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

Wide I/O
Palladium Memory Model
User Guide

Document Version: 2.0

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

Wide I/O Memory Model

1. Introduction

The Cadence Palladium Wide I/O Model is based on the JESD229 specification (December 2011).

Different sizes from 1Gb 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. Configurations

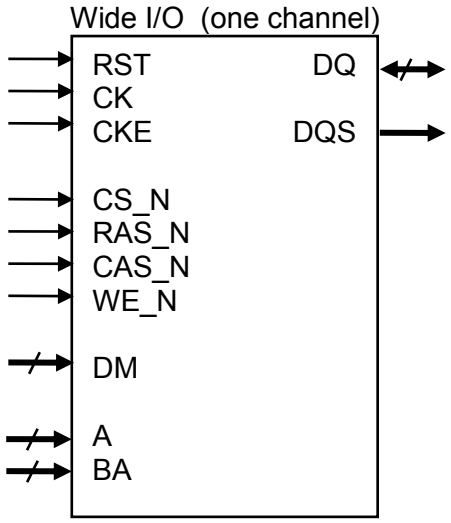
The following table lists the possible configurations. Not all configurations are available from all vendors. Please consult the appropriate vendor site for details on the parts they offer.

Device Density	Density / Channel	Address		
		BA	Row	Column
1Gb	256Mb	BA0 – BA1	RA0 – RA11	CA0 – CA6
2Gb	512Mb	BA0 – BA1	RA0 – RA12	CA0 – CA6
4Gb	1Gb	BA0 – BA1	RA0 – RA13	CA0 – CA6
8Gb	2Gb	BA0 – BA1	RA0 – RA14	CA0 – CA6
16Gb	4Gb	TBD	TBD	TBD
32Gb	8Gb	TBD	TBD	TBD

The auto pre-charge bit is A10

4. Model Block Diagram

The widths of the A, BA, DM, DQ, and DQS buses are dependent on the density of the part being used.

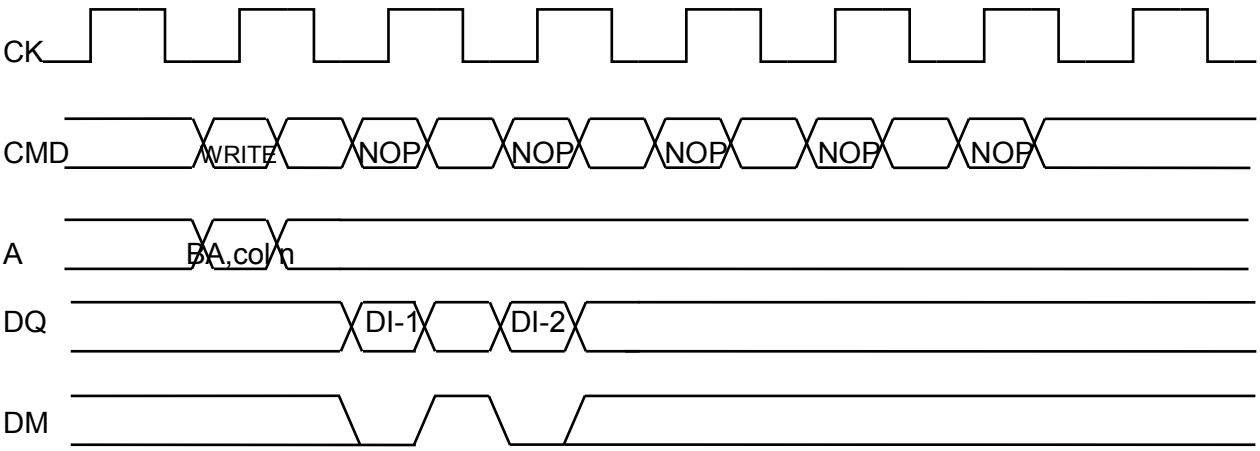


NAME	DESCRIPTION
CK_t[a:d]	Clock
CKE[0:3] [a:d]	Clock Enable
CS[0:3]_n[a:d]	Chip Select
RAS_n[a:d]	Command Inputs
CAS_n[a:d]	Command Inputs
WE_n[a:d]	Command Inputs
A[0:16] [a:d]	Address Inputs
BA[0:1] [a:d]	Bank Inputs
RST[0:3]_n	Reset Inputs
DQ[0:127] [a:d]	Data Inputs/Outputs
DQS[0:7]_t[a:d]	Data Strobe
DM[0:15] [a:d]	Input Data Mask

5. Waveforms for Read/Write

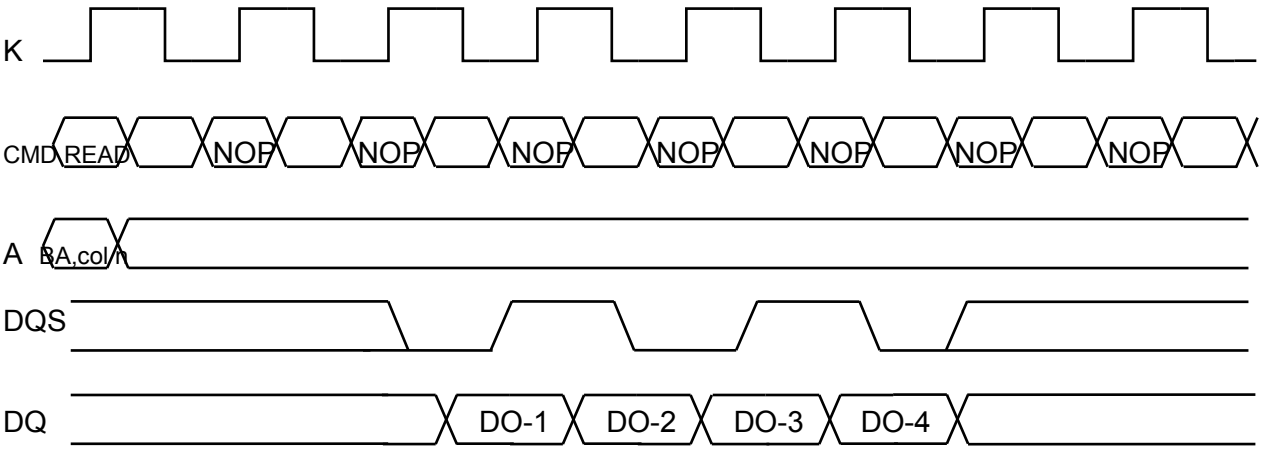
5.1. Write

Here is an example of timing sequence for write operation, setting WL = 1, Burst Length = 2.



5.2. Read

Here is an example of timing sequence for write operation, setting RL = 2, Burst Length = 4.



6. Address mapping

The array of the Wide I/O 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, ROW, COL}

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

7. Register Definitions

In the Wide I/O there are four registers, the Mode Register #0, Mode Register #1, Mode Register #2 and Mode Register #3. The Palladium Wide I/O model implements all four Mode Registers.

7.1. Mode Register #0

The mode register #0 is implemented in the Wide I/O model. The model supports the following parameter values.

BA1	**BA0**	**A15-A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
0	0	0	Write Recovery			TM	Read Latency			0	Burst Length		

A16	BA1	BA0	
0	0	0	MR0
0	0	1	MR1
0	1	0	MR2
0	1	1	MR3

A10	A9	A8	Write Latency
0	0	1	WL=1
0	1	0	WL=2(Option)
Others			Reserved

A7	Mode
0	Normal
1	Test

A6	A5	A4	Read Latency
0	0	0	Reserved
0	0	1	Reserved
0	1	0	RL=2
0	1	1	RL=3
1	0	0	RL=4(Option)
Others			Reserved

A2	A1	A0	Burst Length
0	0	0	Reserved
0	0	1	2
0	1	0	4
Others			Reserved

7.2. Mode Register #1

The mode register #1 is implemented in the Wide I/O model. The model does not support any contents but is required to be present as it is part of the initialization sequence.

7.3. Mode Register #2

The mode register #2 is implemented in the Wide I/O model. The model supports the following parameter values.

BA1	BA0	Ai – A10	A9	A8	A7	A6	A5	A4 - A0
1	0	0	nWR		TM	Drive Strength		0

A16	BA1	BA0	
0	0	0	MR0
0	0	1	MR1
0	1	0	MR2
0	1	1	MR3

A9	A8	nWR
0	0	Reserved(2)
0	1	nWR=3
1	0	Reserved(4)
1	1	Reserved

A7	Mode
0	Normal
1	Test

A6	A5	Drive strength
0	0	Full
0	1	Weak
1	0	Reserved
1	1	Reserved

7.4. Mode Register #3

The mode register #3 is implemented in the Wide I/O model. The model supports the following parameter values.

A16	BA1	BA0	A15 – A0
0	1	1	PASR

With Partial Array Self Refresh (PASR), the self refresh may be restricted to a variable portion of the total array. The PASR detail scheme will be defined. Data outside the defined area will be lost. Address bits A0 to A15 are used to set PASR.

8. Commands

The Wide I/O model accepts the following commands:

- Deselect
- NOP
- Mode Register Set
- Active
- Precharge
- Precharge All
- Read
- Read with Auto Precharge
- Write
- Write with Auto Precharge
- Burst Terminate
- Auto Refresh

9. Initialization Sequence

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

1. Assert RST
2. De-assert RST
3. Start clocks
4. Wait for CKE to asserted
5. Write to Mode Register 0
6. Write to Mode Register 1
7. Write to Mode Register 2
8. Write to Mode Register 3

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.

10. 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_wideio_1gb_128bit_4ch_1rank \
    ./jedec_wideio_1gb_128bit_4ch_1rank.vp \
    -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 jedec_wideio_1gb_128bit_4ch_1rank.vp
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f verilog jedec_wideio_1gb_128bit_4ch_1rank.vg
hdlSynthesize -memory -keepVhdlCase -keepRtlSymbol -keepAllFlipFlop
jedec_wideio_1gb_128bit_4ch_1rank
……

2)
vavlog jedec_wideio_1gb_128bit_4ch_1rank.vp

vaelab -keepRtlSymbol -keepAllFlipFlop -outputVlog
jedec_wideio_1gb_128bit_4ch_1rank.vg jedec_wideio_1gb_128bit_4ch_1rank
……
```

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-

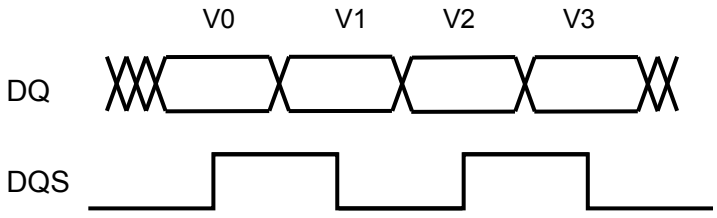
ICE can remove flops with constant inputs and merge equivalent FF. The picture above is modified a bit when ICE ATB mode (–atb) is used since then a constant input FF is only optimized out when there is no initial value for it or the initial value is the same as the constant input value.

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.

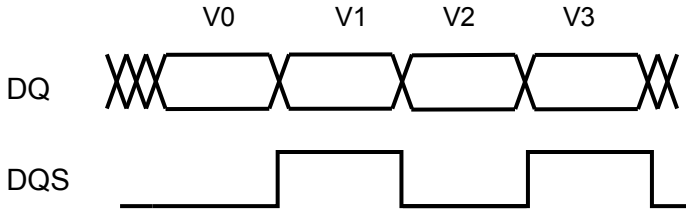
If the recommended compile script includes the aforementioned options, the user must include them to avoid affecting functionality of the design.

11. 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.

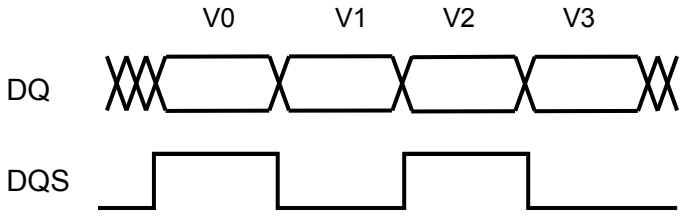


For DDR models provided by Cadence for Palladium, if the design drives DQ and DQS signals with the above timing, the DDR memory will behave correctly. However, to obtain this timing in Palladium, the fastest design clock must toggle twice as frequently as the DQS signal. If this faster clock is not needed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

Revision History

The following table shows the revision history for this document

Date	Version	Revision
October 2012	1.0	Initial version
July 2014	1.1	Repaired doc title property. Added revision history. Updated legal. Removed watermark.
September 2014	1.2	Remove version from UG file name. Update UXE / IXE documentation reference titles. Updated “Handling DQS …” section.
November 2014	1.3	Remove emulation capacity info. Update related publications list.
July 2015	1.4	Update Cadence naming on front page
September 2015	1.5	Adding Compile section
January 2016	1.6	Update for Palladium-Z1 and VXE
July 2016	1.7	Remove hyphen in Palladium naming
September 2017	1.8	Update reference specification
January 2018	1.9	Modify header and footer
July 2018	2.0	Update for new utility library