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

Universal Flash Storage (UFS) 2.1
Palladium Memory Model
User Guide

Document Version: 2.6

Document Date: July 2018

Contents

List of Figures

List of Tables

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

UFS Memory Model

1. Introduction

The Cadence Palladium Universal Flash Storage (UFS) Palladium memory model is based upon the *Universal Flash Storage v2.1 (UFS 2.1) JESD220C* (March 2016) specification and the *Specification for Unified Protocol (UniPro^{SM}) Version 1.6* (6 August 2013) and the *Specification for M-PHY Version 3.0* (26 July 2013) and *Universal Flash Storage (UFS) Card Extension v1.0 JESD220-2* (March 2016).

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. Acronyms

Table 1: UFS Model Acronyms

NAME	Descriptions
ACK	Acknowledgment
AFC	Acknowledgment and Flow Control
CFG	Configuration
CRC	Cyclical Redundancy Checking
CReq	Credit Transmit Request
CSD	Controlled Segment Dropping
DL	Data Link
DME	Device Management Entity
E2E FC	End-to-End Flow Control
EOB	End of Burst
EOF	End of Frame
EOM	End of Message
FC	Flow Control
HS	High Speed
HS MODE	M-PHY mode of operation for High Speed Transmission
IDN	Identifier for UFS descriptors, attributes, flags
LBA	Logical Block Address
LS MODE	M-PHY mode of operation for Low Speed transmission
LU	Logical Unit
LUN	Logical Unit Number
LUs	Logical Units
MIB	Management Information Base
MIPI	Mobile Industry Processor Interface
MTU	Maximum Transfer Unit
M-RX	M-PHY electrical Receiver
M-TX	M-PHY electrical Transmitter
NAC	Negative Acknowledgement Control
NL	Network Layer
PA	PHY Adapter
PDU	Protocol Data Unit
PHY	Physical Layer
PWM	Pulse-Width-Modulation
RReq	Reset Link Request
RX	Receiver
RPMB	Replay Protected Memory Block
RMMI	Reference M-PHY Module Interface
SOB	Start of Burst
SOF	Start of Frame
SOM	Start of Message
SSU	Start Stop Unit SCSI command
TC0	Traffic Class 0
TC1	Traffic Class 1
TL	Transport Layer
TX	Transmitter
UFS	Universal Flash Storage
UniPro	Unified Protocol
UPIU	UFS Protocol Information Unit
WLUN, W-LUN	Well-known Logical Unit Number

4. Features

The Universal Flash Storage (UFS) Palladium memory model is based upon the *Universal Flash Storage v2.1 (UFS 2.1) JESD220C* (March 2016) specification and the *Specification for Unified Protocol (UniPro^{SM}) Version 1.6* (6 August 2013) and the *Specification for M-PHY Version 3.0* (26 July 2013) and *Universal Flash Storage (UFS) Card Extension v1.0 JESD220-2* (March 2016). The supported features of this model have been thoroughly tested according to the *Universal Flash Storage (UFS) Test specification JESD224* (March 2013). The unsupported features have not been tested. If the user would like to review the test results they should inquire to Cadence's HSV (Palladium) Support team with a specific request.

Table 2: Features List of UFS Model

FEATURE	SUPPORT	NOTE
UNIPRO		
Unipro Options and Tunable Parameters (M-PHY Spec. Annex A and Unipro Spec. Annex D)		
RMMI interface	Yes	The interface width can be configured to 1, 2 or 4 parallel symbols
CPort interface	Yes	The interface width is fixed 4 bytes and only support 1 CPort as UFS required
PHY interface	No	The serial PHY interface is not supported yet
Unipro PHY Adapter (PA) Layer (Unipro Spec. Section 5)		
HS Mode	Yes	Support HS Gear1-3
LS Mode	Yes	Support LS PWM0-7
Multiple PHY Data Lanes	Yes[12]	Support Max 2 Lanes
M-PHY Data Scrambling	Yes	5.7.3.5
Skip Symbol Insertion	Yes	5.7.3.6
Link Startup Sequence	Yes	5.7.8
PA Layer Re-initialization	Yes	5.7.10
Lane Synchronization	Yes	5.7.11
Link Configuration Procedure	Yes	5.7.12
PA Hibernate	Yes	5.7.13
PA Layer MIB	Yes	5.8
Remote Attribute SET and GET	Yes[1]	5.7.14
PHY testing	No	5.7.15 Not required for UFS
Unipro Data Link (DL) Layer (Unipro Spec. Section 6)		
Receive Preemption	Yes	6.6.1.1
TC0 Data Frame	Yes	Only support TC0
DL Layer Flow Control	Yes	6.6.7
Retransmission	Yes	6.6.8
DL Layer Initialization	Yes	6.7
DL Layer MIB	Yes	6.8
Send Preemption	No	6.6.1.1 Not required for UFS
TC1 Data Frame	No	6.6.6 Not required for UFS
Unipro Network Layer (Unipro Spec. Section 7)		
Short Header	Yes	7.8.2.1
Destination Device ID	Yes	7.8.2.2
Network Layer MIB	Yes	7.14
Unipro Transport Layer (Unipro Spec. Section 8)		
Segmentation and Reassembly	Yes	8.11.3, 8.11.4
UniPro Test Feature	Yes	8.15

FEATURE	SUPPORT	NOTE
Transport Layer MIB	Yes	8.17
End-to-End (E2E) Flow Control	No	8.5, 8.11.8 Not required for UFS
CPort Safety Valve (CSV)	No	8.11.9 Not required for UFS
Controlled Segment Dropping (CSD)	No	8.11.10 Not required for UFS
Unipro Device Management Entity (DME) (Unipro Spec. Section 9)		
Attribute SET and GET	Yes	9.2
Reset (Code Reset/ Warm Reset/ EndPointReset)	Yes	9.3
DME Boot Procedures	Yes	9.11
Hibernate	Yes	9.5
DME Power Mode Change	Yes	9.6
DME MIB	Yes	9.12
UFS		
UFS Resets (UFS Spec. Section 7.1)		
Reset event	Yes[12]	7.1.1, 7.1.2
EndPointReset	Yes	7.1.3
Logical Unit Reset	Yes	7.1.4
UFS Power Modes (UFS Spec. Section 7.4)		
Active	Yes	7.4.1.1
Idle	Yes	7.4.1.2
Pre-Active	Yes	7.4.1.3
Sleep	Yes	7.4.1.4
Pre-Sleep	Yes	7.4.1.5
Powerdown	Yes	7.4.1.6
Pre-Powerdown	Yes	7.4.1.7
UFS Protocol Information Unit (UPIU) (UFS Spec. Section 10.7)[2]		
COMMAND UPIU	Yes[2]	10.7.1
RESPONSE UPIU	Yes	10.7.2
DATA OUT UPIU	Yes	10.7.3
DATA IN UPIU	Yes	10.7.4
READY TO TRANSFER UPIU	Yes	10.7.5
TASK MANAGEMENT REQUEST UPIU	Yes	10.7.6
TASK MANAGEMENT RESPONSE UPIU	Yes	10.7.7
QUERY REQUEST UPIU	Yes	10.7.8
QUERY RESPONSE UPIU	Yes	10.7.9
REJECT UPIU	Yes	10.7.10
NOP OUT UPIU	Yes	10.7.11
NOP IN UPIU	Yes	10.7.12
TASK MANAGEMENT FUNCTION	Yes	10.9.8
QUERY FUNCTION	Yes	10.9.9
UFS Native Commands (UFS Spec. Section 11.2)		
UFS NATIVE COMMANDS	n/a	Not defined in this version of spec.
UFS Protocol Layer -- SCSI Commands (UFS Spec. Section 11.3)		
INQUIRY Command	Yes	11.3.2
MODE SELECT (10) Command	Yes	11.3.3
MODE SENSE (10) Command	Yes	11.3.4
READ (6) Command	Yes	11.3.5
READ (10) Command	Yes	11.3.6
READ (16) Command	Yes	11.3.7
READ CAPACITY (10) COMMAND	Yes	11.3.8
READ CAPACITY (16) COMMAND	Yes	11.3.9
START STOP UNIT	Yes	11.3.10

FEATURE	SUPPORT	NOTE
TEST UNIT READY COMMAND	Yes	11.3.11
REPORT LUNS COMMAND	Yes	11.3.12
VERIFY (10)	Yes	11.3.13
WRITE (6) COMMAND	Yes	11.3.14
WRITE (10) COMMAND	Yes	11.3.15
WRITE (16) COMMAND	Yes	11.3.16
REQUEST SENSE COMMAND	Yes	11.3.17
FORMAT UNIT COMMAND	Yes	11.3.18
PRE-FETCH (10) COMMAND	Yes[4]	11.3.19
PRE-FETCH (16) COMMAND	Yes[4]	11.3.20
SECURITY PROTOCOL IN COMMAND	Yes	11.3.21
SECURITY PROTOCOL OUT COMMAND	Yes	11.3.22
SEND DIAGNOSTIC COMMAND	Yes[6]	11.3.23
SYNCHRONIZE CACHE (10) COMMAND	Yes[4]	11.3.24
SYNCHRONIZE CACHE (16) COMMAND	Yes[4]	11.3.25
UNMAP COMMAND	Yes	11.3.26
READ BUFFER COMMAND	Yes[3]	11.3.27
WRITE BUFFER COMMAND	Yes[3]	11.3.28
UFS Mode Pages (UFS Spec. Section 11.4)		
UFS Mode Pages	Yes[8]	11.4
UFS Vital product data parameters (UFS Spec. Section 11.5)		
UFS Vital product data parameters	Yes[9]	11.5
UFS Security (UFS Spec. Section 12)		
Secure Mode	Yes[5]	12.2.3.1, 12.2.3.2, 12.2.3.3, 12.2.3.4
Device Data Protection	Yes	12.3
RPMB	Yes[10]	12.4
UFS Malware Protection (UFS Spec. Section 12.5)		
Malware Protection	No	12.5
UFS Functional (UFS Spec. Section 13)		
UFS Boot	Yes[12]	13.1
Logical Unit Management	Yes	13.2
Logical Block Provisioning	Yes	13.3
UFS Cache	Yes[4]	13.5
Production State Awareness	No	13.6
UFS Functional : Host Device Interaction (UFS Spec. Section 13.4)		
Command Queue: Inter-LU Priority	Yes	13.4.3
Background Operations Mode	No	13.4.4
Power Off Notification	Yes	13.4.5
Dynamic Device Capacity	Partial	13.4.6.1.1, up to MMP_LU<0-7>_ADDR_BITS; Dynamic Device Capacity Procedural Flow (section 13.4.6.1.2) is not supported
Data Reliability	No	13.4.7
Real-Time Clock Information	No	13.4.8
Context Management	No	13.4.9
System Data Tag Mechanism	No	13.4.10
Exception Events Mechanism	No	13.4.11
Queue Depth Definition	Yes[7]	13.4.12
Device Life Span Mode	No	13.4.13
UFS Descriptors, Flags, and Attributes (UFS Spec. Section 14)		

FEATURE	SUPPORT	NOTE
UFS Descriptors, Flags, and Attributes	Yes	Access by Query Function (10.7.9)
UFS Card	Yes[12]	

Note:
1) The UFS Host controller can use DME_PEER_SET and DME_PEER_GET to set or get the device UniPro attributes.
2) Initiator ID (IID) in UPIU and Command Priority (CP) flag fields in COMMAND UPIU are supported.
3) The default buffer size is 1MB, the user can change the buffer size by setting the value of parameter 'buffer_bits'. The model supports only 1 buffer for each LU, the buffer ID is 0. "Mode = 01h" (Vendor Specific) is not supported.
4) The model can accept the cache related commands (Prefetch10, Prefetch16, Synchronize Cache10 and Synchronize16) and send out response, but no real cache and no functionality are implemented. There is no read/write performance change between use with or without cache.
5) The data protection mode for secure mode operation is not supported.
6) The model can accept the 'Send Diagnostic' command with SELFTEST of 1 and parameter list length of 0 and send out response, but no internal operation is implemented.
7) The model implements a shared queue with depth 64 and there is a numerical limit (16) for pending tasks for each LU.
8) The vendor specific mode pages(PF=0) are not supported, and the subpages are not supported. Only the SWP in Control Mode Page, WCE and RCD in Caching Mode Page are changeable; the other fields are not changeable. There is no functional affect from the values set in the three mode pages.
9) Only the Mode Page Policy VPD is supported. All logic units share mode pages.
10) For RPMB, the Message Authentication Code (MAC) calculation (HMAC SHA-256) and check is not supported. For Request type message, the MAC field (from host) is not checked; for Response type message, the MAC field (from device) is not calculated and is filled with all '0'. Only Security Protocol = 'hEC (JEDEC UFS application) and 'h00 (Security protocol information) are supported, other values are not supported.
11) For error conditions, only sense key with values 00h (NO SENSE), 02h (NOT READY), 05h (ILLEGAL REQUEST), 07h (DATA PROTECT), 0Bh (ABORTED COMMAND) are supported, other sense key values are not supported.
12) For UFS card, the HW reset and boot feature are not required and unsupported. The model supports card detection and up to 1 lane.

5. Verilog Macro Defines

The following table lists the Verilog macro `define(s) required by the Palladium UFS model.

Table 3: Verilog Defines

Macro Name	Value range	Required(1)	Purpose
MMP_LU0_ADDR_BITS	>= 8	No	Logical Unit 0 size
MMP_LU1_ADDR_BITS	>= 8	No	Logical Unit 1 size
MMP_LU2_ADDR_BITS	>= 8	No	Logical Unit 2 size
MMP_LU3_ADDR_BITS	>= 8	No	Logical Unit 3 size
MMP_LU4_ADDR_BITS	>= 8	No	Logical Unit 4 size
MMP_LU5_ADDR_BITS	>= 8	No	Logical Unit 5 size
MMP_LU6_ADDR_BITS	>= 8	No	Logical Unit 6 size
MMP_LU7_ADDR_BITS	>= 8	No	Logical Unit 7 size
MMP_LG_MEM_LU0	-	No	Enable Large memory support for LU0
MMP_LG_MEM_LU1	-	No	Enable Large memory support for LU1
MMP_LG_MEM_LU2	-	No	Enable Large memory support for LU2
MMP_LG_MEM_LU3	-	No	Enable Large memory support for LU3
MMP_LG_MEM_LU4	-	No	Enable Large memory support for LU4

MMP_LG_MEM_LU5	-	No	Enable Large memory support for LU5
MMP_LG_MEM_LU6	-	No	Enable Large memory support for LU6
MMP_LG_MEM_LU7	-	No	Enable Large memory support for LU7

NOTE: A UFS device contains one or more logical units. Therefore, per specification, at least one of the macros needs to be defined.

When the value of MMP_LUx_ADDR_BITS is set to larger than 30, the user needs to define the macro MMP_LG_MEM_LUx to enable the large memory support for that Logic Unit. When the value of MMP_LUx_ADDR_BITS is smaller than or equal 30, the user should NOT define the macro MMP_LG_MEM_LUx.

6. Configurations

The UFS model supports up to 8 Logical Units (LUs) with each Logical Unit (LU) supporting the size (MMP_LUx_ADDR_BITS address bits x 32bits memory width) of storage capacity. The size of each LU can be defined with the macros in Table 3.

MMP_LU0_ADDR_BITS and MMP_LU1_ADDR_BITS are defined by default but the values can be modified by editing ufs_controller.vp directly or on the command line by using the appropriate define option at compile time. During runtime QUERY REQUEST UPIU can be issued to write to the configuration descriptor to change the value of dNumAllocUnits for each LU's unit descriptor to use a smaller portion of the storage capacity. The user should not define a Verilog macro to a value of '0'. If an LU is not going to be used, it is recommended to comment it out as shown in ufs_controller.vp for LU 2 to LU 7. An LU of size 0 still requires significant hardware resources and will needlessly increase the gate count.

Table 4: UFS Configuration

UFS Model	Logic Unit	Size
JEDEC UFS	LU0	1GB (MMP_LU0_ADDR_BITS=28)
	LU1	1GB (MMP_LU1_ADDR_BITS=28)
	LU2-LU7	Disabled
Micron UFS mtfc32gamakam	LU0	16GB (MMP_LU0_ADDR_BITS=32)
	LU1	16GB (MMP_LU1_ADDR_BITS=32)
	LU2-LU7	Disabled
JEDEC UFS Card	LU0	1GB (MMP_LU0_ADDR_BITS=28)
	LU1	1GB (MMP_LU1_ADDR_BITS=28)
	LU2-LU7	Disabled

7. Model Block Diagram

The following diagram shows the UFS Model, which includes two parts: the UFS Device and UniPro. The UniPro receives data from the RMMI interface and, after processing, and then passes it through the Cport Interface to the UFS Device. The UFS Device works with UniPro through the DME interface to process UFS commands and manage functions. The UFS Model supports a maximum of 2 lanes.
For UFS Card the model supports up to 1 lane.

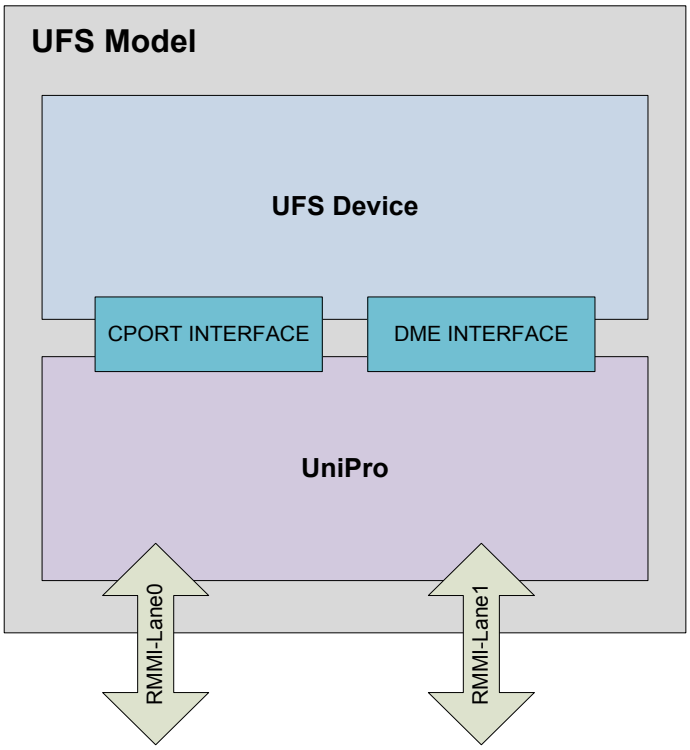


Figure 1: UFS Model Block Diagram

8. Input/Output (I/O) Diagram

The IO signals include model common signals (clock and reset), standard RMMI M-TX and M-RX signals, and extra signals for clock frequency switching. While only 1 lane of the RMMI interface signals is shown in the diagram below, there are a total of two lanes supported in the Palladium UFS Model.
For UFS Card the model supports up to 1 lane. The hardware reset pin ‘sys_reset’ is not supported, and the pin ‘card_detection’ is supported.

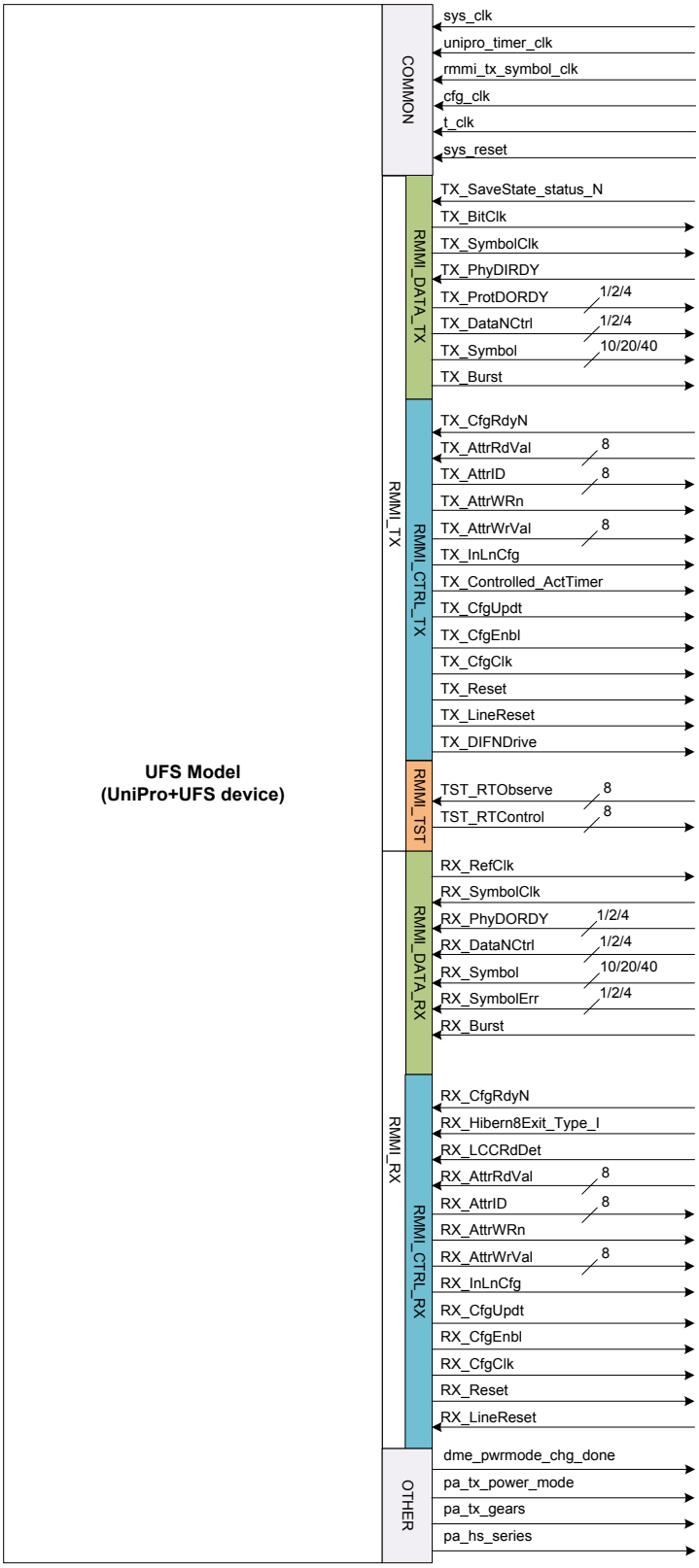


Figure 2: UFS Model I/O Diagram

9. I/O Signal Description

The table below lists and describes the model I/O signals. The RMMI interface signal is identical to the M-TX and M-RX defined in M-PHY specification. For UFS card the model supports up to 1 lane.

Table 5: UFS Model I/O Signals

NAME	TYPE	DESCRIPTION
COMMON IO signals		
sys_clk	Input	Clock for core function
unipro_timer_clk	Input	Clock for UniPro stack timer
rmmi_tx_symbol_clk	Input	Clock for RMMI TX symbol clock
cfg_clk	Input	Clock for UniPro attribute configuration
t_clk	Input	Clock for Cport interface
sys_reset	Input	Reset signal for UFS stack, active low Not required for UFS Card and unsupported for UFS Card
card_detection	Input	ONLY supported for UFS Card It must first drive 0 and then drive 1 to act same as reset. The Unipro Link Startup sequence will start when card_detection changes to 1.
RMMI_DATA_RX – Lane0		
RX_RefClk_0	Output	Reference Clock. It is the same as sys_clk in the model
RX_SymbolClk_0	Input	Symbol Clock All M-RX-DATA interface signals are synchronous with this signal. The M-RX may disable RX_SymbolClk generation when the M-RX is not in LINE-CFG, PWM-BURST, SYS-BURST, or HS-BURST states. The M-RX shall provide the minimum number of cycles to transfer all M-RX data to the Protocol Layer. At the end of BURST, the M-RX shall provide a minimum of two additional clock cycles beyond the de-assertion of RX_Burst. In HS-MODE and SYS-MODE, RX_SymbolClk shall have a period of 10 UI for a 10-bit RX_Symbol bus, 20 UI for a 20-bit RX_Symbol bus, or 40 UI for a 40-bit RX_Symbol bus. In PWM-MODE, RX_SymbolClk shall have a period of 10 T_{PWM-RX} for a 10-bit RX_Symbol bus, 20 T_{PWM-RX} for a 20-bit RX_Symbol bus, or 40 T_{PWM-RX} for a 40-bit RX_Symbol bus. The behavior of RX_SymbolClk must be glitch-free even when this signal is being enabled or disabled. The M-RX shall not provide an RX_SymbolClk “1” or “0” pulse with a duration less than one-quarter of the nominal RX_SymbolClk period.
RX_PhyDORDY_0	Input	PHY Data Output Ready RX_PhyDORDY indicates data is available in the corresponding RX_Symbol bus range. The width of RX_PhyDORDY is one, two or four bits depending on the RX_Symbol bus width of 10, 20, or 40 bits, respectively. Each bit in RX_PhyDORDY corresponds to a 10b8b symbol in RX_Symbol bus. RX_PhyDORDY bitRX_Symbol bits (10b8b enabled) 0 bits[9:0] (bits[7:0]) 1 bits[19:10] (bits[15:8]) 2 bits[29:20] (bits[23:16]) 3 bits[39:30] (bits[31:24]) The M-RX shall set each bit of RX_PhyDORDY to “1” for every RX_SymbolClk cycle that the corresponding RX_Symbol bus range contains new data.

NAME	TYPE	DESCRIPTION
		The M-RX shall set each bit of RX_PhyDORDY bit to “0” for every RX_SymbolClk cycle that the corresponding RX_Symbol bus range does not contain new data. The Protocol Layer shall always be ready to consume the data from the M-RX.
RX_DataNCtrl_0	Input	RX_DataNCtrl indicates the type of symbol on the indicated range of RX_Symbol. The width of RX_DataNCtrl is one, two or four bits depending on the RX_Symbol bus width of 10, 20, or 40 bits, respectively. RX_DataNCtrl are mapped the same as RX_PhyDORDY. The M-RX shall set the corresponding bit of RX_DataNCtrl to “0” when the related RX_Symbol bus range carries a data symbol. The M-RX shall set the corresponding bit of RX_DataNCtrl to “1” when the related RX_Symbol bus range carries a control symbol or a reserved symbol which was erroneously received (see RX_SymbolErr definition). The M-RX shall set all bits of RX_DataNCtrl to “0” when 10b8b decoding is bypassed.
RX_Symbol_0	Input	RX_Symbol is used for BURST data transfer to the Protocol Layer. The contents of this bus depend on the interface width (10, 20 or 40 bits, corresponding to 1, 2 and 4 parallel symbols, respectively), and also on whether or not the 10b8b decoding function is bypassed. When the 10b8b decoding function is disabled, RX_Symbol carries the raw data as received on the LINEs, parallelized according to the implemented width. The LSb of RX_Symbol shall correspond to the earliest received bit. When the 10b8b decoding function is enabled, only the 8, 16, or 32 LSbs of RX_Symbol are used to carry the decoded DATA or control symbol. The M-RX shall set the remaining MSbs to “0”.
RX_SymbolErr_0	Input	The width of RX_SymbolErr is one, two or four bits depending on the RX_Symbol bus width of 10, 20, or 40 bits, respectively. The M-RX shall set each bit of RX_SymbolErr to “1” for one RX_SymbolClk cycle when any of the following conditions on the corresponding RX_Symbol bus range are “TRUE”: • The 3b4b sub-block is in error while decoding the related 8b10b symbol received over the LINE • The 5b6b sub-block is in error while decoding the related 8b10b symbol received over the RLINE • The Running Disparity algorithm computes an RD error for the related 8b10b symbol received over the LINE • The related 8b10b symbol received over the LINE is a reserved symbol. The M-RX shall set all bits of RX_SymbolErr to “0” for all other conditions. RX_Symbol shall carry, in the corresponding bus range, the remapped payload byte. The M-RX shall set all bits of RX_SymbolErr to “0” when 10b8b decoding is bypassed.
RX_Burst_0	Input	RX_Burst provides a framing window to the Protocol Layer for received BURSTs. The M-RX shall set RX_Burst to “1” when it detects the start of a PREPARE period. The M-RX shall set RX_Burst to “0” when it detects any of the BURST exit conditions and all 8b10b payload data has been sent to the Protocol Layer via RX_Symbol.
RMMI_CTRL_RX – Lane0		
RX_CfgRdyN_0	Input	RX_CfgRdyN indicates the M-RX cannot process a register write command to its effective configuration bank.

NAME	TYPE	DESCRIPTION
		The M-RX shall set this signal to “1” in the same RX_CfgClk cycle that triggers its internal FSM exit from SLEEP, STALL, or HIBERN8 state to any other state. The M-RX may also set this signal to “1” while it is processing a Protocol-issued change to its effective configuration bank. The M-RX shall set this signal to “0” when its internal FSM is in SLEEP, STALL, or HIBERN8 state and the MODULE is ready to accept a register write command to any register of its effective configuration bank. For an RX_Reset (local RESET command, the M-RX shall set RX_CfgRdyN to “1” asynchronously. If the Protocol Layer issues write commands to the M-RX effective configuration bank (including RX_CfgUpdt) while RX_CfgRdyN is set to “0”, the M-RX shall process those commands immediately. If the Protocol Layer issues write commands to the M-RX effective configuration bank while RX_CfgRdyN is set to “1”, the specific M-RX behavior is dependent on the command itself addressing an OFFLINE-SET or INLINE-SET attribute. The M-RX shall execute a write command to an OFFLINE-SET Attribute in the effective configuration bank. The M-RX shall redirect a write command to an INLINE-SET attribute in the effective configuration bank to the associated shadow register. On the next transition of RX_CfgRdyN from “1” to “0”, the M-RX automatically updates the associated effective register. The M-RX shall not ignore a write command or Rx_CfgUpdt request from the Protocol Layer except in UNPOWERED and DISABLED states, or when local RESET is asserted. The M-RX shall respond to read commands from the Protocol Layer regardless of the value of RX_CfgRdyN. The M-RX shall process register write commands to its shadow memory bank regardless of the value of RX_CfgRdyN.
RX_Hibern8Exit_Type_I_0	Input	*RX_Hibern8Exit_Type-I* indicates the M-RX is exiting HIBERN8. The M-RX sets *RX_Hibern8Exit_Type-I* to “1” when it detects a DIF-Z to DIF-N transition on the LINE. The M-RX sets *RX_Hibern8Exit_Type-I* to “0” when the M-RX is in either HIBERN8 or DISABLED state. A Type-I implementation shall include this signal.
RX_LCCRdDet_0	Input	RX_LCCRdDet indicates the M-RX received at least one LCC-READ sequence, add has updated all corresponding attributes of the pending LCC-READ sequences. Upon exiting LCC-MODE following a LCC-READ sequence, the M-RX shall set RX_LCCRdDet to “1” for a single RX_CfgClk cycle.
RX_AttrRdVal_0	Input	RX_AttrRdVal carries the attribute value read from an M-RX-MIB attribute specified by RX_AttrID. The M-RX-MIB attribute value should be held on this bus until a subsequent read command is issued by the protocol. The M-RX shall provide the specified attribute value within one-half of the RX_CfgClk period.
RX_AttrID_0	Output	RX_AttrID carries the AttributeID of M-RX Configuration attributes for read or write operations, or M-RX Capability attribute or OMCS Status Attributes for read operation.
RX_AttrWRn_0	Output	RX_AttrWRn specifies the operation, read or write, to perform on an M-RX-MIB attribute. The Protocol Layer shall set RX_AttrWRn to “0” to indicate a read operation, or “1” to indicate a write operation.
RX_AttrWrVal_0	Output	RX_AttrWrVal carries the attribute value to write to an M-RX-MIB attribute specified by RX_AttrID.
RX_InLnCfg_0	Output	RX_InLnCfg is used in conjunction with RX_AttrWRn and RX_CfgEnbl to direct an M-RX-MIB write operation to the M-RX's

NAME	TYPE	DESCRIPTION
		shadow memory bank, or to the M-RX's effective configuration bank. RX_InLnCfg should be set to “1”.
RX_CfgUpdt_0	Output	RX_CfgUpdt transfers the contents of the INLINE-CR registry to the effective configuration bank during a SAVE state. The Protocol Layer shall set RX_CfgUpdt to “1” for a single RX_CfgClk cycle to trigger the upload of the entire M-RX shadow memory contents to the effective configuration bank. The Protocol Layer shall move the MODULE into a SAVE state, if not already in a SAVE state, before the new settings become effective. While RX_InLnCfg allows for single attributes to be written directly to the M-RX effective configuration bank, e.g. dithering control between HS-BURSTs, RX_CfgUpdt allows the Protocol Layer to make configuration changes to the M-RX's shadow memory sequentially, then make the changes effective atomically. RX_CfgUpdt indicates to the MODULE, independent of MODULE state, the completion of the requested configuration settings for effectuating configuration change requests.
RX_CfgEnbl_0	Output	Config Enable The Protocol Layer shall set RX_CfgEnbl to “1” for a single RX_CfgClk cycle to perform an attribute read, or write, operation. The Protocol Layer shall set RX_CfgEnbl, RX_AttrID, RX_AttrWRn, RX_InLnCfg and RX_AttrWrVal in the same RX_CfgClk cycle.
RX_CfgClk_0	Output	Control Interface Clock. All M-RX-CTRL interface signals, with the exception of RX_Reset and *RX_Hibern8Exit_Type-I*, are synchronous with this signal. The exact frequency of RX_CfgClk is implementation specific. Choice of frequency should consider squelch detection, adequate measurement of $T_{LINE\text{-}RESET}$, LCC-READ event signaling and minimizing interface access latencies. RX_CfgClk may change frequency depending on state, but is expected to be available in all M-RX states except DISABLED and UNPOWERED.
RX_Reset_0	Output	RX_Reset is the active-high asynchronous reset for all logic inside the M-RX. RX_Reset implements the local RESET function. The Protocol Layer shall set RX_Reset to “1” for at least 100 ns.
RX_LineReset_0	Input	RX_LineReset indicates the status of the LINE-RESET action in the M-RX. M-RX shall set RX_LineReset to “1” when LINE-RESET is detected. M-RX shall set RX_LineReset to “0” once it has transitioned to the LINE-RESET exit state.
RMMI_DATA_TX – Lane0		
TX_SaveState_status_N_0	Input	*TX_SaveState_Status_N* indicates the M-TX is entering or exiting a SAVE state. The Protocol Layer can use this signal to understand when the M-TX is not transmitting PREPARE, SYNC, HOB, PAYLOAD, TOB, BURST Extension or LINE-CFG information. The M-TX sets *TX_SaveState_Status_N* to “0” when it enters into a SAVE state. The M-TX sets *TX_SaveState_Status_N* to “1” when it exits a SAVE state. A Type-I implementation shall include this signal. Though not required, a Type-II implementation should include this signal.
TX_BitClk_0	Output	Bit Clock This clock is the same as sys_clk in the model.
TX_SymbolClk_0	Output	Symbol Clock All M-TX-DATA interface signals are synchronous with this signal. The Protocol Layer may disable TX_SymbolClk generation when the M-TX is not in LINE-CFG, PWM-BURST, SYS-BURST, or HS-BURST states. For this purpose, the Protocol Layer shall read the M-TX FSM state attribute.

NAME	TYPE	DESCRIPTION
		In HS-MODE and SYS-MODE, TX_SymbolClk shall have a period of 10 UI for a 10-bit TX_Symbol bus, 20 UI for a 20-bit TX_Symbol bus, or 40 UI for a 40-bit TX_Symbol bus. In PWM-MODE, TX_SymbolClk shall have a period of 10 T_{PWM_TX} for a 10-bit TX_Symbol bus, 20 T_{PWM_TX} for a 20-bit RX_Symbol bus, or 40 T_{PWM_TX} for a 40-bit TX_Symbol bus. The behavior of TX_SymbolClk must be glitch-free even when this signal is being enabled or disabled. TX_SymbolClk shall not have a “1” or “0” pulse with a duration less than one-quarter of the nominal TX_SymbolClk period.
TX_PhyDIRDY_0	Input	PHY Data Input Ready TX_PhyDIRDY indicates the M-TX is ready to accept new data on the TX_Symbol bus. The M-TX shall set TX_PhyDIRDY to “1” when the M-TX is ready to consume data. The M-TX shall set TX_PhyDIRDY to “0” when the M-TX is busy. The Protocol Layer should not update TX_Symbol while TX_PhyDIRDY is “0” and TX_Burst is “1”.
TX_ProtDORDY_0	Output	Protocol Data Output Ready TX_ProtDORDY indicates data is available in the corresponding TX_Symbol bus range. The width of TX_ProtDORDY is one, two or four bits depending on the TX_Symbol bus width of 10, 20, or 40 bits, respectively. Each bit in TX_ProtDORDY corresponds to a range of bits in the TX_Symbol bus. TX_ProtDORDY Bits TX_Symbol bits (8b10b enabled) 0 bits[9:0] (bits[7:0]) 1 bits[19:10] (bits[15:8]) 2 bits[29:20] (bits[23:16]) 3 bits[39:30] (bits[31:24]) When a TX_Symbol bus range contains new data, the Protocol Layer shall set the corresponding bit of TX_ProtDORDY to “1” for each TX_SymbolClk cycle. When a TX_Symbol bus range does not contain new data, the Protocol Layer shall set the corresponding bit of TX_ProtDORDY to “0” for each TX_SymbolClk cycle. When the M-TX 8b10b encoding function is bypassed, the Protocol Layer shall set the applicable bits of TX_ProtDORDY to “1” for each TX_SymbolClk cycle during a BURST.
TX_DataNCtrl_0	Output	TX_DataNCtrl indicates the type of symbol on the indicated range of TX_Symbol. The width of the TX_DataNCtrl is one, two or four bits depending on the TX_Symbol bus width of 10, 20, or 40 bits, respectively. The bits of TX_DataNCtrl are mapped the same as the bits of TX_ProtDORDY. The Protocol Layer shall set the corresponding bit of TX_DataNCtrl to “0” when the related TX_Symbol bus range carries a data symbol. The Protocol Layer shall set the corresponding bit of TX_DataNCtrl to “1” when the related TX_Symbol bus range carries a control symbol. The Protocol Layer should set all bits of TX_DataNCtrl to “0” when 8b10b encoding is bypassed. The M-TX shall ignore all bits of TX_DataNCtrl when 10b8b decoding is bypassed.
TX_Symbol_0	Output	TX_Symbol is used for BURST data transfer to the M-TX. The contents of this bus depend on the interface width (10, 20 or 40 bits, corresponding to 1, 2 and 4 parallel symbols, respectively), and also on whether the 8b10b encoding function in the M-TX is bypassed. When the M-TX 8b10b encoding function is bypassed, TX_Symbol carries the raw data to send on the LINEs,

NAME	TYPE	DESCRIPTION
		parallelized according to the implemented width. The LSb of TX_Symbol shall correspond to the earliest transmitted bit. When the M-TX 8b10b encoding function is enabled, only the 8, 16, or 32 LSbs of TX_Symbol are used to carry the unencoded DATA or control symbol. The M-TX shall ignore the unused MSbs of TX_Symbol. The Protocol Layer should set the unused MSbs to “0”. TX_Symbol is accepted by the M-TX on every TX_SymbolClk cycle in which TX_ProtDORDY, TX_PhyDIRDY and TX_Burst are “1”.
TX_Burst_0	Output	TX_Burst initiates a BURST. The Protocol Layer shall set TX_Burst to “1” to initiate a BURST, and hold the value for the duration of the BURST. Once TX_Burst is set to “1”, the M-TX shall send the PREPARE sequence (and SYNC sequence in the case of a HS-BURST), followed by data or FILLER symbols. If any bit of TX_ProtDORDY is set to “1”, the M-TX shall send the data present on the corresponding TX_Symbol bus range. If any bit of TX_ProtDORDY is set to “0”, the M-TX shall send one FILLER for each TX_ProtDORDY bit set to 0. Once TX_Burst is set to “0”, the M-TX shall send the TAIL-OF-BURST sequence.
RMMI_CTRL_TX – Lane0		
TX_CfgRdyN_0	Input	Identical behavior to RX_CfgRdyN
TX_AttrRdVal_0	Input	Identical behavior to RX_AttrRdVal
TX_AttrID_0	Output	Identical behavior to RX_AttrID
TX_AttrWRn_0	Output	Identical behavior to RX_AttrWRn
TX_AttrWrVal_0	Output	Identical behavior to RX_AttrWrVal
TX_InLnCfg_0	Output	Identical behavior to RX_InLnCfg
TX_Controlled_ActTimer_0	Output	Not used in the model
TX_CfgUpdt_0	Output	Identical behavior to RX_CfgUpdt
TX_CfgEnbl_0	Output	Identical behavior to RX_CfgEnbl
TX_CfgClk_0	Output	Identical behavior to RX_CfgClk
TX_Reset_0	Output	Identical behavior to RX_Reset
TX_LineReset_0	Output	TX_LineReset triggers the M-TX to issue a LINE-RESET condition. When TX_Controlled_ActTimer is set to “0”, or TX_Controlled_ActTimer is not implemented, the Protocol Layer shall set TX_Burst to “0” and wait for *T*ACTIVATE after the M-TX sets TX_SaveState_Status_N to “0” before it sets TX_LineReset to “1” for one TX_CfgClk cycle. After the Protocol Layer sets TX_LineReset to “1” for one TX_CfgClk cycle, the M-TX shall immediately drive the LINE-RESET condition. When TX_Controlled_ActTimer is set to “1”, after the Protocol Layer sets TX_LineReset to “1” for one TX_CfgClk cycle, the M-TX shall immediately drive DIF-N for *T*ACTIVATE, irrespective of its current state, before it drives the LINE-RESET condition.
TX_DIFNDrive_0	Output	Not used in the model
RMMI_TST – Lane0		
TST_RTObserve_0	Input	Not used in the model
TST_RTControl_0	Output	Not used in the model
RMMI_DATA_RX – Lane1		
RX_RefClk_1	Output	Identical behavior to equivalent signal in Lane0
RX_SymbolClk_1	Input	Identical behavior to equivalent signal in Lane0
RX_PhyDORDY_1	Input	Identical behavior to equivalent signal in Lane0
RX_DataNCtrl_1	Input	Identical behavior to equivalent signal in Lane0
RX_Symbol_1	Input	Identical behavior to equivalent signal in Lane0
RX_SymbolErr_1	Input	Identical behavior to equivalent signal in Lane0
RX_Burst_1	Input	Identical behavior to equivalent signal in Lane0

NAME	TYPE	DESCRIPTION
RMMI_CTRL_RX – Lane1		
RX_CfgRdyN_1	Input	Identical behavior to equivalent signal in Lane0
RX_Hibern8Exit_Type_I_1	Input	Identical behavior to equivalent signal in Lane0
RX_LCCRdDet_1	Input	Identical behavior to equivalent signal in Lane0
RX_AttrRdVal_1	Input	Identical behavior to equivalent signal in Lane0
RX_AttrID_1	Output	Identical behavior to equivalent signal in Lane0
RX_AttrWRn_1	Output	Identical behavior to equivalent signal in Lane0
RX_AttrWrVal_1	Output	Identical behavior to equivalent signal in Lane0
RX_InLnCfg_1	Output	Identical behavior to equivalent signal in Lane0
RX_CfgUpdt_1	Output	Identical behavior to equivalent signal in Lane0
RX_CfgEnbl_1	Output	Identical behavior to equivalent signal in Lane0
RX_CfgClk_1	Output	Identical behavior to equivalent signal in Lane0
RX_Reset_1	Output	Identical behavior to equivalent signal in Lane0
RX_LineReset_1	Input	Identical behavior to equivalent signal in Lane0
RMMI_DATA_TX – Lane1		
TX_SaveState_status_N_1	Input	Identical behavior to equivalent signal in Lane0
TX_BitClk_1	Output	Identical behavior to equivalent signal in Lane0
TX_SymbolClk_1	Output	Identical behavior to equivalent signal in Lane0
TX_PhyDIRDY_1	Input	Identical behavior to equivalent signal in Lane0
TX_ProtDORDY_1	Output	Identical behavior to equivalent signal in Lane0
TX_DataNCtrl_1	Output	Identical behavior to equivalent signal in Lane0
TX_Symbol_1	Output	Identical behavior to equivalent signal in Lane0
TX_Burst_1	Output	Identical behavior to equivalent signal in Lane0
RMMI_CTRL_TX – Lane1		
TX_CfgRdyN_1	Input	Identical behavior to equivalent signal in Lane0
TX_AttrRdVal_1	Input	Identical behavior to equivalent signal in Lane0
TX_AttrID_1	Output	Identical behavior to equivalent signal in Lane0
TX_AttrWRn_0	Output	Identical behavior to equivalent signal in Lane0
TX_AttrWrVal_1	Output	Identical behavior to equivalent signal in Lane0
TX_InLnCfg_1	Output	Identical behavior to equivalent signal in Lane0
TX_Controlled_ActTimer_1	Output	Identical behavior to equivalent signal in Lane0
TX_CfgUpdt_1	Output	Identical behavior to equivalent signal in Lane0
TX_CfgEnbl_1	Output	Identical behavior to equivalent signal in Lane0
TX_CfgClk_1	Output	Identical behavior to equivalent signal in Lane0
TX_Reset_1	Output	Identical behavior to equivalent signal in Lane0
TX_LineReset_1	Output	Identical behavior to equivalent signal in Lane0
TX_DIFNDrive_1	Output	Identical behavior to equivalent signal in Lane0
RMMI_TST – Lane1		
TST_RTObserve_1	Input	Identical behavior to equivalent signal in Lane0
TST_RTControl_1	Output	Identical behavior to equivalent signal in Lane0
Other IO signals		
dme_pwrmode_chg_done	Output	Used for UniPro clock switch When the dme_pwrmde_cfg_done is high, it indicates the power mode change sequence done and the user can start clock frequency switch according to power_mode and gears and series.
pa_tx_power_mode	Output	Used for Unipro clock switch Defined same as PA Layer attribute “PA_PWRMode”
pa_tx_gears	Output	Used for Unipro clock switch Defined same as PA Layer attribute “PA_TxGear”
pa_hs_series	Output	Used for Unipro clock switch Defined same as PA Layer attribute “PA_HSSeries”

10. Connection with RMMI interface without MPHY

The host controller and UFS device can be connected with the RMMI interface without MPHY. The following is an instantiation example for the connection.
It uses a 20bit RMMI symbol interface and has a unipro_timer_clk frequency of 125MHz. The parameters “RX_NUM_OF_PARALLEL_SYMBOLS” and “TX_NUM_OF_PARALLEL_SYMBOLS” are the number of parallel symbols for RMMI RX/TX data and are set to 2. The parameter “TCK” is the number of cycles of the unipro_timer_clk and is set to 8.

```
jedec_ufs_stack
    #(.RX_NUM_OF_PARALLEL_SYMBOLS(2), // RMMI RX parallel symbols
      .TX_NUM_OF_PARALLEL_SYMBOLS(2), // RMMI TX parallel symbols
      .TCK(8)                         // CYCLE of unipro_timer_clk
     )
ufs_stack_inst
(
// ----------- Clocks and Reset ------------------------
    .sys_clk                    (sys_clk                ), //I
    .unipro_timer_clk           (unipro_timer_clk       ), //I
    .rmmi_tx_symbol_clk         (rmmi_tx_symbol_clk     ), //I
    .cfg_clk                    (cfg_clk                ), //I
    .t_clk                      (t_clk                  ), //I
    .sys_reset                  (sys_reset              ), //I
// ----------- RMMI RX Data Interface - LANE0 ----------
    .RX_RefClk_0                (                       ), //O
    .RX_SymbolClk_0             (tx_symbolclk0_host     ), //I
    .RX_PhyDORDY_0              (tx_protdordy0_host     ), //I
    .RX_DataNCtrl_0             (tx_datanctrl0_host     ), //I
    .RX_Symbol_0                (tx_symbol0_host        ), //I
    .RX_SymbolErr_0             ('b0                    ), //I
    .RX_Burst_0                 (tx_burst0_host         ), //I
// ----------- RMMI RX Control Interface - LANE0 -------
    .RX_CfgRdyN_0               (1'b1                   ), //I
    .RX_Hibern8Exit_Type_I_0    (1'b0                   ), //I
    .RX_LCCRdDet_0              (1'b0                   ), //I
    .RX_AttrRdVal_0             (8'h0                   ), //I
    .RX_AttrID_0                (                       ), //O
    .RX_AttrWRn_0               (                       ), //O
    .RX_AttrWrVal_0             (                       ), //O
    .RX_InLnCfg_0               (                       ), //O
    .RX_CfgUpdt_0               (                       ), //O
    .RX_CfgEnbl_0               (                       ), //O
    .RX_CfgClk_0                (                       ), //O
    .RX_Reset_0                 (                       ), //O
    .RX_LineReset_0             (tx_linereset0_host     ), //I
// ----------- RMMI TX Data Interface - LANE0 ----------
    .TX_SaveState_status_N_0    (1'b0                   ), //I
    .TX_BitClk_0                (                       ), //O
    .TX_SymbolClk_0             (rx_symbolclk0_host     ), //O
    .TX_PhyDIRDY_0              (TX_Burst_0_d           ), //I
    .TX_ProtDORDY_0             (rx_phydordy0_host      ), //O
    .TX_DataNCtrl_0             (rx_datanctrl0_host     ), //O
    .TX_Symbol_0                (rx_symbol0_host        ), //O
    .TX_Burst_0                 (rx_burst0_host         ), //O
// ----------- RMMI TX Control Interface - LANE0 -------
    .TX_CfgRdyN_0               (1'b1                   ), //I
    .TX_AttrRdVal_0             (8'h0                   ), //I
    .TX_AttrID_0                (                       ), //O
    .TX_AttrWRn_0               (                       ), //O
```

```
    .TX_AttrWrVal_0             (                       ), //O
    .TX_InLnCfg_0               (                       ), //O
    .TX_Controlled_ActTimer_0   (                       ), //O
    .TX_CfgUpdt_0               (                       ), //O
    .TX_CfgEnbl_0               (                       ), //O
    .TX_CfgClk_0                (                       ), //O
    .TX_Reset_0                 (                       ), //O
    .TX_LineReset_0             (                       ), //O
    .TX_DIFNDrive_0             (                       ), //O
// ----------- RMMI Test Ext Interface - LANE0 ---------
    .TST_RTObserve_0            (8'h0                   ), //I
    .TST_RTControl_0            (                       ), //O
// ----------- RMMI RX Data Interface - LANE1 ----------
    .RX_RefClk_1                (                       ), //O
    .RX_SymbolClk_1             (tx_symbolclk1_host     ), //I
    .RX_PhyDORDY_1              (tx_protdordy1_host     ), //I
    .RX_DataNCtrl_1             (tx_datanctrl1_host     ), //I
    .RX_Symbol_1                (tx_symbol1_host        ), //I
    .RX_SymbolErr_1             ('b0                    ), //I
    .RX_Burst_1                 (tx_burst1_host         ), //I
// ----------- RMMI RX Control Interface - LANE1 -------
    .RX_CfgRdyN_1               (1'b1                   ), //I
    .RX_Hibern8Exit_Type_I_1    (1'b0                   ), //I
    .RX_LCCRdDet_1              (1'b0                   ), //I
    .RX_AttrRdVal_1             (8'h0                   ), //I
    .RX_AttrID_1                (                       ), //O
    .RX_AttrWRn_1               (                       ), //O
    .RX_AttrWrVal_1             (                       ), //O
    .RX_InLnCfg_1               (                       ), //O
    .RX_CfgUpdt_1               (                       ), //O
    .RX_CfgEnbl_1               (                       ), //O
    .RX_CfgClk_1                (                       ), //O
    .RX_Reset_1                 (                       ), //O
    .RX_LineReset_1             (tx_linereset1_host     ), //I
// ----------- RMMI TX Data Interface - LANE0 ----------
    .TX_SaveState_status_N_1    (1'b0                   ), //I
    .TX_BitClk_1                (                       ), //O
    .TX_SymbolClk_1             (rx_symbolclk1_host     ), //O
    .TX_PhyDIRDY_1              (TX_Burst_1_d           ), //I
    .TX_ProtDORDY_1             (rx_phydordy1_host      ), //O
    .TX_DataNCtrl_1             (rx_datanctrl1_host     ), //O
    .TX_Symbol_1                (rx_symbol1_host        ), //O
    .TX_Burst_1                 (rx_burst1_host         ), //O
// ----------- RMMI TX Control Interface - LANE0 -------
    .TX_CfgRdyN_1               (1'b1                   ), //I
    .TX_AttrRdVal_1             (8'h0                   ), //I
    .TX_AttrID_1                (                       ), //O
    .TX_AttrWRn_1               (                       ), //O
    .TX_AttrWrVal_1             (                       ), //O
    .TX_InLnCfg_1               (                       ), //O
    .TX_Controlled_ActTimer_1   (                       ), //O
    .TX_CfgUpdt_1               (                       ), //O
    .TX_CfgEnbl_1               (                       ), //O
    .TX_CfgClk_1                (                       ), //O
    .TX_Reset_1                 (                       ), //O
    .TX_LineReset_1             (                       ), //O
    .TX_DIFNDrive_1             (                       ), //O
// ----------- RMMI Test Ext Interface - LANE0 ---------
    .TST_RTObserve_1            (8'h0                   ), //I
    .TST_RTControl_1            (                       ), //O
// ----------- Output Signals For Clock Switch ---------
    .dme_pwrmode_chg_done       (                       ), //O
    .pa_tx_power_mode           (                       ), //O
```

```
    .pa_tx_gears                (                       ), //O
    .pa_hs_series               (                       )  //O
);
```

Notes:
1. For the usage and frequency relationship of the input clocks, please refer to the section UFS Model Clocks.
2. All the connection signals suffixed with *_host are refer to the Host controller RMMI interface signals.
3. The connection signals 'TX_Burst_0_d' is one cycle shift delay of TX_Burst_0 (output from model) with TX_SymbolClk_0, which is used to mimic input 'TX_PhyDIRDY_0'.
Doing the same for 'TX_Burst_1_d'.

```
always @(posedge TX_SymbolClk_0) TX_Burst_0_d <= TX_Burst_0;
always @(posedge TX_SymbolClk_1) TX_Burst_1_d <= TX_Burst_1;
```

4. All the unused output pins can be floating.
5. The last 4 output signals are used for clock switch if needed.

11. Model Parameter Descriptions

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

Table 6: User Adjustable Parameters

User Adjustable Parameter	Default Value	Description
RX_NUM_OF_PARALLEL_SYMBOLS	2	Number of parallel symbols for RMMI RX data bus. Valid values are 1, 2, 4
TX_NUM_OF_PARALLEL_SYMBOLS	2	Number of parallel symbols for RMMI TX data bus. Valid values are 1, 2, 4
TCK	8	Cycle of unipro_timer_clk, used for timers' counter
buffer_bits	20	The UFS buffer size (2^buffer_bits)

The following table provides some information about exposed parameters and localparams that are NOT user adjustable. On rare occasion the user may find one of these parameters or localparam needs adjusting for their configuration. If this case arises, please contact Cadence emulation or MMP support.

Table 7: Visible Non-User-Adjustable Parameters & Localparams

Parameter / Localparam	Default Value	Description

CPORT_TX_DATA_N	4	Cport TX data width in bytes
CPORT_RX_DATA_N	4	Cport RX data width in bytes
RX_SYMBOL_WIDTH	10* RX_NUM_OF_PARALLEL_SYMBOLS	RMMI RX data width
TX_SYMBOL_WIDTH	10* TX_NUM_OF_PARALLEL_SYMBOLS	RMMI TX data width
CPORT_TX_DATA_WIDTH	8* CPORT_TX_DATA_N	CPort TX data width
CPORT_RX_DATA_WIDTH	8* CPORT_RX_DATA_N	CPort RX data width
block_bit0 block_bit1 … block_bit7	12	The value of bLogicalBlockSize0 bLogicalBlockSize1 … bLogicalBlockSize7. Note: these localparameters are in the file ufs_controller.vp

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.

12. Address Mapping

Users generally need to pay attention to the address mapping when they access the main array with Palladium runtime load and dump commands. The address mapping for load and dump commands can be different from that in the read/write commands. For models with programmable array widths, such as this model, the address mapping for accessing the core memory array can become complicated by the fact that even though the real memory device implements an 8bit array width, for the large memory model configurations the data need to be reformatted into 32 bit load and dump files for these runtime commands. This section explains the reason for this implementation and then suggests a procedure that alleviates the need for data modification.

The allocated palladium memory array width is 32bits and the size within each LU is 2**MMP_LU<0-7>_ADDR_BITS x 4 bytes. UFS addressing is in terms of LBA (Logical Block Address). The size of each logical block depends on the value of unit descriptor (UFS spec. 14.1.6.5) bLogicalBlockSize and each logical block is 2**bLogicalBlockSize bytes. The default value is 0Ch, which corresponds to 4 Kbyte per logical block. As an example, if a read command specifies an LBA of 5 and a transfer length of 6 blocks, the command is specifying memory array locations 20480 to 45055. If the user wants to preload the array at LBA 5 the data file needs to specify the address @1400h (5000h>>2).

The hierarchy of the memory array for each LU is
ufs_stack_inst.ufs_device.lu0_mem, ufs_stack_inst.ufs_device.lu1_mem … ufs_stack_inst.ufs_device.lu7_mem.

If the macro ‘MMP_LG_MEM_LUx’ is defined for large memory support, i.e. when ‘MMP_LUx_ADDR_BITS’ is larger than 30, the memory array of the logic unit will be split into multiple memory array and its hierarchy will be different. See the section 23 on Large Memory Support for more details.

Palladium runtime memory load and dump operations support various data formats which can vary the user's load/dump performance and data file content.

For programmable array widths that are larger than 8bits, the user has the option to take advantage of the Palladium utility memTran to convert a memory data file formatted in 8bits wide data—i.e., a readmemh format—to a headerless binary formatted file—i.e. raw2 format—that can be loaded via the memory –load command without the user needing to be aware of or concerned about the memory width.

The steps are as follows for an example using readmemh formatted input:

1. User creates a "readmemh" formatted, or other Palladium supported format," data file with 8bits wide data
2. User converts the readmemh file using Palladium's *memTran* utility into a file with "pd_raw2" format.
3. User loads the memory at runtime using the command 'memory –load %pd_raw2' to load the data file into memory. Because this command does not care about the memory width and simply streams the data into the assigned memory at the specified start address, the user can perform the memory access and load without modifying the data file from the standard readmemh format. The user does need to remain cognizant of the preload start address if the start address is not address 0. The start address is the physical address of the memcore that is a 32bit memory address. The description of how to calculate the physical address for the memory was discussed above.

To look at a specific example, the user may have a memdata file called test_load.h that contains 8 bits wide data. The following memTran command converts the test_load.h file to raw2 formatted data (pd_raw2) in a file called test_load.bin and then compares the content of the two files.

```
memTran -translate %readmemh memdata/test_load.h %pd_raw2 test_load.bin -width 8 -depth 256
memTran -compare   %readmemh memdata/test_load.h %pd_raw2 test_load.bin -width 8 -depth 256
```

During runtime, the memory –load command can be used as below:

```
memory -load %pd_raw2 sd_card_256m.dut.mem -file test_load.bin –nochecksize
```

And the memory –dump command as below:

```
memory -dump %pd_raw2 sd_card_256m.dut.mem -file test_dump.bin
```

The dump command may be followed by the appropriate memTran commands to convert the binary formatted file back to 8-bit width format.

```
memTran -translate %pd_raw2 test_dump.bin %readmemh test_dump.bin.postconv.h -width 64
memTran -compare %pd_raw2 test_dump.bin %readmemh test_dump.bin.postconv.h -width 64 -depth
335
```

An additional consideration is that load/dump operations are typically faster with binary formatted files such as raw2 formats.

Please reference the UXE/VXE user guide and command reference manual for more detailed information regarding the memTran utility which translates between raw format

and other file formats. Referencing the following sections, among others, may be helpful: “Compiling and Running Designs with Memories,” “Using Memory Streaming,” and “Translating Memory Format Using the memTran Utility.”

13. UniPro Attribute Definitions

The table below lists the attributes and reset values for each layer of UniPro and M-PHY. The user may configure the UniPro Master's attributes with UniPro Slave default attribute values. The attributes can be SET or GET by the UniPro Master by using DME_PEER_SET and DME_PEER_GET commands.
For M-PHY, only the attributes can be accessed, no other M-PHY function implemented.

Table 8: UniPro and M-PHY Attributes

Attribute Name	Attribute ID	Reset Value
PA Layer Attributes		
PA_ActiveTxDataLanes	0x1560	1
PA_TxTailingClocks	0x1564	3
PA_ActiveRxDataLanes	0x1580	1
PA_PHY_Type	0x1500	1
PA_AvailTxDataLanes	0x1520	1
PA_AvailRxDataLanes	0x1540	1
PA_MinRxTrailingClocks	0x1543	0
PA_TxHsG1SyncLength	0x1552	0x004F
PA_TxHsG1PrePareLength	0x1553	15
PA_TxHsG2SyncLength	0x1554	0x004F
PA_TxHsG2PrePareLength	0x1555	15
PA_TxHsG3SyncLength	0x1556	0x004F
PA_TxHsG3PrePareLength	0x1557	15
PA_TxMk2Extension	0x155A	0
PA_PeerScrambling	0x155B	0
PA_TxSkip	0x155C	0
PA_TxSkipPeriod	0x155D	250
PA_Local_TX_LCC_Enable	0x155E	1
PA_Peer_TX_LCC_Enable	0x155F	1
PA_ConnectedTxDataLanes	0x1561	0
PA_TxGear	0x1568	1
PA_TxTermination	0x1569	0
PA_HSSeries	0x156A	1
PA_PWRMode	0x1571	0x55
PA_ConnectedRxDataLanes	0x1581	0
PA_RxGear	0x1583	1
PA_RxTermination	0x1584	0
PA_Scrambling	0x1585	0
PA_MaxRxPWMGear	0x1586	1
PA_MaxRxHSGear	0x1587	0
PA_PACPReqTimeout	0x1590	63
PA_PACPReqEoBTimeout	0x1591	15
PA_LogicalLaneMap	0x15A1	0xE4E4
PA_SleepNoConfigTime	0x15A2	15
PA_StallNoConfigTime	0x15A3	255
PA_SaveConfigTime	0x15A4	250
PA_RxHSUnterminationCapability	0x15A5	0
PA_RxLSTerminationCapability	0x15A6	0
PA_Hibern8Time	0x15A7	128
PA_TActive	0x15A8	1
PA_LocalVerInfo	0x15A9	0
PA_Granularity	0x15AA	4
PA_MK2ExtensionGuardBand	0x15AB	0
PA_PWRModeUserData[0 to 11]	0x15B0 to 0x15BB	0

Attribute Name	Attribute ID	Reset Value
PA_PACPFrameCount	0x15C0	0
PA_PACPErrorCount	0x15C1	0
PA_PHYTestControl	0x15C2	0
DL Layer Attributes		
DL_TC0TXFCThreshold	0x2040	9
DL_FC0ProtectionTimeOutVal	0x2041	8191
DL_TC0ReplayTimeOutVal	0x2042	65535
DL_AFC0ReqTimeOutVal	0x2043	32767
DL_AFC0CreditThreshold	0x2044	127
DL_TC0OutAckThreshold	0x2045	0
DL_TC1TXFCThreshold	0x2060	9
DL_FC1ProtectionTimeOutVal	0x2061	8191
DL_TC1ReplayTimeOutVal	0x2062	65535
DL_AFC1ReqTimeOutVal	0x2063	32767
DL_AFC1CreditThreshold	0x2064	0
DL_TC1OutAckThreshold	0x2065	0
DL_TxPreemptionCap	0x2000	0
DL_TC0TxMaxSDUSize	0x2001	DL_SYMBOL_MTU
DL_TC0RxInitCreditVal	0x2002	128
DL_TC0TxBufferSize	0x2005	128
DL_PeerTC0Present	0x2046	1
DL_PeerTC0RxInitCreditVal	0x2047	128
DL_TC1TxMaxSDUSize	0x2003	DL_SYMBOL_MTU
DL_TC1RxInitCreditVal	0x2004	0
DL_TC1TxBufferSize	0x2006	0
DL_PeerTC1Present	0x2066	0
DL_PeerTC1RxInitCreditVal	0x2067	0
Network Layer Attributes		
N_DeviceID	0x3000	1
N_DeviceID_valid	0x3001	1
N_TC0TxMaxSDUSize	0x3020	N_MTU
N_TC1TxMaxSDUSize	0x3021	N_MTU
Transport Layer Attributes		
T_PeerDeviceID	0x4021	0
T_PeerCPortID	0x4022	0
T_ConnectionState	0x4020	1
T_TrafficClass	0x4023	0
T_ProtocolID	0x4024	0
T_CPortFlags	0x4025	6
T_TxTokenValue	0x4026	32
T_RxTokenValue	0x4027	32
T_LocalBufferSpace	0x4028	0
T_PeerBufferSpace	0x4029	0
T_CreditsToSend	0x402A	0
T_CPortMode	0x402B	1
T_NumCPorts	0x4000	1
T_NumTestFeatures	0x4001	1
T_TC0TxMaxSDUSize	0x4060	T_MTU
T_TC1TxMaxSDUSize	0x4061	T_MTU
T_TstSrcOn	0x4081	0
T_TstSrcPattern	0x4082	0
T_TstSrcIncrement	0x4083	1
T_TstSrcMessageSize	0x4084	256
T_TstSrcMessageCount	0x4085	256
T_TstSrcInterMessageGap	0x4086	0
T_TstCPortID	0x4080	0
T_TstDstOn	0x40A1	0
T_TstDstErrorDetectionEnable	0x40A2	0

Attribute Name	Attribute ID	Reset Value
T_TstDstPattern	0x40A3	0
T_TstDstIncrement	0x40A4	1
T_TstDstMessageCount	0x40A5	0
T_TstDstMessageOffset	0x40A6	0
T_TstDstMessageSize	0x40A7	256
T_TstDstFCCredits	0x40A8	256
T_TstDstInterFCTokenGap	0x40A9	0
T_TstDstInitialFCCredits	0x40AA	256
T_TstDstErrorCode	0x40AB	0
DME Attributes		
DME_DDBL1_Revision	0x5000	0
DME_DDBL1_Level	0x5001	0
DME_DDBL1_DeviceClass	0x5002	0
DME_DDBL1_ManufacturerID	0x5003	0
DME_DDBL1_ProductID	0x5004	0
DME_DDBL1_Length	0x5005	0
M-PHY Attributes		
TX_HSMODE_Capability	0x0001	1
TX_HSGEAR_Capability	0x0002	3
TX_PWMG0_Capability	0x0003	1
TX_PWMGEAR_Capability	0x0004	7
TX_Amplitude_Capability	0x0005	3
TX_ExternalSYNC_Capability	0x0006	1
TX_HS_Unterminated_LINE_Drive_Capability	0x0007	1
TX_LS_Terminated_LINE_Drive_Capability	0x0008	1
TX_Min_SLEEP_NoConfig_Time_Capability	0x0009	1
TX_Min_STALL_NoConfig_Time_Capability	0x000A	1
TX_Min_SAVE_Config_Time_Capability	0x000B	1
TX_REF_CLOCK_SHARED_Capability	0x000C	1
TX_PHY_MajorMinor_Release_Capability	0x000D	0
TX_PHY_Editorial_Release_Capability	0x000E	1
TX_Hibern8Time_Capability	0x000F	1
TX_Advanced_Granularity_Capability	0x0010	0
TX_Advanced_Hibern8Time_Capability	0x0011	1
TX_HS_Equalizer_Setting_Capability	0x0012	3
TX_MODE	0x0021	1
TX_HSRATE_Series	0x0022	1
TX_HSGEAR	0x0023	1
TX_PWMGEAR	0x0024	1
TX_Amplitude	0x0025	2
TX_HS_SlewRate	0x0026	0
TX_SYNC_Source	0x0027	0
TX_HS_SYNC_LENGTH	0x0028	15
TX_HS_PREPARE_LENGTH	0x0029	15
TX_LS_PREPARE_LENGTH	0x002A	15
TX_HIBERN8_Control	0x002B	1
TX_LCC_Enable	0x002C	1
TX_PWM_BURST_Closure_Extension	0x002D	32
TX_BYPASS_8B10B_Enable	0x002E	0
TX_DRIVER_POLARITY	0x002F	0
TX_HS_Unterminated_LINE_Drive_Enable	0x0030	0
TX_LS_Terminated_LINE_Drive_Enable	0x0031	0
TX_LCC_Sequencer	0x0032	0
TX_Min_ActivateTime	0x0033	15
TX_PWM_G6_G7_SYNC_LENGTH	0x0034	15
TX_Advanced_Granularity_Step	0x0035	0
TX_Advanced_Granularity	0x0036	15
TX_HS_Equalizer_setting	0x0037	0
TX_FSM_State	0x0041	0

Attribute Name	Attribute ID	Reset Value
MC_Output_Amplitude	0x0061	1
MC_HS_Unterminated_Enable	0x0062	0
MC_LS_Terminated_Enable	0x0063	0
MC_HS_Unterminated_LINE_Drive_Enable	0x0064	0
MC_LS_Terminated_LINE_Drive_Enable	0x0065	0
RX_HSMODE_Capability	0x0081	1
RX_HSGEAR_Capability	0x0082	3
RX_PWMG0_Capability	0x0083	1
RX_PWMGEAR_Capability	0x0084	7
RX_HS_Unterminated_Capability	0x0085	1
RX_LS_Terminated_Capability	0x0086	1
RX_Min_SLEEP_NoConfig_Time_Capability	0x0087	1
RX_Min_STALL_NoConfig_Time_Capability	0x0088	1
RX_Min_SAVE_Config_Time_Capability	0x0089	1
RX_REF_CLOCK_SHARED_Capability	0x008A	1
RX_HS_G1_SYNC_LENGTH_Capability	0x008B	0x4F
RX_HS_G1_PREPARE_LENGTH_Capability	0x008C	15
RX_LS_PREPARE_LENGTH_Capability	0x008D	15
RX_PWM_Burst_Closure_Length_Capability	0x008E	31
RX_Min_ActivateTime_Capability	0x008F	1
RX_PHY_MajorMinor_Release_Capability	0x0090	0
RX_PHY_Editorial_Release_Capability	0x0091	1
RX_Hibern8Time_Capability	0x0092	1
RX_PWM_G6_G7_SYNC_LENGTH_Capability	0x0093	0x4F
RX_HS_G2_SYNC_LENGTH_Capability	0x0094	0x4F
RX_HS_G3_SYNC_LENGTH_Capability	0x0095	0x4F
RX_HS_G2_PREPARE_LENGTH_Capability	0x0096	15
RX_HS_G3_PREPARE_LENGTH_Capability	0x0097	15
RX_Advanced_Granularity_Capability	0x0098	0
RX_Advanced_Hibern8Time_Capability	0x0099	1
RX_Advanced_Min_ActivateTime_Capability	0x009A	1
RX_MODE	0x00A1	1
RX_HSRATE_Series	0x00A2	1
RX_HSGEAR	0x00A3	1
RX_PWMGEAR	0x00A4	1
RX_LS_Terminated_Enable	0x00A5	0
RX_HS_Unterminated_Enable	0x00A6	0
RX_Enter_HIBERN8	0x00A7	1
RX_BYPASS_8B10B_Enable	0x00A8	0
RX_Termination_Force_Enable	0x00A9	0
RX_FSM_State	0x00C1	0
OMC_TYPE_Capability	0x00D1	0
MC_HSMODE_Capability	0x00D2	1
MC_HSGEAR_Capability	0x00D3	3
MC_HS_START_TIME_Var_Capability	0x00D4	0
MC_HS_START_TIME_Range_Capability	0x00D5	0
MC_RX_SA_Capability	0x00D6	1
MC_RX_LA_Capability	0x00D7	1
MC_LS_PREPARE_LENGTH	0x00D8	15
MC_PWMG0_Capability	0x00D9	1
MC_PWMGEAR_Capability	0x00DA	7
MC_LS_Terminated_Capability	0x00DB	1
MC_HS_Unterminated_Capability	0x00DC	1
MC_LS_Terminated_LINE_Drive_Capability	0x00DD	1
MC_HS_Unterminated_LINE_Drive_Capability	0x00DE	1
MC_MFG_ID_Part1	0x00DF	0
MC_MFG_ID_Part2	0x00E0	0
MC_PHY_MajorMinor_Release_Capability	0x00E1	0
MC_PHY_Editorial_Release_Capability	0x00E2	1

Attribute Name	Attribute ID	Reset Value
MC_Vendor_Info_Part1	0x00E3	0
MC_Vendor_Info_Part2	0x00E4	0
MC_Vendor_Info_Part3	0x00E5	0
MC_Vendor_Info_Part4	0x00E6	0

Note: the Unipro and M-PHY attributes registers are ‘keepNet’ by default in the file ‘dme_mib.vp’ and the user can use ‘force’ to change its value.

14. UFS Descriptors, Attributes, and Flags

The tables in the subsections below list the descriptors, attributes, and flags in the UFS device.

The user may configure some of the device and unit descriptors by using Query Request UPIU to write the configuration descriptor. However not all descriptor parameters can be modified. Only the configuration descriptor can be written to in order to effect device and unit descriptor changes. The configuration descriptor table lists configurable device and unit descriptor parameters.

Similarly, some attributes and flags are read-only or write-only. Attributes and flags are also accessed by Query Request UPIU. Query Response UPIUs return readable parameter values.

UFS Device Descriptor

The table in this subsection lists the device descriptor in the UFS model.

Table 9: UFS Device Descriptor

Offset	Size	Name	MDV (1)	User Conf. (2)	Description
00h	1	bLength	40h	No	Size of this descriptor
01h	1	bDescriptorType	00h	No	Device Descriptor Type Identifier
02h	1	bDevice	00h	No	Device type 00h: Device Others: Reserved
03h	1	bDeviceClass	00h	No	UFS Device Class 00h: Mass Storage Others: Reserved
04h	1	bDeviceSubClass	JEDEC: 02h Micron: 00h	No	UFS Mass Storage Subclass Bits (0/1) specify as follows: Bit 0: Bootable / Non-Bootable Bit 1: Embedded / Removable Bit 2: Reserved Others: Reserved Examples: 00h: Embedded Bootable 01h: Embedded Non-Bootable 02h: Removable Bootable 03h: Removable Non-Bootable
05h	1	bProtocol	00h	No	Protocol supported by UFS Device 00h: SCSI Others: Reserved
06h	1	bNumberLU	JEDEC: 01h Micron: 00h	Yes (3)	Number of Logical Units bNumberLU does not include well known logical unit
07h	1	bNumberWLU	04h	No	Number of Well know Logical Units
08h	1	bBootEnable	00h	Yes	Boot Enable Indicate whether the device is enabled for boot 00h: Boot feature disabled 01h: Bootable feature enabled Others: Reserved
09h	1	bDescrAccessEn	00h	Yes	Descriptor Access Enable Indicate whether the Device Descriptor can be read after the partial initialization phase of the boot sequence

Offset	Size	Name	MDV (1)	User Conf. (2)	Description
					00h: Device Descriptor access disabled 01h: Device Descriptor access enabled Others: Reserved
0Ah	1	bInitPowerMode	01h	Yes	Initial Power Mode bInitPowerMode defines the Power Mode after device initialization or hardware reset 00h: UFS-Sleep Mode 01h: Active Mode Others: Reserved
0Bh	1	bHighPriorityLUN	7Fh	Yes	High Priority LUN bHighPriorityLUN defines the high priority logic unit Valid values are: from 0 to 7, and 7Fh 7Fh: all logical unit have the same priority
0Ch	1	bSecureRemovalType	00h	Yes	Secure Removal Type 00h: information removed by an erase of the physical memory 01h: information removed by overwriting the addressed locations with a single character followed by an erase 02h: information removed by overwriting the addressed locations with a character, its complement, then a random character 03h: information removed using a vendor define mechanism Others: Reserved
0Dh	1	bSecurityLU	01h	No	Support for security LU 00h: not support 01h: RPMB Others: Reserved
0Eh	1	bBackgroundOpsTermLat	JEDEC: 00h Micron: 05h	No	Background Operations Termination Latency bBackgroundOpsTermLat defines the maximum latency for the termination of ongoing background operations. When the device receives a COMMAND UPIU with a transfer request, the device shall start the data transfer and send a DATA IN UPIU or a RTT UPIU within the latency declared in bBackgroundOpsTermLat. The latency is expressed in units of 10 ms (e.g., 01h=10ms, FFh=2550ms). The latency is undefined if the value of this parameter is zero.
0Fh	1	bInitActiveICCLevel	00h	Yes	Initial Active ICC Level bInitActiveICCLevel defines the bActiveICCLevel value after power on or reset. Valid range from 00h to 0Fh.
10h	2	wSpecVersion	0210h	No	Specification version Bits[15:8] = Major version in BCD format Bits[7:4] = Minor version in BCD format Bits[3:0] = Version suffix in BCD format Example: 3.21=0321h
12h	2	wManufactureDate	0000h	No	Manufacturing Date BCD version of the device manufacturing date, i.e. August 2010 = 0810h
14h	1	iManufacturerName	00h	No	Manufacturer Name Index to the string which contains the Manufacturer Name.
15h	1	iProductName	01h	No	Product Name Index to the string which contains the Product Name.
16h	1	iSerialNumber	02h	No	Serial Number Index to the string which contains the Serial Number.
17h	1	iOemID	03h	No	OEM ID

Offset	Size	Name	MDV (1)	User Conf. (2)	Description
					Index to the string which contains the OEM ID.
18h	2	wManufacturerID	JEDEC: 0000h Micron: 002Ch	No	Manufacturer ID Manufacturer ID as defined in JEDEC standard JEP106 “Standard Manufacturer’s Identification Code”.
1Ah	1	bUD0BaseOffset	10h	No	Unit Descriptor 0 Base Offset Offset of the Unit Descriptor 0 configuration parameters within the Configuration Descriptor.
1Bh	1	bUDConfigPLength	10h	No	Unit Descr. Config. Param. Length Total size of the configurable Unit Descriptor parameters.
1Ch	1	bDeviceRTTCap	02h	No	RTT Capability of device Maximum number of outstanding RTTs supported by device. The minimum value is 2.
1Dh	2	wPeriodicRTCUpdate	0000h	Yes	Frequency and method of Real-Time Clock update. Bits [15:10] Reserved Bits [9] TIME_BASELINE 0h: Time elapsed from the previous dSecondsPassed update. 1h: Absolute time elapsed from January 1st 2010 00:00. NOTE: If the host device has a Real Time Clock it should use TIME BASELINE = ‘1’. If the host device has no Real Time Clock it should use TIME BASELINE = ‘0’. Bits [8:6] TIME_UNIT 1. 0x0 = Undefined 2. 0x1 = Months 3. 0x2 = Weeks 4. 0x3 = Days 5. 0x4 = Hours 6. 0x5 = Minutes 7. 0x6 = Reserved 8. 0x7 = Reserved Bits [5:0] TIME_PERIOD If TIME_UNIT is 0 TIME_PERIOD is ignored and the period between RTC update is not defined. All fields are configurable by the host.
1Fh	1	bUFSFeaturesSupport	01h	No	UFS Features Support This field indicates which features are supported by the device. A feature is supported if the related bit is set to one. bit[0]: Field Firmware Update (FFU) bit[1]: Production State Awareness (PSA) bit[2]: Device Life Span Others: Reserved Bit 0 shall be set to one.
20h	1	bFFUTimeout	JEDEC: 00h Micron: 0Ah	No	Field Firmware Update Timeout The maximum time, in seconds, that access to the device is limited to or not possible through any ports associated due to execution of a WRITE BUFFER command. A value of zero indicates that no timeout is provided.
21h	1	bQueueDepth	40h	No	Queue Depth 0: The device implements the per-LU queueing architecture. 1.. 255: The device implements the shared queueing architecture. This parameter indicates the depth of the shared queue. If bLUQueueDepth>0 for any LU (except RPMB LU), then bQueueDepth shall be 0.
22h	2	wDeviceVersion	0002h	No	Device Version This field provides the device version

Offset	Size	Name	MDV (1)	User Conf. (2)	Description
24h	1	bNumSecureWPArea	20h	No	Number of Secure Write Protect Areas This value specifies the total number of Secure Write Protect Areas supported by the device. The value shall be equal to or greater than bNumberLU and shall not exceed 32 (bNumberLU ≤ bNumSecureWPArea ≤ 32).
25h	4	dPSAMaxDataSize	JEDEC: 000000h Micron: 140000h	No	PSA Maximum Data Size This parameter specifies the maximum amount of data that may be written during the pre-soldering phase of the PSA flow. The value indicates the total amount of data for all logical units with bPSASensitive = 01h. Value expressed in units of 4 Kbyte.
29h	1	bPSAStateTimeout	JEDEC: 00h Micron: 12h	No	PSA State Timeout This parameter specifies the command maximum timeout for a change in bPSAState state. 00h means undefined. Otherwise, the formula to calculate the max timeout value is: Production State Timeout = 100us * 2^bPSAStateTimeout For example: 01h means 100us x 2^1 = 200us 02h means 100us x 2^2 = 400us 17h means 100us x 2^23 = 838.86s
2Ah	1	iProductRevisionLevel	04h	No	Product Revision Level Index to the string which contains the Product Revision Level
2Bh	5	Reserved			Reserved
30h	16	Reserved			Reserved for Unified Memory Extension standard

NOTE 1 The column “MDV” (Manufacturer Default Value) specifies parameter values after device manufacturing. Some parameters may be configured by the user writing the Configuration Descriptor.

NOTE 2 “User Conf.” column specifies which fields can be configured by the user writing the Configuration Descriptor. “Yes” means that the field can be configured. “No” means that the field is a capability of the device and cannot be changed by the user. The desired value shall be set in the equivalent parameter of the Configuration Descriptor.

NOTE 3 bNumberLU field is calculated by the device based on bLUEnable field in the Unit Descriptors.

UFS Configuration Descriptor

The tables in this subsection list the UFS model’s configuration descriptor format and configurable device and unit descriptor parameters.

Table 10: Configuration Descriptor Format

Offset	Description
00h … (B-1)h	Configuration Descriptor header and Device Descriptor configuration parameters
(B)h … (B+L-1)h	Unit Descriptor 0 configurable parameters
(B+L)h … (B+2*L-1)h	Unit Descriptor 1 configurable parameters
…	…
(B+7*L)h	

… (B+8*L-1)h	Unit Descriptor 7 configurable parameters

NOTE 1 B is offset of the Unit Descriptor 0 configurable parameters within the Configuration Descriptor. L is the total size of the configurable Unit Descriptor parameters.

Table 11: Configuration Descriptor Header and Device Descriptor Configurable Parameters

Offset	Size	Name	MDV (1,2)	Description
00h	1	bLength	90h	Size of this descriptor
01h	1	bDescriptorType	01h	Configuration Descriptor Type Identifier
02h	1	Reserved		
03h	1	bBootEnable		Boot Enable Enables to boot feature.
04h	1	bDescrAccessEn		Descriptor Access Enable Enables access to the Device Descriptor after the partial initialization phase of the boot sequence.
05h	1	bInitPowerMode		Initial Power Mode Configures the power mode after device initialization or hardware reset.
06h	1	bHighPriorityLUN		High Priority LUN Configures the high priority logical unit
07h	1	bSecureRemovalType		Secure Removal Type Configures the secure removal type.
08h	1	bInitActiveICCLevel		Initial Active ICC Level Configures the ICC level in Active mode after device initialization or hardware reset.
09h	2	wPeriodRTCUpdate		Frequency and method of Real-Time clock update (see Device Descriptor).
0Bh:0Fh	5	Reserved		

NOTE 1 The column “MDV” (Manufacturer Default Value) specifies parameter values after device manufacturing.
NOTE 2 See “Device Descriptor” table for the default parameters value set by the device manufacturer.

Table 12: Unit Descriptor Configurable Parameters

Offset	Size	Name	Description
00h	1	bLUEnable	Logical Unit Enable
01h	1	bBootLunID	Boot LUN ID
02h	1	bLUWriteProtect	Logical Unit Write Protect
03h	1	bMemoryType	Memory Type
04h	4	dNumAllocUnits	Number of Allocation Units Number of allocation units assigned to the logical unit. The value shall be calculated considering the capacity adjustment factor of the selected memory type.
08h	1	bDataReliablility	Data Reliability
09h	1	bLogicalBlockSize	Logical Block Size
0Ah	1	bProvisioningType	Provisioning Type
0Bh	2	wContextCapabilities	Context Capabilities
0Dh:0Fh	3	Reserved	

UFS Unit Descriptor

The table in this subsection lists the UFS model's unit descriptor. Each LU has its own unit descriptor.

Table 13: Unit Descriptor

Offset	Size	Name	MDV (1)	User Conf. (2)	Description
00h	1	bLength	23h	No	Size of this descriptor
01h	1	bDescriptorType	02h	No	Unit Descriptor Type Identifier
02h	1	bUnitIndex	00h to 07h	No	Unit Index One for each logical unit
03h	1	bLUEnable	LU0-LU1 01h LU2-LU7 00h	Yes	Logical Unit Enable 00h: Logical Unit disabled 01h: Logical Unit enabled Others: Reserved
04h	1	bBootLunID	00h	Yes	Boot LUN ID 00h: Not bootable 01h: Boot LU A 02h: Boot LU B Others: Reserved
05h	1	bLUWriteProtect	00h	Yes	Logical Unit Write Protect 00h: LU not write protected 01h: LU write protected when fPowerOnWPEn=1 02h: LU permanently write protected when fPermanentWPEn=1 03h: Reserved (for UFS Security Extension) Others: Reserved
06h	1	bLUQueueDepth	00h	No	Logical Unit Queue Depth 0 : LU queue not available (shared queuing is used) [1.. 255] : LU queue depth If any bQueueDepth>0, bLUQueueDepth shall be 0.
07h	1	bPSASensitive	JEDEC: 00h Micron: 01h	No	00h: LU is not sensitive to soldering 01h: LU is sensitive to soldering Others: Reserved
08h	1	bMemoryType	00h	Yes	Memory Type bMemoryType defines logical unit memory type. 00h: Normal Memory 01h: System code memory type 02h: Non-Persistent memory type 03h: Enhanced memory type 1 04h: Enhanced memory type 2 05h: Enhanced memory type 3 06h: Enhanced memory type 4 Others: Reserved
09h	1	bDataReliability	00h	Yes	Data Reliability bDataReliability defines the device behavior when a power failure occurs during write operation to the logical unit. 00h: the logical unit is not protected. Logical unit's entire data might be lost as a result of a power failure during a write operation. 01h: logical unit is protected. Logical unit's data is protected against power failure. Others: Reserved
0Ah	1	bLogicalBlockSize	0Ch	Yes	Logical Block Size The size of addressable logical blocks is equal the result of exponentiation with as base the number two and as exponent the bLogicalBlocksize value: 2**bLogicalblocksize (i.e., bLogicalBlockSize=0Ch corresponds to 4 Kbyte Logical Block Size).

Offset	Size	Name	MDV (1)	User Conf. (2)	Description
					Its minimum value is 0Ch, which corresponds to 4 Kbyte.
0Bh	8	qLogicalBlockCount	00h	Yes (3)	Logical Block Count Total number of addressable Logical Blocks in the LU in Logical Block Size unit.
13h	4	dEraseBlocksize	00h (4)	No	Erase Block Size In number of Logical Blocks
17h	1	bProvisioningType	00h	Yes	Provisioning Type 00h: Thin Provisioning is disable (default) 02h: Thin Provisioning is enabled and TPRZ=0 03h: Thin Provisioning is enabled and TPRZ=1 Others: Reserved
18h:1Fh	8	qPhyMemResourceCount	00h	No	Physical Memory Resource Count Total physical memory resource available in the logical unit. Value expressed in units of Logical Block Size.
20h	2	wContextCapabilities	00h	Yes	Bits [3:0]: MaxContextID is the maximum amount of contexts that the LU supports simultaneously. The sum of all MaXContextID must not exceed bMaxContexIDNumber. Bits [6:4]: LARGE_UNIT_MAX_MULTIPLIER_M1 is the highest multiplier that can be configured for Large Unit contexts, minus one. Large Unit contexts may be configured to have a multiplier in the range: 1 ≤ multiplier ≤ (LARGE_UNIT_MAX_MULTIPLIER_M1 + 1). This field is read only. Bit [15:7]: Reserved.
22h	1	bLargeUnitGranularity_M1	00h	No	Granularity of the Large Unit, minus one. Large Unit Granularity = 1MB * (bLargeUnitGranularity_M1 + 1)

NOTE 1 The column “MDV” (Manufacturer Default Value) specifies parameter values after device manufacturing. Some fields may be configured by the user writing the Configuration Descriptor.
NOTE 2 “User Conf.” column specifies which fields can be configured by the user writing the Configuration Descriptor. “Yes” means that the field can be configured. “No” means that the field is a capability of the device and cannot be changed by the user. The desired value shall be set in the equivalent parameter of the Configuration Descriptor.
NOTE 3 qLogicalBlockCount can be configured by setting the dNumAllocUnits parameter of the Configuration Descriptor.
NOTE 4 dEraseBlockSize value is updated automatically by the device after device configuration.
NOTE 5 the default value of bLogicalBlockSize is ‘h0C and it can’t be updated by the configuration descriptor. The user can adjust the value at compile time by changing the local parameters ‘block_bit0’, … ‘block_bit7’ in the file ‘ufs_controller.vp’. The user can also use ‘force *ufs_stack_inst.ufs_device.bLogicalBlockSize0’…*‘force *ufs_stack_inst.ufs_device.bLogicalBlockSize7’ to change its value at runtime after keepNet the signal when compile.*

UFS Attributes

The tables in this subsection list the attributes in the UFS device as well as access property definitions.

Table 14: UFS Attribute Access Properties

Access	Property	Description
Read	Read	The attribute can be read.
	Write only	The attribute cannot be read.
Write	Read only	The attribute cannot be written.
	Write once	The attribute can be written only one time during the device lifetime, the value is kept after power cycle or any type of reset.
	Persistent	The attribute can written multiple times, the value is kept after power cycle or any type of reset event.
	Volatile	The attribute can be written multiple times. The attribute is set to the default value after power cycle or any type of reset event.

	Power on reset	The attribute can be written only one time. The attribute is set to the default value after power cycle or hardware reset event.

Table 15: UFS Attributes

IDN	Name	Access Property	Size	Type(1) #Ind.(2) #Sel.(3)	MDV (4)	Description
00h	bBootLunEn	Read/ Persistent	1 Byte	D	00h	Boot LUN Enable 00h: Boot disabled 01h: Enabled boot from Boot LU A 02h: Enabled boot from Boot LU B All others: Reserved When bBootLunEn = 00h the boot feature is disabled, the device behaves as if bBootEnable would be equal to 0
01h	Reserved					
02h	bCurrentPowerMode	Read only	1 Byte	D	11h see Note 5	Current Power Mode 00h: Idle mode 10h: Pre-Active mode 11h: Active mode 20h: Pre-Sleep mode 22h: UFS-Sleep mode 30h: Pre-PowerDown mode 33h: UFS-PowerDown mode Others: Reserved (5)
03h	bActiveICCLevel	Read/ Persistent	1 Byte	D	00h See Note 6	Active ICC Level bActiveICCLevel defines the maximum current consumption allowed during Active Mode. 00h: Lowest Active ICC level … 0Fh: Highest Active ICC level Others: Reserved Valid range from 00h to 0Fh. (6)
04h	bOutOfOrderDataEn	Read/ Write once	1 Byte	D	00h	Out of Order Data transfer Enable 00h: Out-of-order data transfer is disabled. 01h: Out-of-order data transfer is enabled. Others: Reserved This bit shall have effect only when bDataOrdering = 01h
05h	bBackgroundOpStatus	Read only	1 Byte	D	00h	Background Operations Status Device health status for background operation 00h: Not required 01h: Required, not critical 02h: Required, performance impact 03h: Critical Others: Reserved
06h	bPurgeStatus	Read only	1 Byte	D	00h	Purge Operation Status 00h: Idle (purge operation disabled) 01h: Purge operation in progress 02h: Purge operation stopped prematurely 03h: Purge operation completed successfully 04h: Purge operation failed due to logical unit queue not empty 05h: Purge operation general failure Others: Reserved When the bPurgeStatus is equal to the values 02h, 03h, 04h or 05h, the bPurgeStatus is automatically cleared to 00h (Idle) the first time that it is read.
07h	bMaxDataInSize	Read/ Persistent	1 Byte	D	78h	Maximum Data-In Size Maximum data size in a DATA IN UPIU.

IDN	Name	Access Property	Size	Type(1) #Ind.(2) #Sel.(3)	MDV (4)	Description
					See Note 7	Value expressed in number of 512 Byte units. bMaxDataIn Size shall not exceed the bMaxInBufferSize parameter. bMaxDataInSize = bMaxInBufferSize when the UFS is shipped. This parameter can be written by the host only when all LU task queues are empty. (7,8)
08h	bMaxDataOutSize	Read/ Persistent	1 Byte	D	78h	Maximum Data-Out Size Maximum data size in a DATA OUT UPIU. Value expressed in number of 512 Byte units. bMaxDataOutSize shall not exceed the bMaxOutBufferSize parameter. bMaxDataOutSize = bMaxOutBufferSize when the UFS device is shipped. This parameter can be written by the host only when all LU task queues are empty. (8)
09h	dDynCapNeeded	Read only	4 Bytes	A 8 (LUN) 0	0000 0000h	Dynamic Capacity Needed The amount of physical memory needed to be removed from the physical memory resource pool of the particular logical unit, in units of bOptimalWriteBlockSize. (9)
0Ah	brefClkFreq	Read/ Write once	1 Byte	D	01h	Reference Clock Frequency value 0h: 19.2MHz 1h: 26MHz 2h: 38.4MHz 3h: 52MHz Others: Reserved
0Bh	bConfigDescrLock	Read/ Write once	1 Byte	D	00h	Configuration Descriptor Lock 0h: Configuration Descriptor not locked 1h: Configuration Descriptor locked Others: Reserved
0Ch	bMaxNumOfRTT	Read/ Persistent	1 Byte	D	02h	Maximum current number of outstanding RTTs in device that is allowed. bMaxNumOfRTT shall not exceed the bDeviceRTTCap parameter. This parameter can be written by the host only when all LU task queues are empty.
0Dh	wExceptionEventControl	Read/ Persistent	2 Bytes	D	0000h	Each bit, if set to '1' by the host, enables the assertion of the relevant exception event bit, allowing the device to raise the EVENT_ALERT bit in the Device Information field in the Response UPIU: Bit 0: DYNCAP_EVENT_EN Bit 1: SYSPOOL_EVENT_EN Bit 2: URGENT_BKOPS_EN Bit 3 – 15: Reserved
0Eh	wExceptionEventStatus	Read only	2 Bytes	D	0000h	Each bit represents an exception event. A bit will be set only if the relevant event has occurred (regardless of the wExceptionEventControl status). Bit 0: DYNCAP_NEEDED Bit 1: SYSPOOL_EXHAUSTED Bit 2: URGENT_BKOPS Bit 3 – 15: Reserved
0Fh	dSecondsPassed	Write only	4 Bytes	D	0000 0000h	Bits[31:0]: Seconds passed from TIME BASELINE (see wPeriodicRTCUpdate in Device Descriptor)
10h	wContextConf	Read/ Volatile	2 Bytes	A 8 (LUN) 15 (ID)	0000h	INDEX specifies the LU number, SELECTOR specifies the Context ID within the LU. Valid values are 01h – Fh. **Bit[15:8]: RFU** **Bit[7:6]: Reliability mode** 00h: MODE0 (normal)

IDN	Name	Access Property	Size	Type(1) #Ind.(2) #Sel.(3)	MDV (4)	Description
						01h: MODE1 (non-Large Unit, reliable mode or Large Unit unit-by-unit mode) 02h: MODE2 (Large Unit, one-unit-tail mode) 03h: Reserved **Bit[5:3]: Large Unit multiplier** If Large Unit context is set, this field defines the Large Unit size, else it is ignored **Bit[2]: Large Unit context** 00h: context is not following Large Unit rules 01h: Context follows Large Unit rules **Bit[1:0]: Activation and direction mode** 00b: Context is closed and is no longer active 01b; Context is configured and activated as a write-only context and according to the rest of the bits in this configuration register. 10b: Context is configured and activated as a read-only context and according to the rest of the bits in this configuration register. 11b: Context is configured and activated as a read/write context and according to the rest of the bits in this configuration register.
11h	Obsolete					
12h	Reserved					Reserved for Unified Memory Extension specification
13h	Reserved					Reserved for Unified Memory Extension specification
14h	bDeviceFFUStatus	Read Only	1 Byte	D	00h	Device FFU Status 00h: No information 01h: Successful microcode update 02h: Microcode corruption error 03h: Internal error 04h: Microcode version mismatch 05h-FEh: Reserved 0FFh: General Error
15h	bPSAState	Read/ Persistent	1 Byte	D	00h	00h: 'Off'. PSA feature is off. 01h: 'Pre-soldering'. PSA feature is on, device is in the pre-soldering state. 02h: 'Loading Complete' PSA feature is on. The host will set to this value after the host finished writing data during pre-soldering state. 03h: 'Soldered'. PSA feature is no longer available. Set by the Device to indicate it is in postsoldering state. This attribute unchangeable after it is in 'Soldered' state.
16h	dPSADataSize	Read/ Persistent	8 Bytes	D	00 … 00h	The amount of data that the host plans to load to all logical units with bPSASensitive set to 1.

NOTE 1 The type "D" identifies a device level attribute, while the type "A" identifies an array of attributes. If Type = "D", the attribute is addressed setting INDEX = 00h and SELECTOR = 00h.
NOTE 2 For array of attributes, "# Ind." specifies the amount of valid values for the INDEX field in QUERY REQUEST/RESPONSE UPIU. If # Ind = 0, the attribute is addressed setting INDEX = 00h.
NOTE 3 For array of attributes, "# Sel." specifies the amount of valid values for the SELECTOR field in QUERY REQUEST/RESPONSE UPIU. If # Sel = 0, the attribute is addressed setting SELECTOR = 00h.
NOTE 4 The column "MDV" (Manufacturer Default Value) specifies attribute values after device manufacturing.
NOTE 5 bCurrentPowerMode value after device initialization may be: 20h (Pre-Sleep mode) or 22h (UFS-Sleep mode) if bInitPowerMode = 00h, or 11h (Active Mode) if bInitPowerMode = 01h.
NOTE 6 bActiveICCLevel after power on or reset is equal bInitActiveICCLevel parameter value included in the Device Descriptor. bInitActiveICCLevel is equal to 00h after device manufacturing and can be configured during system integration.
NOTE 7 bMaxDataInSize = bMaxInBufferSize when the UFS device is shipped.
NOTE 8 If the host attempts to write this Attribute when there is at least one logical unit with command queue not empty, the operation shall fail, and Response field in the QUERY RESPONSE UPIU shall be set to FFh ("General failure").
NOTE 9 dDynCapNeeded is composed by eight elements, one for each logical unit. The desired element shall be selected assigning the LUN to INDEX field of QUERY REQUEST UPIU.
NOTE 11 bDeviceFFUStatus value is kept after power cycle, hardware reset or any other type of reset. This attribute may change value when a microcode activation event occurs.

UFS Flags

The tables in this subsection below list the flags in the UFS device as well as access property definitions.

Table 16: UFS Flags Access Properties

Access	Property	Description
Read	Read	The flag can be read.
	Write only	The flag cannot be read.
Write	Read only	The flag cannot be written.
	Write once	The flag can be written only one time during the device lifetime, the value is kept after power cycle or any type of reset event.
	Persistent	The flag can be written multiple times, the value is kept after power cycle or any type of reset event.
	Volatile	The flag can be written multiple times. The flag is set to the default value after power cycle or any type of reset event.
	Set only	The flag can be only be set one (or zero) by the host and cleared to zero (or one) by the device. The flag is cleared after power cycle or any type of reset event.
	Power on reset	The flag can be written only one time. The flag is set to the default value after power cycle or hardware reset event.

Table 17: UFS Flags

IDN	Name	Type	Type(1) # Ind.(2) # Sel.(3)	Default	Description
00h	Reserved				
01h	fDeviceInit	Read/ Set only	D	0	Device Initialization Host set fDeviceInit flag to initiate device initialization after boot process is completed. Device resets flag when device initialization is completed. 0b: Device initialization completed or not started yet. 1b: Device initialization in progress.
02h	fPermanentWPEn	Read/ Write once	D	0	Permanent Write Protection Enable fPermanentWPEn enables permanent write protection on all logical units configured as permanent protected; it cannot be toggled or cleared once it is set. 00h: Permanent write protection disabled 01h: Permanent write protection enabled
03h	fPowerOnWPEn	Read/ Power on reset	D	0	Power On Write Protection Enable fPowerOnWPEn enables the write protection on all logical units configured as power on write protected. If fPowerOnWPEn is equal to one and the device receive a Query Request to clear or toggle this flag, the Query Request shall fail and Response field shall be set to “F8h (Parameter already written). The device shall set fPowerOnWPEn to zero in the event of power cycle or hardware reset. 0b: Power on write protection disabled. 1b; Power on write protection enabled.
04h	fBackgroundOpsEn	Read/ Volatile	D	1	Background Operations Enable 0b: Device is not permitted to run background operations. 1b: Device is permitted to run background operations.
05h	fDeviceLifeSpanModeEn	Read/ Volatile	D	0	Device Life Span Mode 0b: Device Life Span Mode is disabled. 1b: Device Life Span Mode is enabled.

IDN	Name	Type	Type(1) # Ind.(2) # Sel.(3)	Default	Description
06h	fPurgeEnable	Write only/ Volatile	D	0	Purge Enable 0b: Purge operation is disabled. 1b: Purge operation is enable. This flag shall only be set when the command queue of all logical units are empty and the bPurgeStatus is 00h (Idle). fPurgeEnable is automatically cleared by the UFS device when the operation completes or an error condition occurs. fPurgeEnable can be cleared by the host to interrupt an ongoing purge operation.
07h	Reserved				Reserved
08h	fPhyResourceRemoval	Read/ Persistent	D	0	Physical Resource Removal The host sets this flag to one to indicate that the dynamic capacity operation shall commence upon device EndPointReset or hardware reset. The device shall reset this flag to zero after completion of dynamic capacity operation. The host cannot reset this flag.
09h	fBusyRTC	Read only	D	0	Busy Real Time Clock 0b: Device is not executing internal operation related to RTC 1b: Device is executing internal operation related to RTC When this flag is set to one, it is recommended for the host to not send commands to the device.
0Ah	Reserved				Reserved for Unified Memory Extension specification
0Bh	fPermanentlyDisableFwUpdate	Read/ Write once	D	0	Permanently Disable Firmware Update 0b: The UFS device firmware may be modified 1b: The UFS device shall permanently disallow future firmware updates to the UFS device

NOTE 1 The type “D” identifies a device level flag, while the type “A” identifies an array of flags. If Type = “D”, the flag is addressed setting INDEX = 00h and SELECTOR = 00h.
NOTE 2 For array of flags, ““# Ind.” specifies the amount of valid values for the INDEX field in QUERY REQUEST/RESPONSE UPIU. If # Ind = 0, the flag is addressed setting INDEX = 00h.
NOTE 3 For array of flags, ““# Sel.” specifies the amount of valid values for the SELECTOR field in QUERY REQUEST/RESPONSE UPIU. If # Sel = 0, the flag is addressed setting SELECTOR = 00h.

Other descriptors such as geometry and string descriptors are not configurable. Please refer to UFS spec for more details on those items.

15. UFS Commands

Currently the UFS model accepts the following subset of UFS SCSI commands:

INQUIRY
READ (6)
READ (10)
READ (16)
READ CAPACITY (10)
READ CAPACITY (16)
REPORT LUNS
REQUEST SENSE
START STOP UNIT
TEST UNIT READY
UNMAP
VERIFY (10)
WRITE (6)
WRITE (10)
WRITE (16)
FORMAT UNIT
PRE-FETCH (10)
PRE-FETCH (16)
SEND DIAGNOSTIC
SYNCHRONIZE CACHE (10)
SYNCHRONIZE CACHE (16)
READ BUFFER
WRITE BUFFER
MODE SELECT (10)
MODE SENSE (10)
SECURITY PROTOCOL IN
SECURITY PROTOCOL OUT

16. UniPro Initialization Sequence

The Unipro initialization sequence is comprised of two portions: a Physical Adapter (PA) layer link startup sequence and Data Link (DL) layer link startup sequence. The controller (UniPro Master) should follow the sequences described below to ensure that initialization is completed. The initialization sequence is implemented based on the UniPro1.6 version of the specification.

16.1.Physical Adapter (PA) Layer Link Startup Sequence

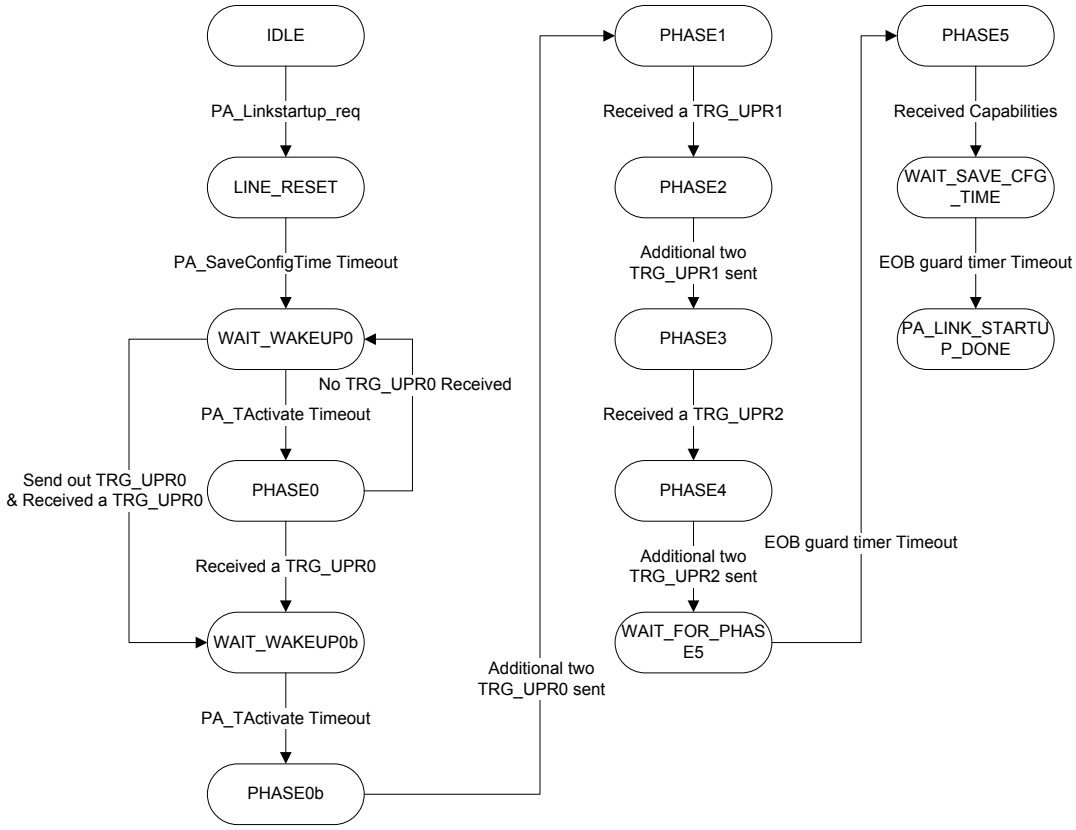


Figure 3: Physical Adapter (PA) Link Startup Sequence

Table 18: PA Link Startup Sequence State Description

STATE	DESCRIPTION
IDLE	Wait PA_Linkstartup_req to start PA Link Startup Sequence
LINE_RESET	Issue a line reset to M-PHY
WAIT_WAKEUP0	Wait for a period of PA_TActivate to wake-up M-RXs from HIBERN8
PHASE0	Begin burst and send a TRG_UPR0
WAIT_WAKEUP0b	Wait for a period of PA_TActivate
PHASE0b	Send two additional TRG_UPR0s
PHASE1	Send a TRG_UPR1
PHASE2	After received TRG_UPR1, send two additional TRG_UPR1
PHASE3	Send a TRG_UPR2
PHASE4	After received TRG_UPR2, send two additional TRG_UPR2 and close burst
WAIT_FOR_PHASE5	Wait EOB guard time before begin a new burst
PHASE5	Send PACP_CAP_EXT1_ind and PACP_CAP_ind
WAIT_SAVE_CFG_TIME	Wait PA_SaveConfigTime
PA_LINK_STARTUP_DONE	PA Link Startup finished

16.2.Data Link (DL) layer link startup sequence

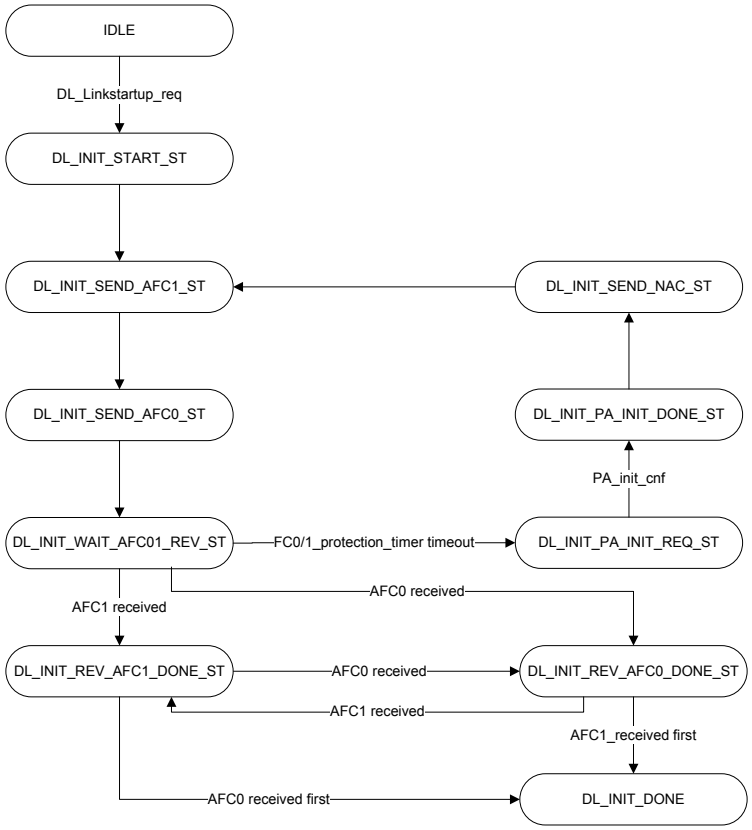


Figure 4: Data Link (DL) Layer Link Startup Sequence

Table 19: Data Link (DL) Layer Link Startup Sequence State Description

STATE	DESCRIPTION
IDLE	Wait DL_Linkstartup_req to start DL Link Startup Sequence
DL_INIT_START_ST	Start DL layer Link Startup sequence
DL_INIT_SEND_AFC1_ST	Send AFC1 frame with Rreq=1
DL_INIT_SEND_AFC0_ST	Send AFC0 frame with Rreq=1
DL_INIT_WAIT_AFC01_REV_ST	Wait AFC0/1 reception
DL_INIT_REV_AFC1_DONE_ST	AFC1 frame received
DL_INIT_REV_AFC0_DONE_ST	AFC0 frame received
DL_INIT_PA_INIT_REQ_ST	After AFC1 and AFC0 sent out and FC0/1_protection_timer timeout without received AFC1/0, start PA_init sequence
DL_INIT_PA_INIT_DONE_ST	PA init sequence done
DL_INIT_SEND_NAC_ST	Send NAC frame

17. UFS Boot Code Download

After the UniPro initialization sequence completes, the UFS host may initiate the following boot code download process. This process can be executed only if the bBootEnable parameter is set in the device descriptor.

1. UFS host sends NOP OUT UPIU.
2. UFS device responds with NOP IN UPIU.
3. UFS host sends QUERY REQUEST UPIU to read bBootEnable parameter.
4. UFS device responds with QUERY RESPONSE UPIU.
5. UFS host issues a TEST UNIT READY command to the Boot well known logical unit.
6. Boot well known logical unit returns RESPONSE UPIU.
7. UFS host issues one or more SCSI READ commands(s).
8. Boot well known logical unit returns boot code in one or more DATA IN UPIU(s).
9. UFS host sets the fDeviceInit flag to “01h” to enable device to complete initialization.
10. UFS device resets the fDeviceInit flag after initialization is complete.
11. UFS host polls the fDeviceInit flag to check for completion of process.
12. When the fDeviceInit flag is reset, the device is ready for normal operation.

18. UniPro Power Mode Change Sequence

The following Figure 5: UniPro Layer Power Mode Change Sequence shows the UniPro PA layer power mode change sequence. After reset, the default power mode is PWM-G1 and only lane0 is active. The user can use the power mode change sequence to change the power mode to HS mode and used lanes from 1 to 2, also the user can enable scrambling after UniPro is in Linkup state. Please refer to UniPro v1.6 specification for additional detail.

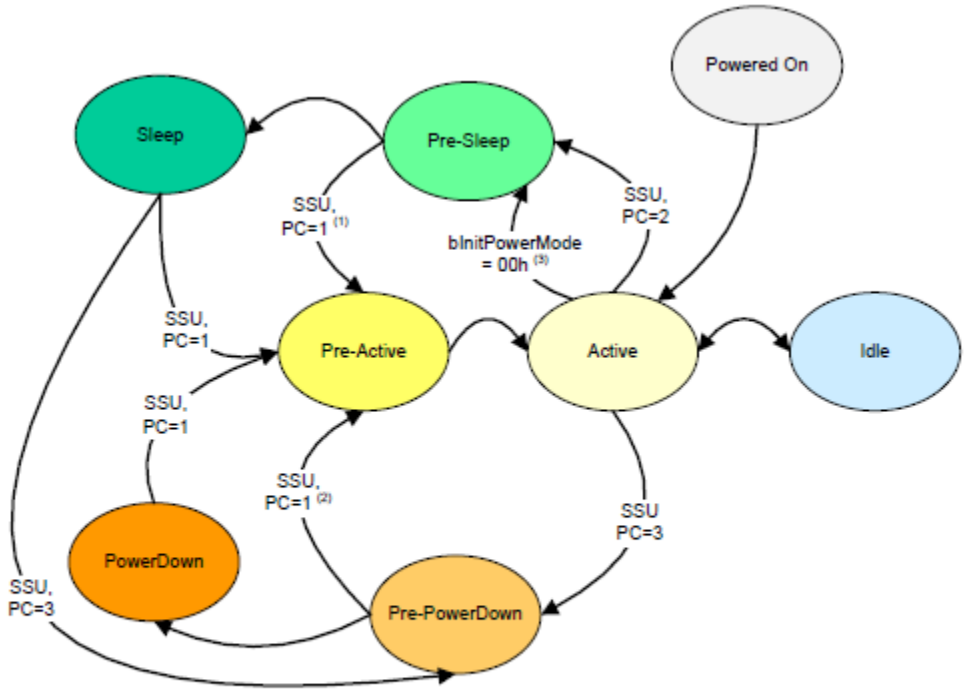
Figure 5: UniPro Layer Power Mode Change Sequence

Table 20: UniPro Layer Power Mode Change Sequence State Description

STATE	DESCRIPTION
IDLE	Wait PACP_PWR_REQ frame reception to start PA Power Mode Change Sequence
PA_WAIT_PA_LM_PWR_MODE_RSP_ST	Send pa_lm_pwr_mode_ind to DME and Wait pa_lm_pwr_mode_rsp after reveived PACP_PWR_REQ frame and tx_mode and rx_mode is not 3
PA_DL_PAUSE_IND_REMOTE_ST	Send pa_dl_pause after received pa_lm_pwr_mode_rsp
PA_PWR_CFG_MOD_PEER_ST	Config M-PHY attribute after pa_dl_pause_rsp
PA_PWR_BURST_START_REMOTE_ST	Start a new burst when burst is inactive
PACP_PWR_CNF_ST	Send PACP_PWR_cnf frame
PACP_PWR_WAIT_PEER_EOB_REMOTE_ST	Wait EOB reception
PACP_PWR_BURST_END_REMOTE_ST	End burst after received peer EOB
PA_PWR_CHG_REMOTE_DONE_ST	PA power mode change sequence finished

19. UFS Power Modes

There following diagram, Figure 6: Power Mode State Machine, from the UFS spec shows the relationship amongst the different UFS power modes.



(1) This transition may occur only if the SSU command that caused the transition to Pre-Sleep had IMMED set to one.
(2) This transition may occur only if the SSU command that caused the transition to Pre-PowerDown had IMMED set to one.
(3) This automatic transition shall occur at the end of device initialization if bInitPowerMode = 00h.

Figure 6: Power Mode State Machine

20. UFS Model Clocks

There are a total of 7 input clocks which the user needs to drive. The table below, Table 21, lists the clock names and their functionality in the model. The following table, Table 22, shows an example frequency of these clocks with HS Gear-2A Mode.

Table 21: UFS Model Clocks

CLOCK NAME	DECRIPTION
sys_clk	The sys_clk is used for model core function. Its frequency should set same to the rmmi_tx_symbol_clk with 2-Symbol RMMI Data Width.
unipro_timer_clk	The unipro_timer_clk is used as the UniPro timer clock, and its frequency is 125MHz. The parameter TCK is the cycles of the clock and is set to 8.
rmmi_tx_symbol_clk	The rmmi_tx_symbol_clk is used for RMMI TX symbol transmit. Its frequency depends on Burst Mode (HS or LS) and Gears and RMMI TX Data Width.
cfg_clk	The cfg_clk is used for UniPro and M-PHY attribute configuration and its frequency is set to be the same as sys_clk.
t_clk	The t_clk is used for CPort data transmission and its frequency is set same to sys_clk.
RX_SymbolClk_0	The RX_SymbolClk_0 is the RMMI Lane0 RX symbol clock. Its frequency depends on Burst Mode (HS or LS) and Gears and RMMI RX Data Width.
RX_SymbolClk_1	The RX_SymbolClk_1 is the RMMI Lane1 RX symbol clock. Its frequency is set to be the same as RX_SymbolClk_0.

Table 22: Example for UFS Model Clocks with High Speed (HS) Gear-2A Mode

CLOCK NAME	RMMI Symbol Width		
	1	2	4
sys_clk	125MHz	125MHz	125MHz
unipro_timer_clk	125MHz	125MHz	125MHz
rmmi_tx_symbol_clk	250MHz	125MHz	62.5MHz
cfg_clk	125MHz	125MHz	125MHz
t_clk	125MHz	125MHz	125MHz
RX_SymbolClk_1	250MHz	125MHz	62.5MHz
RX_SymbolClk_1	250MHz	125MHz	62.5MHz

The clocks which have a frequency that is related to the Burst Mode (HS or LS) and Gears can change its frequency according to the currently working Burst Mode and Gears after finishing a PA power mode change sequence (The I/O signal “dme_pwrmode_chg_done” goes high). The UniPro is working in PWM-G1 mode default after reset. The UFS model will also work when the user does not change the clock frequency after performing a power mode change.

21. Limitations

1. No real cache implemented, no read/write performance change with/without cache configured.
2. For RPMB, the Message Authentication Code (MAC) calculation (HMAC SHA-256) and check are not supported.
3. Features that are not supported are indicated in Table 2.

22. 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. The *_filelist.f, also included in the release, lists all the protected RTL files that are shown in Table 22. All the RTL files reside in the release folder “ufs_common” and “*Vendor*/*_ufs_model”.

By default, the UFS Model has the Verilog macro MMP_LU0_ADDR_BITS set to a value of ‘28’, as in “`define MMP_LU0_ADDR_BITS 28”, in order to open the LU0 memory source and function. By default, both LU0 and LU1 are enabled. The user can use the IXCOM option “+define+MMP_LU2_ADDR_BITS=28” to enable LU2. The available Logic Unit range is from LU0 to LU7.

```
ixcom -64bit +sv –ua \
    +dut+jedec_ufs_stack \
    +libext+.v+.vp \
    -incdir ../ufs_common \
    -f ./jedec_ufs_filelist.f \
    ./jedec_ufs_model/jedec_ufs_stack.v \
    -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 an example for Palladium classic ICE synthesis:

1)

```
hdlInputFile -i ../ufs_common
hdlInputFile -f ./jedec_ufs_filelist.f
hdlInputFile -add ./jedec_ufs_model/jedec_ufs_stack.v
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f Verilog jedec_ufs_stack.vg
hdlSynthesize -memory -keepVhdlCase -keepRtlSymbol -keepAllFlipFlop
jedec_ufs_stack
……
```

2)

```
vavlog -IncDir ./ufs_common -
f ./jedec_ufs_filelist.f ./jedec_ufs_model/jedec_ufs_stack.v
```

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

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.

Table 23: UFS Model RTL File List

FILE NAME	DESCRIPTIONS
unipro_macro_defines.v	Define Configurations and Constant values for UniPro
unipro_afifo.vp	Asynchronous FIFO
unipro_sfifo.vp	Synchronous FIFO
pa_boot_procedure.vp	PA layer boot procedure
pa_decomposition.vp	Parse PACP frames
pa_initialization.vp	PA Layer initialization
pa_lane_descrambling.vp	Descrambling for each RX Lane data symbols
pa_lane_scrambling.vp	Scrambling for each TX Lane data symbols
pa_multi_lane_deskew.vp	Multi-lanes deskew alignment
pa_power_management.vp	PA Layer power mode change procedure
pa_rx_buffer.vp	Transfer RMMI RX data symbols width from 8bits/16bits/32bits to core processing 16bits
pa_tx_buffer.vp	Transfer RMMI TX data symbols width from core processing 16bits to 8bits/16bits/32bits
pa_tx_schedule.vp	Schedule PACP frame transmission
mmp_unipro_pa_top.vp	The top module of PA Layer
dl_decomposition.vp	Parse DL layer RX frames
dl_flow_ctrl.vp	Processing DL Layer Flow Control function
dl_initialization.vp	DL Layer initial sequence
dl_rx_buf.vp	Store DL Layer RX data frames
dl_rx_queue.vp	Store DL Layer RX data frame's pointer info into queue
dl_tx_data_buf.vp	Store DL Layer TX data frames
dl_tx_data_queue.vp	Store DL Layer TX data frame's pointer info into queue
dl_tx_queue.vp	Store DL Layer TX Control frame's pointer info into queue
dl_tx_schedule.vp	Schedule DL Layer data and Control frames transmission
mmp_unipro_dl_top.vp	The top module of DL Layer

FILE NAME	DESCRIPTIONS
mmp_unipro_nl_top.vp	Network Layer function processing
tl_err_check.vp	Transport Layer error checking
tl_reassembling.vp	Reassemble packets into message
tl_segmentation.vp	Segment message into packets
tl_test_cp_adapter.vp	Transport Layer Test Feature Cport adapter
tl_test_traffic_analyzer.vp	Transport Layer Test Feature message analyzer
tl_test_traffic_generator.vp	Transport Layer Test Feature message generator
mmp_unipro_tl_top.vp	The top module of Transport Layer
mmp_unipro_dme_ctrl.vp	The control functions of DME Layer
mmp_unipro_dme_mib.vp	UniPro MIB SET and GET function
mmp_unipro_dme_top.vp	The top module of DME Layer
mmp_unipro_top.vp	The top module of UniPro stack
*_ufs_controller.vp	The UFS commands processing and UFS device controller functions. It is in the directory vendor/*_ufs_model.
mmp_ufs_gen_mem.vp	Used for large memory configurations to generate multiple arrays
mmp_ufs_submem.vp	Used for large memory configurations to generate multiple arrays
*_ufs_stack.v	The top module of UFS model, which includes UFS device and UniPro. It is in the directory vendor/*_ufs_model.
*_ufs_card.v	The top module of UFS card model. There is no *_ufs_stack.v for the UFS card model. It is in the directory vendor/*_ufscard_model.

Table 24: UFS Model Preload Data File List

DATA FILE NAME	DESCRIPTIONS
*_attr.dat	For Attributes
*_cfg_desc.dat	For Configuration Descriptor
*_dev_desc.dat	For Device Descriptor
*_ds0_inq.dat	For Inquiry Response Data
*_ds0_desc.dat	For Unit Descriptor - LU0
*_ds1_desc.dat	For Unit Descriptor - LU1
*_ds2_desc.dat	For Unit Descriptor - LU2
*_ds3_desc.dat	For Unit Descriptor - LU3
*_ds4_desc.dat	For Unit Descriptor - LU4
*_ds5_desc.dat	For Unit Descriptor - LU5
*_ds6_desc.dat	For Unit Descriptor - LU6
*_ds7_desc.dat	For Unit Descriptor - LU7
*_flag.dat	For Flags
*_geo_desc.dat	For Geometry Descriptor
*_int_desc.dat	For Interconnect Descriptor
*_mfr_desc.dat	For Manufacturer Descriptor
*_oem_desc.dat	For OEM ID String Descriptor
*_pow_desc.dat	For Power Parameters Descriptor
*_prd_desc.dat	For Product Name String Descriptor
*_rpm_desc.dat	For RPMB Unit Descriptor
*_rpm_inq.dat	For RPMB Inquiry Response Data
*_ser_desc.dat	For Serial Number String Descriptor
*_dvh_desc.dat	For Device Health Descriptor
*_rev_desc.dat	For Product Revision Level String Descriptor

The user should preload all the initial dat files as listed in Table 24 at very beginning after download the UFS model into Palladium. These dat files are used for the UFS model descriptors/attributes/flags initialization. As described in the UFS spec, some flags/attributes values, such as ‘bBootLunEn’, are kept after power cycle or any type of reset. These values will not change after the first reset (UFS port signal ‘sys_reset’) in the UFS model.
Below is a scripting example snippet showing the preloading of the dat files into the jedec UFS Model. The user can change the UFS Model instance name “ufs_stack_inst” and the dat files in vendor/*_ufs_dat directory as needed.

```
……
xc xt0 zt0 run
xc memory -load %readmemh ufs_stack_inst.ufs_device.dev_desc_d -file jedec_ufs_dev_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.cfg_desc_d -file jedec_ufs_cfg_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds0_desc_d -file jedec_ufs_ds0_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds1_desc_d -file jedec_ufs_ds1_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds2_desc_d -file jedec_ufs_ds2_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds3_desc_d -file jedec_ufs_ds3_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds4_desc_d -file jedec_ufs_ds4_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds5_desc_d -file jedec_ufs_ds5_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds6_desc_d -file jedec_ufs_ds6_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds7_desc_d -file jedec_ufs_ds7_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.geo_desc_d -file jedec_ufs_geo_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.dm_attr_array_d -file jedec_ufs_attr.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.dm_flag_array_d -file jedec_ufs_flag.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.rpm_desc_d -file jedec_ufs_rpm_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.pow_desc_d -file jedec_ufs_pow_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.int_desc_d -file jedec_ufs_int_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.mfr_desc_d -file jedec_ufs_mfr_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.prd_desc_d -file jedec_ufs_prd_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.oem_desc_d -file jedec_ufs_oem_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ser_desc_d -file jedec_ufs_ser_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.dvh_desc_d -file jedec_ufs_dvh_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.rev_desc_d -file jedec_ufs_rev_desc.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds0_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds1_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds2_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds3_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds4_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds5_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds6_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ds7_inq_data -file jedec_ufs_ds0_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.rl_inq_data -file jedec_ufs_rpm_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.ud_inq_data -file jedec_ufs_rpm_inq.dat
xc memory -load %readmemh ufs_stack_inst.ufs_device.rpm_inq_data -file jedec_ufs_rpm_inq.dat
……
```

23. Large Memory Support

MMP model capacity has been limited to 4GBytes (32 bits data width) due to the current 30 bit address width limitation in IXCOM. To work around this constraint, a multiple core memory array generator (mmp_ufs_gen_mem.vp) has been incorporated into these large sized memory models that split larger core memory arrays into multiple 4GBytes arrays. The user needs to define the macro 'MMP_LG_MEM_LUx' to enable this large memory support if the macro 'MMP_LUx_ADDR_BITS' is set to larger than 30.

23.1. Preloading Multiple Arrays

With a single array of 30 bits or less the address mapping and hierarchical path is described in Section 12 Address Mapping.

In scenarios with array address space (MMP_LUx_ADDR_BITS) 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:

ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_0_.u1 .memcore
address 0 1G-1 (32bits data width)
ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_1_.u1 .memcore
address 1G 2G-1 (32bits data width)
ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_2_.u1 .memcore
address 2G 3G-1 (32bits data width)
ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_3_.u1 .memcore
address 3G 4G-1 (32bits data width)

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 ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_0_.u1.memcore -file
mem0.dat

memory -load %readmemh ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_1_.u1 .memcore -file
mem1.dat

memory -load %readmemh ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_2_.u1 .memcore -file
mem2.dat

memory -load %readmemh ufs_stack_inst.ufs_device.lu0_mem.\multiple.array_3_.u1 .memcore -file
mem3.dat
```

24. UFS Debugging

The UFS model is complex and therefore the associated problem of debugging issues is likewise complex. Below is a list of recommended debugging techniques and tips that the user may use in isolating a problem.

- For issues that are may not be UFS specific please review the *Memory Model Portfolio FAQ for All Models User Guide.*

- **Waveform debugging:** signal and sequences
  - Check that the clock and reset signals are correctly driven.
  - Check that the PA layer link startup sequence finished successfully.
    Related signals for checking: *i_pa_lm_linksup_req* and *o_pa_lm_linksup_cnf* and *pa_boot_cstate* in module *pa_boot_procedure*.
  - Check that the DL layer Link Startup sequence finished successfully.
    Related signals for checking: *i_dl_lm_linkstartup_req* and *o_dl_lm_linkstartup_cnf* and *dl_init_cstate* in module *dl_initialization*.
  - Check that the PA power mode change sequence finished successfully if needed.
    Related signals for checking: *pacp_pwr_req_received* and *pa_lm_pwr_mode_changed_remote_ind* and *pa_pwr_change_remote_cstate* in module *pa_power_management*.
  - Check that the UniPro state is in LinkUp state when sending packets.
    Related signals for check: *unipro_working_cstate* in module *dme_ctrl*.
  - Check if the Cport packet is correct or not.
    Related signals for check: *cport tx* and *cport_rx* signals in module mmp_*unipro_top*.

- **Golden waveform:** A package with a reference waveform is available which shows the following command sequence:
  - PA Link startup sequence, refer to UG and Unipro v1.6 spec for detail
  - DL Link startup sequence, refer to UG and Unipro v1.6 spec for detail
  - PA PowerMode Change sequence, refer to UG and Unipro v1.6 spec for detail
  - UFS NOP OUT and NOP IN command
  - UFS Write6 and Read6 command
  - UFS Write Descriptor and Read Descriptor command
  - UFS Write Attribute and Read Attribute command
  - UFS Read Capacity command, Inquiry command, Request Sense command
  - UFS Start Stop Unit command, Test Unit Ready command
  - UFS Verify10 command, Report Luns command
  - UFS Write Buffer command, Read Buffer command
  - UFS PreFetch10 command, Synchronize Cache10 command
  - UFS Format Unit command, Send Diagnostic command
  - UFS Unmap command
  - UFS Mode Select and Mode Sense command
  - UFS RPMB Operation – Authentication Key Programming
  - UFS RPMB Operation – Read Counter Value
  - UFS RPMB Operation – Authenticated Data Write
  - UFS RPMB Operation – Authenticated Data Read

- **Debug Display:** The Palladium UFS 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. Some of the debug information displayed when the Debug Display macro options are enabled includes:

    - UniPro states
    - PA layer Link Startup sequence
    - DL layer Link Startup sequence
    - PA power mode change sequence
    - PACP frame send and receive information
    - DL DATA/AFC/NAC frame send and receive information
    - DL credit update information
    - Attribute Set/Get information
    - CRC error /Invalid field error
    - UPIU transfer sequence
    - Logical Unit data transfer sequence
    - Boot code transfer sequence
    - Logical Unit command processing states
    - Task Management function processing states
    - Query Request processing states
    - Report LUNS WLUN command processing states
    - UFS Device WLUN command processing states
    - UFS RPMB WLUN command processing states
    - Error Response information

Revision History

The following table shows the revision history for this document

Date	Version	Revision
December 2014	1.0	Initial release
July 2015	1.1	Update Cadence naming on front page
September 2015	1.2	Add vavlog/vaelab script for ice compile and options note for synthesis.
January 2016	1.3	Update for Palladium-Z1 and VXE
March 2016	1.4	Add main memory width and path information
March 2016	1.5	Update the address mapping and bLogicalBlockSize information
June 2016	1.6	Remove BETA watermark for MR release of UFS 2.0 parts
June 2016	1.7	Remove hyphens from Palladium platform names.
August 2016	1.8	Update for UFS2.1 features for Phase-I
September 2016	1.9	Update for DME_PEER_SET/GET feature support
October 2016	2.0	Add large memory configuration support
February 2017	2.1	Add notes about preloading dat files at very beginning after download the UFS model
March 2017	2.2	Update for UFS2.1 features for Phase-II Update for UFS Card information Change model name to lower case
August 2017	2.3	Update macro with the prefix MMP_
October 2017	2.4	Update module name to be much unique; Remove BETA watermark. Move UFS 2.1 to non-BETA (MR) level
January 2018	2.5	Modify header and footer
July 2018	2.6	Update for new utility library Update the UFS instantiation information for RMMI interface connection.