -64 --ncsim \
    -sv_lib ../../../utils/cdn_mmp_utils/lib/64bit/libMMP_utils.so -- \
    -input auto_xedebug.tcl
```

The script below shows two example for Palladium classic ICE synthesis:

```
1)
hdlInputFile wideio2_pd.vp
hdlInputFile jedec_wio2_8Gb_1h_8ch_8x64die.v
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f verilog jedec_wio2_8Gb_1h_8ch_8x64die.vg
hdlSynthesize -memory -keepVhdlCase -keepRtlSymbol -keepAllFlipFlop
jedec_wio2_8Gb_1h_8ch_8x64die
……

2)
vavlog wideio2_pd.vp jedec_wio2_8Gb_1h_8ch_8x64die.v

vaelab -keepRtlSymbol -keepAllFlipFlop -outputVlog
jedec_wio2_8Gb_1h_8ch_8x64die.vg jedec_wio2_8Gb_1h_8ch_8x64die
……
```

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

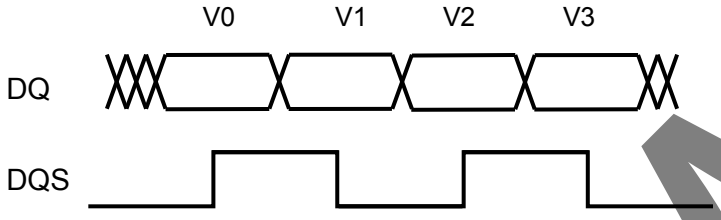
modified a bit when ICE ATB mode (–atb) is used since then a constant input FF is only optimized out when there is no initial value for it or the initial value is the same as the constant input value.

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.

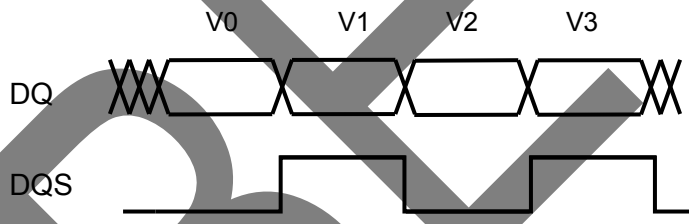
If the recommended compile script includes the aforementioned options, the user must include them to avoid affecting functionality of the design.

13. 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.

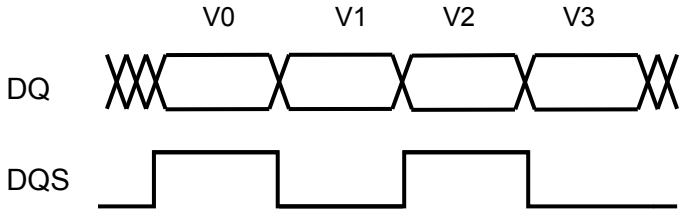


For DDR models provided by Cadence for Palladium, if the design drives DQ and DQS signals with the above timing, the DDR memory will behave correctly. However, to obtain this timing in Palladium, the fastest design clock must toggle twice as frequently as the DQS signal. If this faster clock is not needed 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 to 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
```

3. Revision History

The following table shows the revision history for this document

Date	Version	Revision
November 2013	1.0	Initial release
July 2014	1.1	Repaired doc property title. Updated legal.
September 2014	1.2	Remove version from UG file name. Update UXE / IXE documentation reference titles. Added paragraph about flow independent delay cell in “Handling DQS …” section.
November 2014	1.3	Remove emulation capacity info. Update related publications list.
June 2015	1.4	Adding WIO2 JEDEC spec support
July 2015	1.5	Update Cadence naming on front page
September 2015	1.6	Adding Compile section
January 2016	1.7	Update for Palladium-Z1 and VXE
July 2016	1.8	Remove hyphen in Palladium naming
January 2018	1.9	Modify header and footer
July 2018	2.0	Update for new utility library