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

GDDR5M
Palladium Memory Model
User Guide

Document Version: **2.1**

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

GDDR5M Memory Model

1. Introduction

The Cadence Palladium GDDR5M x8 and x16 models are based on the JC-42.3C committee item number 1733.72A which complements the GDDR5 JESD212 JEDEC specification.

There are eight configurations with size ranging from 2Gb to 16Gb.

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.

Memory Size	2Gb	2Gb	4Gb	4Gb	8Gb	8Gb	16Gb	16Gb
Data Width	x8	x16	x8	x16	x8	x16	x8	x16
Banks Address	BA[2:0]	BA[2:0]	BA[2:0]	BA[2:0]	BA[3:0]	BA[3:0]	BA[3:0]	BA[3:0]
Row Address	A[14:0]	A[13:0]	A[15:0]	A[14:0]	A[14:0]	A[13:0]	A[15:0]	A[14:0]
Column Address	A[6:0]	A[6:0]	A[6:0]	A[6:0]	A[7:0]	A[7:0]	A[7:0]	A[7:0]
Auto precharge	A[8]	A[8]	A[8]	A[8]	A[8]	A[8]	A[8]	A[8]
Page Size	1K	2K	1K	2K	2K	4K	2K	4K

Note:

The column address notation does not include the lower three address bits as the burst order is always fixed for READ and WRITE commands. Burst length is 8.

4. Verilog Macro Defines

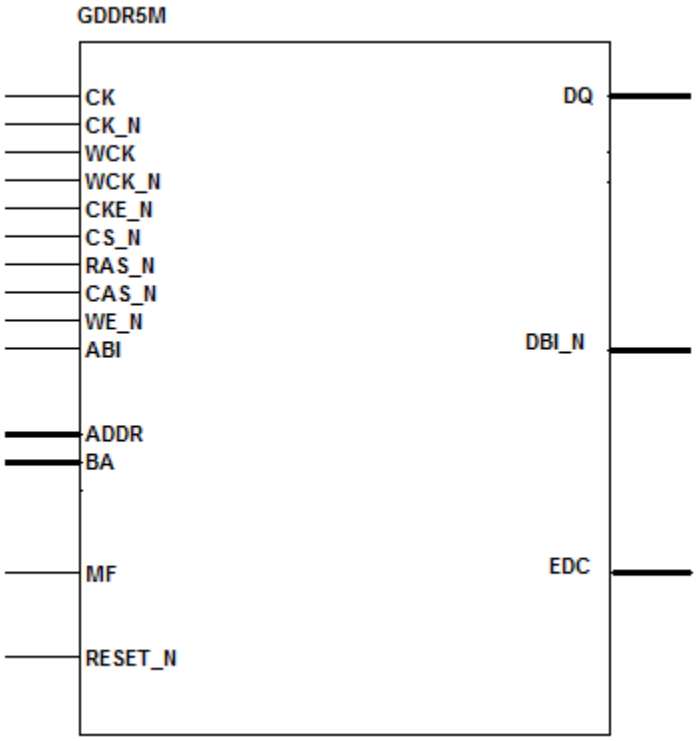
The following table lists the Verilog macro defines related to the MMP GDDR5M model.

As described in the section “Large Memory Support,“ each MMP GDDR5M model that exceeds 30 bits of address width needs to incorporate the multiple core memory array generator (mmp_gen_mem_2rp.vp) into the memory build. In these cases, the file mmp_gen_mem_2rp.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_2rp.vp or overridden on the command with the define option. Users of GDDR5M 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.

`define Macro Purpose	Optional Verilog `define	Default Value
Large memory support	MMP_LG_MEM_BITS	30 Range 25 .. 30

5. Model Block Diagram

The width of the ADDR, BA, DBI_N, EDC, and DQ buses are dependent on the density and data width of the part being used.



6. Address mapping

The array of the GDDR5M 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,3'b000}

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

The array name in the model hierarchy is: Array

7. Register Definitions

In the GDDR5M there are nine Mode Registers (MR0-7, MR15). The Palladium GDDR5M model implements all nine Mode Registers. However not all features are supported in the model.

7.1. Mode Register 0

The mode register is implemented in the GDDR5M model. The x8 model supports the following parameter values. RFU must be programmed to 0 during MRS command, for all mode registers.

BA[3:0]	A[11:8]	A[7]	A[6:3]	A[2:0]
MR Select	Write Recovery	Test Mode	CAS Latency	Write Latency

BA[3:0]	MR Select
0000	MR0
0001	MR1
0010	MR2
0011	MR3
0100	MR4
0101	MR5
0110	MR6
0111	MR7
1111	MR15

Bit 11	Bit 10	Bit 9	Bit 8	Write Recovery
X	X	X	X	Not supported

Bit 7	Test Mode	
0	Normal	Supported
1	Test Mode	Not supported

Bit 6	Bit 5	Bit 4	Bit 3	CAS Latency
0	0	0	0	5
0	0	0	1	6
0	0	1	0	7
0	0	1	1	8
0	1	0	0	9
0	1	0	1	10
0	1	1	0	11
0	1	1	1	12
1	0	0	0	13
1	0	0	1	14
1	0	1	0	15
1	0	1	1	16
1	1	0	0	17
1	1	0	1	18

1	1	1	0	19
1	1	1	1	20

Bit 2	Bit 1	Bit 0	Write Latency
0	0	0	Reserved
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	7

7.2. Mode Register 1

The mode register 1 is implemented in the GDDR5M model. The model supports the following parameter values.

BA[3:0]	A[11]	A[10]	A[9]	A[8]	A[7]	A[6]	A[5:4]	A[3:2]	A[1:0]
MR Select	RFU	ABI	WDBI	RDBI	RFU	Cal Upd	Addr / Cmd Termination	Data Termination	Drive Strength

Calibration Update, Addr / Cmd Termination, Data Termination, and Drive Strength are not applicable to the GDDR5M Palladium Model.

Bit 10	ABI
0	On
1	Off

Bit 9	WDBI
0	On
1	Off

Bit 8	RDBI
0	On
1	Off

7.3. Mode Register 2

The mode register 2 is implemented in the GDDR5M model. However the model does not support any of the parameter values.

BA[3:0]	A[11:9]	A[8:6]	A[5:3]	A[2:0]
MR Select	Address/ Command Termination Offset	Data and WCK Termination Offset	OCD Pullup Driver Offset	OCD Pulldown Driver Offset

7.4. Mode Register 3

The mode register 3 is implemented in the GDDR5M model. The model does not support any contents but is required to be present as it is part of the initialization sequence.

BA[3:0]	A[11:10]	A[9:8]	A[7:6]	A5	A4	A3	A2	A[1:0]
MR Select	Bank Groups	WCK Termination	DRAM Info	RDQS Mode	WCK2 CK	WCK23 INV	WCK01 INV	Self-Refresh

7.5. Mode Register 4

The mode register 4 is implemented in the GDDR5M model. The model supports the following parameter values, except EDC Inv and hold pattern.

BA[3:0]	A11	A10	A9	A[8:7]	A[6:4]	A[3:0]
MR Select	EDC Inv	WR CRC	RD CRC	CRC Read Latency	CRC Write Latency	EDC Hold Pattern

Bit 10	WR CRC
0	On
1	Off

Bit 9	RD CRC
0	On
1	Off

Bit 8	Bit 7	CRC Read Latency
0	0	Reserved
0	1	1
1	0	2
1	1	Reserved

Bit 6	Bit 5	Bit 4	CRC Write Latency
0	0	0	Reserved
0	0	1	8
0	1	0	9
0	1	1	10
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

7.6. Mode Register 5

The mode register 5 is implemented in the GDDR5M model. The model does not support any contents but is required to be present as it is part of the initialization sequence.

BA[3:0]	A[11:6]	A[5:3]	A2	A1	A0
MR Select	RAS	RFU	LP3	LP2	RFU

7.7. Mode Register 6

The mode register 6 is implemented in the GDDR5M model. The model does not support any contents but is required to be present as it is part of the initialization sequence.

BA[3:0]	A[11:8]	A[7:4]	A[3:1]	A[0]
MR Select	VREFD Offset Device ID 0 and 1	VREFD Offset Device ID 2 and 3	RFU	WCK PIN

7.8. Mode Register 7

The mode register 7 is implemented in the GDDR5M model. The model does not support any contents but is required to be present as it is part of the initialization sequence.

BA{3:0]	A[11:10]	A[9:8]	A[7]	A[6]	A[5]	A[4]	A[3]	A[2:0]
MR Select	DCC	VDD Range	Half VREFD	Temp Sense	DQ PreA	Auto Sync	LF Mode	RFU

7.9. Mode Register 15

The mode register 15 is implemented in the GDDR5M model. The model supports the following parameter values.

BA[3:0]	A[11:10]	A[9]	A[8]	A[7:0]
MR Select	Address Training	MRE MF1	MRE MF0	X

Bit 11	Bit 10	Address Training
X	0	Off
0	1	Train rising CK edge
1	1	Train rising /CK edge

Bit 9	MR Enable MF=1
0	Enable
1	Disable

Bit 8	MR Enable MF=0
0	Enable
1	Disable

8. Commands

The GDDR5M model accepts the following commands:

- Deselect
- NOP
- Mode Register Set
- Activate
- Read
- Read with AP
- Load Fifo
- Read Training
- Write
- Write with AP
- Write with Single Byte Mask
- Write with AP, Single Byte Mask
- Write with Double Byte Mask
- Write with AP, Double Byte Mask
- Write Training
- Precharge
- Precharge All
- Refresh

9. Initialization Sequence

The GDDR5M 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_N
2. De-assert RESET_N with DBI_N and MF set for the desired device ID
3. Start clocks CK and CK_N
4. Wait for CKE_N to asserted
5. Issue at least 2 NOP commands
6. Issue a Precharge All command followed by NOP commands
7. Issue MRS command to MR15 for address training mode (optional)
8. Start clocks WCK and WCK_N
9. Issue MRS commands to the Mode Registers in any order
10. Issue two Refresh commands followed by NOP commands

Generally there is no ordering required for step 9 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. The device ID chosen in step 2 determines the write mask mapping for x8 models.

10. Synthesis and Compilation of the Model

The model is provided as a protected RTL source file for the core memory along with a selection of wrapper files that correspond to various configurations. Please see section 3 Model Configurations. The gddr5x_pd.vp file need to be synthesized prior to compiling for Palladium, either with HDL-ICE in Classic ICE flow or with IXCOM flow.

An IXCOM compile example is provided below:

```
ixcom   -clean -ua \
        -timescale 1ps/1ps \
        +dut+gddr5m_jedec_<size> +sv \
        +libext+.v+.sv+.vp \
        -y ../rtl \
        +incdir+../rtl \
        -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 scripts below show two examples for Palladium classic ICE synthesis:
1)

```
 hdlInputFile -add gddr5m_jedec_<size>.vp
 hdlImport -full -2001 -l qtref
```

```
hdlOutputFile -add -f Verilog gddr5m_jedec_<size>.vg
hdlSynthesize -memory -keepRtlSymbol -keepAllFlipFlop
gddr5m_jedec_<size>
……
```

2)

```
vavlog      gddr5m_jedec_<size>.vp
vaelab      -keepRtlSymbol -keepAllFlipFlop -outputVlog
            gddr5m_jedec_<size>.vg gddr5m_jedec_<size>
```

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.

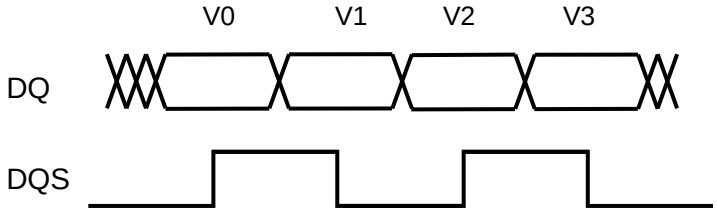
11. Limitations

The GDDR5M model does not support the following features:

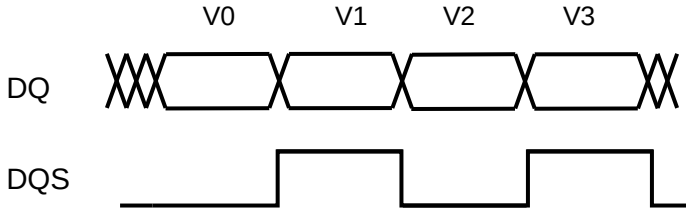
Test Mode
RDQS Mode
DRAM Info
WCK INV
EDC Inv and Hold Pattern
Low Power Modes

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.

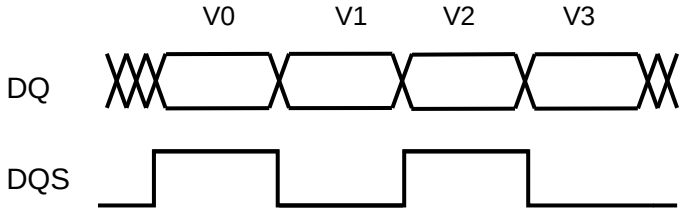


For DDR models provided by Cadence for Palladium, if the design drives DQ and DQS signals with the above timing, the DDR memory will behave correctly. However, to obtain this timing in Palladium, the fastest design clock must toggle twice as frequently as the DQS signal. If this faster clock is not needed 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. 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_2rp.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_2rp.vp automatically defines Verilog macro MMP_LG_MEM_BITS with a value of ‘30’ by default.

13.1 Preloading Multiple Arrays

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

tb_top.rtl_module.gddr5m_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.gddr5m_i0.mem1.\multiple.array_0_.u1 .memcore addresses 0 .. 1G - 1
tb_top.rtl_module.gddr5m_i0.mem1.\multiple.array_1_.u1 .memcore addresses 1G .. 2G - 1
tb_top.rtl_module.gddr5m_i0.mem1.\multiple.array_2_.u1 .memcore addresses 2G .. 3G - 1
tb_top.rtl_module.gddr5m_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.gddr5m_i0.mem1.multiple.array_0_.u1.memcore -file
mem0.dat

memory -load %readmemh
tb_top.rtl_module_w.rtl_module.gddr5m_i0.mem1.multiple.array_1_.u1.memcore -file
mem1.dat

memory -load %readmemh
tb_top.rtl_module_w.rtl_module.gddr5m_i0.mem1.multiple.array_2_.u1.memcore -file
mem2.dat

memory -load %readmemh
tb_top.rtl_module_w.rtl_module.gddr5m_i0.mem1.multiple.array_3_.u1.memcore -file
mem3.dat
```

13.2 Models with Multiple Arrays

Not every GDDR5M model exceeds the 30 bit IXCOM memory limit. Below is a list of models with address width exceeding 30 bits and the number of arrays generated in the core memory.

Model Name	BA	ROW	COL	Total bits	Arrays
gddr5m_jedec_16gb_8	4	16	11	31	2

13.3 IXCOM Compilation

For large memory models that exceed 30 bits of address width, include the file mmp_gen_mem_2rp.vp in the build as shown below. This file will incorporate the multiple core memory array generator into the build and the Verilog macro MMP_LG_MEM_BITS will automatically be defined.

```
vlan      -64bit +sv tb.v \
          <model_name>.vp \
          mmp_gen_mem_2rp.vp mmp_submem_2rp.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


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

13.4 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_2rp.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.

14. Revision History

The following table shows the revision history for this document

Date	Version	Revision
Oct 2012	1.0	Initial Release
July 2014	1.1	Repaired doc property title. Updated legal. Removed watermark.
September 2014	1.2	Remove version from UG file name. Updated “Handling DQS …” section.
November 2014	1.3	Remove emulation capacity info. Update related publications list.
July 2015	1.4	Document large memory support
July 2015	1.5	Update Cadence naming on front page
August 2015	1.6	Replaced gen_mem_2rp.vp with mmp_gen_mem_2rp.vp and added mmp_submem_2rp.vp
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
February 2018	2.0	Add synthesis and compile section
July 2018	2.1	Update for utility library