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

# DDR2
# Palladium Memory Model
# User Guide

**Document Version:** 2.4

**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

# DDR2 Memory Model

## 1. Introduction

The Cadence Palladium DDR2 Model is based on the JEDEC specification JESD79-2F.

The model is available in several configurations with model sizes to match real devices manufactured by the following vendors: Micron, Hynix and Samsung.

Different sizes from 256Mb up to 2Gb 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, 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

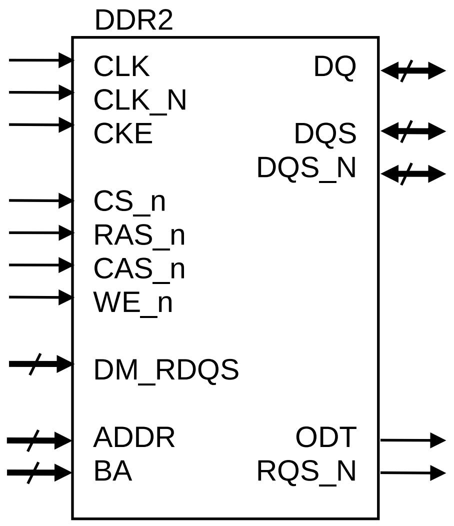
The following table lists the possible configurations. Not all configurations are available from all vendors. Please consult the appropriate vendor site for details on the parts they offer.

| Data Width | | 256Mb | 512Mb | 1Gb | 2Gb |
|---|---|---|---|---|---|
| | Banks | 4 | 4 | 8 | 8 |
| X4 | Row Address | A[12:0] | A[13:0] | A[13:0] | A[14:0] |
| | Column Address | A[11,9:0] | A[11,9:0] | A[11,9:0] | A[11,9:0] |
| X8 | Row Address | A[12:0] | A[12:0] | A[13:0] | A[14:0] |
| | Column Address | A[9:0] | A[9:0] | A[9:0] | A[9:0] |
| X16 | Row Address | A[12:0] | A[12:0] | A[12:0] | A[13:0] |
| | Column Address | A[8:0] | A[9:0] | A[9:0] | A[9:0] |

The auto pre-charge bit is A10

## 4. Model Block Diagram

The widths of the Addr, Ba, Dm, Dq, and Dqs buses are dependent on the density of the part being used.



## 5. Address mapping

The array of the DDR2 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.

The array name in the model hierarchy is: memcore

If {row,ba,col} addressing is needed instead of the default {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.

## 6. Register Definitions

In the DDR2 there are four registers, the Mode Register and three Extended Mode Register. The Palladium DDR2 model only implements the Mode Register and the first Extended Mode Register.

### 6.1. Mode Register

The mode register is implemented in the DDR2 model. The model supports the following parameter values.

| 15-13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | PD | Write Recovery | | | DLL Reset | Mode | CAS Latency | | | Burst Type | Burst Length | | |

| Bit 12 | Power Down | |
|---|---|---|
| 0 | Fast Exit | Not Supported |
| 1 | Slow Exit | Not supported |

| Bit 11 | Bit 10 | Bit 9 | Write Recovery |
|---|---|---|---|
| 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 6 | Bit 5 | Bit 4 | CAS Latency |
|---|---|---|---|
| 0 | 0 | 0 | Not supported |
| 0 | 0 | 1 | Not supported |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

| Bit 3 | Burst Type | |
|---|---|---|
| 0 | Sequential | Supported |
| 1 | Interleaved | Supported |

| Bit 2 | Bit 1 | Bit 0 | Burst Length |
|---|---|---|---|
| 0 | 0 | 0 | Not supported |
| 0 | 0 | 1 | Not supported |

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | Not supported |
| 1 | 0 | 1 | Not supported |
| 1 | 1 | 0 | Not supported |
| 1 | 1 | 1 | Not supported |

## 6.2. Extended Mode Register #1

The extended mode register #1 is implemented in the DDR2 model. The model supports the following parameter values.

| **15-13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | QOFF | RDQS Enable | DQS_N Enable | OCD Cal | | | RTT2 | Additive Latency | | | RTT1 | DIC | DLL Enable |

QOFF, OCD Cal, RTT2, RTT1, DIC and DLL Enable are not applicable to the DDR2 Palladium Model.

| **Bit 11** | **RDQS Enable** | |
|---|---|---|
| 0 | No | Supported |
| 1 | Yes | Supported |

| **Bit 10** | **DQS_N Enable** | |
|---|---|---|
| 0 | No | Supported |
| 1 | Yes | Supported |

| **Bit 5** | **Bit 4** | **Bit 3** | **Additive Latency** |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | Not supported |

## 6.3. Extended Mode Register #2 and #3

The extended mode registers #2 and #3 are not supported.

## 7. Commands

The DDR2 model accepts the following commands:

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

## 8. Compilation 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 compilation. An example of the command for compilation (including synthesis) and run of this model in the IXCOM flow is shown below.

```
ixcom -64 +sv -ua +dut+<model_name> \
    ./<model_name>.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 examples for Palladium classic ICE synthesis:

```
1)
hdlInputFile <model_name>.vp
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f verilog <model_name>.vg
hdlSynthesize -memory -keepVhdlCase -keepRtlSymbol -keepAllFlipFlop
<model_name>
……

2)
vavlog <model_name>.vp

vaelab -keepRtlSymbol -keepAllFlipFlop -outputVlog <model_name>.vg
<model_name>
……
```

**NOTE:** It is common for Palladium flows to require –keepallFlipFlop since it removes optimizations which are in place by default.   For example, without –keepAllFlipFlop, HDL-

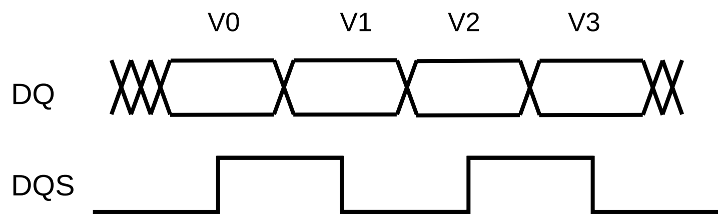
ICE can remove flops with constant inputs and merge equivalent FF. The picture above is modified a bit when ICE ATB mode ( –atb) is used since then a constant input FF is only optimized out when there is no initial value for it or the initial value is the same as the constant input value.

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.

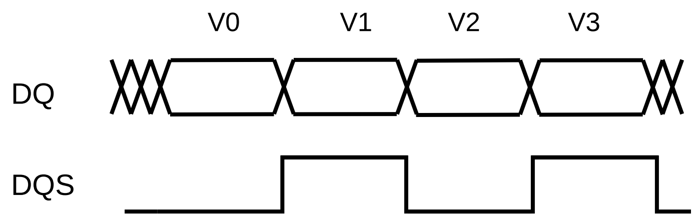
If the recommended compile script includes the aforementioned options, the user must include them to avoid affecting functionality of the design.

## 9. 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.

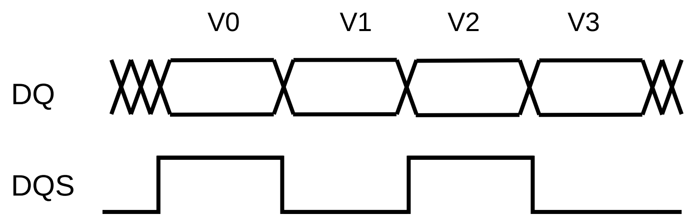


For DDR models provided by Cadence for Palladium, if the design drives DQ and DQS signals with the above timing, the DDR memory will behave correctly. However, to obtain this timing in Palladium, the fastest design clock must toggle twice as frequently as the DQS signal. If this faster clock is not needed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

## 10. Configuration Parameters

There are a few parameters that can be set to adjust the configuration of a model. One limitation is the sum of bank_addr_width, row_addr_width, and col_addr_width should be less than 32 bits due to the 2G address limit in HDL-ICE. The following parameters are available in the protected rtl version (<model_name>.vp file) of a model:

```
parameter    data_bits = 16;  // width of DQ bus
parameter    addr_bits = 13;  // row address width
parameter    bank_addr_width = 3;  // bank address width
parameter    row_addr_width = addr_bits;  // row address 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
```

## 11. 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.

### 11.1. 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.

## 12. 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 sequence:

| | |
|---|---|
| Cmd | Current commands |
| WL | Write latency |
| RL | Read latency |
| Mwe | Window of the written data |
| do_read | Preamble followed by read data |

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

  (1) MRS (cmd=0) sequences changing WL and RL (write and read latency)

  (2) Write (cmd=4) and Read (cmd=5) sequences with different WL and RL:

- **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>.mode_reg | MR | W |
| <model_name>.xtnd_mode_reg | EMR(1) | W |

## 13. Revision History

The following table shows the revision history for this document

| Date | Version | Revision |
|---|---|---|
| March 2010 | 1.0 | Initial Release |
| June 2011 | 1.1 | Added release levels description |
| March 2012 | 1.2 | Updated EMR description |
| June 2014 | 1.3 | Added a section on Configuration Parameters. Added +define+MMP_RBC to select {row,bank,col} addressing. |
| July 2014 | 1.4 | Repaired doc property title. Updated legal. |
| September 2014 | 1.5 | 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 | 1.6 | Remove emulation capacity info. |
| January 2015 | 1.7 | Added MMP and ECC section. Update related publications list. |
| July 2015 | 1.8 | Update Cadence naming on front page |
| January 2016 | 1.9 | Update for Palladium-Z1 and VXE |
| July 2016 | 1.7 | Remove hyphen in Palladium naming |
| December 2016 | 2.0 | Add "Manual Configuring of DDRx/LPDDRx" section |
| October 2017 | 2.1 | Modify "bank" vs "ba" references in Address Mapping section. |
| January 2018 | 2.2 | Modify header and footer |
| June 2018 | 2.3 | Update Manual Configuring description in Debugging section |
| July 2018 | 2.4 | Update for new utility library |