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

Memory Model Portfolio Debug Display User Guide

Document Version: 2.7

Document Date: June 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. Memory Model Debug Display Capability

1. Introduction

The Cadence Palladium Memory Model Portfolio has for a selected subset of the more complex and more popular of its models a debug methodology called MMP Debug Display that is based on the Verilog system task $display. Using the $display system task allows this feature to be used in software simulation on IXCOM, hardware acceleration on Palladium with IXCOM and also with Classic ICE mode.

Since there is a standard approach ensuring that all MMP models have the same look and feel for this capability, this standalone user guide provides details on the method and format of the debug display information.

The user is provided with both compile and runtime controls that allow them to enable or disable this feature. In addition the controls allow the user to enable or disable specific types of debug messages.

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. Debug Display Highlights

- The MMP debug display feature is supported in these Palladium modes:
    - Classic ICE
    - IXCOM software simulation
    - IXCOM hardware simulation

- For MMP debug display support models should synthesize the protected rtl model (*.vp) with needed $display options.
    - IXCOM flow + Display Debug requires the +gfifoDisp+[<dut_mod_name>[+...]] option.
    - ICE flow + Display Debug requires HDL-ICE to be used in ATB mode

- MMP debug display messaging uses Verilog system task $display

- MMP debug display compilation can be disabled using a Verilog macro define

- MMP debug display messaging issues an initial banner display with model family, name, and configuration info

- MMP debug display runtime control at the beginning of a run supports $test$plusargs system function

- MMP debug display runtime control supports force and release commands upon `mmp_debug_display_*` registers

- MMP debug display messaging for writeable configuration and mode registers upon initial value and upon change of value via either command or forcing

- MMP debug display supports enabling and disabling of messaging at the level of memory type during the compile phase

- MMP debug display supports enabling and disabling of messaging during runtime at level of memory type and of memory model instance

4. Supported Models

The models listed below have Debug Display support:

- DDR2
- DDR3
- DDR4
- LPDDR2
- LPDDR3
- LPDDR4, LPDDR4 x8, LPDDR4X
- LPDDR5, LPDDR5 x8
- MRAM
- UFS
- HMC 2.0
- HBM
- Hyperflash
- ONFI 3.0 / 3.2 / 4.0
- SPI NAND
- QSPI Verilog (Winbond and Micron parts)
- Most Cellular RAM / PSRAM
- HyperRam
- GDDR6
- Toshiba Toggle DDR 2.0 models
- Toshiba Toggle DDR 3.0 models
- eMMC 5.0
- OctaRAM

5. Enabling and Disabling $display in IXCOM During Compile

MMP debug display messaging is based on the Verilog system task $display.

The $display support for IXCOM is provided through use of the IXCOM command-line option +gfifoDisp+[<dut_mod_name>[+...]] to map existing $display task calls for specified DUT modules such as the MMP model. Note that without the specification of <dut_mod_name>, all $display-family task calls in all DUT modules would get transformed. An example command line option is shown below:

```
ixcom -64bit +sv +dut+jedec_lpddr4_8Gb \
-ua -incdir ../sim -timescale 1ps/1ps \
+gfifoDisp+jedec_lpddr4_8Gb \
<path>/jedec_lpddr4_8Gb.vp \
 . . .
```

Additionally, if the MMP model being compiled in, or enabled, for Debug Display includes multiple module files, the user should specify ALL of the model file names in the gfifoDisp option specification in order to pick up the $display messaging in all relevant modules including that messaging which lies within protected areas.

The +gfifoDisp option is the recommended IXCOM display support for MMP debug display. The option may be removed from compile when not debugging relevant memory models or one may use the display debug compile arguments described below to achieve finer resolution control over Debug Display logic.

5.1. Full Disabling of Debug Display Messaging During Compilation

For those MMP models which support the MMP Debug Display feature, the MMP Debug Display related logic is compiled into the database by default at compile time. One method for compile time disabling, or removal, of debug display messaging logic for MMP modules involves disabling all debug display messaging across all models. To perform this full disabling, the user can define the Verilog macro MMP_NO_DEBUG_DISPLAY in the IXCOM or ICE compile script. An example that employs the gfifoDisp option globally for general messaging but disables MMP debug display messaging is shown below:

```
ixcom   -64bit +sv +dut+hbm_top -ua \
        ../rtl/rtl_code/hbm_pd.v \
        ../rtl/rtl_code/hbm_mem.v \
        ../rtl/rtl_code/hbm_ieee1500_wrapper.v \
        ../rtl/rtl_code/hbm_channel.v \
        ../rtl/rtl_code/hbm_top.v \
                ../tb/tb_lane_repair.sv \
        +define+MMP_HBM_PSEUDO_CHANNEL \
        +define+MMP_HBM_ECC_SUPPORT \
        +define+MMP_NO_DEBUG_DISPLAY \
        +gfifoDisp
```

Disabling the Debug Display feature may save critical emulation resources or prevent any performance degradation due to output messaging.

5.2. Type Based Enabling and Disabling of Debug Display Messaging During Compilation

For added flexibility and finer control resolution the MMP Debug Display feature also supports compile time enabling and disabling of the feature based on model type.

To disable the MMP Debug Display feature logic from being included during the compile phase for one particular type of MMP model, the user can define the Verilog macro MMP_<MODEL_TYPE>_NO_DEBUG_DISPLAY at compile time.

An example directive to disable at compile time only the LPDDR4 model's Debug Display messaging is as follows:

```
+define+MMP_LPDDR4_NO_DEBUG_DISPLAY
```

The mappings from relevant MMP "MODEL_TYPE" to the Verilog macro required for compile time enable/disable is shown below.

Table 1: Compile Phase Verilog Macros for Type Based Enable/Disable of Debug Display

MMP Catalog Model Type	Compile Phase Verilog Macro
DDR2 DDR2 DIMM	MMP_DDR2_NO_DEBUG_DISPLAY
DDR3 DDR3 UDIMM DDR3 RDIMM	MMP_DDR3_ NO_DEBUG_DISPLAY
DDR4 DDR4 DIMM DDR4 LRDIMM	MMP_DDR4_ NO_DEBUG_DISPLAY
LPDDR2	MMP_LPDDR2_NO_DEBUG_DISPLAY
LPDDR3	MMP_LPDDR3_NO_DEBUG_DISPLAY
LPDDR4 LPDDR4X LPDDR4 x8	MMP_LPDDR4_NO_DEBUG_DISPLAY
LPDDR5	MMP_LPDDR5_NO_DEBUG_DISPLAY
GDDR6	MMP_GDDR6_ NO_DEBUG_DISPLAY
MRAM	MMP_MRAM_ NO_DEBUG_DISPLAY
UFS	MMP_UFS_ NO_DEBUG_DISPLAY
HMC	MMP_HMC_ NO_DEBUG_DISPLAY
HBM	MMP_HBM_ NO_DEBUG_DISPLAY
HyperFlash	MMP_HYPERFLASH_NO_DEBUG_DISPLAY
ONFI 3.0 NAND	MMP_ONFI3_ NO_DEBUG_DISPLAY
ONFI 3.2 NAND	MMP_ONFI3_ NO_DEBUG_DISPLAY
ONFI 4.0 NAND	MMP_ONFI4_ NO_DEBUG_DISPLAY

Toggle DDR 2.0	MMP_TDDR2_NO_DEBUG_DISPLAY
Toggle DDR 3.0	MMP_TDDR3_NO_DEBUG_DISPLAY
Quad SPI Flash Verilog Models	MMP_QSPI_ NO_DEBUG_DISPLAY
SPI NAND	MMP_SPINAND_NO_DEBUG_DISPLAY
Cellular RAM: ADMUX PSRAM	MMP_APSRAM_ NO_DEBUG_DISPLAY
Cellular RAM: OPIDDR PSRAM	MMP_OPSRAM_ NO_DEBUG_DISPLAY
Cellular RAM: PSRAM x8	MMP_PSRAMX8_NO_DEBUG_DISPLAY
Cellular RAM: OctaRAM	MMP_OCTARAM_NO_DEBUG_DISPLAY
Cellular RAM: OSPI PSRAM	MMP_OSPI_PSRAM_NO_DEBUG_DISPLAY
HyperRam	MMP_HYPERRAM_ NO_DEBUG_DISPLAY
eMMC5.0	MMP_EMMC50_ NO_DEBUG_DISPLAY

6. Enabling and Disabling $display in Classic ICE During Compile

MMP Debug Display is based on the Verilog system task $display. MMP debug display messaging is supported through synthesis of the protected RTL (*.vp) format of MMP model in Classic ICE flows. The $display support for Classic ICE is provided through the Acceleratable Testbench (ATB) mode of HDL-ICE synthesis. ATB mode is enabled with the –atb switch for the hdlImport command (`hdlImport –atb`) which then permits synthesis of modules with ATB supported constructs such as the $ display instances in some MMP models.

ATB generates a runtime script that contains runtime XEL commands which are to be called at XeDebug runtime after download and after the `configPM` command. The name of the runtime file is specified using either the `–outputQel <filename>` option with the vaelab command or using the `hdlOutputFile –f qel <filename>` command. For example:

```
hdlImport -2001 -atb -full -l qtref

hdlOutputFile -add -f qel runtime_init_atb.qel
```

or

```
vavlog -atb ../rtl/spi_nand_pd.v ../tb/top.v

vaelab -keepRtlSymbol -keepAllFlipFlop -outputQel
runtime_init_atb.qel -outputVlog top.vg top
```

To call the runtime script the user issues the hdlConfig run-time XEL command. This script will then initialize ATB related registers and memories. See the UXE User Guide for additional details about ATB mode.

It is mentioned above that the debug display runtime control leverages system function $test$plusargs.

There are several options for controlling the ATB features. Please see the UXE User Guide for more information.

6.1. Full Disabling of Debug Display Messaging During Compilation

For those MMP models which support the MMP Debug Display feature, the MMP Debug Display related logic is compiled into the database by default at compile time. One method for compile time disabling, or removal, of debug display messaging logic for MMP modules involves disabling all debug display messaging across all models. To perform this full disabling, the user can define the Verilog macro MMP_NO_DEBUG_DISPLAY in the ICE compile script. An example of how to define this macro in an ICE synthesis script that enables globally the ATB features while also disabling MMP debug display messaging is shown below:

```
vavlog \
  -atb -sv \
  ../rtl/rtl_code/hbm_pd.v \
  ../rtl/rtl_code/hbm_mem.v \
  ../rtl/rtl_code/hbm_ieee1500_wrapper.v \
  ../rtl/rtl_code/hbm_channel.v \
  ../rtl/rtl_code/hbm_top.v \
  ./tb_ice_atb.sv \
  +define+MMP_HBM_PSEUDO_CHANNEL \
  +define+MMP_HBM_ECC_SUPPORT \
  +define+MMP_NO_DEBUG_DISPLAY
```

Disabling the Debug Display feature may save critical emulation resources or prevent any performance degradation due to output messaging.

6.2. Type-Based Enabling and Disabling of Debug Display Messaging During Compilation

For added flexibility and finer control resolution the MMP Debug Display feature also supports compile time enabling and disabling of the feature based on model type.
See the IXCOM section above on Type Based Enabling and Disabling of Debug Display Messaging During Compilation.

7. Runtime Control of Debug Display Messaging

Various aspects of Debug Display messaging may be controlled during runtime.

7.1. Initial Banner: Messaging and Control

The initial banner display informs users of basic information about the MMP model in use. The initial banner is output ahead of all the other Debug Display messages during runtime.

An example of the initial banner for a LPDDR4 2-channel model is shown below:

```
Time: 0.0 ns MMP LPDDR4 Init Banner: ===================================
Time: 0.0 ns MMP LPDDR4 Init Banner:  Family:    LPDDR4
Time: 0.0 ns MMP LPDDR4 Init Banner:  Model:     lpddr4_2ch v3.0
Time: 0.0 ns MMP LPDDR4 Init Banner:  Size:      4 Gb
Time: 0.0 ns MMP LPDDR4 Init Banner: ===================================
```

The initial banner messaging portion of the Debug Display feature can be enabled and disabled by configuring the following parameter:

Parameter Name	Default Value	Description
init_banner_on	1’b1	0: Turn off Initial Banner Display 1: Turn on Initial Banner Display

For a model that includes multiple cores in its wrapper, only one of the multiple cores is enabled to display an initial banner. For multi-core models this banner displays information about the model rather than information about the core. As with single core models, to turn off the initial banner messaging of such a model, set the init_banner_on parameter of this core from 1’b1 to 1’b0.

7.2. $test$plusargs Controlled Messaging

The $test$plusargs support in IXCOM and HDL-ICE is configured to allow the user to control various debug display conditions during runtime without recompiling the design. A set of $test$plusargs functions in the MMP model control a corresponding set of Verilog conditions that enable or disable aspects of debug display messaging as described in the table below.

To change an argument at the beginning of an IXCOM run that invokes irun use the same syntax as $test$plusargs uses in IES simulation. An example is shown below:

```
irun -64bit -R -messages -tcl \
+MMP_DEBUG_DISPLAY +MMP_DEBUG_DISPLAY_STATE \
+MMP_DEBUG_DISPLAY_ERROR +MMP_DEBUG_DISPLAY_DATA \
-input ./ixcom_run_debuginfo.tcl
```

To change an argument at the beginning of an IXCOM run that invokes xeDebug use the “−−ncsim” option to pass to xeDebug the same syntax as $test$plusargs uses in IES simulation. Note the concluding “−−“. An example is shown below:

```
xeDebug --ncsim +MMP_DEBUG_DISPLAY +MMP_DEBUG_DISPLAY_STATE
+MMP_DEBUG_DISPLAY_ERROR +MMP_DEBUG_DISPLAY_DATA --
```

To change the argument value during a Classic ICE run add +<arg> to the hdlConfig command in the same manner as above:

```
debug .
host .
download
configPM -stb
hdlConfig runtime_init_atb.qel +MMP_DEBUG_DISPLAY
+MMP_DEBUG_DISPLAY_STATE +MMP_DEBUG_DISPLAY_ERROR
+MMP_DEBUG_DISPLAY_DATA
```

Arguments control runtime debug display messaging as described in the table below:

$test$plusargs Argument	**Debug Display Messaging Behavior**
MMP_DEBUG_DISPLAY	Enable the overall MMP debug display feature
MMP_DEBUG_DISPLAY_NO_TIME	If SET, removes the $time Verilog task from the messages. Time is displayed correctly in IXCOM when buffered display is not used. If buffered display is configured, the user may wish to disable the timestamp form the messaging. Time is displayed as cycle count in classic ICE.
MMP_DEBUG_DISPLAY_NO_INST	If SET, removes the format specification for the hierarchical path to the memory module instance. For cases where the memory instance name may be a long string that could cause formatting issues this argument can be set to obtain messaging without the instance path.
MMP_DEBUG_DISPLAY_STATE	Command decoding and state machine information such as 1) Initial sequence states And 2) Bank states transition(Precharge/Activate)
MMP_DEBUG_DISPLAY_ERROR	Error conditions. Error messaging may be invoked from various state checks for commands.
MMP_DEBUG_DISPLAY_DATA	Display information for memory accesses to the core memory array within the model. The message is displayed when the actual write or read occurs at the lowest level in the model.

To obtain MMP debug display messaging, note that at least two arguments must be set: the MMP_DEBUG_DISPLAY argument to enable the general feature and any additional arguments are desired. Applying the MMP_DEBUG_DISPLAY argument by itself will not result in any messaging.

For runtime control in the middle of an acceleration or emulation run—i.e. past the initialization and after some amount of run—the user can force and release as needed these mmp_debug_display registers with the resulting effect that the related messaging content will be enabled and disabled. This runtime control allows the user to optimize a run during periods of slower performance as may be seen with significant message output. For example, the user might consider forcing the mmp_debug_display registers to disable messaging initially and force the appropriate registers to an enabled state when the time of the problem occurs.

Example of debug messaging displayed when data is being read from the memcore array

```
Time: 238328.0 ns MMP LPDDR3 Inst: tb.inst0 Cmd: READ Array: memcore
Addr: 0x300d103 [Bnk: 0x6 Row: 0x0068 Col: 0x103] -- Data 0xe94e5d87
```

Example of a state machine transitioning to the Precharge state for Bank 0

```
Time: 238353.0 ns MMP LPDDR3 Inst: tb.inst0 FSM: Bank0 State: PRE
```

Example of an activate command being decoded

```
Time: 235908.0 ns MMP LPDDR3 Inst: tb.inst0 Cmd: ACT
```

7.3. Type Based $test$plusargs Debug Display Messaging

In addition to the fully on or fully off $test$plusargs discussed above, the Debug Display feature supports model type based and model instance based $test$plusargs.

MMP models with Debug Display support are separated into MODEL_TYPE according to the table shown below. For each model type the Debug Display messaging for that type can be controlled by specifying the desired set or subset of MODEL_TYPE $test$plusargs at runtime.

Table 2: Runtime $TEST$PLUSARGS Control Signals for Type Based Enable/Disable of Debug Display Messaging

MMP Catalog Model Type	**Runtime $test$plusargs Arguments**
	NOTE: '*' denotes explicit specification of the set or a subset of these arguments: +MMP_<MODEL TYPE>_DEBUG_DISPLAY +MMP_<MODEL TYPE>__DEBUG_DISPLAY_STATE +MMP_<MODEL TYPE>__DEBUG_DISPLAY_ERROR +MMP_<MODEL TYPE>__DEBUG_DISPLAY_DATA
DDR2 DDR2 DIMM	MMP_DDR2_DEBUG_DISPLAY*
DDR3 DDR3 UDIMM DDR3 RDIMM	MMP_DDR3_DEBUG_DISPLAY*
DDR4 DDR4 DIMM DDR4 LRDIMM	MMP_DDR4_DEBUG_DISPLAY*
LPDDR2	MMP_LPDDR2_DEBUG_DISPLAY*
LPDDR3	MMP_LPDDR3_DEBUG_DISPLAY*
LPDDR4 LPDDR4X LPDDR4 x8	MMP_LPDDR4_DEBUG_DISPLAY*
LPDDR5	MMP_LPDDR5_DEBUG_DISPLAY*
GDDR6	MMP_GDDR6_DEBUG_DISPLAY*
MRAM	MMP_MRAM_DEBUG_DISPLAY*
UFS	MMP_UFS_DEBUG_DISPLAY*
HMC	MMP_HMC_DEBUG_DISPLAY*
HBM	MMP_HBM_DEBUG_DISPLAY*
HyperFlash	MMP_HYPERFLASH_DEBUG_DISPLAY*
ONFI 3.0 NAND	MMP_ONFI3_DEBUG_DISPLAY*
ONFI 3.2 NAND	MMP_ONFI3_DEBUG_DISPLAY*
ONFI 4.0 NAND	MMP_ONFI4_DEBUG_DISPLAY*
Toggle DDR 2.0	MMP_TDDR2_DEBUG_DISPLAY*
Toggle DDR 3.0	MMP_TDDR3_DEBUG_DISPLAY*
Quad SPI Flash Verilog Models	MMP_QSPI_DEBUG_DISPLAY*
SPI NAND	MMP_SPINAND_DEBUG_DISPLAY*
Cellular RAM: ADMUX PSRAM	MMP_APSRAM_DEBUG_DISPLAY*
Cellular RAM: OPIDDR PSRAM	MMP_OPSRAM_DEBUG_DISPLAY*
Cellular RAM: PSRAM x8	MMP_PSRAMX8_DEBUG_DISPLAY*
Cellular RAM: OctaRAM	MMP_OCTARAM_DEBUG_DISPLAY*
Cellular RAM: OSPI PSRAM	MMP_OSPI_PSRAM_DEBUG_DISPLAY*

HyperRam	MMP_HYPERRAM_DEBUG_DISPLAY*
eMMC5.0	MMP_EMMC50_DEBUG_DISPLAY*

To enable all of the relevant debug display messaging for a specific MMP model type, add the arguments of MMP_< MODEL_TYPE>_DEBUG_DISPLAY* where the asterisk denotes specifying all of the below arguments:

```
+MMP_<MODEL TYPE>_DEBUG_DISPLAY
+MMP_<MODEL TYPE>__DEBUG_DISPLAY_STATE
+MMP_<MODEL TYPE>__DEBUG_DISPLAY_ERROR
+MMP_<MODEL TYPE>__DEBUG_DISPLAY_DATA
```

This specification will activate overall Debug Display for the type as well as Debug Display state messaging, Debug Display error messaging, and Debug Display data messaging for that model type.

For example, switching on debug display messaging for only the LPDDR4 model during runtime would specify the below arguments in place of the generic Display Debug $test$plusargs arguments for the various flow examples shown above.

```
+MMP_LPDDR4_DEBUG_DISPLAY
+MMP_LPDDR4_DEBUG_DISPLAY_STATE
+MMP_LPDDR4_DEBUG_DISPLAY_ERROR
+MMP_LPDDR4_DEBUG_DISPLAY_DATA
```

One can also specify a subset of this group of arguments to further constrain the messaging categories and content during runtime as long as the argument `+MMP_<MODEL TYPE>_DEBUG_DISPLAY` is among the specification set.

7.4. Instance Based $test$plusargs Debug Display Messaging

MMP models with Debug Display support can also be controlled at runtime at the instance level by specifying the desired set or subset of instance level $test$plusargs. Instance level $test$plusargs include the hierarchical path to the model of interest. The instance level specification permits users to control messaging for models that include multiple cores—such as a LPDDR4 with 2 channels or a DDR4 LRDIMM with 16 DDR4 models nested within—as well as individual model instances in cases where a single model part is instantiated many times.

To enable debug display messaging of a particular model instance, one needs to specify the set of $test$plusargs arguments
MMP_INST_DEBUG_DISPLAY*=<INSTANCE_PATH> where the asterisk denotes specifying all of the below arguments:

```
+MMP_INST_DEBUG_DISPLAY=<INSTANCE_PATH>
+MMP_INST_DEBUG_DISPLAY_STATE=<INSTANCE_PATH>
+MMP_INST_DEBUG_DISPLAY_ERROR=<INSTANCE_PATH>
+MMP_INST_DEBUG_DISPLAY_DATA=<INSTANCE_PATH>
```

The above specification will activate overall the Debug Display feature for the model instance as well as Debug Display state messaging, Debug Display error messaging, and Debug Display Data messaging for that model instance. As mentioned above, MMP debug display messaging requires that at least two arguments must be set:

Some examples of resulting debug display messages are as below:

```
Time: 5.0 ns MMP LPDDR4 Inst: tb.mmp0.lpddr4 MR0 initial value is 0x00
Time: 5.0 ns MMP LPDDR4 Inst: tb.mmp0.lpddr4 MR1 initial value is 0x00
Time: 5.0 ns MMP LPDDR4 Inst: tb.mmp0.lpddr4 MR2 initial value is 0x00
```

To enable debug display messaging of one memory core inside a particular multicore model, such as a two-channel LPDDR4 model or a DDR4 LRDIMM model, users need to specify the $test$plusargs argument using the hierarchical path to the memory part instance plus the core instance.

Example: arguments for switching on debug display messaging that is constrained to Channel A of a two channel LPDDR4:

```
+MMP_INST_DEBUG_DISPLAY=tb.mmp0.lpddr4_chA
```

A few of the resulting debug display messages are below:

```
Time: 5.0 ns MMP LPDDR4 Inst: tb.mmp0.lpddr4_chA MR0 initial value is 0x00
Time: 5.0 ns MMP LPDDR4 Inst: tb.mmp0.lpddr4_chA MR1 initial value is 0x00
Time: 5.0 ns MMP LPDDR4 Inst: tb.mmp0.lpddr4_chA MR2 initial value is 0x00
```

In this example of instance level specification channel A of the 2-channel LPDDR4 model will print relevant messaging and channel of the same model will not print any Debug Display messaging.

7.5. Messaging During Manual Configuration of MMP Models

Many MMP models that support Debug Display also support 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*.

The Debug Display feature offers additional support to this manual configuration scenario by helping the user to monitor the values of forcible registers through debug display messages. The initial values of these forcible registers will be displayed in messaging after the model's initial banner and ahead of any other debug display messages. Once these registers are changed, whether by protocol command or by forcing, the previous and changed values will also be messaged.

These registers include:

- The debug display runtime control registers `mmp_debug_display*` discussed above

- Any init_done register for the model which (if present) indicates the completion of an initial sequence
- Any writeable configuration registers, writeable mode registers, or writeable fields in mixed R/W configuration registers.

Example: messaging for a configuration register initial value for the LPDDR4 model

```
……
Time: 5.0 ns MMP LPDDR4 Debug Display : display = 1, state = 1, no_time =
0, error = 1, data = 1, no_inst = 1
Time: 5.0 ns MMP LPDDR4 MR0 initial value is 0x00
Time: 5.0 ns MMP LPDDR4 MR1 initial value is 0x00
Time: 5.0 ns MMP LPDDR4 MR2 initial value is 0x00
……
Time: 5.0 ns MMP LPDDR4 init_done initial value is 0x0
……
```

Example: messaging for a changed value in a configuration register of an LPDDR4 model:

```
……
Time: 2400430.0 ns MMP LPDDR4 init_done value is changed from 0 to 1
Time: 2405440.0 ns MMP LPDDR4 MR2 is changed from 0x00 to 0x80
……
```

8. Logging of $test$plusargs Arguments

Note that the invoking program records the user's valid $test$plusargs :

xeDebug.log:

```
DOWNLOADED
XE> configPM -stb
XE> host -enableTIF
XE> hdlConfig mmp_debuginfo.qel +MMP_DEBUG_DISPLAY
+MMP_DEBUG_DISPLAY_STATE +MMP_DEBUG_DISPLAY_ERROR
+MMP_DEBUG_DISPLAY_DATA
```

Irun.log:

```
irun(64): 13.20-s008: (c) Copyright 1995-2014 Cadence Design Systems, Inc.
TOOL:   irun(64)       13.20-s008: Started on <time_date stamp>
irun
    -64bit
    -R
    -messages
    -tcl
    +MMP_DEBUG_DISPLAY
    +MMP_DEBUG_DISPLAY_STATE
    +MMP_DEBUG_DISPLAY_ERROR
    +MMP_DEBUG_DISPLAY_DATA
    -input ./ixcom_run_debuginfo.tcl

  User defined plus("+") options:
    +MMP_DEBUG_DISPLAY
    +MMP_DEBUG_DISPLAY_STATE
    +MMP_DEBUG_DISPLAY_ERROR
    +MMP_DEBUG_DISPLAY_DATA
```

3. Revision History

The following table shows the revision history for this document

Date	Version	Revision
August 2014	1.0	Initial release
March 2015	1.1	Update related publications list. Update some runtime information. Listed supported models.
May 2015	1.2	Minor modifications.
July 2015	1.3	Update Cadence naming on front page
August 2015	1.4	Add Verilog QSPI and ONFI 4.0 to supported list.
September 2015	1.5	Remove *.vgp references; correct command line synthesis example
January 2016	1.6	Update for Palladium-Z1 and VXE. Add information about compile time Verilog define MMP_NO_DEBUG_DISPLAY
May 2016	1.7	Added Advanced PSRAM and QSPI W25Q256JW to supported list
July 2016	1.8	Updated "Enabling $Display in IXCOM …" section to note the need to specify all model files in the iscdisp option
September 2016	1.9	Added PSRAM OPIDDR APS* to supported list.
November 2016	2.0	Added HyperRam model to supported list
February 2017	2.1	Added GDDR6 model to supported list. Updated copyright.
June 2017	2.2	Added Toggle DDR 2.0 (BiCS3) to supported list
September 2017	2.3	Added eMMC 5.0 to supported list
November 2017	2.4	Added PSRAM x8 and LPDDR4X, LPDDR4 x8 to supported list
December 2017	2.5	Added LPDDR5 to supported list.
April 2018	2.6	Added OctaRAM to supported list Update the instance name in the log example
June 2018	2.7	Added Toshiba Toggle TDDR 3.0 to supported list Added subsections and descriptions for enhancements: • Initial banner messaging • Messaging for writeable configuration and mode registers • Compile phase enabling and disabling of messaging at the level of memory type • Runtime enabling and disabling of messaging at level of memory type and of memory model instance • gfifoDisp option support Added OSPI_PSRAM to supported lists & tables.