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

# LPDDR3
# Palladium Memory Model
# User Guide

**Document Version:** 2.9

**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. LPDDR3 Memory Model

## 1. Introduction

The Cadence Palladium LPDDR3 Model is based on the JEDEC specification for LPDDR3 (JESD209-3B).

The model only supports the S8 Type.

The model is initially only available in a few configurations based on the generic specification from the JEDEC spec. Currently, only a few different sizes 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. Features

The LPDDR3 Palladium memory model is based on the JEDEC specification for LPDDR3 (JESD209-3B). Below table lists which features are supported and which are unsupported. Please refer to the *JESD209-3B* for detailed information.

**Table 1: Features List of LPDDR3 Model**

| FEATURE | SUPPORT | NOTE |
|---|---|---|
| Initialization sequence | Yes | |
| Activate command | Yes | |
| Burst Read operation | Yes | |
| Burst Write operation | Yes | |
| Write Data Mask | Yes | |
| Precharge/auto-precharge operation | Yes | |
| Refresh/self-refresh command | Yes | |
| Partial Array Self-Refresh (PASR) | No | |
| Mode Register Read command | Yes | |
| Mode Register Write command | Yes | |
| DQ Calibration | Yes | |
| Write Leveling | Yes | |
| CA Training | Yes | |
| ZQ Calibration | Partial | Only support value of 0xFF, Calibration command after initialization |

# 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.

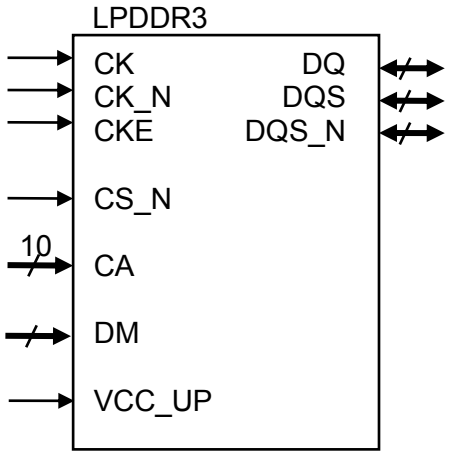
**Table 2: Model Configurations**

| | | 2Gb | 4Gb | 8Gb | 16Gb | 32Gb |
|---|---|---|---|---|---|---|
| Data Width | Type | S8 | S8 | S8 | S8 | S8 |
| | Banks | 8 | 8 | 8 | 8 | 8 |
| x16 | Row Address | - | A[13:0] | A[14:0] | A[14:0] | TBD |
| | Column Address | - | A[10:0] | A[10:0] | A[11:0] | TBD |
| x32 | Row Address | A[13:0] | A[13:0] | A[14:0] | A[14:0] | TBD |
| | Column Address | A[8:0] | A[9:0] | A[9:0] | A[10:0] | TBD |

## 5. Model Block Diagram

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

Additional VCC_UP pin is used to model the power on condition. If modeling the power on condition is desired then initially drive this pin low and then assert high after some time to indicate power on. If not needed then simply tie this pin high.




## 6. Model Parameter Descriptions

The following table provides details on the user adjustable parameters for the Palladium LPDDR3 Memory Model. These parameters may be modified when instantiating an LPDDR3 wrapper or, if necessary, by modifying the HDL parameter declarations and default values which are exposed for access and debug visibility.

**Table 3: User Adjustable Parameters**

| User Adjustable Parameter | Default Value | Description |
|---|---|---|
| data_bits | 32 | Data bus width |
| addr_bits | 10 | CA bus width |
| bank_addr_width | 3 | Bank address width |
| row_addr_width | 14 | Row address width |
| col_addr_width | 10 | Column address width |
| tDQSCK | 1 | Timing parameter tDQSCK |

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.

**Table 4: Visible Non-User-Adjustable Localparam**

| Localparam | Default Value | Description |
|---|---|---|
| byte_width | 8 | Byte bits |
| mode_reg_width | 8 | Mode register address width |
| num_bytes | 4 | Data width in bytes |
| total_addr_bits | bank_addr_width+row_addr_width +col_addr_width | Memory capacity address width |
| banks | 1<<bank_addr_width | Number of banks |
| mem_depth | 64'b1 << total_addr_bits | memory core depth |

Note that there are additional exposed localparams in the model hdl that are not described here nor intended to be described here. These additional localparams are exposed for debugging purposes only and will not be described herein.

## 7. Address mapping

The array of the LPDDR3 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.

# 8. Register Definitions

In the LPDDR3 specification there are up to 256 registers defined. The LPDDR3 Palladium model implements the following registers.

**Table 5: Registers Implemented in Model**

| Register Number | Register Name | Read/Write |
|---|---|---|
| 0 | Device Info | R |
| 1 | Device Feature 1 | W |
| 2 | Device Feature 2 | W |
| 3 | IO Config 1 | W |
| 4 | SDRAM Refresh | R |
| 5 | Basic Config 1 | R |
| 6 | Basic Config 2 | R |
| 7 | Basic Config 3 | R |
| 8 | Basic Config 4 | R |
| 10 | IO Calibration | W |
| 16 | PASR Bank | W |
| 17 | PASR Seg | W |
| 32 | DQ Calibration Pattern A | R |
| 40 | DQ Calibration Pattern B | R |
| 41 | CA_Training_1 | W |
| 42 | CA_Training_2 | W |
| 48 | CA_Training_3 | W |
| 63 | Reset | W |

## 8.1. MR0 Device Info Register

The device info register is implemented in the LPDDR3 model. The model supports the following bits.

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| RL3 | N/A | | | | | | DAI |

## 8.2. MR1 Device Feature 1 Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| nWR | | | N/A | | BL | | |

Note: nWRE is Bit 4 of MR2.

| Bit 7 | Bit 6 | Bit 5 | nWRE | nWR |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 3 |
| 1 | 0 | 0 | 0 | 6 |

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 8 |
| 1 | 1 | 1 | 0 | 9 |
| 0 | 0 | 0 | 1 | 10 |
| 0 | 0 | 1 | 1 | 11 |
| 0 | 1 | 0 | 1 | 12 |
| All Others | | | | Not Supported |

| Bit 2 | Bit 1 | Bit 0 | Burst Length |
|---|---|---|---|
| 0 | 0 | 0 | Not supported |
| 0 | 0 | 1 | Not supported |
| 0 | 1 | 0 | Not supported |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | Not supported |
| 1 | 0 | 1 | Not supported |
| 1 | 1 | 0 | Not supported |
| 1 | 1 | 1 | Not supported |

## 8.3. MR2 Device Feature 2 Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Write Leveling | WL Select | N/A | nWRE | RL & WL | | | |

| Bit 6 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | RL/WL |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | RL=3/WL=1 |
| 0 | 0 | 1 | 0 | 0 | RL=6/WL=3 |
| 0 | 0 | 1 | 1 | 0 | RL=8/WL=4 |
| 0 | 0 | 1 | 1 | 1 | RL=9/WL=5 |
| 0 | 1 | 0 | 0 | 0 | RL=10/WL=6 |
| 0 | 1 | 0 | 0 | 1 | RL=11/WL=6 |
| 0 | 1 | 0 | 1 | 0 | RL=12/WL=6 |
| 0 | 1 | 1 | 0 | 0 | RL=14/WL=8 |
| 0 | 1 | 1 | 1 | 0 | RL=16/WL=8 |
| 1 | 0 | 0 | 0 | 1 | RL=3/WL=1 |
| 1 | 0 | 1 | 0 | 0 | RL=6/WL=3 |
| 1 | 0 | 1 | 1 | 0 | RL=8/WL=4 |
| 1 | 0 | 1 | 1 | 1 | RL=9/WL=5 |
| 1 | 1 | 0 | 0 | 0 | RL=10/WL=8 |
| 1 | 1 | 0 | 0 | 1 | RL=11/WL=9 |
| 1 | 1 | 0 | 1 | 0 | RL=12/WL=9 |
| 1 | 1 | 1 | 0 | 1 | RL=14/WL=11 |
| 1 | 1 | 1 | 1 | 0 | RL=16/WL=13 |
| | All others | | | | Not supported |

## 8.4. MR3 IO Config Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| N/A | | | | DS | | | |

Writing to this register has no functional affect on the LPDDR3 model.

## 8.5. MR4 SDRAM Refresh Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| SDRAM Refresh Rate | | | | | | | |

The function controlled by this register is not supported in LPDDR3 model. Read this register 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 specification for list of manufacturer codes. The model currently supports Samsung (0x01) and Micron (0xFF).

## 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 |
|---|---|---|---|---|---|---|---|
| IO Width | | Density | | | | Type | |

| Bit 1 | Bit 0 | Type |
|---|---|---|
| 0 | 0 | Reserved |
| 0 | 1 | Reserved |
| 1 | 0 | Reserved |

| | | |
|---|---|---|
| 1 | 1 | S8 SDRAM |

| Bit 5 | Bit 4 | Bit 3 | Bit 2 | Density |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 4Gb |
| 0 | 1 | 1 | 1 | 8Gb |
| 1 | 0 | 0 | 0 | 16Gb |
| 1 | 0 | 0 | 1 | 32Gb |
| All others | | | | Not supported |

| Bit 7 | Bit 6 | IO Width |
|---|---|---|
| 0 | 0 | X32 |
| 0 | 1 | X16 |
| 1 | 0 | Not Used |
| 1 | 1 | Not Used |

## 8.10.MR10 Calibration Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| **Calibration Code** | | | | | | | |

The initialization sequence requires a write of 0xFF to this register. All other values have no effect on the model.

## 8.11.MR16/MR 17 PASR Bank and Seg Registers

These register are implemented in the model but their contents have no effect on the function of the model.

## 8.12.MR32 DQ Calibration Pattern A

This value of MR32 is 8’b10101010.

## 8.13.MR40 DQ Calibration Pattern B

This value of MR40 is 8’b00110011.

## 8.14.MR41 CA Training_1 Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| **CA Training Enable** | | | | | | | |

A write to this register causes entry to the CA training mode.

## 8.15.MR42 CA Training_2 Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| **CA Training Exit** | | | | | | | |

A write to this register causes exit of the CA training mode.

### 8.16. MR48 CA Training_3 Register

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CA Training Enable | | | | | | | |

A write to this register causes entry to the CA training mode.

### 8.17. MR63 Reset Register

This register is implemented in the model where the MRW RESET command (MR63) is an integral part of the initialization sequence. This initialization function is the only function MR63 has in this model. The register content is *don't care*.

## 9. Commands

The LPDDR3 model accepts the following commands:

- Deselect
- NOP
- Mode Register Write
- Mode Register Read
- Activate
- Precharge
- Precharge All
- Read
- Read with AP
- Write
- Write with AP
- Self Refresh

The following table is from the JEDEC Specification.

| SDRAM Command | SDR Command Pins | | | DDR CA pins (10) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CKE | | CS_N | CA0 | CA1 | CA2 | CA3 | CA4 | CA5 | CA6 | CA7 | CA8 | CA9 | CK_t EDGE |
| | CK_t(n-1) | CK_t(n) | | | | | | | | | | | | |
| MRW | H | H | L | L | L | L | L | MA0 | MA1 | MA2 | MA3 | MA4 | MA5 | |
| | | | X | MA6 | MA7 | OP0 | OP1 | OP2 | OP3 | OP4 | OP5 | OP6 | OP7 | |
| MRR | H | H | L | L | L | L | H | MA0 | MA1 | MA2 | MA3 | MA4 | MA5 | |
| | | | X | MA6 | MA7 | X | | | | | | | | |
| Refresh (per bank) | H | H | L | L | L | H | L | X | | | | | | |
| | | | X | X | | | | | | | | | | |
| Refresh (all bank) | H | H | L | L | L | H | H | X | | | | | | |
| | | | X | X | | | | | | | | | | |
| Enter Self Refresh | H | L | L | L | L | H | X | | | | | | | |
| | X | | X | X | | | | | | | | | | |
| Activate (bank) | H | H | L | L | H | R8 | R9 | R10 | R11 | R12 | BA0 | BA1 | BA2 | |
| | | | X | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R13 | R14 | |
| Write (bank) | H | H | L | H | L | L | RFU | RFU | C1 | C2 | BA0 | BA1 | BA2 | |
| | | | X | AP[3] | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | |
| Read (bank) | H | H | L | H | L | H | RFU | RFU | C1 | C2 | BA0 | BA1 | BA2 | |
| | | | X | AP[3] | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | |
| Precharge [11] (per bank, all bank) | H | H | L | H | H | L | H | AB | X | X | BA0 | BA1 | BA2 | |
| | | | X | X | X | X | X | X | X | X | X | X | X | |
| Enter Deep Power Down | H | L | L | H | H | L | X | | | | | | | |
| | X | | X | X | | | | | | | | | | |
| NOP | H | H | L | H | H | H | X | | | | | | | |
| | | | X | X | | | | | | | | | | |
| Maintain PD, SREF, DPD (NOP) see note 4 | L | L | L | H | H | H | X | | | | | | | |
| | | | X | X | | | | | | | | | | |
| NOP | H | H | H | X | | | | | | | | | | |
| | | | X | X | | | | | | | | | | |
| Maintain PD, SREF, DPD see note 4 | L | L | X | X | | | | | | | | | | |
| | | | X | X | | | | | | | | | | |
| Enter Power Down | H | L | H | X | | | | | | | | | | |
| | X | | X | X | | | | | | | | | | |
| Exit PD, SREF, DPD | L | H | H | X | | | | | | | | | | |
| | X | | X | X | | | | | | | | | | |

# 10. Initialization Sequence

The LPDDR3 model requires that the memory controller follows the initialization sequence as documented in the specification. The sequence basically entails the following steps:

1. Wait for CKE to asserted
2. Write to Mode Register 63 (MRW Reset Command)
3. Complete CA Training (Optional)
4. Write to Mode Register 10 (ZQ Calibration)

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.

# 11. 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 -64 +sv -ua +dut+jedec_lpddr3_16gb_32 \
    ./jedec_lpddr3_16gb_32.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_lpddr3_16gb_32.vp
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f verilog jedec_lpddr3_16gb_32.vg
hdlSynthesize -memory -keepVhdlCase -keepRtlSymbol -keepAllFlipFlop
jedec_lpddr3_16gb_32
……

2)
vavlog jedec_lpddr3_16gb_32.vp

vaelab -keepRtlSymbol -keepAllFlipFlop -outputVlog
jedec_lpddr3_16gb_32.vg jedec_lpddr3_16gb_32
……
```

**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.

## 12. SWI Smart Memory Interface

This LPDDR3 core model (lpddr3_pd.vp) 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 6: 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 7: 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 7: 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 8: 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 8: 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 9: 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 9: 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 |

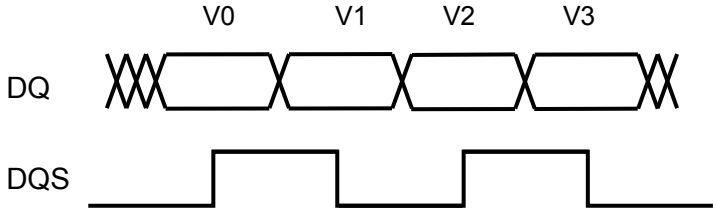
The SWI Smart Memory interface has dependencies on the inclusion of external file(s). For additional details about purpose and content of any Smart Memory related external files, please consult the SWI documentation and other Hybrid Solution documentation.

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

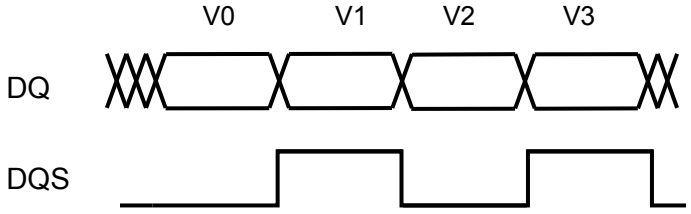
```
`endif
```

# 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.

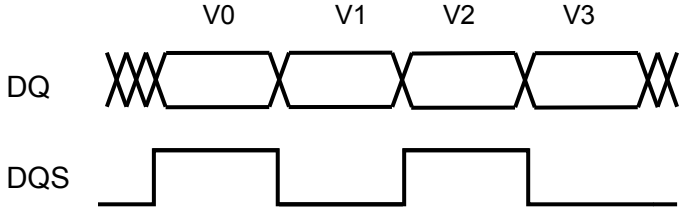



For DDR models provided by Cadence for Palladium, if the design drives DQ and DQS signals with the above timing, the DDR memory will behave correctly. However, to obtain this timing in Palladium, the fastest design clock must toggle twice as frequently as the DQS signal. If this faster clock is not needed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

## 14. Debugging

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

- For issues that are may not be model specific please review the *Memory Model Portfolio FAQ for All Models User Guide.*
- **Golden waveform:** A package with a reference waveform is available which shows the following command sequence(s):

(1) Initialization sequence:
CKE is low first --> at least 1 CKE high or NOP --> MRW MR63
[ --> MRW MR41 --> MRW MR48 --> MRW MR42 ] (CA training sequence is optional)
--> MRW MR10 --> init_done
(2) Additional command sequence includes:
MRR MR0
MRR MR5
MRR MR8
MRR MR32
MRR MR40
MRW MR1
MRW MR2
ACT
WRITE
READ
PRE
CA training sequence

- **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 10: 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>.MR1_reg | MR1 | W |
| <model_name>.MR1_set | | |
| <model_name>.MR2_reg | MR2 | W |
| <model_name>.MR2_set | | |
| <model_name>.MR3_reg | MR3 | W |
| <model_name>.MR3_set | | |
| <model_name>.MR10_reg | MR10 | W |
| <model_name>.MR10_set | | |
| <model_name>.MR16_reg | MR16 | W |
| <model_name>.MR16_set | | |
| <model_name>.MR17_reg | MR17 | W |
| <model_name>.MR17_set | | |
| <model_name>.MR41_reg | MR41 | W |
| <model_name>.MR41_set | | |
| <model_name>.MR42_reg | MR42 | W |
| <model_name>.MR42_set | | |
| <model_name>.MR48_reg | MR48 | W |
| <model_name>.MR48_set | | |
| <model_name>.init_done | [Not Applicable] | 1'b1 indicates initialization is complete |

Note: The user needs to force the MR#_set to be 1'b1 when forcing MR#_reg.

# 3. Revision History

The following table shows the revision history for this document

| Date | Version | Revision |
|---|---|---|
| Sep 2012 | 1.0 | Initial release |
| Oct 2012 | 1.1 | Added complete device table |
| April 2014 | 1.2 | Added a section on Configuration Parameters. |
| June 2014 | 1.3 | Added +define+MMP_RBC to select {row,bank,col} addressing. |
| July 2014 | 1.4 | Add info about MR63 in register definition section. Repair doc properties title. Added new legal. |
| September 2014 | 1.5 | Remove version from UG file name. |
| September 2014 | 1.6 | Adding Model parameter descriptions section. Added paragraph about flow independent delay cell in "Handling DQS …" section. |
| November 2014 | 1.7 | Remove emulation capacity info. |
| January 2015 | 1.8 | Adding support extra RL/WL based on JESD209-3B. Update related publications list. |
| March 2015 | 1.9 | Add section on SWI Smart Memory interface |
| June 2015 | 1.10 | Remove BST command from LPDDR3 model |
| July 2015 | 1.11 | Update Cadence naming on front page<br>Adding DQ Calibration support |
| September 2015 | 1.12 | Adding feature table and compile section |
| January 2016 | 2.0 | Update for Palladium-Z1 and VXE |
| April 2016 | 2.1 | Update SM interface sm_we |
| July 2016 | 2.2 | Remove hyphen in Palladium naming |
| December 2016 | 2.3 | Add "Manual Configuring of DDRx/LPDDRx" section |
| May 2017 | 2.4 | Update "Manual Configuring of DDRx/LPDDRx" section |
| August 2017 | 2.5 | Update for new SM interface |
| October 2017 | 2.6 | Modify "bank" vs "ba" references in Address Mapping section. |
| January 2018 | 2.7 | Modify header and footer |
| May 2018 | 2.8 | Add section for Manual configuration |
| July 2018 | 2.9 | Update for new utility library |