Application Report
SPRACI6–November 2018

AM65xx System Performance

Jian Wang

ABSTRACT

The AM65xx family of devices incorporated multiple unique features to achieve the best system latency, bandwidth, and functional safety demanded by industrial applications. This application report provides an overview of the system interconnect and configuration capabilities as well as key subsystem latency and bandwidth throughput benchmarks, so a system integrator can leverage these features to optimize their application to achieve desired real-time performance.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The AM65xx family of processors are targeted for high-performance industrial networking and control applications, where a highly-integrated processor is achieved with bandwidth performance, low-latency, functional safety, and security, all in a single-chip architecture. To support these features, a flexible system interconnect is required to provide enough bandwidth and configuration capability, so a system can be purposely optimized for a certain class of applications. This document starts with the introduction of the system interconnect, where various configuration capabilities are presented, then provides subsystem performance benchmarks followed with example optimization techniques.

2 Device Overview

The internal interconnect of the AM65x is optimized to address the demand from modern complex control systems, where several traditionally conflicting performance vectors are to be met simultaneously. These conflicting vectors are:

- Advanced microprocessors with high-performance caches, multicore coherency and memory management (MMU)
- CPU and system memory virtualization, IO virtualization
- Security firewalls
- Additional heterogeneous processor cores with unified memory map
- High-bandwidth, latency-critical interfaces such as PCI Express (PCIe), multi-port real-time Giga-bit Ethernet and graphics processor
- High-bandwidth interfaces such as camera, display and USB ports
- Execution isolation to ensure freedom from interference, for functional safety
- Domain isolation to ensure freedom from interference, for functional safety

A multi-level hierarchical interconnect architecture is thus introduced to the device, with auxiliary side-band fields such as routing ID, addressing type, and firewall controls to support these functions.

2.1 Interconnect Topology

Figure 1 shows a top level data view of the interconnect networks in the device, where names of the top level cross-bar switches are shown in the brackets for clarity. These cross-bar switches are generally named SCR[M/P]_[WIDTH]_[CLOCK_DOMAIN]_[Main IP], where SCR stands for Switched Central Resource (SCR). There is no need to decode these parameters in detail. The voltage domains are also marked in the diagram for reference, though the focus of this document is on data interconnect.

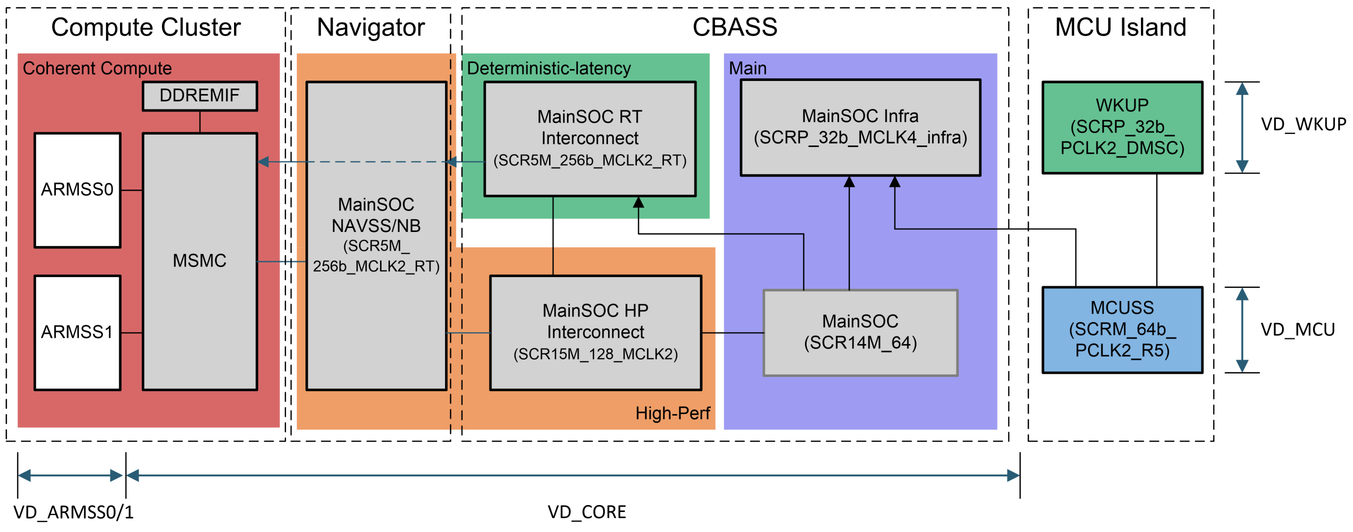
Compute Cluster
Navigator
CBASS
MCU Island
Coherent Compute
DDREMIF
ARMSS0
ARMSS1
MSMC
MainSOC NAVSS/NB (SCR5M_256b_MCLK2_RT)
Deterministic-latency
MainSOC RT Interconnect (SCR5M_256b_MCLK2_RT)
Main
MainSOC Infra (SCRP_32b_MCLK4_infra)
MainSOC HP Interconnect (SCR15M_128_MCLK2)
High-Perf
MainSOC (SCR14M_64)
WKUP (SCRP_32b_PCLK2_DMSC)
MCUSS (SCRM_64b_PCLK2_R5)
VD_WKUP
VD_MCU
VD_ARMSS0/1
VD_CORE

Figure 1. Domain-Level View of AM65x Device Interconnect

From topology hierarchical view, the system interconnect can be partitioned as the Compute Cluster, Navigator, CBASS, and MCU Island. Each of the subsystems may have one or multiple interconnect fabrics:

- Compute Cluster – this is the main coherent interconnect fabric, consists of the two dual-core A53 MPCore subsystems, the Multicore Shared Memory Controller (MSMC), MSMC RAM, and DDREMIF. This fabric ensures that accessing to either the MSMC RAM or the DDR memory space is cache-coherent to the MPU cores
- Navigator - part of the High Performance fabric. It supports multipath bridges between peripherals and accelerators to the main CPU and memory subsystems. Several high-performance System-on-Chip (SoC) infrastructure components reside inside the Navigator, such as UDMA, Mailboxes, Interrupt aggregators, Timer Managers, and so forth. These shared resources can be utilized by multiple subsystems simultaneously
- CBASS - hierarchical interconnect of the main SOC of multiple fabrics, including:
  - MainSOC High Performance Interconnect - Several high-bandwidth interfaces such as USB, camera (CAL), display (DSS), crypto (SA2_UL), are connected to this fabric
  - Deterministic latency fabric – supports real-time Ethernet (ICSSG), GPU and PCIe High-Priority port. Bus masters on this fabric can use a dedicated routing path to access the MSMC memory, not interfered by the rest of system traffic.
  - Main interconnect – supports the rest of the subsystem components and interfaces
- MCU Island – includes the dual-lockstep R5 (MCU) and a select set of SoC subsystems and peripherals. This island can be independently active while the main domain is powered off.
- Wakeup Subsystem – a small subsystem that keep a wakeup domain ON, when the rest of the SoCs are powered off. The wakeup subsystem can be viewed as part of the MCU Island.

The power, clocking and reset architecture mandates that the WKUP, MCUSS, and Main SOC domains be hierarchically enabled, where WKUP is required for MCUSS or MainSOC to be active, and WKUP and MCUSS are required for Main SOC to be active. These dependencies are depicted in Figure 2. For detailed power state definition, see the *Power* chapter in the *AM65x Sitara™ Processors Technical Reference Manual*.

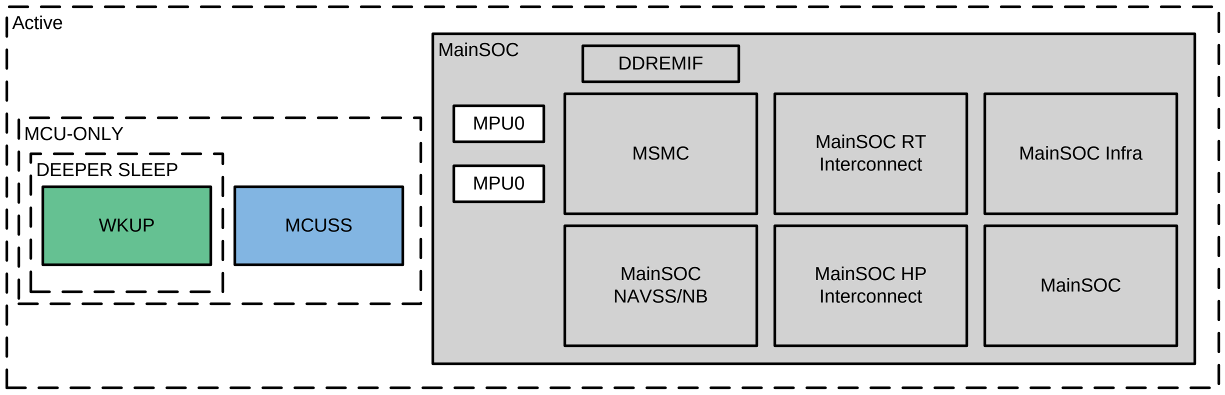

Figure 2. Domain Dependencies

Figure 3 shows the topology view of the MCUSS domain and end-points connected to this domain.

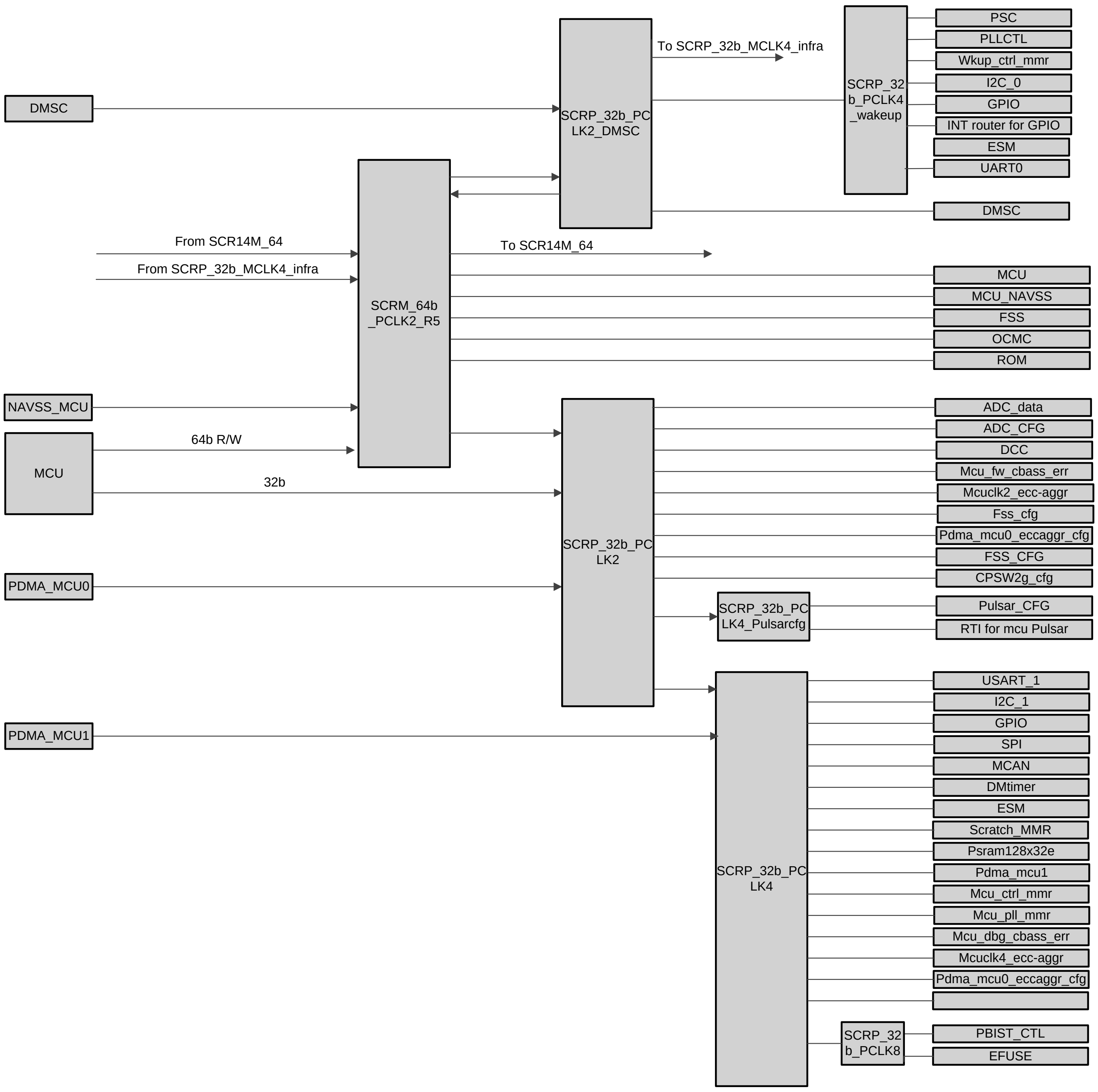

Figure 3. WKUP/MCU Interconnects

Figure 4 shows the topology view of the Main domain and end-points connected to this domain.

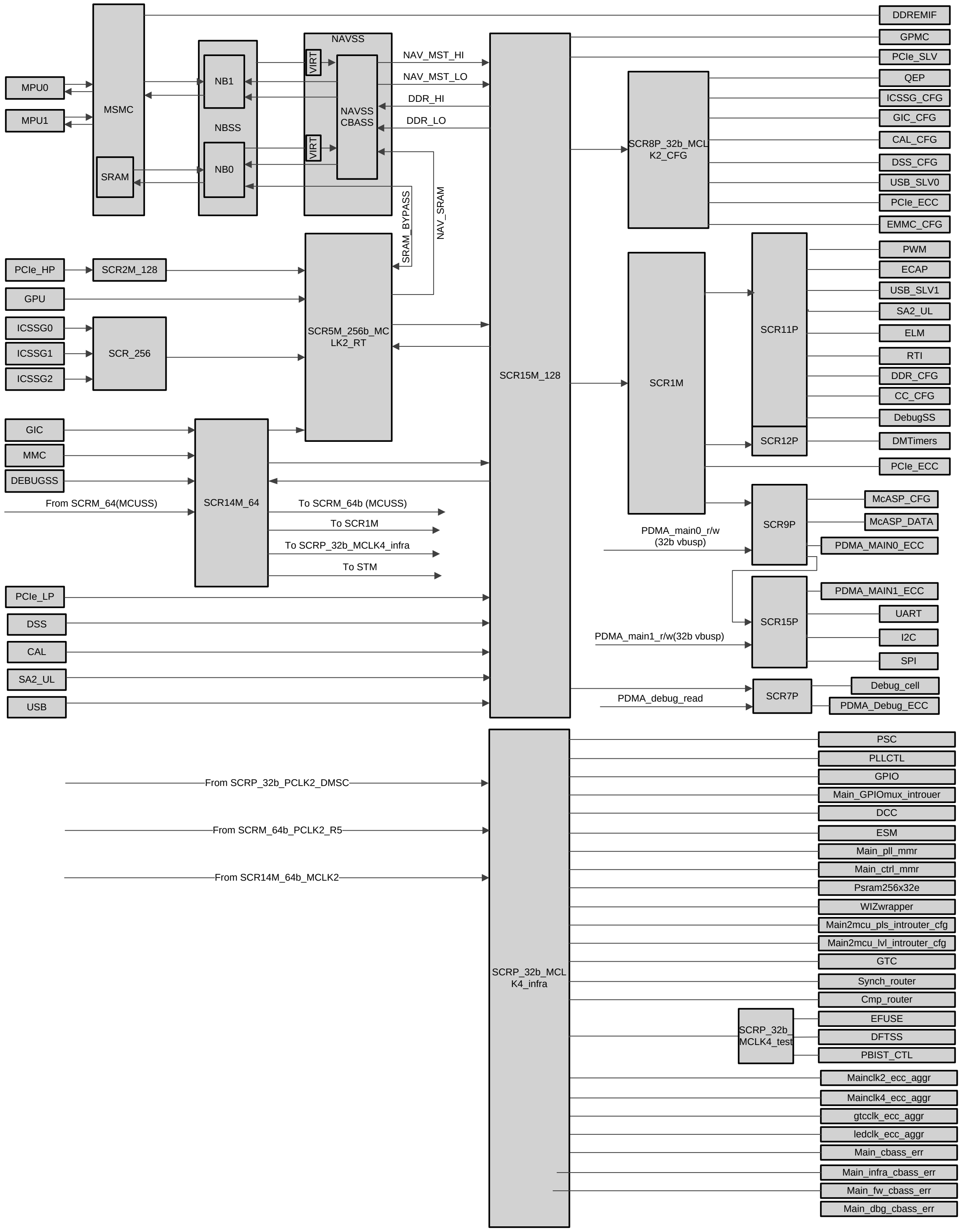

Figure 4. Main Interconnects

2.2 Interconnect Bandwidth

Table 1 shows raw bus bandwidth of key masters when connecting to their immediate slave ports in the interconnect. Table 2 gives bus bandwidth of selected slaves in the interconnect. Actual data throughput depending on specific master and slaves, as well as simultaneous bus traffic in the system. They are provided in Section 6.

Table 1. Bandwidth for System Masters

Master	To Slave Port	Bus Width (bit)	Bus Speed (Mhz)	Bandwidth (MByte/s)
ARMSS0/1	MSMC	256	800-1100	25600-35200
GPU	CBASS	256	250	8000
ICSSG	CBASS	256	250	8000
MCU	MCU_CBASS	64	200	1600
DMSC	WKUP_CBASS	32	200	800
PCIe HP	CBASS	128	250	4000
PCIe LP	CBASS	128	250	4000
DSS	CBASS	128	250	4000
USB	CBASS	128	250	4000
CAL	CBASS	128	250	4000
SA2_UL	CBASS	128	250	4000
MMC	CBASS	128	250	4000
NAVSS	MSMC RAM	256	500	16000
NAVSS	MSMC DDR	256	500	16000
CBASS (x4)	NAVSS	128	250	4 x 4000

Table 2. Bandwidth of Key Slaves

Slave	From Master Port	Bus Width (bit)	Bus Speed (Mhz)	Bandwidth (MByte/s)
MSMC SRAM	NAVSS	256	500	8000
DDREMIF	MSMC	256	500	8000
GPMC	CBASS	32	250	1000
PCIe Slave	CBASS	128	250	4000
McASP	CBASS	32	250	1000
USB Slave	CBASS	32	128	2000
ADC Data	MCU_CBASS	32	200	800
FSS (Slave 0/1)	MCU_CBASS	64	200	1600
MCAN	MCU_CBASS	32	100	800
OCMC	MCU_CBASS	64	200	1600

3 Interconnect Configurations

Sideband signals and controls are available within the interconnect to support priority, tracking, and security firewalls of the on-chip network. These controls are typically controlled by a set of memory mapped registers (MMR) and can be either preassigned, or programmable system startup time or dynamically.

Table 3. Interconnect Configuration

Controls	MMR	Location	Purpose	Programming
RouteID	Device Assigned	---	transaction tracking	---
ORDERID	QoS MMR(4-bit)	Master Endpoints	Quality-of-Service	Static/Init time
PRIORITY	QoS MMR(4-bit)	Master Endpoints	Quality-of-Service	Static/Init time
QOS	QoS MMR(4-bit, not used)	Master Endpoints	Quality-of-Service	Static/Init time
Atype	VirtID MMR/UDMA	Master/UDMA	Virtualization	Static/Dynamic
VirtID	VirtID MMR/UDMA	Master/UDMA	Virtualization	Static/Dynamic
Firewall	FW MMR	Master/Slave	Security	Static/Dynamic

The following subsections provide an overview of these configurations and the system applications of these configurations in Section 4.

3.1 Route ID (RouteID)

The RouteID is a 12-bit sideband signal used by the interconnect to identify endpoint masters to route return status and data back to the transaction initiators. Each end point master has one unique RouteID assigned to it, and there are no user level controls to reassign them. RouteID has no other usage.

3.2 Order ID (OrderID)

OrderID is a new sideband signal added to the interconnect, to order execution of the transactions and select parallel paths. Parallel paths are defined as multiple paths between a master and a designated slave port in the interconnect, where the slave address viewed from the master is the same, regardless which path the master use to address the slave. In this situation, the master uses OrderID to identify which path to take for the initiated transactions. Only a selected set of master and slaves support OrderID. OrderIDs can be configured using MMR registers associated with the master. They must be configured during system initialization and should not be dynamically changed.

3.3 Priority

Priority sets the arbitration priorities across the CBASS interconnect. The CBA protocols use a 3-bit, 8 level priority scheme, where the value of 0 means the highest priority. Additionally, the Escalated Priority (ePriority) signal is output by the master to inform the arbitration logic of the relative urgency of any pending requests on this interface. Some masters use QoS MMR to control the Priority setting.

3.4 Quality-of-Service (QOS)

The 4-bit QOS field in the QOS MMR is not used in the device interconnect. Therefore, any settings in this field will be ignored.

3.5 Address Type (Atype) and Virtualization ID (VirtID)

The interconnect supports physical and intermediate address types. Transactions are routed using physical address decoding. Each slave has one or multiple address regions assigned to it. The intermediate address needs to be translated to the physical before routing. The intermediate address is used to support the address translation method through a Peripheral Virtualization Unit (PVU). The translation table used by the PVU is identified by the Virtualization ID (VirtID). VirtID is configured by the user for a given master or DMA channel.

The following Atype values are supported in the device:

- 0: pointers are physcial address
- 1: Intermediate addresss
- 2: virtual address

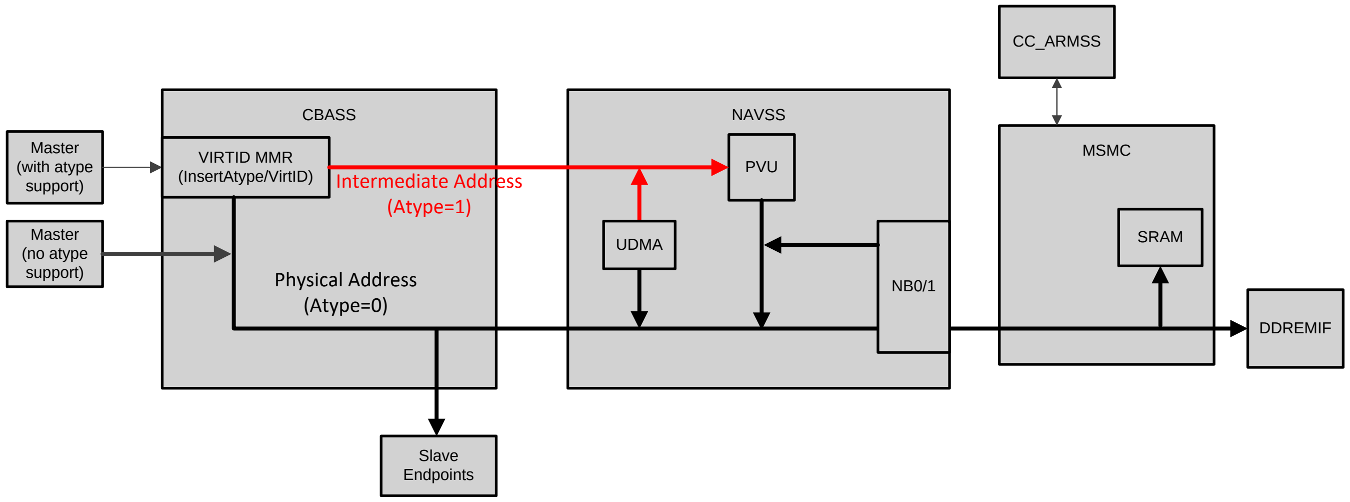
CC_ARMSS
CBASS
NAVSS
MSMC
Master (with atype support)
VIRTID MMR (InsertAtype/VirtID)
Intermediate Address (Atype=1)
PVU
Master (no atype support)
UDMA
SRAM
Physical Address (Atype=0)
NB0/1
DDREMIF
Slave Endpoints

Figure 5. Address Translation With PVU

The translation table Figure 5 shows that PVU translates the intermediate addresses issued by the masters or UDMA, then the translated physical addresses are decoded to address memory or slaves. Note that ARMSS will only issue the physical address, which is either the direct address or via MMU translations.

3.6 Firewall Controls

Firewall controls are used to protect data and configuration spaces by managing accesses to these memory regions. MMRs for firewalls are called Initiator Security Control (ISC) registers. There are firewall controls for most of the slave endpoints and selected masters in the system.

3.7 Summary of Interconnect Configurations

Table 4 gives a summary of interconnect configurations for bus masters in the device.

Table 4. Summary of Interconnect Configurations for Bus Masters

Master	Multi-Channel	ISC	QoS	VirtID/Atype	Controls	Priority/ ePriority	OrderID
CAL	Yes	Yes	Yes	Yes	QoS MMR	IP Internal FIFO	IP MMR
DebugSS	Yes	Yes	Yes	N/A			QoS MMR
DMSC	No	No	No	No			
EMMC	No	Yes	Yes	Yes			QoS MMR
ICSSG	Yes	Yes	Yes	No			QoS MMR
GPU	No	Yes	Yes	Yes			QoS MMR
DSS	Yes	Yes	Yes	Yes	QoS MMR	DSS	QoS MMR
PCIe	Yes	Yes	Yes	Yes	ReqID->VirtID	QoS MMR	QoS MMR
PDMA	Yes	No	No	No			
MCU	No	Yes	Yes	No	N/A	QoS MMR	QoS MMR
SA2_UL	No	Yes	Yes	Yes	N/A	QoS MMR	QoS MMR
USB	No	Yes	Yes	Yes	QoS MMR	QoS MMR	QoS MMR

4 End-to-End Quality of Service

For a given system, system-wide quality-of-service and resource management is governed by QoS configurations of the interconnect, MSMC bandwidth management, and DDR QoS configurations.

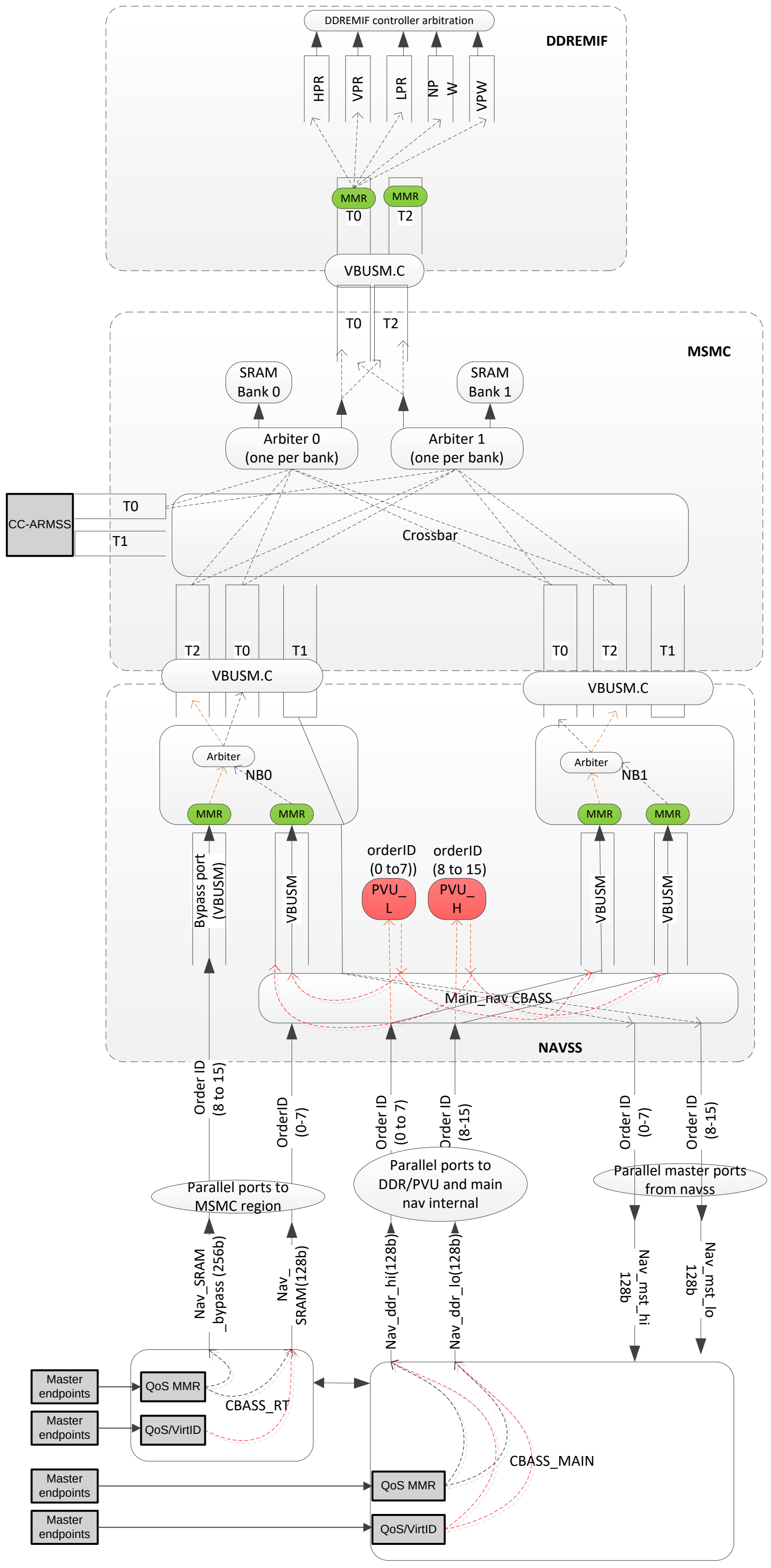

Figure 6. QoS Based Routing

Figure 6 shows data routing paths between the SoC levels CBASS, NAVSS, MSMC and DDR. NAVSS is the central module of the high-performance interconnect. It interfaces with the SoC level CBASS on the SoC side and with MSMC in the coherent fabric from ARMSS side. The DDR interfaces directly with the MSMC.

4.1 Routing Between CBASS, NAVSS and MSMC

There are a total of four parallel paths from SOC CBASS to the NAVSS: two of them dedicated to the DDR space and the other two dedicated to the RAM space in MSMC. These parallel paths are separated based on OrderID groups, where OrderID[7:0] goes to one routing path and OrderID[15:8] goes to another routing path.

There are two North bridges in NAVSS, named as NB0 and NB1, as shown in Figure 6. They serve as gateways between the SoC and the compute cluster, which contains the main application processors and coherent memory subsystems. Besides protocol conversions, NB0/1 also bridges port mapping, parallel paths mapping, muxing and fragmentation, and the reassembly function to allow MSMC to execute out of orders.

NB0 only provides the transaction flows from the SoC to the compute cluster. All of the transactions routed to the north bridges are addressed to the MSMC SRAM region. NB1 provides all the transactions addressed to DDR from the SoC level. In addition, it also provides the path for the processor to access the rest of the SoC.

Each north bridge connects to two VBUSM ingress paths (from the SoC to the Compute Cluster). Each VBUSM ingress interface can be configured to map to either thread 0 or thread 2. There is an arbiter inside the North bridge to multiplex the transactions based on thread and priority. The arbiter picks thread 2 first. If both ports have the same thread ID, it uses priority to break the tie. If there is a transaction with the same thread ID and priority level, it uses round robin. All of the transactions sent to the North bridge are fragmented into 128B burst for both read and write. The north bridge has read assembly buffers to return the read data in the order of receiving it.

Inside NAVSS, transactions with OrderID[15:8] for the SRAM path are directly connected to NB0 instead of connecting to the main NAVSS CBASS. This is to optimize access latency from the SoC level to the MSMC SRAM, such as ICSSG and PCIe. The other SRAM path combines transactions from the NAVSS master interfaces and transactions from the SoC level to SRAM. Due to the OrderID partition, no transactions to the MSMC SRAM initiated by the main NAVSS should use OrderID value 8-15, these traffic will be terminated.

All transactions from the application processors use thread 1.

MSMC provides credits for each thread. In addition, there is one credit reserved for thread 2 usage to make sure thread 2 transactions are not stalled for a long duration of time.

There are two banks SRAM in the MSMC. Each bank requires one arbiter. All transaction routed to MSMC are split into these two arbiters based on the address stride. The arbiter further determines whether the transactions are routed to the MSMC SRAM or DDR, or it needs to trigger coherent transactions.

4.2 Routing Between MSMC and DDREMIF

DDREMIF controller supports five priority levels:

- HPR: High priority read
- VPR: Variable priority read
- LPR: Low priority read
- NPW: Normal priority write
- VPW: Variable priority write

There are two independent QoS configuration registers on the DDREMIF subsystem to map transactions threads to the 5 priority queues and handle the transactions coming from these two threads from MSMC. In addition, the DDREMIF control also allows transactions from certain RouteID range maps to different sets of queues.

5 Example System Configurations

Given the flexibility of configurations, it is obvious that a common set of QoS configuration cannot address different performances and the latency requirements of each application. Instead, considerations and guidelines of a few base application profiles are provided so that you can further optimize these configuration profiles based on your detailed system.

5.1 *Industrial Networking Profile*

The Industrial Networking Profile ensures deterministic latency for traffic initiated by the PCIe High-priority (HP) master port, and deterministic latency from DMA transactions between the ICSSG firmware-based switch and the MSMC SRAM. OrderIDs of the PCIe IP port should assign between 8-15, so the bypass port can be used. It may be desirable to map this traffic to Thread 2 in the MSMC; the rest of the system traffic can use default QoS configurations.

Table 5. CBASS, NAVSS, and DDR QoS Configuration Scheme for Industrial Networking Profile Use Cases

Endpoints	Configuration	Use Case 1 Industrial NIC Card	Use Case 2 Industrial PLC
PCIe HP QOS MMR	EPRIORITY	Default	Default
PCIe HP QOS MMR	OrderID	8-15	Default
PCIe HP QOS MMR	QoS	Not used	Not used
ICSSGx QoS MMR	EPRIORITY	Default	Default
ICSSGx QoS MMR	OrderID	8-15	Default
ICSSGx QoS MMR	QoS	Not used	Not used
NB0	Port/Thread mapping	Bypass Port->Thread 2	Bypass Port->Thread 2
NB1	DDREMIF QOS mapping	Default	Default

5.2 *Multimedia Application Processor Profile*

This class of applications requires high-bandwidth image sensor data capture, MPU processing of images, graphics processing based on GPU, and then displays on the DSS interface.

Table 6. CBASS, NAVSS, and DDR QoS Configuration Scheme for Multimedia Processor Profile

Endpoints	Configuration	Guideline
CAL QOS MMR	EPRIORITY	Default
CAL QOS MMR	OrderID	8-15 (DDR path)
CAL QOS MMR	QoS	Not used
DSS QOS MMR	EPRIORITY	High
DSS QOS MMR	OrderID	8-15 (DDR path)
DSS QOS MMR	QoS	Not used
GPU QoS MMR	EPRIORITY	Default
GPU QoS MMR	OrderID	8-15 (DDR path)
GPU QoS MMR	QoS	Not used
NB0	Port/Thread mapping	Default
NB1	Port/Thread mapping	OrderID[15:8]->T2; OrderID[7:0]->T0
DDREMIF	LPR/HPR and cos_level mapping	Default + optimization [(1)]

(1) Optimization depending on image sensor data transfer patter vs. DSS resolution.

6 Subsystem Performance

In this section, latency and bandwidth data for key subsystems is provided. These data are based on combined system analysis, simulation, or actual silicon tests. Additional performance data may be added in future releases as they become available.

6.1 ARMSS

ARMSS0 and ARMSS1 are symmetrically integrated in the device, where each of the subsystem includes a dual-core Cortex A53 MPCore cluster with 512KB shared L2 cache. The two clusters can also be configured as a coherent quad-core A53, with the support of MSMC for coherency.

standard benchmarks, system latency and bandwidth performance data for these subsystem is provided.

6.1.1 System Access Latency

Table 7 gives estimated access latency from the A53 processors to the key slave endpoints in the device. These data are based on simulation and architecture analysis and, therefore, should be treated as estimates only. All read latency numbers include the round-trip delay of the command and data between the master and slave end points. All write latency numbers include both write data and write confirmation from the slave, so for peripheral write operations, actual latency for the peripheral should be about half of the value shown in the table. DDR latency numbers includes delays due to the external device, and is assumed to be DDR4 running at 1600MT/s and the page already open.

Table 7. A53 Access Latency to System Slave Endpoints (1GHz)

Slave Endpoints	Read (ns)	Write With Response
MSMC SRAM	40	40
DDR (HP)	136	98
DDR (LP)	138	100
PCIe	140	128
ICSSG	144	152
GIC	156	164
GPMC	156	168
HRPWM	---	224
EQEP	180	---
McASP	172	180

6.1.2 Instruction Cache Bandwidth

Figure 7 shows A53 bandwidth consumption due to cache misses based on the test data collected when one A53 core is running at different speed, with the program code residing in MSMC SRAM or DDR. The program code is purposely designed to execute as long linear code, with no loops, branching or data loading. This test program causes continuous cache misses and reports memory bandwidth utilization during execution.

It is observed that when the A53 core runs at 1 Ghz, placing the program code in the MSMC SRAM can support up to 2.4GB/s program fetch bandwidth. On the other hand, the program residing in DDR can support ~1.1GB/s fetch bandwidth. Therefore, for general linear control programs that require frequent cache misses, the MSMC SRAM is the preferred storage location for the code.

It is also observed that when A53 runs at a slower speed, such as 500 MHz or 250 MHz, the cache bandwidth reduces accordingly. Bandwidth utilization maximizes in tests with 32-256 KB code size. This is due to the fact that prefetch fully kicks in at these code sizes, and the effect of prefetch for larger code sizes becomes less significant compared to DDR bandwidth limitations.

The Arm prefetch memory (PRFM) was experimented on different frequencies and various sizes of using the configuration MSMC level 3 cache, but there was no significant effect on the cache bandwidth observed in these tests.

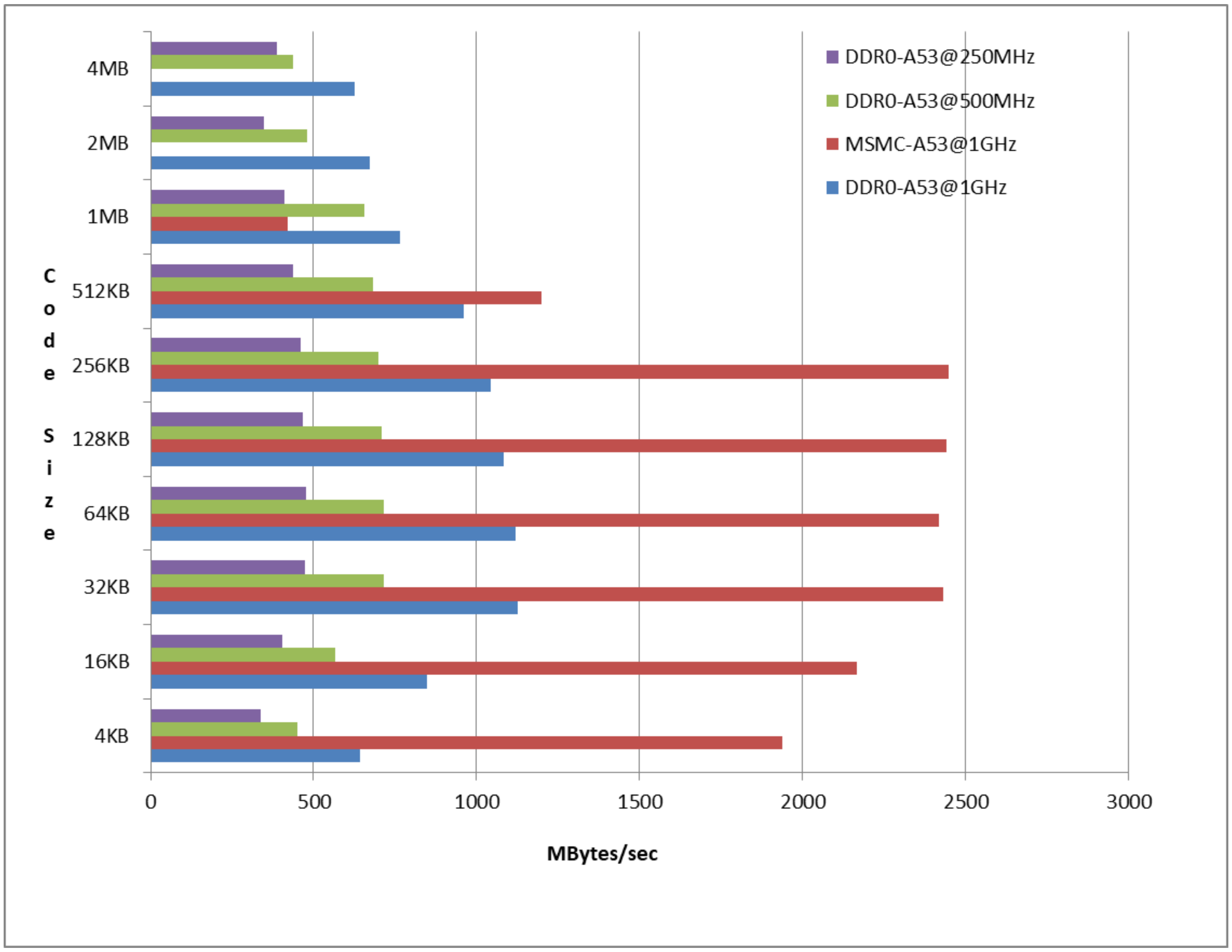

Figure 7. DDR Bandwith Consumption Due to A53 Instruction Fetch

6.1.3 Standard Software Benchmarks

Table 8 shows several standard software benchmarks. These benchmarks are collected from the initial processor SDK release, on the device EVM with A53 running at 800 MHz and DDR running at 1333 MT/s. Further details on test conditions and future updates can be accessed online at Processor SDK Performance Guide.

Table 8. Standard Benchmarks

Benchmark	Test Condition (A53/DDR)	Performance/Score
Dhrystone_per_second	800 MHz, 1333 MT/s	4,255,319
Dhrystone_per_mhz	800 MHz, 1333 MT/s	3.05
Whestone	800 MHz, 1333 MT/s	2000
Linpack	800 MHz, 1333 MT/s	333,253
LMBench: lat_mem_rd-stride128-sz50	800 MHz, 1333 MT/s	3.76
LMBench: lat_mem_rd-stride128-sz62k	800 MHz, 1333 MT/s	8.02
LMBench: lat_mem_rd-stride128-sz1000k	800 MHz, 1333 MT/s	44.43
LMBench: bw_mem-rdwr-2mb	800 MHz, 1333 MT/s	781.12 (min 474.89, max 1087.35)

6.2 DDR and MSMC SRAM Memory Bandwidth

Table 9 shows sustained memory bandwidth using DRU block transfers to the copycopy block of data from the source to the destination locations. Memory type is also shown in the table. All memory types are running at 1600MT/s data rate. Tests are performed independently without additional background traffic, and all QoS settings are default.

Table 9. Memory Throughput With DRU Transfers

Source	Destination	BLK Size	DDR Type	Throughput	Utilization
DDR	DDR	1024	DDR3L	3329	52%
MMC	DDR	1024	DDR3L	3942	62%
DDR	MSMC	1024	DDR3L	3484	54%
MSMS	MSMC	1024	MSMC	14564	Internal to MSMC

Memory bandwidth performance is also provided using UDMA transfers in Table 10. It is noted that bandwidth utilization is lower with UDMA, as it is commonly used by the peripheral to access memory.

Table 10. Memory Throughput With UDMA

Source	Destination	BLK Size (KB)	DDR Type	Bandwidth MB/s	Utilization
DDR	DDR	2048	DDR4	1009	16%
MSMC	DDR	1024	DDR4	993	16%
DDR	MSMC	1024	DDR4	544	8%
DDR	DDR	2048	LPDDR4	1057	17%
MSMC	DDR	1024	LPDDR4	993	16%
DDR	MSMC	1024	LPDDR4	575	9%
DDR	DDR	1024	DDR3L	1047	16%
MSMC	DDR	1024	DDR3L	992	16%
DDR	MSMC	1024	DDR3L	571	9%
MSMC	MSMC	512	MSMC	1971	24%

6.3 FSS

The Flash Subsystem (FSS) supports NOR flash devices using Octal-SPI, Quad-SPI, or Hyperbus interface protocols. It is typically used for fast boot or program storage for execution-in-place (XIP).

The read throughout gives worst case (100% cache misses) XIP performance when the processor cores run the program from the flash without first loading the program to internal memory. Performance of real applications in XIP will be dominated by cache miss rate. To ensure data integrity and protect code security, the inline ECC and on-the-fly decryption are supported on the Octal-SPI interface. Table 11 provides read throughput and read latency for Octal-SPI protocol. Theoretic throughput values used in the table are based on the payload data net of ECC parity and OTFA MAC codes. Differences between measured throughput and theoretic bandwidth are due to dummy clock cycles and arbitration overhead.

Table 11. OSPI Read Throughput

	Test Configuration	Raw Bandwidth (MB/s)	Theoretic Throughput	Measured Throughput (MB/s)	Measured Throughput (%)
Read	R5 core with cache enabled, reads OSPI@166MHz, Double data rate	316.6	316.6	156	49%
Read with ECC	R5 core with cache enabled, reads OSPI@166MHz, Double data rate, ECC enabled	316.6	281 (1)	158	56%
Read with OTFA	R5 core with cache enabled, reads OSPI@166MHz, Double data rate, OTFA enabled	316.6	281 (2)	128	45%
Read with OTFA+ECC	R5 core with cache enabled, reads OSPI@166MHz, Double data rate, OTFA and ECC enabled	316.6	253 (3)	105	41%

(1) ECC adds a 4-byte parity code per 32 bytes of payload data.
(2) OTFA adds a 4-byte Message Authentication Code (MAC) code per 32 bytes of payload data.
(3) ECC and OTFA adds total of 8 bytes for parity and MAC code per 32 bytes of payload data.

For control applications, it may also be interesting to understand R5 core cache latency from Octal-SPI when cache miss happens. Table 12 shows the breakdown of R5 to the OSPI IP boundary as well as the controller latency. This table assumes R5 runs at 400 MHz and OSPI runs at 166 Mhz I/O clock with double data rate. Total latency is the sum of the two segments.

Table 12. Cache Miss Latency of Octal-SPI When in XIP Mode

XIP Mode	R5 to OSPI via Interconnect (ns)	Controller Latency (ns)	Total Load to Use latency (ns)
Read	92.5	108	203
Read with ECC	92.5	470	565
Read with OTFA	92.5	518	613
Read with ECC and OTFA	92.5	602	697

When ECC or OTFA is enabled, the controller works with 32 bytes block size data to calculate parity and perform decryption; therefore, additional latency is incurred.

7 References

1. *AM654x, AM652x Sitara™ Processors Data Manual*
2. *AM65x/DRA80xM Processors Technical Reference Manual*
3. Processor SDK Performance Guide