FPGA Design Document

Rev: A0-09

15 Feb 2015

FPGA design document

For

FSLU_NVME

Revision History:

Date	Rev No.	Description	By
10-July-2015	A0-00	First Draft version	Abilash Joseph
14-July-2015	A0-01	Diagrams updated	Joseph George
12-August-2015	A0-02	Control Status Registers Updated	Nikhil Kannanthra
13-August-2015	A0-03	NVME Register implementation details updated	Irshad pp
14-August-2015	A0-04	Interrupt Mask Register and Descriptor Table Size Register Updated	Nikhil Kannanthra
24-August-2015	A0-05	Updated as per the changes requested by software team.	Nikhil Kannanthra
25-August-2015	A0-06	Updated the Descriptor structure	Nikhil Kannanthra
30-October-2015	A0-07	Design update after bring-up	Nikhil Kannanthra
04-November-2015	A0-08	Appendix Updated	Antony Mathew
15-February-2015	A0-09	Updated Descriptor structure and Descriptor OCM BAR Mapping	Shivam Porwal

Table of Contents

List of Figures

List of Tables

1 High level hardware block diagram

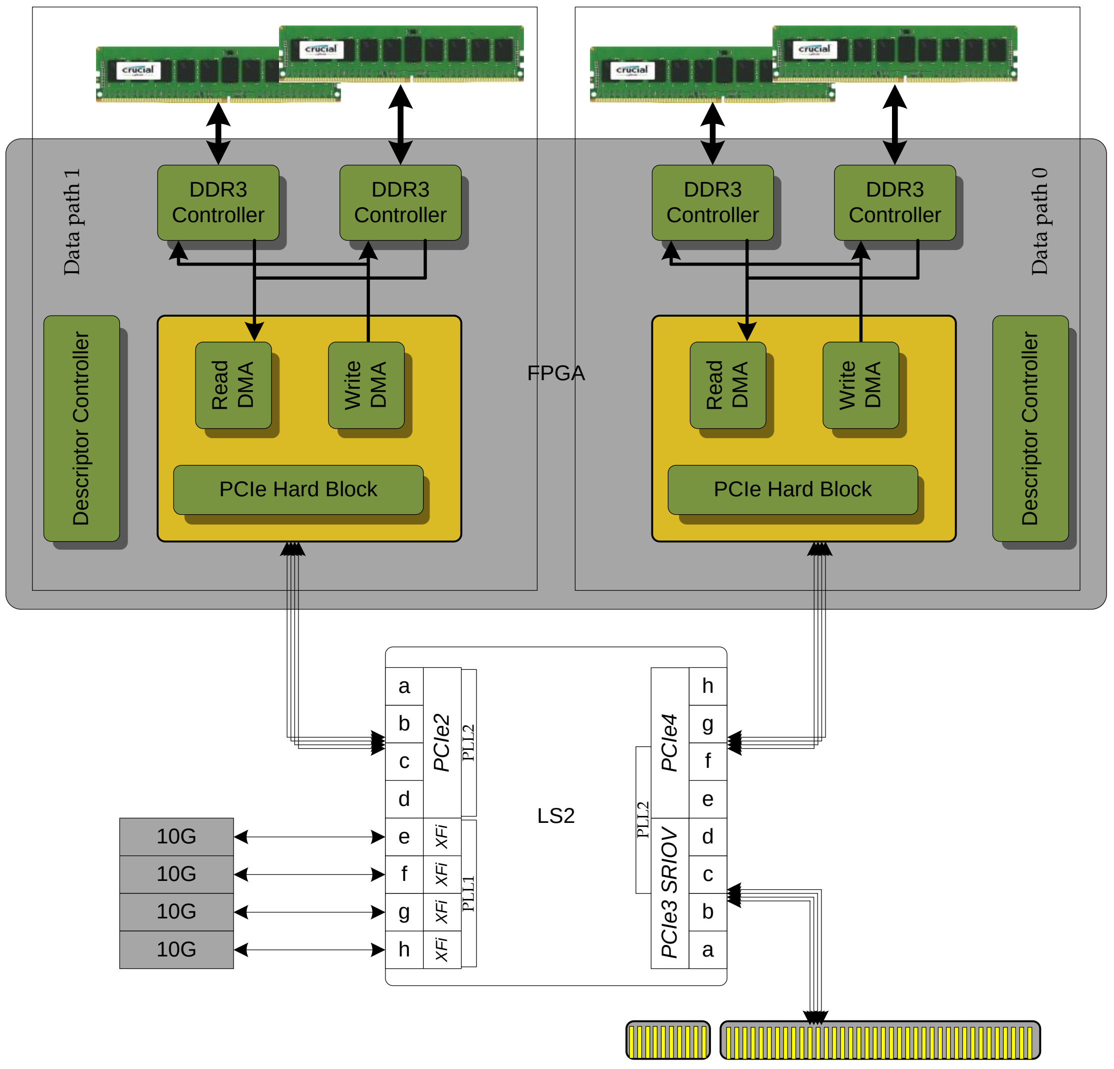


Figure 1: HW block Diagram

2 FPGA Logic Diagram

Descriptors are kept inside an OCM and these OCM's are mapped to PCIe BARn space. Each OCM will have its own BASE ADDRESS. Descriptor controller will read the OCM and initiate the DMA transaction. NVMe Controller register space is provided inside the FPGA and is in second PCIe x4 Interface.

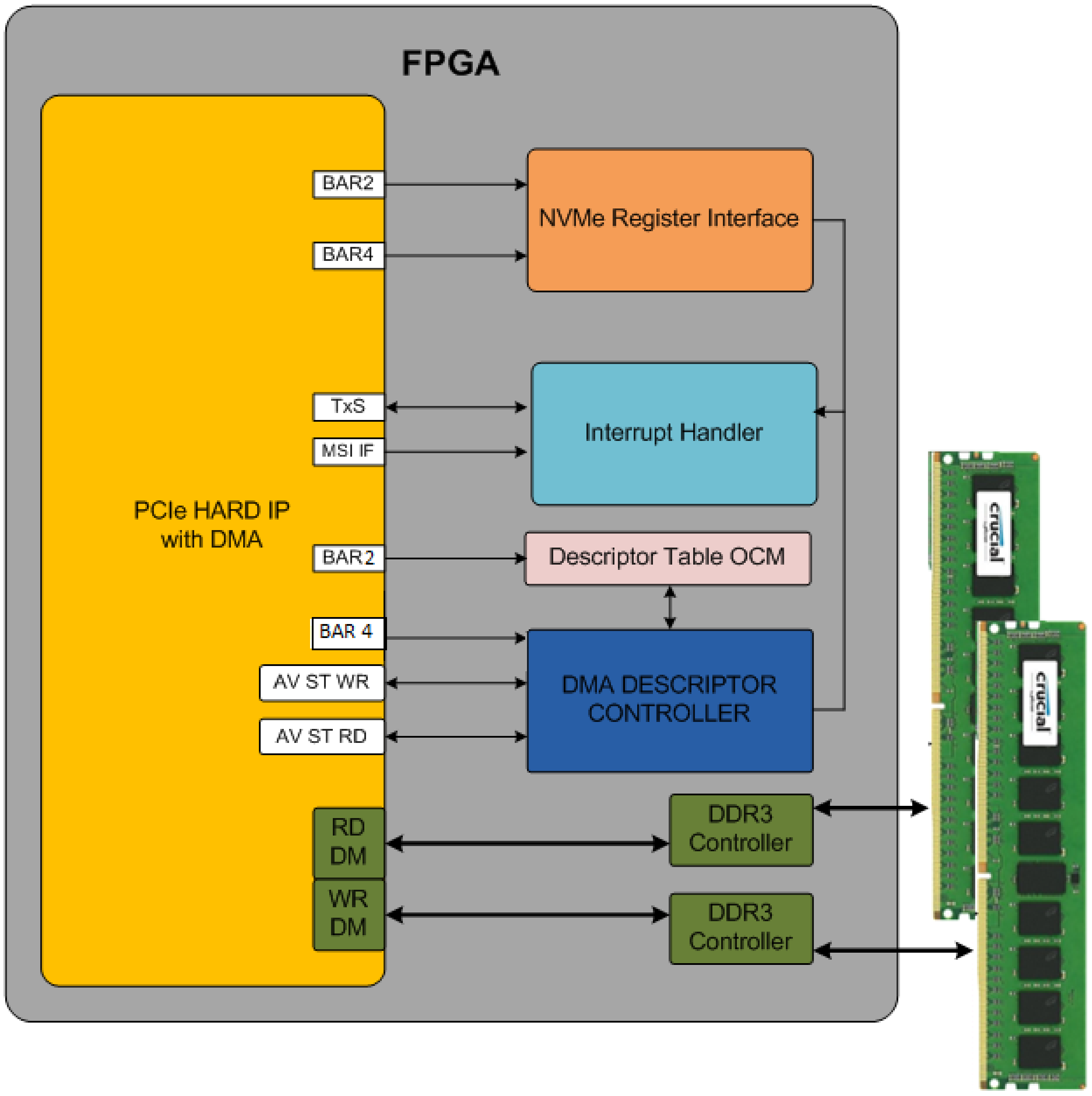


Figure 2: FPGA Block Diagram

3 DIMM memory structure

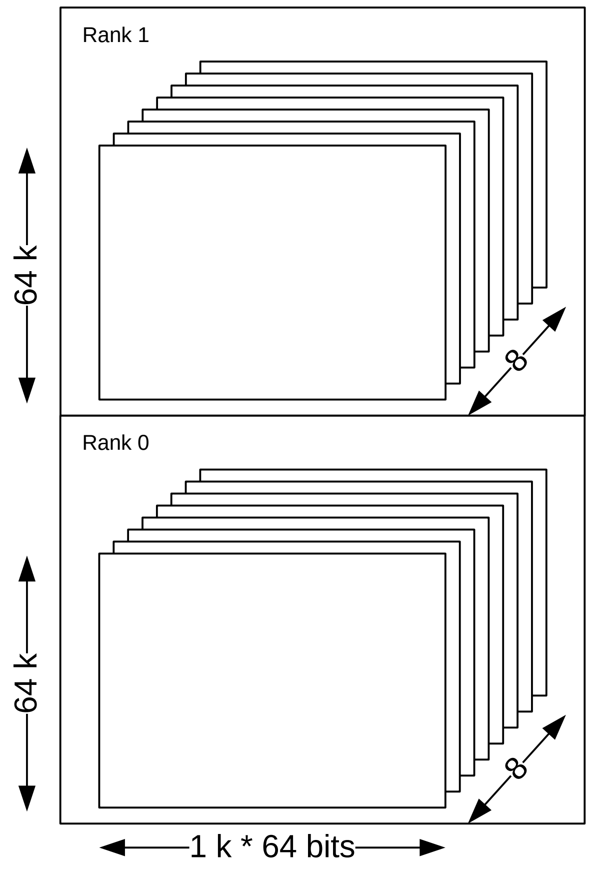


Figure 3: DIMM Memory structure

Interface width(bits)	64
no of Colum	1*1024
no of Rows	64*1024
no of Bank	8
no of Rank	2
Total Capacity(Giga Bytes)	8

Table 1: DIMM Size Calculation

3.1 DIMM memory handling in FPGA

- <Chip Select><Row><Bank><Column> addressing scheme is selected for better performance
- The rank select and row address MSB bit are hard coded for each descriptor.

4 Descriptor Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Des 0	FPGA DDR3 Address Lower																															
	FPGA DDR3 Address Upper																															
	Memory Address LSB																															
	Memory Address MSB																															
	RSV														Length in data words																	
	RSV														Ownership	Descriptor ID											Dma cmp	Skip	EOC	Dma type	Hold	IRQ Bit

Figure 4: Descriptor structure

4.1 Descriptor field description

- **FPGA DDR3 Address Lower :** 32 bit lower address for 4 GB addressing
- **FPGA DDR3 Address Upper :** We are using **2** bits from these 32 bits along with Lower address to address upto 16 GB
- **Memory address LSB :** Lower 32 bits address of Host Memory
- **Memory address MSB :** Upper 32 bits address of Host Memory
- **Interrupt bit:** Host can set this bit to 1 to raise the interrupt after that particular descriptor operation is completed.
- **Hold bit:** Host can set this bit to one to hold the next descriptor fetch till the start signal from host.
- **DMA Type :** To select commands for the DDR(Only Read and Write in case of DDR3
- **Length in Data Words :** Length of data transmission in Data Words
- **EOC bit:** Software can write a '1' to identify the end descriptor in a command
- **Skip bit:** Software can write a '1' to skip the specific descriptor
- **DMA Complete:** When a DMA is successfully completed this bit is set to '1'.
- **Descriptor ID :** ID of the descriptor
- **Ownership bit:** To set the ownership of the descriptor between FPGA and processor. If this bit is set to 1 by host, then the ownership is transferred to FPGA DMA Descriptor Controller, Descriptor controller can fetch and execute that descriptor. After completing a descriptor operation FPGA will update the status of operation in to the status field and set this bit to 0 to transfer the ownership back to processor.

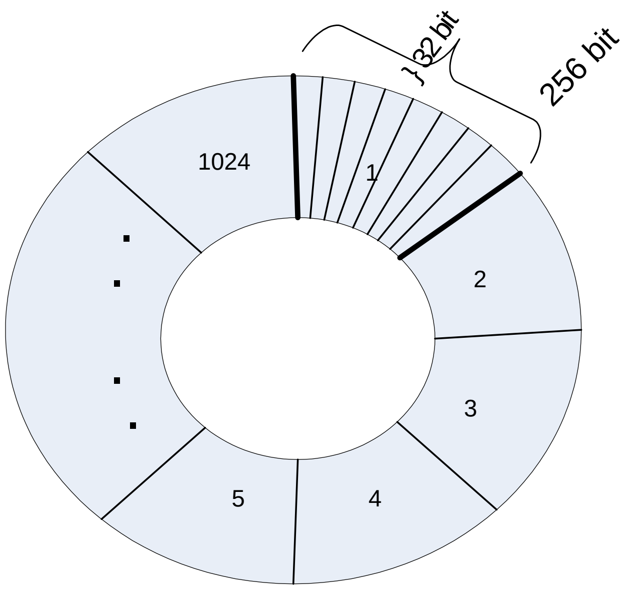


Figure 5: Descriptor table structure in OCM

5 Memory Mapping of PCIe BAR Space for DMA controller

PCIe BAR 4 mapping for DMA is as shown in figure below. For both the PCIe we are keeping the same BAR mapping.

Base Address(BAR4)	Field
0X0000000000002000	Control and status register
0X0000000000002002	Descriptor Table Size Register

Base Address(BAR2)	Field
0X0000000000020000	Descriptor Table 1

Table 2: Memory mapping of CSR and Descriptor Tables

5.1 Control and status register (CSR) for descriptor controller (At offset 0x002000)

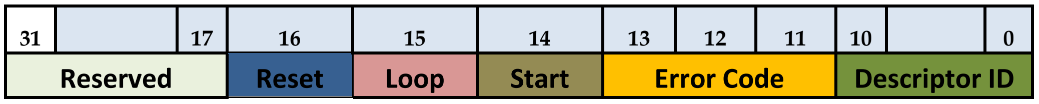


Figure 6: Control and status register

- **Descriptor ID:** This 8 bit field specifies which descriptor in the Descriptor table is the cause for Interrupt.
- **Error Code:** This three bit field specifies the type of error.

Error Code			Description
0	0	0	No IRQ
0	0	1	Write Time Out
0	1	0	Read Time Out
0	1	1	No Error, User Requested Interrupt
1	0	0	No Error, User Requested Interrupt with hold

Table 3: DMA Error Code

- **Start:** Start bit to start DMA transaction for the specified Descriptor controller. Software can write a '1' to start DMA

- **Loop:** If this bit is set to '0' by the software, Descriptor controller stops after reading and processing the 1024 descriptors. Else the Descriptor controller starts again to process from descriptor '0'.
- **Reset:** This bit is set to reset the entire DMA logic as well as descriptor table once an error occurs in the data transfer. Should be set separately for both the PCIe's.

5.2 Handling Interrupts

5.2.1 DMA Interrupts (IRQ0 & IRQ1)

Once an Interrupt condition is occurred, the corresponding Descriptor controller will generate the specific interrupt. There will be separate MSI interrupt for the 2 descriptor controllers in each x4 PCIe IP **(To be implemented)**. Once the interrupt is raised software can read the corresponding CSR to check the status.

5.2.2 NVMe Interrupts (IRQ2 & IRQ3)

Once an Interrupt condition is occurred, single interrupt will be generated and no need to clear the interrupts manually from software side.

MSI Interrupt	Function
IRQ0	Descriptor Controller 0(PCIe0)
IRQ1	Descriptor Controller 1(PCIe1)
IRQ2	NVMe FIFO Interrupt
IRQ3	NVMe Doorbell Interrupt

Table 4: MSI IRQ functions

5.3 Descriptor Table Size Register (At offset 0x002002)

By default descriptor table size will be 1024, which is the maximum value. Software can update this size using the Descriptor table size register, there by the maximum number of descriptors is configurable**.**

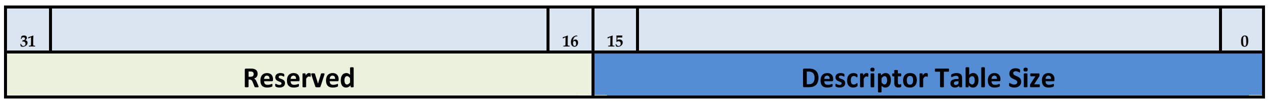


Figure 7 Descriptor Table Size Register

6 NVME Block Diagram

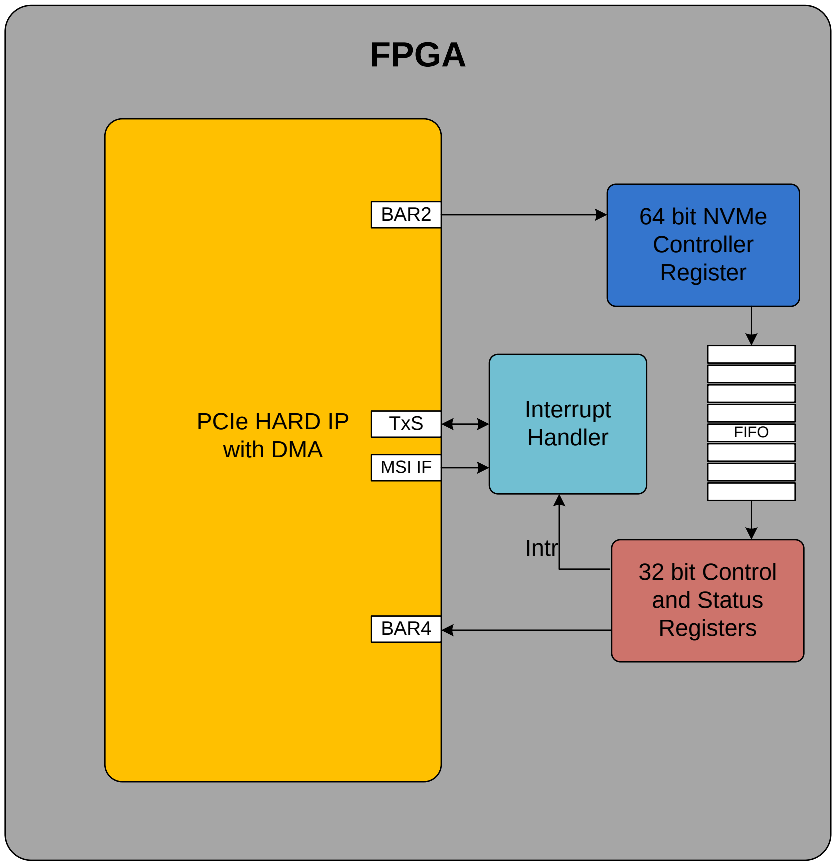


Figure 8 NVME Block Diagram

The design consists of one PCIe end point implementation inside the FPGA. PCIe end point is connected to NVME Controller Register Interface through avalon MM Slave interface (BAR2) of the PCIe hard IP. BAR4 of PCIe hard IP is connected to 32-bit Control/Status registers. PCIe hard IP also receive MSI interrupt source from NVME register interface.

6.1 NVME Controller Registers

This is a 64 bit register bank used to implement NVME Standard Controller registers. It has the following registers.

- Controller Capabilities
- Version
- Interrupt Mask Set
- Interrupt mask clear
- Controller Configuration
- Controller status
- NVME subsystem Reset

- Admin Queue Attributes
- Admin Submission Queue Base Address
- Admin completion Queue Base Address
- Controller Memory Buffer Location
- Controller Memory Buffer Size
- Submission Queue 0 Tail Doorbell (Admin)
- Completion Queue 0 Head Doorbell (Admin)
- 128 Submission Queue Tail Doorbell registers
- 128 Completion Queue Head Doorbell registers

OFFSET ADRESS(64bit)	**SYMBOL**	
0x000	CAP	
0x001	INTMS	VS
0x002	CC	INTMC
0x003	CSTS	Reserved
0x004	AQA	NSSR
0x005	ASQ	
0x006	ACQ	
0x007	CMBSZ	CMBLOC
0x008 … 0x1FF	Reserved	
0x200	CQ0HDBL	SQ0TDBL
0x201	CQ1HDBL	SQ1TDBL
0x202	CQ2HDBL	SQ2TDBL
… 0x280	… CQ128HDBL	… SQ128TDBL

Table 5: NVME Controller Registers

6.2 FIFO64X32

It is an asynchronous FIFO having 64 bit input data width and 32 bit output data width. Whenever a write operation happens to offset register 0x00 to 0x07 and 0x200, then the new data and offset address are written to FIFO. FIFO depth is fixed as 32 K of 32 bit locations (1 MB). For each update there are four entries into the fifo, two entries for new data and two entries for address.

6.3 Control/Status Registers

This is a 32-bit register bank for different Control/Status register. Following registers are included in this register bank. See register descriptions for details.

- FIFO read data
- FIFO read count
- RESET register
- Interrupt register
- Submission queue write status register0
- Submission queue write status register1
- Submission queue write status register2
- Submission queue write status register3

Address	Registers description
0x0000	RESERVED
0x0001	FIFO_READ_DATA
0x0002	FIFO_RD_CNT
0x0003	RST_REG
0x0004	FIFO_REG_INTR
0x0005	QUEUE_STATUS_REG0
0x0006	QUEUE_STATUS_REG1
0x0007	QUEUE_STATUS_REG2
0x0008	QUEUE_STATUS_REG3

Table 6: Control/status registers

6.4 MSI Interrupt Handler

This module handles the Message Signal Interrupt to the PCI hard IP. It generates MSI request, MSI ack, and MSI number based on the interrupt generation from Interrupt register.

7 Register Descriptions

7.1 64-BIT NVME Controller Registers (Bar2 Registers)

These are 64 bit registers used to implement NVME Controller registers. These registers are connected to BAR2 interface in PCIe Hard IP. Both HOST processor and LS2 processor has the write access to these registers with different base addresses. 64-bit Base address for HOST processor and LS2 processor are **0X0000** and **0x0400** respectively. Write mask for HOST processor and LS2 processor are different.

7.1.1 Controller Registers

OFFSET ADRESS(64bit)	SYMBOL		WIDTH	DESCRIPTION
0x000	CAP		64 BIT	Controller Capabilities
0x001	INTMS	VS	64 BIT	INTMS: Interrupt Mask Set VS: Version
0x002	CC	INTMC	64 BIT	CC: Controller Configuration INTMC: Interrupt Mask Clear
0x003	CSTS	Reserved	64 BIT	CSTS: Controller Status
0x004	AQA	NSSR	64 BIT	AQA: Admin Submission Queue Base Address NSSR: NVM Subsystem Reset
0x005	ASQ		64 BIT	Admin Submission Queue Base Address
0x006	ACQ		64 BIT	Admin Completion Queue Base Address

0x007	CMBSZ	CMBLOC	64 BIT	CMBSZ: Controller Memory Buffer Size CMBLOC: Controller Memory Buffer Location
0x008 … 0x1FF	Reserved	64 BIT		
0x200	CQ0HDBL	SQ0TDBL	64 BIT	CQ0HDBL: Completion Queue 0 Head Doorbell (Admin) SQ0TDBL: Submission Queue 0 Tail Doorbell (Admin)
0x201	CQ1HDBL	SQ1TDBL	64 BIT	CQ1HDBL: Completion Queue 1 Head Doorbell SQ1TDBL: Submission Queue 1 Tail Doorbell
0x202	CQ2HDBL	SQ2TDBL	64 BIT	CQ2HDBL: Completion Queue 2 Head Doorbell SQ2TDBL: Submission Queue 2 Tail Doorbell
…	…	…	…	…
0x280	CQ128HDBL	Q128TDBL	64 BIT	CQ128HDBL: Completion Queue 128 Head Doorbell SQ128TDBL: Submission Queue 128 Tail Doorbell

Table 7: NVME Controller registers details

7.1.2 Controller Registers Write mask for Host Processor

Base address (64Bit): 0x00000000

OFFSET ADRESS(64bit)	SYMBOL		HOST PC WRITING MASK
0x000	CAP		0x00 00 00 00 00 00 00 00
0x001	INTMS	VS	0xFF FF FF FF 00 00 00 00
0x002	CC	INTMC	0x00 FF FF F1 FF FF FF FF
0x003	CSTS	Reserved	0x00 00 00 10 00 00 00 00
0x004	AQA	NSSR	0x0F FF 0F FF FF FF FF FF
0x005	ASQ		0xFF FF FF FF FF FF F0 00
0x006	ACQ		0xFF FF FF FF FF FF F0 00
0x007	CMBSZ	CMBLOC	0x00 00 00 00 00 00 00 00
0x008 … 0x1FF	Reserved		0x00 00 00 00 00 00 00 00
0x200	CQ0HDBL	SQ0TDBL	0x00 00 FF FF 00 00 FF FF
0x201	CQ1HDBL	SQ1TDBL	0x00 00 FF FF 00 00 FF FF
0x202	CQ2HDBL	SQ2TDBL	0x00 00 FF FF 00 00 FF FF
… 0x280	… CQ128HDBL	… SQ128TDBL	… 0x00 00 FF FF 00 00 FF FF

Table 8: Write mask for host processor

7.1.3 Controller Registers Write mask for LS2

Base address (64Bit): 0x00000400

OFFSET ADRESS(64bit)	SYMBOL		HOST PC WRITING MASK
0x000	CAP		0xFF FF FF FF FF FF FF FF
0x001	INTMS	VS	0x00 00 00 00 FF FF FF FF
0x002	CC	INTMC	0xFF 00 00 0E 00 00 00 00
0x003	CSTS	Reserved	0xFF FF FF EF 00 00 00 00
0x004	AQA	NSSR	0xF0 00 F0 00 00 00 00 00
0x005	ASQ		0x00 00 00 00 00 00 0F FF
0x006	ACQ		0x00 00 00 00 00 00 0F FF
0x007	CMBSZ	CMBLOC	0xFF FF FF FF FF FF FF FF
0x008 … 0x01FF	Reserved		0x00 00 00 00 00 00 00 00
0x200	CQ0HDBL	SQ0TDBL	0xFF FF 00 00 FF FF 00 00
0x201	CQ1HDBL	SQ1TDBL	0xFF FF 00 00 FF FF 00 00
0x202	CQ2HDBL	SQ2TDBL	0xFF FF 00 00 FF FF 00 00
… 0x280	… CQ128HDBL	… SQ128TDBL	… 0xFF FF 00 00 FF FF 00 00

Table 9: Write mask for LS2 processor

7.2 32-Bit Control/Status Registers (Bar4 Registers)

Base address (32bit): 0x00000000

The below table represent the registers and addresses.

Address	Registers description	Type	Width	Reset Value
0x0000	RESERVED	NA	32 bit	0xFFFFFFFF
0x0001	FIFO_READ_DATA	Read Only	32 bit	0x00000000
0x0002	FIFO_RD_CNT	Read Only	32 bit	0x00000000
0x0003	RST_REG	Write Only	32 bit	0xFFFFFFFF
0x0004	FIFO_REG_INTR	Write Only	32 bit	0x00000000
0x0005	QUEUE_STATUS_REG0	Read/Write	32 bit	0x00000000
0x0006	QUEUE_STATUS_REG1	Read/Write	32 bit	0x00000000
0x0007	QUEUE_STATUS_REG2	Read/Write	32 bit	0x00000000
0x0008	QUEUE_STATUS_REG3	Read/Write	32 bit	0x00000000

Table 10: Control/Status registers

7.2.1 FIFO read address register (FIFO_READ_DATA)

Read operation from this address gives the FIFO data.

Address = 0x001	Bit31	…	Bit2	Bit1	Bit0
FUNCTION	READ_DATA31	…	READ_DATA2	READ_DATA1	READ_DATA0
Access	R	…	R	R	R
Reset Value	0	…	0	0	0

Table 11: FIFO read address register

7.2.2 FIFO read count Register (FIFO_RD_CNT)

This register holds the number of data inside the FIFO.

Address = 0x002	**Bit31**	…	**Bit13**	**Bit12**	**…**	**Bit0**
FUNCTION	NA	…	NA	RD_CNT1	…	RD_CNT0
Access	--	…	--	R	…	R
Reset Value	0	…	--	0	…	0

Table 12: FIFO read count Register

7.2.3 RESET profile register (RST_REG)

Used for resetting different modules inside the register interface.

Address = 0x03	**Bit31**	…	**Bit1**	**Bit0**
FUNCTION	NA	…	NA	FIFO_RESET
Access	--	…	--	W
Reset Value	0	…	0	0

Table 13: RESET profile register

Bit Name	**Value**	**Description**
FIFO_RESET	1	Active high FIFO reset

Table 14: RESET profile register description

7.2.4 Interrupt Register (FIFO_REG_INTR)

A write operation to corresponding bits in this register generates the MSI/LEGACY interrupt through PCIe according to the configuration.

Address = 0x04	Bit31	…	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FUNCTION	NA	…	NA	NA	NA	NA	INTR1	NA
Access	--	…	--	--	--	--	W	W
Reset Value	0	…	0	0	0	0	0	0

Table 15: Interrupt register

Bit Name	Value	Description
INTR1	1	Generate MSI/LEGACY interrupt according to configuration

Table 16: Interrupt registers description

7.2.5 Submission Queue Write Status Register0 (SQ_STATUS_REG0)

A write operation to Submission queue register1 to Submission queue register32 set the corresponding bits in this register.

Address = 0x05	Bit31	…	Bit1	Bit0
FUNCTION	SQ_STATUS_REG0(31)	…	SQ_STATUS_REG0(1)	SQ_STATUS_REG0(0)
Access	RW	…	RW	RW
Reset Value	0	…	0	0

Table 17: Submission queue write status register0

Bit Name	Value	Description
SQ_STATUS_REG0(0)	1	Submission queue register1 is updated
	0	Reset value
...		
SQ_STATUS_REG0(31)	1	Submission queue register32 is updated
	0	Reset value

Table 18: Submission queue write status register0 description

7.2.6 Submission Queue Write Status Register1 (SQ_STATUS_REG1)

A write operation to Submission queue register33 to Submission queue register64 set the corresponding bits in this register.

Address = 0x06	Bit31	…	Bit1	Bit0
FUNCTION	SQ_STATUS_REG1(31)	…	SQ_STATUS_REG1(1)	SQ_STATUS_REG1(0)
Access	RW	…	RW	RW
Reset Value	0	…	0	0

Table 19: Submission queue write status register1

Bit Name	Value	Description
SQ_STATUS_REG1(0)	1	Submission queue register33 is updated
	0	Reset value
...		
SQ_STATUS_REG1(31)	1	Submission queue register64 is updated
	0	Reset value

Table 20: Submission queue write status register1 description

7.2.7 Submission Queue Write Status Register2 (SQ_STATUS_REG2)

A write operation to Submission queue register65 to Submission queue register96 set the corresponding bits in this register.

Address = 0x07	Bit31	…	Bit1	Bit0
FUNCTION	SQ_STATUS_REG2(31)	…	SQ_STATUS_REG2(1)	SQ_STATUS_REG2(0)
Access	RW	…	RW	RW
Reset Value	0	…	0	0

Table 21: Submission queue write status register2

Bit Name	Value	Description
SQ_STATUS_REG2(0)	1	Submission queue register65 is updated
	0	Reset value
…		
SQ_STATUS_REG2(31)	1	Submission queue register96 is updated
	0	Reset value

Table 22: Submission queue write status register2 description

7.2.8 Submission Queue Write Status Register3 (SQ_STATUS_REG3)

A write operation to Submission queue register65 to Submission queue register96 set the corresponding bits in this register.

Bit Name	Value	Description
SQ_STATUS_REG3(0)	1	Submission queue register97 is updated
	0	Reset value
…		
SQ_STATUS_REG3(31)	1	Submission queue register128 is updated
	0	Reset value

Table 23: Submission queue write status register3

Address = 0x08	**Bit31**	…	**Bit1**	**Bit0**
FUNCTION	SQ_STATUS_REG3(0)	…	SQ_STATUS_REG3(0)	SQ_STATUS_REG3(0)
Access	RW	…	RW	RW
Reset Value	0	…	0	0

Table 24: Submission queue write status register3 Description

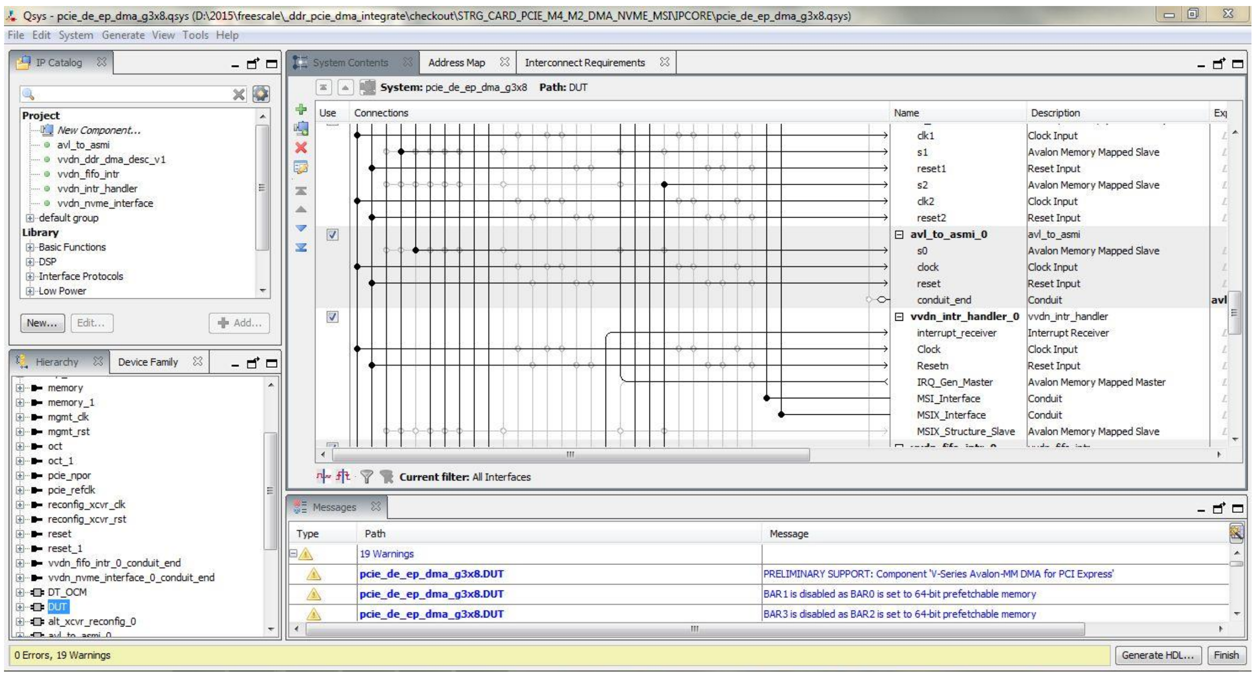
8 References

MT18JSF1G72AZ – 8GBDDR3 SDRAM UDIMM datasheet

9 Appendix

9.1 Qsys Integration

Qsys tool is an integral part of Altera Quartus II FPGA Design/Synthesis Software. It provides all essential tools for legacy/custom IP selection and integration. The tool automates/helps in interfacing various design blocks by providing standard Avalon Interfaces/Handshaking adapters between them. Below shows the standard Qsys window view.



Name	Description
clk1	Clock Input
s1	Avalon Memory Mapped Slave
reset1	Reset Input
s2	Avalon Memory Mapped Slave
clk2	Clock Input
reset2	Reset Input
avl_to_asmi_0	avl_to_asmi
s0	Avalon Memory Mapped Slave
clock	Clock Input
reset	Reset Input
conduit_end	Conduit
vvdn_intr_handler_0	vvdn_intr_handler
interrupt_receiver	Interrupt Receiver
Clock	Clock Input
Resetn	Reset Input
IRQ_Gen_Master	Avalon Memory Mapped Master
MSI_Interface	Conduit
MSIX_Interface	Conduit
MSIX_Structure_Slave	Avalon Memory Mapped Slave

Messages

Type	Path	Message
⚠	19 Warnings	
⚠	**pcie_de_ep_dma_g3x8.DUT**	PRELIMINARY SUPPORT: Component 'V-Series Avalon-MM DMA for PCI Express'
⚠	**pcie_de_ep_dma_g3x8.DUT**	BAR1 is disabled as BAR0 is set to 64-bit prefetchable memory
⚠	**pcie_de_ep_dma_g3x8.DUT**	BAR3 is disabled as BAR2 is set to 64-bit prefetchable memory

0 Errors, 19 Warnings

Qsys Window View

Sytem Contents Tab : Illustrates the possible connections between Logical Blocks in design graphically. Also provides options to add/export interfaces to user design.

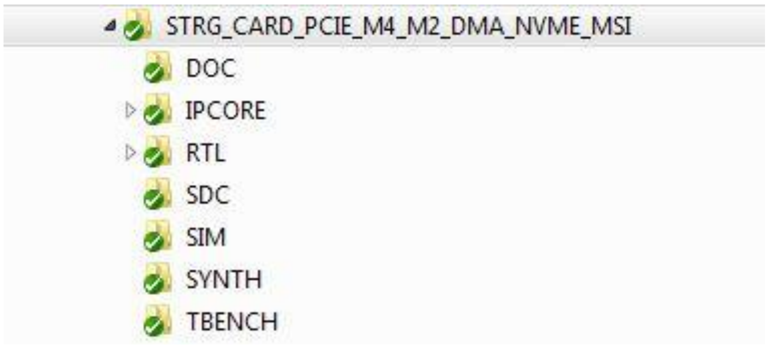
Address Map Tab : Provide a table wise structure to modify the address mapping of each interface connected b/w IP blocks.

Messages Tab: Provides textual message based feedback on the current interconnections. It intimates messages with different severity levels, so that designer can adjust/modify design to optimum.

IP Catalog: Lists all the compatible IPs available in Qsys IP Library. It can also list custom IPs added by developers.

9.2 Folder Structure

This section explains the Folder Structure used in SVN repository for the updation of files used for FPGA Design. Below shows an example screen shot of a project folder structure.



STRG_CARD_PCIE_M4_M2_DMA_NVME_MSI
DOC
IPCORE
RTL
SDC
SIM
SYNTH
TBENCH

Figure :Folder Structure

DOC	: Contains all design documents of FPGA design. FDD, Flash Controller Design, Serial Flash Image Upgradation Documents are updated here.
IPCORE	: Contains all design component files associated with Qsys/Quartus tools. This folder only holds files generated by tools for current design. It contains all tool generated IP component files (like .qip,.qsys,.tcl,.v,.vhd files). It may contain sub-directory for each separate IPs.
RTL	: Contains verilog/vhdl files developed as part of the design by VVDN. All files from this folder are explicitly added to design file list in Quartus Tool.
SDC	: The .sdc file (synopsis design constraints file) used for the project is updated here.
SIM	: All simulation scripts (.tcl,.do files) are updated here. These files relate to a Simulation tool like ModelSIM
SYNTH	: Files specific to the synthesis tool are kept here. The Quartus Project file (.qpf) and Quartus Settings File (.qsf) are updated here.
TBENCH	: All testbenches files are kept in this folder.