G4M-Main AHB Verification and Validation Report DRAFT

Revision: 0.3

Date: September 30th, 2014

Table of Figures

1.

Tables

Revision

Version	Date	Modified by	Changes
0.1	9th OCT 2012	TD	Initial Version
0.2	11th Oct 2012	TD	Updated with use cases
0.3	29th Jan 2013	TD	Added phase2 related Usecase observations
0.4	30th Sept 2014	SW	Report Version

Glossary

Term	Description
G4 MSS	G4 Micro-controller Sub-System
GPIO	General Purpose Input/Output
APB	Advanced Peripheral Bus
AHB	Advanced High-performance Bus
CAN	Controller Area Network
Cortex-M3	ARM processor in G4MSS
Use Case	A specific configuration of a use model.
Use Model	A reference platform designed from configurable h/w and s/w components, which is targeted at a focused application. Use Models demonstrate one of several possible ways of integrating IPs to build a system and one of several ways in which it could used by the customer.
FPGA	Field Programmable Gate Array - an IC with reconfigurable h/w
FIC	Fabric Interface Controller
GENMON	Fabric logic which is capable of writing into and reading from LSRAM/uSRAM

REFERENCES

1. A4P5000 Chip-Level SAC Spec.doc
2. G4_Fabric Interface to FPGA_SAC.doc
3. G4_MMUART_SAC.doc
4. G4M_SWITCH_SAC.doc
5. http://hoppin/wsvn/G4.G4_data/docs/SACSpecs/trunk/SmartFusion2_design_docs.html?usemime=1
6. G4M Memory Sub System PDV plan reviewed
7. GENMON for A4P5000 LSRAM & uSRAM module specification
8. G4M_AHB_SWITCH_PDV_PLAN.docx
9. G4M_AHB_SWITCH_PDV_RESULTS.docx

1 Introduction

G4M Switch is a multi-layer AHB matrix, which works purely as an AHB-Lite Matrix. One master may be accessing a slave at the same time as another master is accessing another slave. If more than one master is attempting to access the same slave simultaneously, then arbitration for that slave is performed, according to a particular algorithm. One master will win the arbitration while other masters are held temporarily. This document explains various use models that we are planning to implement to analyze the behavior of the switch. This document explains various validation use models and use cases that form the part of analyzing throughput of various masters which are used to create traffic on AHB switch inside G4M MSS.

1.1 Purpose and Scope

The purpose of this document is to describe the G4M AHB Switch as well as its plans and tests that are performed as a part of silicon validation and verification. All UseCases are elaborated upon and reports of issues during verification are documented. This document is to provide a holistic view of the verification and validation process

1.2 Standards and Requirements

The Design team and the Verification and Validation Team have different reporting structures that are independent of one another.

2 G4M Switch background

The following figure shows how multiple masters and slaves in G4 MSS are connected through Switch. In the figure below, MM stands for Mirrored Master and MS stands for Mirrored Slave. There are totally 10 Mirrored Master interfaces where as there are 6 Mirrored Slave interfaces. Following are the masters and slaves in G4M:

Masters:
MM0 -->G4M CC ICODE bus
MM1 --> G4M CC DCODE bus
MM2 --> G4M CC System bus
MM3 --> HPDMA
MM4 --> FIC32_0
MM5 --> FIC32_1
MM6 --> Ethernet
MM7 --> PDMA
MM8 --> USB
MM9 --> SII Master

Slaves:
MS0 --> eSRAM0
MS1 --> eSRAM1
MS2 --> eNVM0

MS3 --> eNVM1
MS4 --> FIC32_0
MS5 --> AHB to AHB Bridge (connecting MAC, FIC32_1, SYSREG, APB_0, APB_1, APB_2, USB)
MS6 --> DDR Bridge

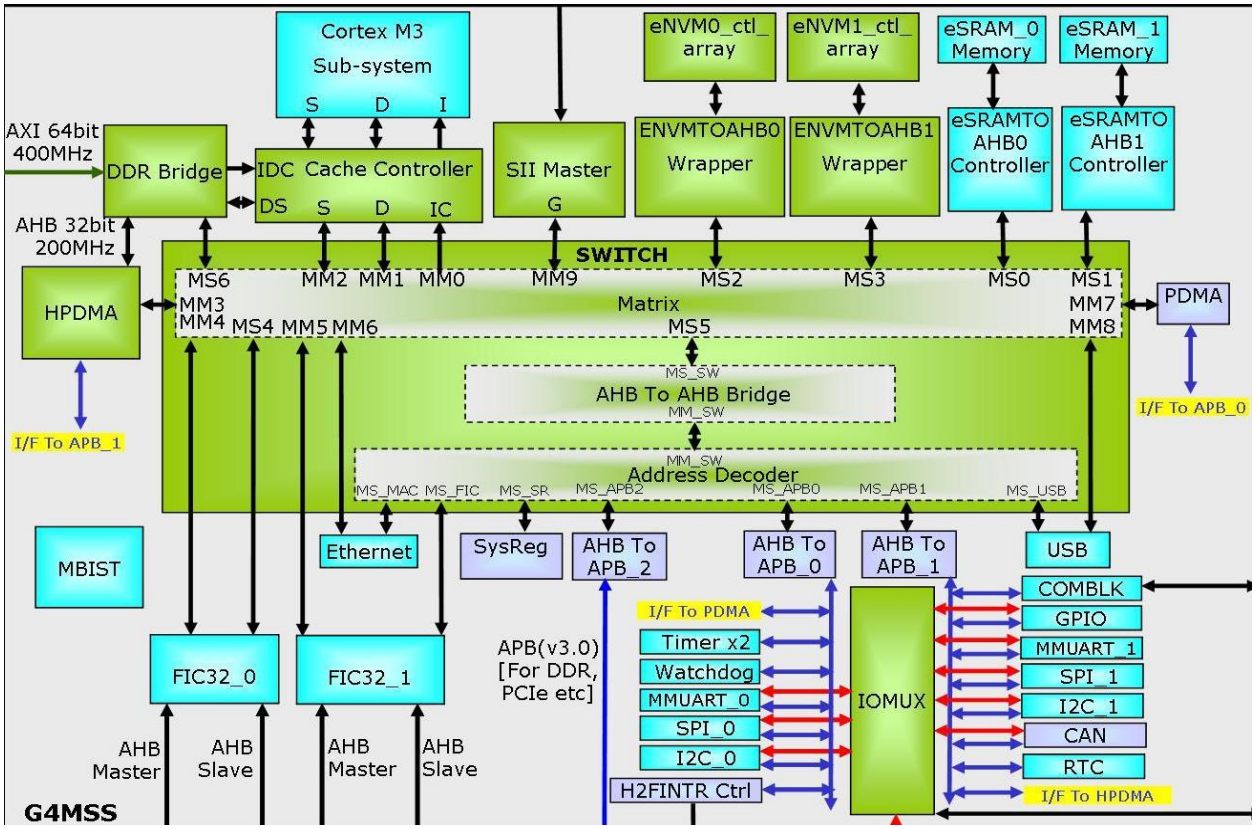

Figure 1 : G4-MSS Block Diagram

The following figure shows how various masters can access different slaves and their priority (Fixed or WRR) in G4M.

	Slave / Master	eSRAM0 [MS0]	eSRAM1 [MS1]	eNVM0 [MS2]	eNVM1 [MS3]	FIC32_0 [MS4]	AHB2AHB Bridge [MS5]							DDR Bridge [MS6]
							MAC_S [MS_MAC]	FIC32_1 [MS_FIC32_1]	SYSREG [MS_SYSREG]	APB_0 [MS_APB0]	APB_1 [MS_APB1]	APB_2 [MS_APB2]	USB_S [MS_USB]	
Priority														
2 [Fixed]	**IC Bus** [MM0]	R	R	R(*)	R(*)	--	--	--	--	--	--	--	--	--
1 [Fixed]	**D-Bus** [MM1]	RW	RW	RW(*)	RW(*)	--	--	--	--	--	--	--	--	--
3 [Fixed]	**S-Bus** [MM2]	RW	RW	RW(*)	RW(*)	RW	RW	RW	RW	RW	RW	RW	RW	--
4 [WRR]	**HPDMA** [MM3]	RW	RW	R(*)	R(*)	RW	--	RW	--	--	--	--	--	--
4 [WRR]	**FIC32_0** [MM4]	RW	RW	RW(*)	RW(*)	RW	RW	RW	RW	RW	RW	RW	RW	RW
4 [WRR]	**FIC32_1** [MM5]	RW	RW	RW(*)	RW(*)	RW	RW	RW	RW	RW	RW	RW	RW	RW
4 [WRR]	**MAC_M** [MM6]	RW	RW	--	--	RW	--	RW	--	--	--	--	--	RW
4 [WRR]	**PDMA** [MM7]	RW	RW	RW(*)	RW(*)	RW	--	RW	--	RW	RW	--	--	RW
4 [WRR]	**USB** [MM8]	RW	RW	--	--	RW	--	RW	--	--	--	--	--	RW
4 [Fixed]	**G** [MM9]	RW	RW	RW(*)	RW(*)	RW	RW	RW	RW	RW	RW	RW	RW	RW

Figure 2 : G4M Switch priority and connections

The following are the different features.

1. Implement an AHB-Lite Multi-Layer switch matrix, AMBA 3 AHB Lite protocol [4].
2. Provide transparent AHB-Lite operation between masters and slaves (i.e. no extra states added to the transactions) in the event of un-contended access to a particular slave.
3. Provide arbitration functionality for access to each slave by candidate masters.
4. Provide high priority for processor bus masters accessing slaves against non-processor bus masters.
5. Provide deterministic latency for the processor bus masters in accessing eSRAM slaves
6. Provide support for programming the value for number of un-interrupted transfer for any master.
7. Provide support for locked AHB-Lite transactions for the required master.
8. Allow 8-bit, 16-bit or 32-bit accesses by masters to slaves.
9. Provide address decoding for full G4MSS.
10. Provide eNVM remap for Cortex-M3.
11. Provide eNVM remap for both soft masters in FPGA fabric.

12. Provide eSRAM remap for Cortex-M3 (in functional mode and in test mode).
13. Provide support for blocking of access to all masters except CM3 master if so configured by firmware.
14. Provide support for blocking of FPGA fabric masters to a defined region of memory space, if so configured by firmware.
15. If MSS present on a virtual part, with eNVM1 block in a powered-down state, then don't allow any accesses to eNVM1 block to be passed to the eNVM1 AHB controller.
16. To restrict access to ENVM based on the size of NVM present in the Design

3 Block Diagram

The below figure describes the A4P5000 PDV setup.

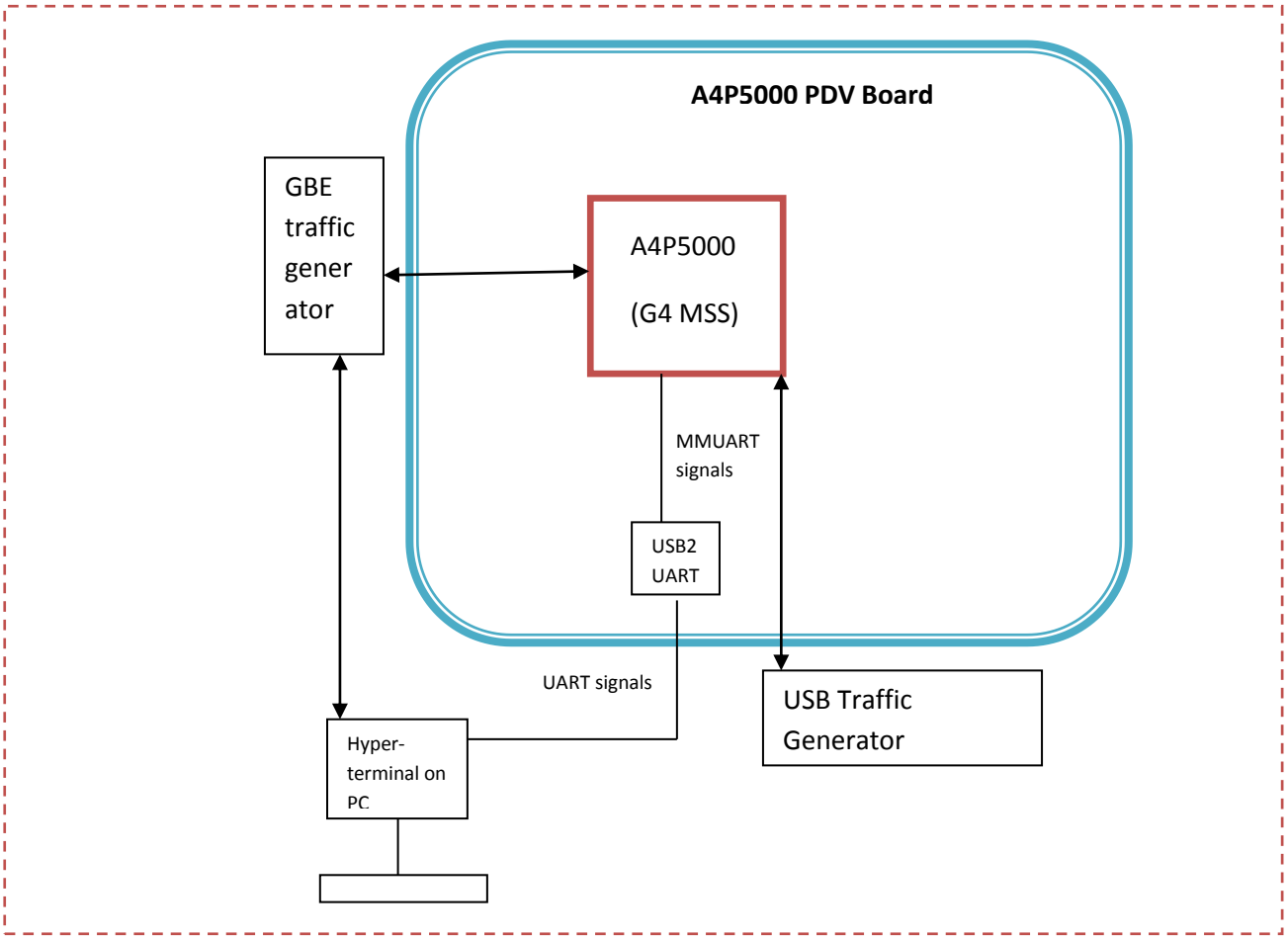

Figure 3 : A4P5000 PDV setup block diagram

4 SYSREG settings

The following SYSREG bits need to be set if you want to access the slaves from masters

MM0_1_2 to MS0/_1/_2/_3/_6 (Cache Controller to eSRAM, eNVM & DDR)

MM4_5 to MS0/_1/_2/_3/_6 (FIC32 to eSRAM, eNVM & DDR)

MM3_6_7_8 to MS0_/1_/2_/3_/6(HPDMA, MAC, PDMA & USB to eSRAM, eNVM & DDR)

MM9 to MS0_/1_/2_/3_/6 (Allowed_R & Allowed_W) (SII Master to eSRAM, eNVM & DDR)

These bits are configured through Libero.

5 Use Models & Implementation Strategy

a. Multiple masters accessing different slaves

Aim

The aim of this use case is to prove that G4M AHB switch is able to handle full traffic situations like multiple masters accessing multiple slaves.

Description

Following are the master – slave access configurations planned for this use case:

- CortexM3 firmware is running out of eSRAM0
- PDMA is writing to MMUART reading from eSRAM1
- HPDMA is writing to DDR bridge
- Fabric master(either CM1 or FSM) is accessing APB peripherals or eSRAM

Using the CortexM3 firmware, we enable all the masters & slaves and establish communication between them. Fabric master CM1 firmware is used to make CM1 access APB peripherals.

Pass criteria

All the transactions have to be passed. All the masters should be able to write to their respective slaves without any hurdles

b. All masters accessing eSRAM0 (Stack & Heap in eSRAM0)

Aim

The aim of this use case is to analyze the thru-put of G4M AHB Switch when multiple masters are accessing same slave simultaneously. Here Stack & Heap are in eSRAM0 and all masters are targeting eSRAM0.

Description

- Here, firmware is running out of eSRAM0.
- CortexM3 is reading eSRAM0
- HPDMA reads from eSRAM0 and writes to DDR bridge
- PDMA reads from eSRAM0 and writes to eSRAM1.
- Fabric master is accessing eSRAM0
- MAC has it's descriptors in eSRAM0. For MAC TX operation, it reads eSRAM0 and transmits the data out
- USB has it's descriptors in eSRAM0. Whenever the file inside USB mass storage device is selected, traffic is generated on eSRAM0.

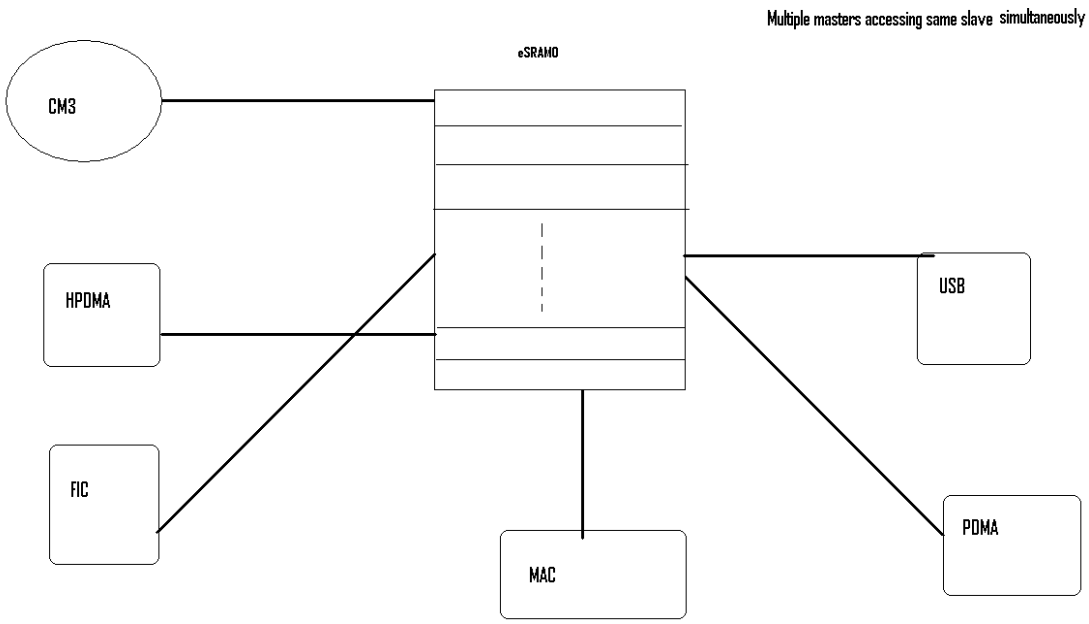

Figure 4 : All masters accessing eSRAM0 (Stack and Heap in eSRAM0)

Flow

We have a global variable set in the USB firmware whenever USB master starts generating traffic on eSRAM0. In the main function, we wait for this global variable to set, and once set, immediately enable

timer and enable all the other masters. Once the timer interrupt occur, we disable all the masters and analyze throughput of various masters.

Thru-put calculations:

1. HPDMA Thru-put = HPDMA_TRANSFER_SIZE * hpdma_loop_var + HPDMA_TRANSFER_SIZE – (pending_counters*4)
2. PDMA Thru-put = (PDMA_BUFFER_SIZE * pdma_loop_var + current loop transfer count)
   Multiplying the above formula with 2 will get AHB switch throughput
3. MAC Thru-put = Reading number of bytes transferred by MAC from MSS MAC registers
4. FIC Thru-put = Number of iterations of while loop * 8
5. USB Thru-put = USB DMA count
6. CM3 Thru-put = Number of while loop iterations

Results table (30us time window)

Master	CM3	HPDMA	PDMA	FIC	MAC	USB
CM3 only						
HPDMA only						
PDMA only						
MAC only						
CM3+HPDMA						
CM3+PDMA						
CM3+FIC						
CM3+MAC						
CM3+USB						
CM3+HPDMA+PDMA						
CM3+HPDMA+PDMA+FIC						
CM3+HPDMA+PDMA+FIC+MAC						
CM3+HPDMA+PDMA+FIC+MAC+USB						

Table 1 : All masters accessing eSRAM0

c. All masters accessing eSRAM1 (Stack and Heap in eSRAM0) – pure round robin – Slave Maximum Latency not configured

Aim

The aim of this use case is to analyze the thru-put of G4M AHB Switch when multiple masters are accessing same slave simultaneously. Here Stack & Heap are in eSRAM0 and all masters are targeting eSRAM1.

Description

- Here, firmware is running out of eSRAM0.
- CortexM3 is reading eSRAM1
- HPDMA reads from eSRAM1and writes to DDR bridge
- PDMA reads from eSRAM1 and writes to eSRAM1.
- Fabric master is accessing eSRAM1
- MAC has it's descriptors in eSRAM1. For MAC TX operation, it reads eSRAM1 and transmits the data out
- USB has it's descriptors in eSRAM1. Whenever the file inside USB mass storage device is selected, traffic is generated on eSRAM1.

Flow

We have a global variable set in the USB firmware whenever USB master starts generating traffic on eSRAM0. In the main function, we wait for this global variable to set, and once set, immediately enable timer and enable all the other masters. Once the timer interrupt occur, we disable all the masters and analyze throughput of various masters.

Results table (30us time window)

Master	CM3	HPDMA	PDMA	FIC	MAC	USB
CM3 only						
HPDMA only						
PDMA only						
MAC only						
CM3+HPDMA						
CM3+PDMA						
CM3+FIC						
CM3+MAC						
CM3+USB						
CM3+HPDMA+PDMA						
CM3+HPDMA+PDMA+FIC						
CM3+HPDMA+PDMA+FIC+MAC						
CM3+HPDMA+PDMA+FIC+MAC+USB						

Table 2 : All masters accessing eSRAM1

d. All masters accessing eSRAM1 (Stack and Heap in eSRAM0) – pure round robin – eSRAM1 Slave Maximum Latency configured (in Libero)

Everything is same as above UseModel except that the eSRAM1 slave maximum latency is configured.

Results table (30us time window)

Master	CM3	HPDMA	PDMA	FIC	MAC	USB
CM3 only						

HPDMA only						
PDMA only						
MAC only						
CM3+HPDMA						
CM3+PDMA						
CM3+FIC						
CM3+MAC						
CM3+USB						
CM3+HPDMA+PDMA						
CM3+HPDMA+PDMA+FIC						
CM3+HPDMA+PDMA+FIC+MAC						
CM3+HPDMA+PDMA+FIC+MAC+USB						

Table 3 : All masters accessing eSRAM1

e. All masters except CM3 accessing eSRAM1 (Stack and heap in eSRAM0)

Aim

The aim of this use case is to analyze the thru-put of G4M AHB Switch when multiple masters are accessing same slave simultaneously. Here Stack & Heap are in eSRAM0 and all masters are targeting eSRAM1 except CM3. CM3 is reading eSRAM0

Description

- Here, firmware is running out of eSRAM0.
- CortexM3 is reading eSRAM0
- HPDMA reads from eSRAM1and writes to DDR bridge
- PDMA reads from eSRAM1 and writes to eSRAM1.
- Fabric master is accessing eSRAM1
- MAC has it's descriptors in eSRAM1. For MAC TX operation, it reads eSRAM1 and transmits the data out
- USB has it's descriptors in eSRAM1. Whenever the file inside USB mass storage device is selected, traffic is generated on eSRAM1.

Flow

We have a global variable set in the USB firmware whenever USB master starts generating traffic on eSRAM0. In the main function, we wait for this global variable to set, and once set, immediately enable timer and enable all the other masters. Once the timer interrupt occur, we disable all the masters and analyze throughput of various masters.

Results table (30us time window)

Master	**CM3**	**HPDMA**	**PDMA**	**FIC**	**MAC**	**USB**

CM3 only						
HPDMA only						
PDMA only						
MAC only						
CM3+HPDMA						
CM3+PDMA						
CM3+FIC						
CM3+MAC						
CM3+USB						
CM3+HPDMA+PDMA						
CM3+HPDMA+PDMA+FIC						
CM3+HPDMA+PDMA+FIC+MAC						
CM3+HPDMA+PDMA+FIC+MAC+USB						

Table 4 : CM3 accessing eSRAM0 & all other masters accessing eSRAM1

f. Weighed Round Robin test

Aim

The aim of this use case is to prove that changing the weight of a particular master impacts the throughput of that particular master.

Description

By default, weight of all the masters is configured as 2. So, every master has got the equal priority and that forms the pure round robin use case. If we change the weight of a particular master, we should observe that the throughput of that particular master has been changed.

6 Use Case Matrix

S.No.	Feature to be emulated	Description	Issues
1	Multiple masters accessing different slaves		
2	All masters accessing eSRAM0 (Stack & Heap in eSRAM0)		
3	All masters accessing eSRAM1 (Stack and Heap in eSRAM0) – pure round robin – Slave Maximum Latency not configured		
4	All masters accessing eSRAM1 (Stack and Heap in eSRAM0) – pure round robin – eSRAM1 Slave Maximum Latency configured (in Libero)		
5	All masters except CM3		

	accessing eSRAM1 (Stack and heap in eSRAM0)		
6	Weighed Round Robin test		
7	eSRAM remap	Will be covered in eSRAM PDV	
8	eNVM remap	Will be covered in eNVM PDV	
9	Memory protection from FPGA Fabric	Will be covered in FIC32 PDV	
10	Multiple Images	Will be covered in eNVM PDV	
11	Fabric running out of eNVM	Will be covered in FIC32 PDV	
12	eNVM protection region side band signals	Will be covered in eNVM PDV	
13	Fabric side band signals	Covered as a part of FIC32 validation.	

Table 5 : Use Case Matrix

7 Hardware used

Board used → G4M Validation board, DVP-102-000304-001-RevB

Silicon used → RevB silicon numbered 61

Libero capture used → 10.9.1.22 & 10.9.2.7_sf2spa

8 AHB Switch

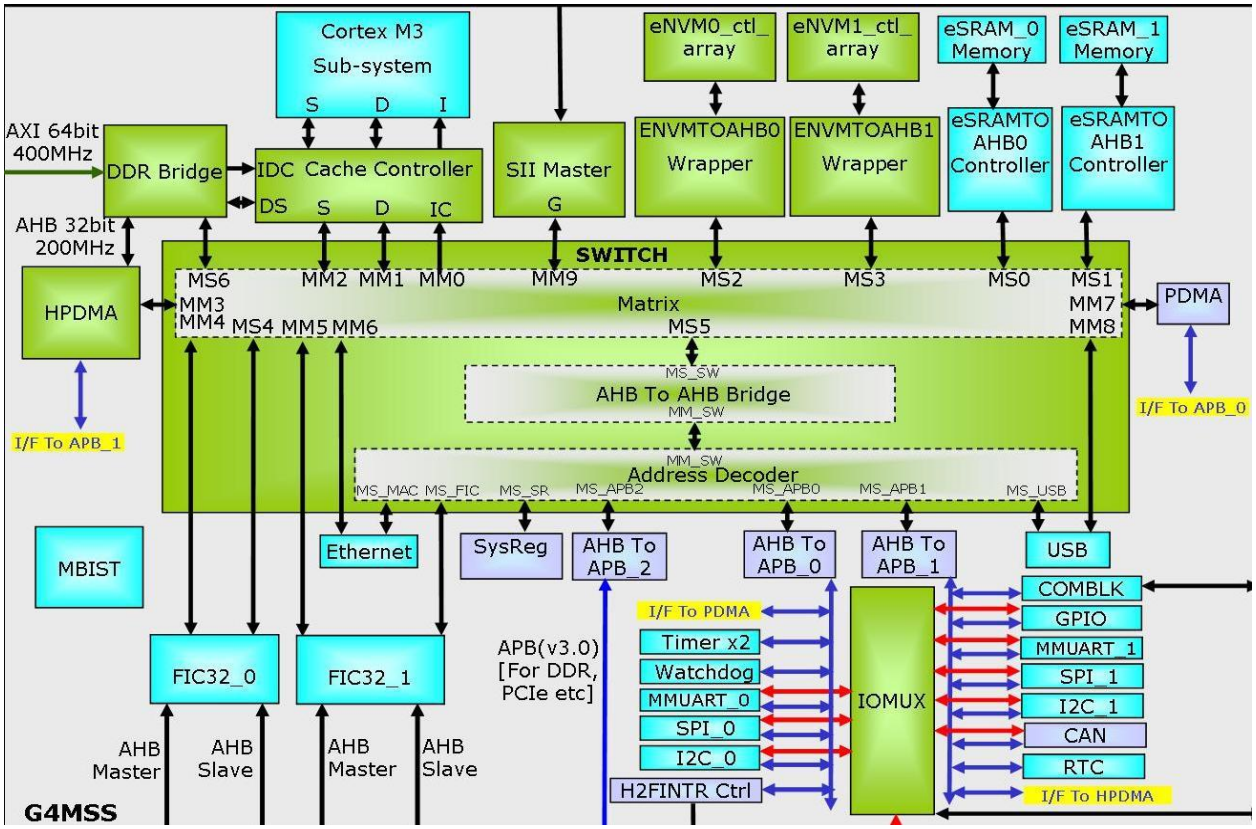

Figure 5 : G4M MSS block diagram

Various masters include:

MM0 -->G4M CC ICODE bus
MM1 --> G4M CC DCODE bus
MM2 --> G4M CC System bus
MM3 --> HPDMA
MM4 --> FIC32_0
MM5 --> FIC32_1
MM6 --> Ethernet
MM7 --> PDMA
MM8 --> USB
MM9 --> SII Master

We used the following WRR masters along with processor master to create traffic on AHB switch:

- HPDMA
- PDMA
- USB
- FIC_0
- MAC

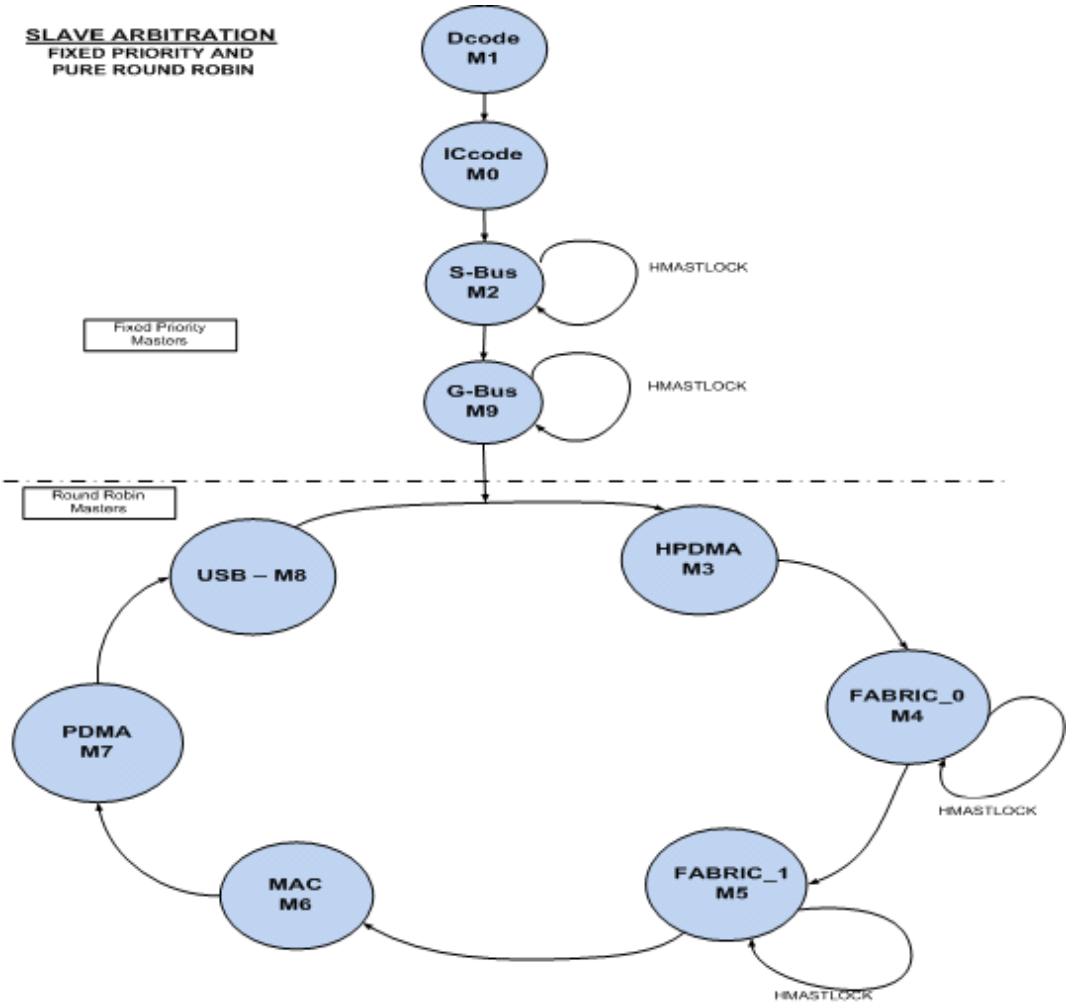

Figure 6 : Pure Round Robin - Fixed Priority slave arbitration

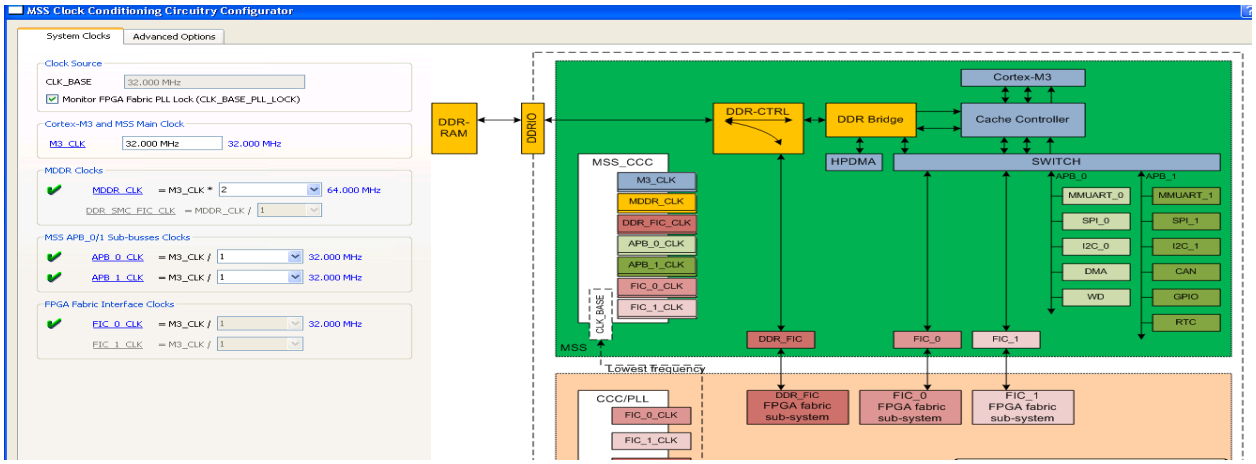

Figure 7: MSS CCC configurator

9 Usemodels & Use cases Test Phase

We executed following use cases to test the throughput of various masters on AHB switch:

9.1 All masters except CortexM3 accessing eSRAM1.

9.1.1 Description

→ Stack & Heap in eSRAM0.

- ➔ 30us time window measurements
- ➔ HPDMA reading from eSRAM1 and writing to DDR (word transfer)
- ➔ PDMA reading from eSRAM1 and writing to eSRAM0 (word transfer)
- ➔ FIC_0 writing to eSRAM1 (Read functionality not implemented in FIC FSM)
- ➔ USB had its descriptors in eSRAM1
- ➔ CortexM3 reading and writing to eSRAM0, so not participating in slave arbitration for eSRAM1.
- ➔ 32MHZ is system clock frequency
- ➔ ESRAM PIPELINE is enabled (means, 2 clock cycles for every read)

9.1.2 Calculation

32MHz clock, means, 32000 clocks for one millisecond

That means 32 clocks for every micro second.

The time frame I am calculating is 30 Microseconds → 30us*32 = 960 clocks

AHB Switch read bandwidth = 960*4 = 3840 bytes of read data

ESRAM PIPELINE is set to 1, which means AHB Switch bandwidth = 3840/2 = 1920 bytes

Master	CM3	HPDMA	PDMA	FIC	USB	Switch Total Throughput
CM3 only	84	0	0	0	0	
HPDMA only	0	728	0	0	0	728
PDMA only	0	0	1048	0	0	1048
FIC only	0	0	0	1064	0	1064
CM3 accessing eSRAM0, HPDMA	84	728	0	0	0	728
CM3 accessing eSRAM0, PDMA	84	0	1048	0	0	1048
CM3 accessing eSRAM0, FIC	84	0	0	1064	0	1064
CM3 accessing eSRAM0, HPDMA+PDMA	84	708	1040	0	0	1748
CM3 accessing eSRAM0, HPDMA+PDMA+FIC	84	540	724	732	0	1996
CM3 accessing eSRAM0, HPDMA+PDMA+FIC+USB	84	452	344	296	468	1560

Table 6 : All masters except CM3 accessing eSRAM1

9.1.3 Observations

1. Total AHB switch read bandwidth is 1920 bytes in 30us. In the above table, we see that total bandwidth has sometimes crossed 1920. This could be because, FIC is writing to eSRAM and it does not take 2 clocks for all the data bytes.

2. When USB is used, we see throughput drop. This could be because, during enumeration USB performs read and write transactions on to eSRAM which may insert idle cycles and finally drop in throughput may be seen.

3. Reviewed on 10 OCT 2012

9.1.4 SVN database

svn://hoppin/IP/PDV/G4_MAIN/G4M_SWITCH/tags/1.0.100/design

9.2 All masters including CortexM3 accessing eSRAM0

9.2.1 Description

→ Stack & Heap in eSRAM0.

- ➔ 30us time window measurements
- ➔ HPDMA reading from eSRAM0 and writing to DDR (word transfer)
- ➔ PDMA reading from eSRAM0 and writing to eSRAM1 (word transfer)
- ➔ FIC_0 writing to eSRAM0 (Read functionality not implemented in FIC FSM)
- ➔ USB had its descriptors in eSRAM0
- ➔ CortexM3 reading eSRAM0 & writing to eSRAM1 participating in slave arbitration for eSRAM0.
- ➔ 32MHZ is system clock frequency
- ➔ ESRAM PIPELINE is enabled (means, 2 clock cycles for every read)

Master	CM3	HPDMA	PDMA	FIC	USB	Total Switch bandwidth
CM3 only	50					50
HPDMA only		756				756
PDMA only			1040			1040
FIC only				1028		1028
CM3 accessing eSRAM0, HPDMA	35	756				791
CM3 accessing eSRAM0, PDMA	43		444			487
CM3 accessing eSRAM0, FIC	43			432		475
CM3 accessing eSRAM0, HPDMA+PDMA	31	684	208			923
CM3 accessing eSRAM0, HPDMA+PDMA+FIC	31	680	168	280		1159
CM3 accessing eSRAM0, HPDMA+PDMA+FIC+USB	24	572	160	224	512	1592

Table 7: All masters including CM3 access eSRAM0; No MAC

9.2.2 Calculation

32MHz clock, means, 32000 clocks for one millisecond

That means 32 clocks for every micro second.

The time frame I am calculating is 30 Microseconds → 30us*32 = 960 clocks

AHB Switch read bandwidth = 960*4 = 3840 bytes of read data

ESRAM PIPELINE is set to 1, which means AHB Switch bandwidth = 3840/2 = 1920 bytes

9.2.3 Observations

1. Total AHB switch read bandwidth is 1920 bytes in 30us. If all the masters are enabled, max switch bandwidth used is 1592.

2. CM3 throughput is getting affected because of other masters as every master is trying to access eSRAM0.

3. Reviewed by Ian on 10 OCT 2012

9.2.4 SVN database

svn://hoppin/IP/PDV/G4_MAIN/G4M_SWITCH/tags/1.0.100/design

9.3 Different masters accessing different slaves

9.3.1 Description

In this use case different masters are enabled and are configured to work with different slaves, meaning, no master is dependent on other master using the bus. In other words there is no slave arbitration in place. Following is the master list:

- CortexM3 running out of eSRAM0
- HPDMA reading from eSRAM1 & writing to DDR2
- PDMA reading from GPIO input register & writing to fabric SRAM

Goal is to create high traffic situation where slave arbitration do not come into picture and still AHB switch handles the traffic as expected.

9.3.2 SVN database

svn://hoppin/IP/PDV/G4_MAIN/G4M_SWITCH/tags/1.0.100/design

9.4 Weighted Round Robin

9.4.1 Description

By default, weight of all the masters is configured as 2. So, every master has got the equal priority and that forms the pure round robin use case. If we change the weight of a particular master, we should observe that the throughput of that particular master has been changed.

9.4.2 Observation

1. With default weight, HPDMA throughput is 728 bytes in 30us time
   Configuring the weight to 4, HPDMA throughput has been changed to 1604 bytes in 30us.
2. With default weight, FIC throughput is 1064 bytes
   Configuring the weight to 2, MAC throughput has been changed to 1112 bytes in 30us.

9.5 Latency effect of eSRAM slave

9.5.1 Description

For eSRAM slaves, maximum latency can be programmed. SW_MAX_LATENCY_ESRAM1 bits need to be programmed with the required latency value. With this parameter, the maximum latency seen by the processor bus can be programmed so when a WRR master with programmable weight greater than the slave max. latency value is accessing an eSRAM slave, then the WRR master will re-arbitrate for the eSRAM slave after doing <max_latency> number of transaction and the access is given to the high priority master requesting the access. In this way the processor bus wait time for the access to eSRAM slave is reduced considerably there by improving the system response.

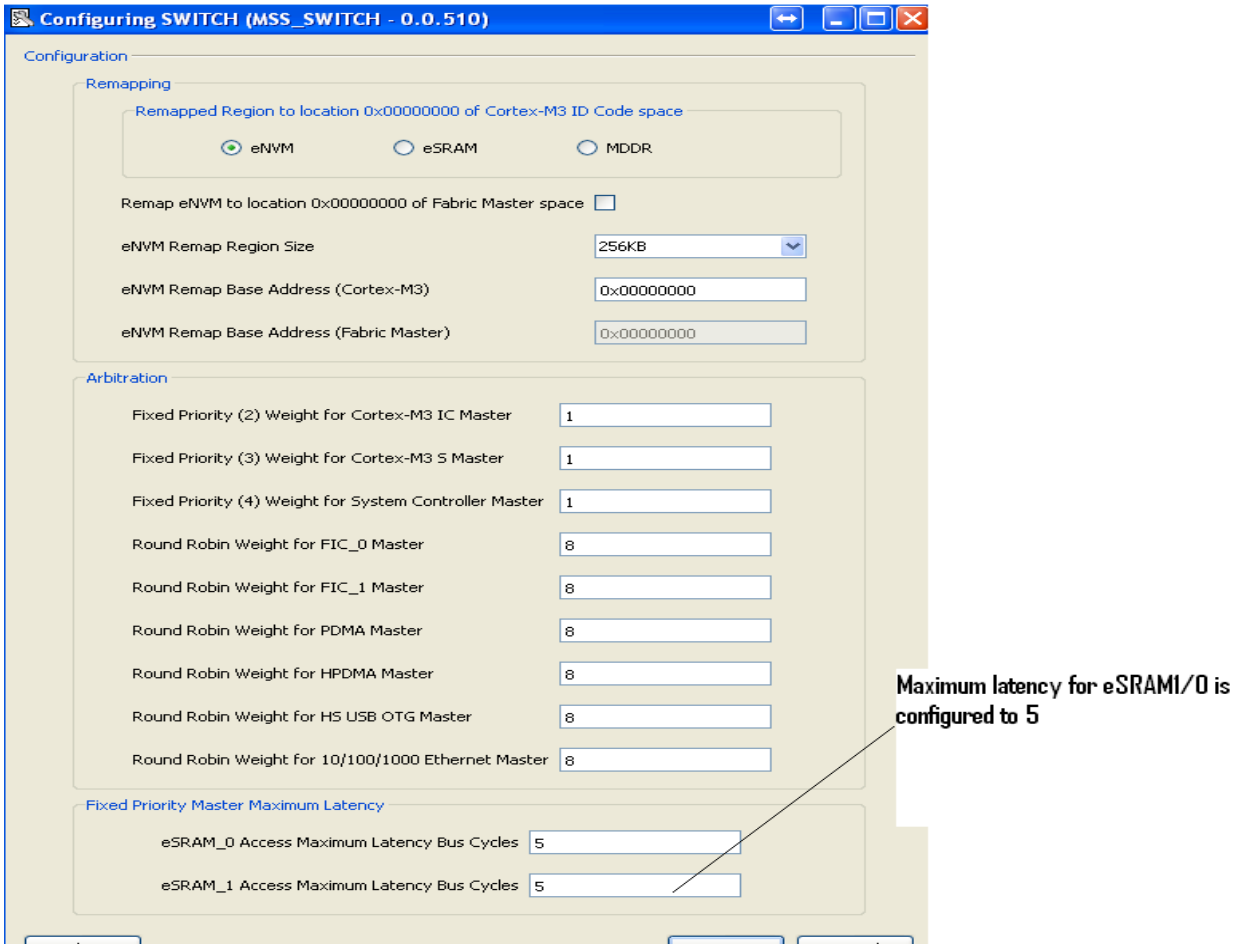

Figure 8 : eSRAM Latency configuration in Libero

9.5.2 Observation

Observed that the slave latency configured in Libero is programmed in to system registers which can be seen using IAR:

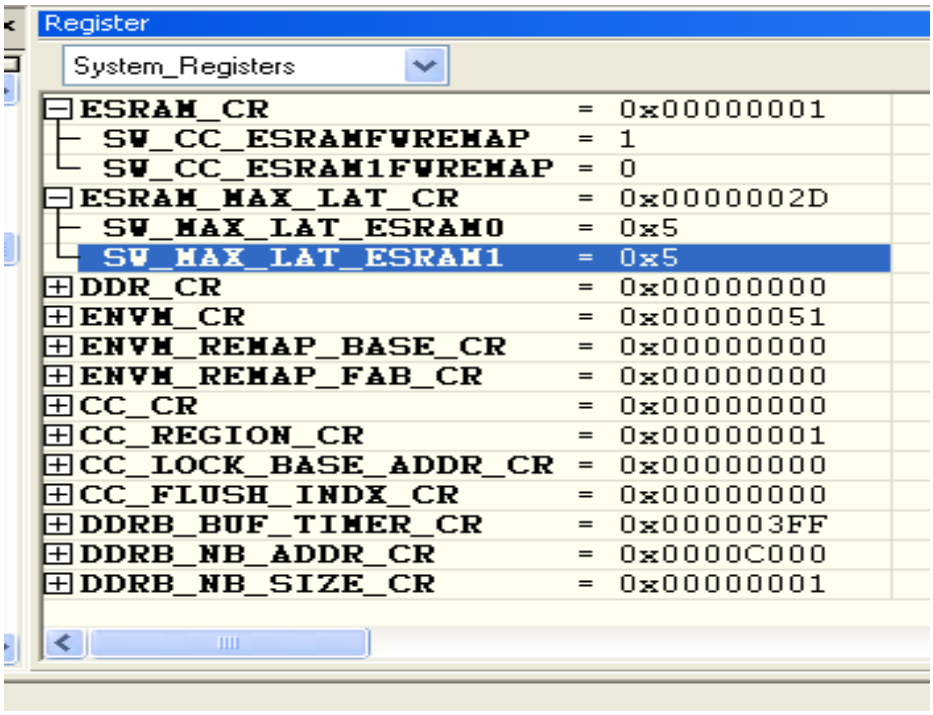

Figure 9 : IAR showing the Libero configuration in system registers for latency values

eSRAM1 latency configured is = 0x05

If PDMA weight is configured as 0x08, PDMA throughput = 21

If PDMA weight is configured as 0x03, PDMA throughput = 13

9.5.3 SVN database

svn://hoppin/IP/PDV/G4_MAIN/G4M_SWITCH/trunk/design/hw/AHB_switch_latency_effect.zip

10 Appendix

Formulae for throughput experiments:

hpdma_thruput = 1024 - hpdma_pending_counters

pdma_thruput = BUFFER size – transfer count

CM3 = loop variable

FIC = Number of memory locations written at destination memory

USB: length_read variable