Application Report
SPRABK5A1—July 2012

Throughput Performance Guide for C66x KeyStone Devices

High-Performance and Multicore Processors

Abstract

This document analyzes various performance measurements of the KeyStone Architecture C66x device. It provides a throughput analysis of the various DSP support peripherals as well as EDMA transfer times to different end-points and memory access.

List of Tables

List of Figures

Terminology Used In This Document

Term	Definition
AIF	Antenna Interface
BCP	Bit Coprocessor
CFG	Configuration
CPPI	Communications Port Programming Interface (now Multicore Navigator)
DSP	Digital Signal Processor
DDR	Dual Data Rate

Term	Definition
EMIF	External Memory Interface Controller
EDMA3	Enhanced Direct Memory Access v3.0
FFTC	Fast Fourier Transform Coprocessor
GPIO	General Purpose Input/Output
I2C	Inter Integrated Circuit
IDMA	Internal Direct Memory Access
LSU	Load Store Unit
MSMC	Multicore Shared Memory Controller
MPU	Memory Protection Unit
NETCP	Network Coprocessor
PA	Packet Accelerator
PCIe	Peripheral Component Interconnect Express
PDSP	Packet Data Structure Processor
PLL	Phase Locked Loop
PKTDMA	Packet DMA
QM, QMSS	Queue Manager, Queue Manager Sub-System
RAC	Receive Accelerator Coprocessor
RSA	Rake Search Accelerator
SA	Security Accelerator
SDMA	Slave Direct Memory Access
SDRAM	Synchronous Dynamic Random Access Memory
SGMII	Serial Gigabit Media Independent Interface
SPI	Serial Peripheral Interface
SRIO	Serial Rapid Input Output
TAC	Transmit Accelerator Coprocessor
TCP3d	Turbo Decode Coprocessor
TCP3e	Turbo Encode Coprocessor
TPDMA	Third Party DMA Engine
TPCC	TPDMA Channel Controller
TPTC	TPDMA Transfer Controller
UART	Universal Asynchronous Receive/Transmit
VCP2	Viterbi Decode Coprocessor
XMC	Extended Memory Controller

1 Introduction

The purpose of this document is to provide throughput performance data for Keystone Architecture C66x devices. This document provides theoretical and measured throughput performance for Keystone memories and peripherals. The peripherals presented in this document are shown in the table below. Please note that not all peripherals are supported by all devices. Please refer to the table or the device-specific data manual to determine which peripherals are supported for a particular device.

In addition to throughput performance data, this document also provides an overview of the TeraNet switch fabric implemented on Keystone C66x devices. Lastly, this document provides a basis for estimating memory access performance and presents theoretical and measured performance data achieved under various operating conditions. Some factors affecting memory access performance also are detailed.

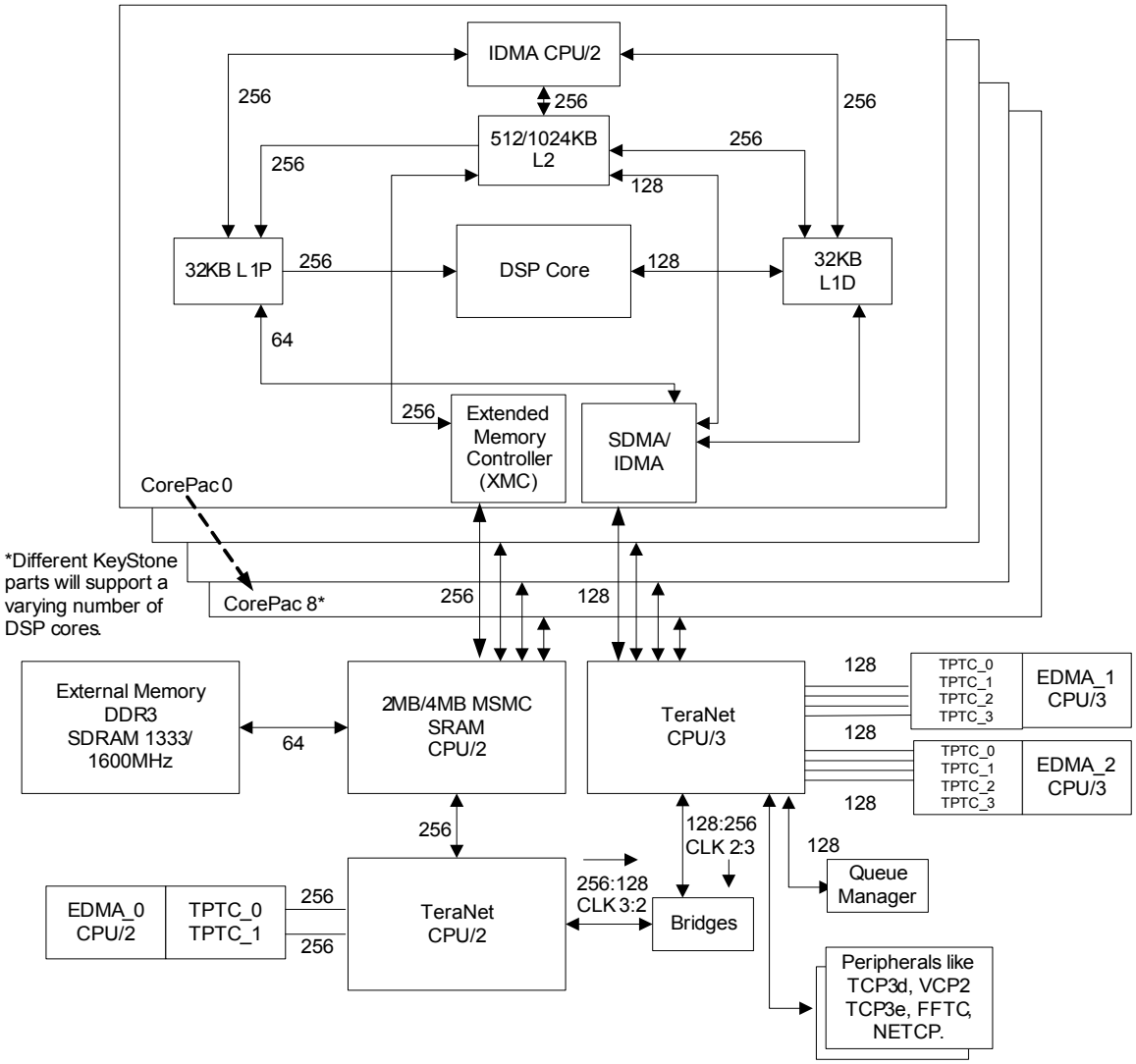
Table 1 Peripherals Supported by Part Number

	TMS320	
	C6670	C6671 C6672 C6674 C6678
"DDR3 Throughput" on page 11	x	x
"EDMA3 Complex Throughput" on page 14	x	x
"BCP Throughput" on page 18	x	
"FFTC Throughput" on page 19	x	
"Gigabit Ethernet Switch Subsystem" on page 22	x	x
"Inter-integrated Circuit (I^2C)" on page 24	x	x
"Multicore Navigator Throughput" on page 27	x	x
"Packet Accelerator (PA)" on page 35	x	x
"PCIe" on page 36	x	x
"RAC Throughput" on page 40	x	
"SPI" on page 41	x	x
"SRIO Throughput" on page 41	x	x
"TAC Throughput" on page 51	x	
"TCP3d Throughput" on page 52	x	
"TCP3e Throughput" on page 53	x	
"UART" on page 57	x	x
"VCP2 Throughput" on page 58	x	
End of Table 1		

2 KeyStone Device Overview

This section focuses on the TeraNet switch fabric that provides the interconnect between C66x DSP cores, peripherals, and memories. The high-level details provided in this section are required to understand the throughput performance of the C66x DSP, because the bus widths and the operating frequencies of each part of the TeraNet directly impact the throughput performance of the connected peripherals or memories. The C66x DSP architecture is shown in Figure 1 and shows the basic connectivity of the CorePac with the TeraNet and the various EDMA transfer controllers.

Figure 1 TeraNet and Memory Access Diagram



2.1 C66x CorePac Overview

This section provides the fundamental details of the C66x CorePac. Additional details about the C66x CorePac can be found in the C66x CorePac User Guide. The C66x CorePac consists of several components including:

- C66x DSP core
- L1 and L2 memories
- external memory controller
- extended memory controller (XMC)
- interrupt controller
- power/sleep controller
- embedded trace buffer
- data trace formatter
- RSA accelerator (core 1 and 2 only)

Each CorePac has the ability to sustain up to 128 bits of load/store operations per cycle to L1D memory and is capable of handling up to 16 GB/second. When accessing data in the L2 memory or external memory, the access rate depends on the memory access pattern and the cache.

Within each CorePac, there is an internal DMA (IDMA) engine that can move data at the frequency rate of DSP/2 with a data width of 256 bits. The IDMA operates in the background of the DSP core activity (i.e., data can be brought into buffer A while the DSP core is accessing buffer B). The IDMA can transfer data only between the L1 and L2 memories and peripheral configuration ports; it cannot access external memory.

2.2 TeraNet Overview

The TeraNet switch fabric provides interconnection between the C66x cores (and their local memories), external memory, the enhanced DMA v3 (EDMA3) controllers, Multicore Navigator, on-chip coprocessors, and high-speed IO. The TeraNet switch fabric allows each of these to operate at maximum efficiency with no blocking or stalling. It allows for concurrent transfers between non-conflicting master/slave pairs and can support a very high total data rate across any endpoint. If transfers line up such that the source or destination memory is the same, then collisions occur and certain transactions will be blocked.

The TeraNet consists of data switch fabric and configuration switch fabric:

- **Data Switch Fabric:** The data switch fabric mainly moves data across the system and is further subdivided into two smaller switch fabrics. One connects high speed masters to slaves via 256-bit data buses running at the DSP/2 frequency. The second data switch fabric connects high-speed masters to slaves via 128-bit data buses running at a DSP/3 frequency.
- **Configuration Switch Fabric:** This switch fabric is mainly used to access peripheral registers. It connects the C66x CorePac and masters on the data switch fabric to slaves via 32-bit configuration buses running at either DSP/3 or DSP/6 frequency.

C66x devices contain three EDMA Channel Controllers: TPCC0, TPCC1, and, TPCC2. Each TPCC can be programmed to move data concurrently in the background without the expense of any DSP cycles. These TPCCs can move data between on-chip L1 and L2 memory, MSMC SRAM, external memory, and the peripherals on the device that support EDMA-based transfer.

The TeraNet also provides connection to the Multicore Navigator which uses a Queue Manager Subsystem (QMSS) and a Packet DMA (PKTDMA) to control and implement high-speed data packet movement within the device. Frequent tasks are commonly offloaded from the host processor to peripheral hardware to increase system performance.

The TeraNet also provides a 64-bit DDR3 interface to support access to external memories that can be used for either data or program memory.

2.3 Memory Access System Overview

This section discusses the C66x memory system. Memory access is critical for applications running on the DSP. Some memories are internal to the CorePac, while other system memories are external to the CorePac. The memory system also provides a 64-bit DDR3 interface for accessing off-chip memory. Internal to each CorePac are the following memories:

- L1D SRAM that can be used as either data memory or data cache or both.
- L1P SRAM that can be used as either program memory or cache or both.
- Local L2 SRAM that can be used as either unified SRAM or unified cache or both.

L1D SRAM, L1P SRAM, L2 SRAM, MSMC SRAM, and DDR3 memories are accessible by all the cores and multiple DMA masters on the device.

External to the CorePac, the C66x DSPs have multicore shared memory (MSMC) SRAM, which is shared between the cores. The MSMC memory can be configured in two ways:

- **SL2 mode**: Shared L2 SRAM mode: L1P/L1D memory will cache MSMC, whereas L2 will not cache requests to MSMC SRAM.
- **SL3 mode**: Level 3 SRAM mode: Both L1P/L1D and L2 memories will cache the MSMC SRAM if it is remapped to an external address using the address extension unit. (For more information on this, see the Multicore Shared Memory Controller (MSMC) for KeyStone Devices User Guide.

A 64-bit EMIF interface is provided for accessing off-chip DDR3 SDRAM, which can be used as data or program memory. Although this interface supports a 64-bit data bus, it can also be configured to operate using a 32-bit or 16-bit data bus.

3 Memory Access Throughput Performance

This section discusses the memory access throughput performance of C66x devices. The bandwidth of a memory copy is determined by the lowest of these three factors:

- Bus bandwidth
- Source throughput
- Destination throughput

Table 2 summarizes the maximum theoretical bandwidth of the C66x core, IDMA, and the EDMA when the device is operating at 1 GHz.

Table 2 Theoretical Bandwidth of Core, IDMA and EDMA

Master	Maximum Bandwidth MB/s	Comments
C66x Core	16000	(128bits)/(8bit/byte)*(1000M)=16000MB/s
IDMA	16000	(256bits)/(8bit/byte)*(1000M/2)=16000MB/s
EDMA0(Single TC)	16000	(256bits)/(8bit/byte)*(1000M/2)=16000MB/s
EDMA1(Single TC)	5333	(128bits)/(8bit/byte)*(1000M/3)=5333MB/s
EDMA2(Single TC)	5333	(128bits)/(8bit/byte)*(1000M/3)=5333MB/s

Table 3 summarizes the maximum theoretical throughput of different memories when the C66x device is operating at 1 GHz. The DDR3 performance assumes that a 64-bit bus width is used and that the external memory is operating at 1333 MHz.

Table 3 Theoretical Bandwidth of Different Memories

Master	Maximum Bandwidth MB/s	Comments
L1D	32000	(256bits)/(8bit/byte)*(1000M)=32000MB/s
L1P	32000	(256bits)/(8bit/byte)*(1000M)=32000MB/s
L2	16000	(256bits)/(8bit/byte)*(1000M/2)=16000MB/s
MSMC RAM	64000	4*(256bits)/(8bit/byte)*(1000M/2)=64000MB/s
DDR3 RAM	10664	(64bits)/(8bit/byte)*(666.5M)*2=10664MB/s

When the C66x core tries to read from or write to different memory endpoints, it consumes some DSP cycles to access the memory region depending on various factors such as prefetching, caching, victim buffer hits, and so on. Section 3.1 and Section 3.2 estimate the number of DSP stalls for accessing different memory endpoints.

Note that MSMC SRAM has 4 memory banks of 256 bits. As mentioned previously, MSMC can be configured in either shared L2 (SL2) mode or shared L3 (SL3) mode.

3.1 Memory Read Performance

The C66x core has an improved pipeline between L1D/L1P and L2 memory controller and this significantly reduces the stall cycles for L1D/L1P cache misses.

Table 4 shows the DSP stalls for accessing different memories when the core tries to read from memory.

Table 4 Memory Read Performance

				DSP Stalls (in Cycles)			
				Single Read		Burst Read	
Source	**L1 Cache**	**L2 Cache**	**Prefetch**	**No Victim**	**Victim**	**No Victim**	**Victim**
ALL	Hit	NA	NA	0	NA	0	NA
Local L2 SRAM	Miss	NA	NA	7	7	3.5	10
MSMC RAM (SL2)	Miss	NA	Hit	7.5	7.5	7.4	11
MSMC RAM (SL2)	Miss	NA	Miss	19.8	20.1	9.5	11.6
MSMC RAM (SL3)	Miss	Hit	NA	9	9	4.5	4.5
MSMC RAM (SL3)	Miss	Miss	Hit	10.6	15.6	9.7	129.6
MSMC RAM (SL3)	Miss	Miss	Miss	22	28.1	11	129.7
DDR RAM (SL2)	Miss	NA	Hit	9	9	23.2	59.8
DDR RAM (SL2)	Miss	NA	Miss	84	113.6	41.5	113
DDR RAM (SL3)	Miss	Hit	NA	9	9	4.5	4.5
DDR RAM (SL3)	Miss	Miss	Hit	12.3	59.8	30.7	287
DDR RAM (SL3)	Miss	Miss	Miss	89	123.8	43.2	183
End of Table 4							

For example, when there is a L1 cache miss, it takes seven DSP cycles for the core to access its local L2 SRAM. Similarly, where there is a L1 cache miss and when the MSMC is configured as SL2/SL3, the DSP stalls depend on whether there is a Hit/Miss in the prefetch buffer. Prefetching reduces the latency gap between local memory and shared (internal/external) memories. Prefetching in the extended memory controller (XMC) helps reduce stall cycles for read accesses to the MSMC and the DDR3 EMIF.

The performance is affected when both L1 and L2 caches contain victims. So, when the MSMC or DDR3 is configured as SL3 memory, it can have a potential double victim performance impact where the victim cache lines have to be evicted from both L1 and L2 caches. When victims are in the cache, burst reads are slower than single reads because reads have to wait for victim write-backs to complete.

The numbers show that MSMC access is slower when it is configured as SL3 instead of SL2. There is a potential double victim involved. This holds true for DDR3 as well.

If DDR3 does not have large cacheable data, it is recommended that it be configured as SL2.

3.2 Memory Write Performance

The C66x CorePac has improved write merging and optimized burst sizes that reduce the stalls to external memory.

The L1D memory controller merges writes, not only to L2 SRAM, but to any address that is allowed to be cached (MAR.PC==1).

Table 5 shows the DSP stalls for accessing different memories when the core tries to write to memory. There are no stalls when the DSP does a single write operation.

Table 5 Memory Write Performance

				DSP Stalls (in Cycles)			
				Single Write		Burst Write	
Source	**L1 Cache**	**L2 Cache**	**Prefetch**	**No Victim**	**Victim**	**No Victim**	**Victim**
ALL	Hit	NA	NA	0	NA	0	NA
Local L2 SRAM	Miss	NA	NA	0	0	1	1
MSMC RAM (SL2)	Miss	NA	Hit	0	0	2	2
MSMC RAM (SL2)	Miss	NA	Miss	0	0	2	2
MSMC RAM (SL3)	Miss	Hit	NA	0	0	3	3
MSMC RAM (SL3)	Miss	Miss	Hit	0	0	6.7	14.6
MSMC RAM (SL3)	Miss	Miss	Miss	0	0	6.7	16.7
DDR RAM (SL2)	Miss	NA	Hit	0	0	4.7	4.7
DDR RAM (SL2)	Miss	NA	Miss	0	0	5	5
DDR RAM (SL3)	Miss	Hit	NA	0	0	3	3
DDR RAM (SL3)	Miss	Miss	Hit	0	0	16	114.3
DDR RAM (SL3)	Miss	Miss	Miss	0	0	18.2	115.5
End of Table 5							

4 DDR3 Throughput

Applies to All KeyStone Devices

The DDR3 module in the C66x DSP is accessible across all cores and other system masters. The DDR3 controller interfaces with most standard DDR3 SDRAM devices. There are two clock domains in the controller. The Command FIFO, Read FIFO and Write FIFO are all on the DSP/2 clock domain. The Memory Mapped Registers (MMR), state machine, and interface to the DDR PHY are all driven by the DDR3 memory clock. The DDR3 controller is interfaced to the MSMC controller and any access made via any of the EDMAs are handled by MSMC which serves as the common management path.

4.1 DDR3 Command Interleaving Latency

The DDR3 controller supports interleaving of commands for maximum efficiency, i.e., the controller will partially execute one command and switch to executing another high priority command before finishing the first command. The READ latencies from DDR3 to L2 and WRITE latencies from L2 to DDR3 are independently run and tested.

The test case details are as follows:

- Testing is done on a 1 GHz TMS320C6670 DSP with DDR3 frequency at 1600 MHz.
- The EDMA configuration used across the tests is an array of contiguous sequences of memory of size 1K bytes.
- Test runs from CorePac0 with L2 memory as one endpoint and DDR3 as the other.
- Data page size is 1KB for different DDR3 modes of operation.
- 2 buffers are set up and populated in DDR3 such that they lie in the same page.
- 2 buffers are set up and populated in DDR3 such that they lie in a different page but in the same bank.

The test process includes the following steps:

1. Wait for the completion of the refresh interval for about 1 ms in a tight loop.
2. Configure two channels (four for 64-bit) on different TCs such that they perform READ/WRITE transaction from the allocated buffers.
3. Trigger channels simultaneously and wait for the completion of both.
4. Aggregate throughput for the transfers is measured.

Table 6 shows the latencies for DDR3 in different modes for a READ command interleave.

Table 6 DDR3 READ Command Interleave Cycle Counts

	Same Page EDMA0 (TC0 and TC1)	Different Page EDMA0 (TC0 and TC1)	Same Page EDMA1 (TC2 and TC3)	Different Page EDMA1 (TC2 and TC3)
CPU Cycle Count (64 bit)	1170	1172	1114	1173
CPU Cycle Count (32 bit)	920	1001	774	768
CPU Cycle Count (16 bit)	1424	1505	981	1043

Table 7 shows the latencies for DDR3 in different modes for interleaved WRITE command.

Table 7 DDR3 WRITE Command Interleave Cycle Counts

	Same Page EDMA0 (TC0 and TC1)	Different Page EDMA0 (TC0 and TC1)	Same Page EDMA1 (TC2 and TC3)	Different Page EDMA1 (TC2 and TC3)
CPU Cycle Count (64 bit)	1338	1422	1121	1121
CPU Cycle Count (32 bit)	835	1000	705	768
CPU Cycle Count (16 bit)	1340	1504	981	1043

4.2 DDR3 Class Of Service Throughput

The commands in the Command FIFO can be mapped to two classes of service (CoS), namely 1 and 2. The mapping based on priority can be done by setting the appropriate values in the Priority to Class of Service Mapping register. By default, all commands will be mapped to Class of Service 2.

The tests were measured on a TMS320C6670 DSP operating at 1 GHz. The scenario details to test the effect of Class of Service assignment to commands for EDMA based transfers are listed below:

- TMS320C6670 DDR3 frequency at 1600 MHz.
- The EDMA configuration used across the tests is an array of contiguous sequences of memory of size 1K bytes.
- Test runs from CorePac0 with L2 memory as one endpoint and DDR3 as the other.
- Data page size is 1KB for different DDR3 modes of operation.
- Two buffers are set up and populated in DDR3 such that they lie in the same page.
- Wait for the completion of the refresh interval for about 1 ms in a tight loop.

The test process includes the following steps:

1. Configure two channels (four for 64-bit) on different TCs such that they perform READ/WRITE transaction from the allocated buffers.
2. Trigger channels simultaneously and wait for the completion of TC1.
3. TC1 throughput for the transfers is measured.
4. The test is run for 3 different CoS settings and the results are compared against each other: Default, TC1 at minimum latency, and TC0 and minimum latency.

Table 8 shows the aggregate throughput for DDR3 in different modes for a READ command priority.

Table 8 DDR3 READ Command Priority Throughput

	Default CoS Settings	TC0 Maximum Latency TC1 Minimum Latency	TC0 Minimum Latency TC1 Maximum Latency
TC1 Throughput in Mbps (64 bit)	2311.5	2348.6	1965.5
TC1 Throughput in Mbps (32 bit)	1943.1	2354	1000
TC1 Throughput in Mbps (16 bit)	1475.5	2354	604.1

Table 9 shows the aggregate throughput for DDR3 in different modes for a WRITE command priority.

Table 9 DDR3 WRITE Command Priority Throughput

	Default CoS Settings	TC0 Maximum Latency TC1 Minimum Latency	TC0 Minimum Latency TC1 Maximum Latency
TC1 Throughput in Mbps (64 bit)	1946.8	1969.2	1973
TC1 Throughput in Mbps (32 bit)	1950.5	1969.2	1692.6
TC1 Throughput in Mbps (16 bit)	1477.6	1692.6	603.4

5 EDMA3 Complex Throughput

Applies to All KeyStone Devices

Various complex throughput tests were performed for measuring the EDMA throughput under different conditions. The following tests calculate the EDMA throughput without any background traffic in the CorePac.

The test process includes the following steps:

1. Get the transfer time for a payload of 32KB/channel (includes overhead).
2. Get the transfer time for 0 bytes - this is called a dummy transfer and closely approximates the overhead.
3. Subtract 2 from 1 to get the transfer time with overhead removed - t3.
4. Throughput = [(32KB * number of channels)/t3] * 1GHz

The basis of these tests is to set up and trigger parallel read and write data transfers bidirectionally for different combinations of TCs of the EDMA. The channel controller is programmed such that each TC performs one read and one write transfer between the TC endpoints bidirectionally on two different logical channels. Once TCs are triggered, wait (poll) for the completion of all transfers and capture the throughput values.

The EDMA configuration uses A-synchronized transfers across all tests and TC combinations with ACNT = 32KB and BCNT = CCNT = 1 per logical channel.

Note that each of the TCs have bidirectional transfers —from endpoint A to endpoint B and the other from endpoint B to endpoint A on two different logical channels. For example, for the TC0 and TC1 testcase, there are a total of four transfers on four different logical channels.

The test case descriptions are as follows:

- **TC0 and TC1 test case:** Trigger TC0 and TC1 in parallel. After the completion of the TC0 and TC1 transfers in both directions, the steady state throughput is calculated.
- **TC2 and TC6 test case:** Trigger TC2 first and after a delta lag TC6 is triggered. After the completion of TC2 and TC6 transfers in both directions, the steady state throughput is calculated.
- **TC3 and TC7 test case:** Trigger TC3 first and after a delta lag TC7 is triggered. After the completion of TC3 and TC7 transfers in both directions, the steady state throughput is calculated.
- **TC3, TC5, TC7, and TC8 test case:** First, TC3 and TC5 are triggered in parallel. After a delta lag, TC7 and TC8 are triggered in parallel. After the completion of all TC transfers in both directions, the steady state throughput is calculated.

Please note that TC0 through TC8 in this document correlate to the Data Manual nomenclature in the following way:

TC0 = EDMA0 TC0
TC1 = EDMA0 TC1
TC2 = EDMA1 TC0
TC3 = EDMA1 TC1
TC6 = EDMA2 TC0
TC7 = EDMA2 TC1
TC8 = EDMA2 TC2

5.1 Scenario 1: EDMA Transfer Between CorePac0 L2 and CorePac1 L2

As mentioned above, the throughput is limited by the lowest of the bus bandwidth, source throughput, and destination throughput.

The TeraNet allows for concurrent transfers between non-conflicting master/slave pairs and can support a very high total data rate across any endpoint. If transfers line up such that the source or destination memory is the same, then collisions occur and certain transactions will be blocked.

In this scenario, the endpoints used are CorePac0 L2 SRAM and CorePac1 L2 SRAM memory. There is contention among TCs as the same endpoint is used by both TCs for each of the bidirectional transfers. The TCs have to use the same SDMA/IDMA port of the same CorePac and the second transaction will have to wait until the first is complete.

So only one TC transfer is done at a time and the theoretical throughput is equivalent to (128bits)/(8bit/byte)×(1000M/3)=5333MB/s. Note that EDMA0 will access the CorePac0 using the DSP/3 128-bit TeraNet switch fabric similar to EDMA1 and EDMA2.

Table 10 shows the measured aggregate throughput of the TCs for this scenario.

Table 10 EDMA Throughput Between CorePac0 L2 and CorePac1 L2 Endpoints

Transaction on TCs	TC0 and TC1	TC2 and TC6	TC3 and TC7	TC3, TC5, TC7, and TC8
Wait for completion of	Both	Both	Both	All
Theoretical Max throughput between TC endpoints (MBPS)	5333.3	5333.3	5333.3	5333.3
Aggregate throughput of TCs fired (MBPS)	5201.3	5276.6	5283.5	5322.5

The testcase setup details are as follows:

Every TC sets up bidirectional data transfer between the same CorePac0 L2 and CorePac1 L2.

5.2 Scenario 2: EDMA Transfer Between Different CorePac L2 and MSMC SRAM

In this scenario, each TC uses a different CorePac source address and there is no contention between TCs. For example TC0 reads from CorePac0 L2, whereas TC1 reads from CorePac1 L2. So the total throughput for any two TCs would be 2× [(128bits)/(8bit/byte)×(1000M/3)] = 10666.66MB/s.

Note—For TC3, TC5, TC7, and TC8 it cannot be
4 × [(128bits)/(8bit/byte)×(1000M/3)]
because it is limited by the MSMC maximum throughput, which is 16000MB/s.

Table 11 shows the measured aggregate throughput of the TCs for this scenario.

Table 11 EDMA transfer between different CorePac L2 and MSMC SRAM

Transaction on TCs	TC0 and TC1	TC2 and TC6	TC3 and TC7	TC3, TC5, TC7, and TC8
Wait for completion of	Both	Both	Both	All
Theoretical Max throughput between TC endpoints (MBPS)	10667	10667	10667	16000
Aggregate throughput of TCs fired (MBPS)	10472.4	10437.3	10553.3	15896.2

The testcase setup details are as follows:

- **TC0 and TC1:** TC0 has bidirectional data transferred between CorePac0 L2 and MSMC. TC1 has bidirectional data transferred between CorePac1 L2 and MSMC.

- **TC2 and TC6:** TC2 has bidirectional data transferred between CorePac0 L2 and MSMC. TC6 has bidirectional data transferred between CorePac1 L2 and MSMC.
- **TC3 and TC7:** TC3 has bidirectional data transferred between CorePac0 L2 and MSMC. TC7 has bidirectional data transferred between CorePac1 L2 and MSMC.
- **TC3, TC5, TC7, and TC8:** Each TC uses a different CorePac to transfer bidirectional data between MSMC and CorePac For example, TC3 uses CorePac0, TC5 uses CorePac1, TC7 uses CorePac2, and TC8 uses CorePac3.

5.3 Scenario 3: EDMA Transfer From MSMC SRAM to MSMC SRAM

In this scenario there is contention between TCs because the source/destination are the same for different TCs, so there is only one transfer happening at a time. Even though the theoretical throughput for MSMC is (256bits)/(8bit/byte)×(1000M/2)=16000MB/s, it cannot do both reads and writes at the same time. It has to wait for the previous read from MSMC to complete and then do the write operation to the MSMC. Therefore, the throughput will be effectively halved and is 16000/2 = 8000MB/s.

Table 12 shows the measured aggregate throughput of the TCs for this scenario.

Table 12 EDMA Throughput Between MSMC SRAM and MSMC SRAM

Transaction on TCs	TC0 and TC1	TC2 and TC6	TC3 and TC7	TC3, TC5, TC7, and TC8
Wait for completion of	Both	Both	Both	All
Theoretical Max throughput between TC endpoints (MBPS)	8000	8000	8000	8000
Aggregate throughput of TCs fired (MBPS)	7961.6	7882.1	7981.5	7981.5

The testcase setup details are as follows:

Every TC sets up bidirectional data transfer between MSMC and MSMC SRAM.

5.4 Scenario 4: EDMA Transfer From MSMC SRAM to DDR3

There is no contention between TCs, but the throughput is limited by the maximum DDR3 bandwidth, which is (64bits)/(8bit/byte)×(666.5M)×2=10664MB/s. Table 13 shows the measured aggregate throughput of the TCs for this scenario.

Table 13 EDMA Throughput Between MSMC SRAM and DDR3

Transaction on TCs	TC0 and TC1	TC2 andTC6	TC3 and TC7	TC3, TC5, TC7, and TC8
Wait for completion of	Both	Both	Both	All
Theoretical Max throughput between TC endpoints (MBPS)	10664	10664	10664	10664
Aggregate throughput of TCs fired (MBPS)	10132.3	10104.2	10193.8	10152.7

The testcase setup details are as follows:

Every TC sets up bidirectional data transfer between the MSMC and DDR3 SRAM.

5.5 Scenario 5: EDMA Transfer From DDR3 to DDR3

There is no contention between TCs, but because the DDR3 EMIF cannot do a read from DDR3 and a write to DDR3 in parallel, the EDMA needs to wait until the previous operation is completed. The DDR3 throughput of 10664MB/s will be halved and will be equivalent to 5332 MB/s. The reason for the low throughput in this case is due to read-write turnaround and page switch overheads. Table 14 shows the measured aggregate throughput of the TCs for this scenario.

Table 14 EDMA Throughput Between DDR3 and DDR3

Transaction on TCs	TC0 and TC1	TC2 and TC6	TC3 and TC7	TC3, TC5, TC7, and TC8
Wait for completion of...	Both	Both	Both	All
Theoretical Max throughput between TC endpoints (MBPS)	5332	5332	5332	5332
Aggregate throughput of TCs fired (MBPS)	3960.5	4138.5	3472.7	4351.9

The testcase setup details are as follows:

Every TC sets up bidirectional data transfer between DDR3 and DDR3 SRAM.

5.6 Scenario 6: EDMA Transfer From Different CorePac L2 to DDR3

In this scenario, each TC uses a different CorePac source address and there is no contention between TCs. However, the throughput is limited by the DDR3 maximum throughput, which is 10664MB/s. Table 15 shows the measured aggregate throughput of the TCs for this scenario.

Table 15 EDMA Throughput Between Different CorePac L2s to DDR3

Transaction on TCs	TC0 and TC1	TC2 and TC6	TC3 and TC7	TC3, TC5, TC7, and TC8
Wait for completion of...	Both	Both	Both	All
Theoretical Max throughput between TC endpoints (MBPS)	10664	10664	10664	10664
Aggregate throughput of TCs fired (MBPS)	10132.3	10158.3	10222.4	10158.3

The testcase setup details are as follows:

- **TC0 and TC1:** TC0 has bidirectional data transferred between CorePac0 L2 and DDR3. TC1 has bidirectional data transferred between CorePac1 L2 and DDR3.
- **TC2 and TC6:** TC2 has bidirectional data transferred between CorePac0 L2 and DDR3. TC6 has bidirectional data transferred between CorePac1 L2 and DDR3.
- **TC3 and TC7:** TC3 has bidirectional data transferred between CorePac0 L2 and DDR3. TC7 has bidirectional data transferred between CorePac1 L2 and DDR3.
- **TC3, TC5, TC7, and TC8:** Each TC uses a different CorePac to transfer bidirectional data between DDR3 and CorePac. i.e., TC3 uses CorePac0, TC5 uses CorePac1, TC7 uses CorePac2, and TC8 uses CorePac3.

6 BCP Throughput

Applies to C6670

The Bit Rate Coprocessor (BCP) is a programmable peripheral for baseband bit processing. It supports FDD LTE, TDD LTE, WCDMA, TD-SCDMA, HSPA, HSPA+, WiMAX 802.16e, and monitoring/planning for LTE-A. The BCP accelerates the Layer1 Downlink processing from transport blocks down to OFDM symbols (LTE, WiMAX) or physical channel data (WCDMA, TD-SCDMA, HSPA/+). It also accelerates Layer1 uplink processing from OFDM symbols to transport blocks for LTE/WiMAX (with the help of Turbo and Convolutional Decoder Accelerators) or the physical channel data to the output of rate dematching or LLR combining for WCDMA, TD-SCDMA, or HSPA/+.

The BCP has three VBUSM master ports for data movement. One is a 128-bit Navigator interface and there are two 128-bit direct I/O interfaces. The BCP runs at DSP/3 clock frequency.

The testcase details for measuring the throughput are as follows:

- Testing is done on a 1GHz TMS320C6670 DSP with DDR3, MSMC SRAM and L2 SRAM as memory endpoints for both input and output data.
- Downlink and Uplink throughput tests are run independently of each other.
- All the packets are enqueued by setting up the descriptors (Host descriptors used for the tests). After enqueing, the logical transmit channel is enabled to initiate transfer.
- Steady-state throughput measurement is performed to remove software overhead by enqueuing 20 input blocks at a time. Once the transmit channel is enabled, the average time taken to process an input block is calculated by averaging the total time taken to process the 20 input blocks.
- Throughput is calculated by dividing the average block processing time by the input blocks size.

BCP LTE Uplink Throughput

Table 16 through Table 17 show the BCP LTE (PUSCH and PDSCH) and WCDMA (HSUPA and HSDPA) throughputs for minimum and maximum transport block sizes, different modulation schemes, and different memory endpoints.

Table 16 BCP LTE Throughput (Part 1 of 2)

Channel	PKTDMA Memory	HARQ Memory	TB Size	Modulation Scheme	Throughput (Mbps)
PUSCH	L2 SRAM	MSMC	136	QPSK	241
	L2 SRAM	DDR3	136	QPSK	241
	DDR3	DDR3	136	QPSK	231
	L2 SRAM	MSMC	75376	64-QAM	846
	L2 SRAM	DDR3	75376	64-QAM	498
	DDR3	DDR3	75376	64-QAM	455

Table 16 BCP LTE Throughput (Part 2 of 2)

Channel	PKTDMA Memory	HARQ Memory	TB Size	Modulation Scheme	Throughput (Mbps)
PDSCH	L2 SRAM	N/A	136	QPSK	278
	MSMC	N/A	136	QPSK	276
	DDR3	N/A	136	QPSK	288
	L2 SRAM	N/A	440	64-QAM	1272
	MSMC	N/A	503	64-QAM	1272
	DDR3	N/A	500	64-QAM	1272
End of Table 16					

Table 17 BCP WCDMA Throughput

Channel	PKTDMA Memory	HARQ Memory	TB Size	Modulation Scheme	Throughput (Mbps)
HSUPA	L2 SRAM	MSMC	18	BPSK	21
	L2 SRAM	DDR3	18	BPSK	21
	DDR3	DDR3	18	BPSK	21
	L2 SRAM	MSMC	22996	4-PAM	332
	L2 SRAM	DDR3	22996	4-PAM	332
	DDR3	DDR3	22996	4-PAM	332
HSDPA	L2 SRAM	N/A	120	N/A	196
	MSMC	N/A	120	N/A	189
	DDR3	N/A	120	N/A	189
	L2 SRAM	N/A	42192	N/A	654
	MSMC	N/A	42192	N/A	654
	DDR3	N/A	42192	N/A	654
End of Table 17					

7 FFTC Throughput

Applies to C6670

The FFTC accelerators on the DSP are used to perform Fourier Transforms and Inverse Fourier Transforms on data. The Packet DMA (PKTDMA) in the FFTC moves data in and out of the accelerator and is directly connected to the 128 bit DSP/3 data switch fabric. The Packet DMA engine and the Queue Manager work together to allow shared access through message passing. (Refer to the Multicore Navigator for KeyStone Devices User Guide for additional information of the Packet DMA.)

Each FFTC has four dedicated transmit queues. To send data to the FFTC, a Packet DMA *packet* is placed on one of the four available queues. Each packet can contain one or more FFT blocks (or tranforms). The built-in scheduler receives the status of each queue and selects packets from the four queues based on the priority. After processing, the FFTC places the results on receive queues as specified by the user.

The throughput of FFTC is measured by how many subcarriers it can support. It is equal to the DFT (FFT) size divided by the FFTC time taken to compute a single block. The capacity depends on various factors, such as DSP frequency and DFT size.

The FFTC tests were measured on a TMS320C6670 DSP operating at 1 GHz. The throughput test details are as follows:

- L2 memory is used as the endpoint for both input and output data.
- Each Packet DMA packet has 3 FFT blocks to keep the FFTC engine pipeline busy. The FFTC is receiving data for one block, computing the transform for another block and transmitting output data for the third block continuously.
- Multiple packets are sent so that the throughput can be maximized.
- LTE Frequency Shift is not enabled in the testing. Note that the throughput will be reduced when it is enabled.
- The steady state processing time shown in Table 18 is the latency of the FFTC engine per block alone. The overall latency also includes PKTDMA overhead, input data transfer and output data transfer to/from the FFTC. If more packets are enqueued, the FFTC engine will be more pipelined. Therefore the total latency for processing all the packets would effectively be the sum of PKTDMA overhead for one packet, input/output data for one block, and FFTC engine processing for all packets.

The test process includes the following steps:

1. All the packets are enqueued by setting up the descriptors (Host descriptors used for the tests). After enqueing, the Packet DMA *transmit channel* is enabled to initiate transfer.
2. Polling is done on the destination queue specified by the user and verifies the packet count received. Time difference between them is measured.
3. Steady-state throughput measurement is performed to remove overheads by sending 10 packets at a time and then sending 20 packets again. Steady-state throughput is calculated by taking the difference between the transfer times of 20 packets and 10 packets and divided by the difference between the number of packets.

Table 18 shows the FFTC throughput for various block sizes.

Table 18 Measured FFTC Throughput Time (Part 1 of 2)

FFT/DFT Size	10 Packets Including Overhead (ns)	20 Packets Including Overhead (ns)	Time To Compute 1 Block Using Steady-State Method (ns)	MSubc/sec
4	2289	4058	58.9	68
8	2433	4332	63.3	126
12	2855	5235	79.3	151
16	2450	4304	61.8	259
24	4061	7557	116.5	206
32	3876	7193	110.5	289
36	4445	8312	128.9	279
48	4642	8681	134.6	357
60	5790	10921	171	351
64	5058	9446	146.2	438
72	8489	16316	260.9	276
96	8371	16010	254.6	377
108	10208	19666	315.2	343
120	11148	21496	344.9	348

Table 18 Measured FFTC Throughput Time (Part 2 of 2)

FFT/DFT Size	10 Packets Including Overhead (ns)	20 Packets Including Overhead (ns)	Time To Compute 1 Block Using Steady-State Method (ns)	MSubc/sec
128	9554	18236	289.4	442
144	11430	21952	350.7	411
180	15285	29599	477.1	377
192	13363	25684	410.7	467
216	22601	44109	716.9	301
240	17448	33723	542.5	442
256	15681	30163	482.7	530
288	23279	45322	734.7	392
300	24046	46820	759.1	395
324	29368	57353	932.8	347
360	32696	63890	1039.8	346
384	28219	54938	890.6	431
432	34466	67309	1094.7	395
480	38555	75375	1227.3	391
512	34350	66832	1082.7	473
540	47734	93547	1527.1	354
576	42414	82815	1346.7	428
600	53468	104856	1712.9	350
648	69796	137385	2252.9	288
720	56857	111343	1816.2	396
768	52322	102084	1658.7	463
864	75229	147668	2414.6	358
900	79682	157814	2604.4	346
960	70805	138595	2259.6	425
972	95019	186870	3061.7	317
1024	64476	125745	2042.3	501
1080	106650	209943	3428.1	315
1152	94350	185142	3026.4	381
1200	95560	187454	3063.1	392
1296	115258	226594	3711.2	349
1536	118452	232298	3794.8	405
2048	148534	291205	4755.7	431
3072	225090	448398	7443.6	413
4096	283090	563960	9363	437
6144	514510	1026520	17067	360
8192	651122	1301384	21675.4	378
End of Table 18				

8 Gigabit Ethernet Switch Subsystem

Applies to All KeyStone Devices

This section discusses the performance of the Gigabit Ethernet (GbE) switch subsystem. The Ethernet switch subsystem is one of three components in the network coprocessor (NETCP) and interacts with the Host DSP through the Packet DMA and the queue manager. The Packet DMA inside the NETCP is the bus master, and is responsible for transferring data between the GbE switch subsystem and Host memory. (Refer to the Multicore Navigator for KeyStone Devices User Guide for additional information of the Packet DMA.)

The Ethernet switch subsystem has one dedicated transmit queue, queue 648, with an attached Packet DMA transmit channel for sending data to the subsystem. For receiving data from the subsystem, receive Packet DMA channels 22 and 23 are used to place data in any of the queues in the queue manager by programming the Packet DMA receive flows 22 and 23. Instead of routing receive data directly to the queue manager, the Ethernet also has the option of providing received data directly to the L2 classify engine of the PA. This allows the PA to perform header classification on the packet before sending the data to the queue manager, and also allows for a wider range of receive queues to be used.

To connect with an Ethernet network, the Gigabit Ethernet switch subsystem supports two SGMII interfaces, each capable of 10/100/1000 megabit per second Ethernet traffic. The interface with the highest throughput is the 1000 megabit (1 gigabit) per second interface. Regardless of the rate, data being transmitted includes the data payload and all packet headers, including the preamble, start-of-frame delimiter, and bit times for interpacket gap. The throughput for only the data payload is highly dependent on the headers that are used in the user specific application, and is not covered by this document.

Assuming that the correct clock frequencies are used, the Ethernet switch subsystem can send and receive data at gigabit speeds for all supported packet sizes, regardless of where the packets are stored in memory. The required clocking frequency for the SGMII and the SerDes modules are discussed in Section 8.1. Additionally, the transmit and receive interfaces each support different throughputs, and are discussed separately. Section 8.2 discusses the performance for the transmit interfaces, and Section 8.3 discusses the performance for the receive interfaces.

8.1 SGMII-SerDes Clocking Considerations

This section discusses the clocking requirements for the SGMII-SerDes, along with some background on 8b/10b encoding. Readers familiar with 8b/10b encoding and clocking for the SGMII interface can skip directly to the throughput performance, discussed in Section 8.2.

Each SGMII module is responsible for encoding and decoding data using the 8b/10b encoding algorithm. For transmit operations, the SGMII module takes eight bits of parallel data, and turns it into 10 bits of parallel encoded data. The SerDes module then takes the 10 bits of encoded parallel data and turns it into 10 bits of encoded serial data, which is then transmitted over the wire. For receive operations, the events happen in the opposite order: the SerDes module will turn the 10 bits of encoded serial data into 10 bits of encoded parallel data, which will then be decoded into 8 bits of parallel data by the SGMII module.

It is important to note that 8b/10b encoding adds an additional 20% overhead to the data being transmitted. To make sure that the 1 gigabit per second data rate can be maintained after adding the overhead due to encoding, the SerDes module must be clocked at a rate of (10/8) * 1 GHz = 1.25 GHz.

8.2 Transmit Throughput Performance

This section discusses the transmit performance of the Ethernet switch subsystem. At a high level, the gigabit Ethernet switch subsystem is able to transmit unicast packets at up to 1 gigabit per second, and transmit multicast packets at up to 2 gigabits per second. More details about these throughputs are described in the paragraphs below.

When transmitting unicast packets, 1 gigabit per second of data that is supported by the Ethernet switch subsystem, and this traffic can be allocated to a single SGMII port or divided between the two SGMII ports. Depending on the traffic pattern, 1 gigabit of unicast traffic will be divided between SGMII0 and SGMII1.

When transmitting multicast packets, up to 2 gigabits per second of data can be transmitted by configuring the address lookup engine in the Ethernet switch to broadcast the packet data to both the SGMII0 port and SGMII1 port simultaneously.

The Ethernet switch subsystem is capable of achieving 1 gigabit per second of unicast packets, and 2 gigabits per second for multicast packets for all packet sizes supported by the subsystem. The minimum supported size is 64 bytes, and the maximum supported size is 9500 bytes (9504 when switch is VLAN aware). When calculating the throughput, an extra 20 bytes of Ethernet data is also included in the calculation for the preamble (7 bytes), start of frame delimiter (1 byte), and inter-packet gap (12 byte times). If the extra 20 bytes of data is included, the maximum supported size becomes 9520 bytes (9524 when switch is VLAN aware).

8.3 Receive Throughput Performance

This section discusses the receive performance of the Ethernet switch subsystem. Each SGMII module in the Ethernet switch subsystem is able to receive data at a maximum rate of one gigabit per second. Since the Ethernet switch subsystem contains two SGMII modules, it is able to support a combined receive throughput of two gigabits per second. Although the Ethernet switch subsystem can support two gigabits per second of Ethernet traffic, it is important to note that the throughput of a single SGMII port cannot exceed one gigabit per second.

The two gigabit per second throughput is based on Ethernet frames of all sizes supported by the subsystem. The minimum supported size is 64 bytes, and the maximum supported size is 9500 bytes (9504 when switch is VLAN aware). When calculating the throughput, an extra 20 bytes of Ethernet data is also included for the preamble (7 bytes), start of frame delimiter (1 byte), and inter-packet gap (12 byte times). If the extra 20 bytes of data is included, the maximum supported size becomes 9520 bytes (9524 when switch is VLAN aware).

9 Inter-integrated Circuit (I^2C)

Applies to All KeyStone Devices

This section discusses the throughput performance for the I^2C peripheral. The Keystone I^2C is compliant with Phillips Semiconductors Inter-IC bus (I^2C bus) specification version 2.1 and is capable of transmitting and receiving data in up to 8-bit data words. The I^2C peripheral connects to the TeraNet through a 32-bit interface. The I^2C peripheral is a bus slave, which means that it is not capable of reading and writing memory, and must rely on another master, such as EDMA, to provide data to the I^2C peripheral.

The I^2C module is connected to the DSP/6 data switch fabric. The DSP/6 clock is divided down inside the I^2C module to derive the I^2C I/O clock frequency. The maximum operating frequency for the I^2C I/O clock is 400 KHz. The remainder of this section discusses the theoretical and measured throughput of the Keystone I^2C peripheral.

9.1 Theoretical Throughput Performance

This section discusses the maximum theoretical throughput for the Keystone I^2C peripheral. The theoretical throughput for the I^2C peripheral is dependent on two main factors, which are the data transfer size, the data word size, and the I/O clock frequency.

The size of the data transfer impacts the I^2C throughput due to a fixed overhead of 11 bits for each data transfer. To begin a data transfer, the I^2C protocol requires a 10 bit header, composed of a 1-bit condition, a 7-bit address, 1-bit read/write specifier, and 1-bit acknowledge. To end a data transfer, the I^2C protocol requires 1-bit stop condition. The 10 header bits and 1 stop bit are required for each data transfer. Since this 11 bits of overhead is fixed for each data transfer, the overall affect will depend on the data transfer size. For small data transfers, the transfer will be dominated by the overhead. For large data transfers, this overhead will be negligible.

In addition to the fixed overhead per transfer, there is additional overhead for each data word that is transferred. This overhead is due to an acknowledge bit, which is generated by the receiving device to indicate that it has successfully received a data word. Since the acknowledge is always 1 bit, the percentage of usable data depends on the size of the data word. A data word can vary in length from a minimum of 1 bit to a maximum of 8 bits. The larger the data word, the lower the percentage of overhead, so the lowest amount of overhead is achieved with 8 bit data words are used.

The last component that affects the throughput is the I^2C I/O clock frequency. The higher the operating rate of the I/O clock, the faster the I^2C module is able to transfer the data. The maximum operating frequency for the I^2C I/O clock is 400 KHz, so the best throughput can be achieved when using this clock rate.

The theoretical throughput for an I^2C data transfer is shown in formula TBD. This formula requires that the values of DataTransferSize and DataWordSize are in bits. The IOfreq parameter is the I^2C I/O clock frequency.

$$\frac{\text{DataTransferSize}}{\text{DataTransferSize} + 11} \times \frac{\text{DataWordSize}}{\text{DataWordSize} + 1} \times \text{IO freq} = I^2C \text{ Throughput}$$

Table 19 and Figure 2 show the theoretical throughput for different data transfer sizes, data word sizes, and I^2C I/O clock frequencies.

Table 19 I^2C Theoretical Throughput

Data Transfer Size (bytes)	Data Transfer Size (bits)	Fixed Header and Footer Overhead (bits)	Data Word Size (bits)	I/O Clock Frequency	Throughput (Kbps)
4	32	11	8	400	264.6
16	128	11	8	400	327.4
64	512	11	8	400	348.1
128	1024	11	8	400	351.8
256	2048	11	8	400	353.6
512	4096	11	8	400	354.6
1024	8192	11	8	400	355.1
4	32	11	6	400	255.1
16	128	11	6	400	315.7
64	512	11	6	400	335.6
128	1024	11	6	400	339.2
256	2048	11	6	400	341.0
512	4096	11	6	400	341.9
1024	8192	11	6	400	342.4
4	32	11	4	400	238.1
16	128	11	4	400	294.7
64	512	11	4	400	313.3
128	1024	11	4	400	316.6
256	2048	11	4	400	318.3
512	4096	11	4	400	319.1
1024	8192	11	4	400	319.6
4	32	11	8	100	66.1
16	128	11	8	100	81.8
64	512	11	8	100	87.0
128	1024	11	8	100	87.9
256	2048	11	8	100	88.4
512	4096	11	8	100	88.6
1024	8192	11	8	100	88.8
End of Table 19					

Figure 2 I^2C Theoretical Throughput

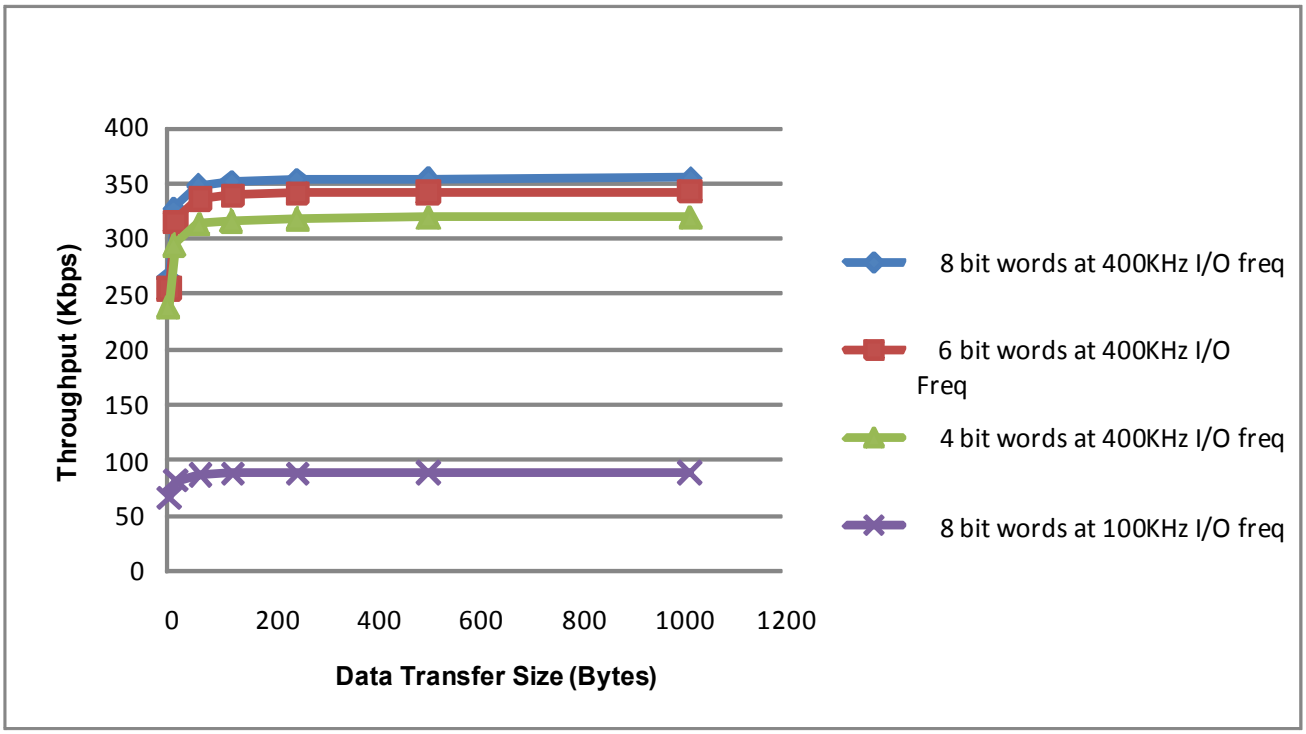


Figure 2 shows the effects of varying the data transfer size, data word size, and I^2C I/O clock frequency on the throughput of the I^2C module. The figure shows that for small data transfers the throughput is greatly reduced due to the fixed overhead required for each data transfer, while for large data transfers, the impact of the fixed overhead is negligible. The figure also shows that the data word size also impacts the throughput, due to the overhead associated with the acknowledge bit. Lastly, the figure shows the throughput for two different operating I/O frequencies.

9.2 Measured Throughput Performance

This section discusses the throughput performance for the Keystone I^2C peripheral. The performance data presented in this section was measured on a TMS320C6670 device. The I^2C performance tests were measured with L2, MSMC, and DDR3 memory endpoints. For all performance tests, the I^2C module was configured for internal loopback, so all data was looped back internally inside the I^2C module.

For these tests, the I^2C I/O clock was operating at the maximum frequency of 400 KHz. EDMA transfers were used to provide the write data to the I^2C module. Similarly, EDMA transfers were used to accept the read data from the I^2C module. Tests were conducted using 256, 512, and 1024 byte data transfers, with 8 data bits per word.

Transfers less than 256 bytes could not be measured accurately due to internal buffering inside the I^2C module. The measured throughput data is shown in Table 20.

Table 20 I^2C Measured Throughput Performance

I/O Clock Frequency (KHz)	Read/Write	Data Transfer Size (Bytes)	Throughput (Kbps)
400	Write	256	354.5
400	Write	512	353.8
400	Write	1024	353.4
400	Read	256	351.7
400	Read	512	352.4
400	Read	1024	352.8
End of Table 20			

10 Multicore Navigator Throughput

The Multicore Navigator uses a Queue Manager Subsystem (QMSS) and a Packet DMA (PKTDMA) to control and implement high-speed data packet movement within the device. This reduces the traditional internal communications load on the device DSPs significantly, increasing overall system performance.

The Multicore Navigator provides a hardware queue manager (8192 queues) and PKTDMA engines. Refer to the Multicore Navigator for KeyStone Devices User Guide for more information.

The key performance data of Multicore Navigator includes delay profiling of Queue PUSH, POP operations, queue pending interrupts and descriptor accumulation interrupts. Also, understanding the bandwidth of infrastructure PKTDMA mode is essential when designing a system.

10.1 Navigator PUSH Latency

This section discusses the latency for queue PUSH operations. From the perspective of a queue manager, the PUSH operation is basically a write. This normally takes only a few cycles, and typically does not stall the master. The queue PUSH operation consists of the following actions:

1. A master in the system (e.g., a DSP core) writes the address of a descriptor to a register for a queue, which generates a PUSH request to the queue manager.
2. The queue manager reads the tail pointer of the queue to get the last descriptor in the queue.
3. The queue manager modifies the linking entry of the last descriptor to make it point to the new descriptor that was pushed into the queue.
4. The queue manager modifies the tail pointer of the queue to make it point to the new descriptor that was pushed into the queue.
5. The queue manager modifies the linking entry of the new descriptor to make it NULL.

An example of the pseudo code that was used to measure the performance of the PUSH operation is shown in Example 1.

Example 1 Pseudo Code for Push Operation Performance

```
preTSC= TimeStampCount;

for(i=0; i< Number_of_Descriptors; i++)
```

```
{
    queueRegs->REG_D_Descriptor= uiDescriptor[i];
}
AverageCycles= (TimeStampCount - preTSC)/ Number_of_Descriptors;
```

End of Example 1

The average DSP cycles for different PUSH operations are shown in Table 21.

Table 21 Multicore Navigator PUSH Latency

No. of Descriptors	VBUSP Space (cycles)	VBUSM Space (cycles)
512	15	14
256	15	13
128	14	11
64	14	7
32	12	1
16	10	1
8	6	1
4	7	1
2	7	1
1	6	1
End of Table 21		

According to the results, a PUSH operation typically takes about 1 to 15 DSP cycles for different sizes. PUSH operation is a write operation, so DSP core may return after it posts the write data into the write buffer. Before the write buffer is full, the DSP may not encounter any stall which is the case when few descriptors are pushed.

On the other hand, if a lot of descriptors are pushed and the write buffer is full with several descriptors, the DSP core will be stalled until there is space in the write buffer for a newer descriptor. The number of cycles stalled is actually the time the queue manager handles one PUSH operation, which is about 15 cycles.

So the conclusion is that if Masters PUSH descriptors with an interval larger than 15 cycles, it will not see any stall. Otherwise, it may be stalled 1-15 DSP cycles.

10.2 Navigator POP Latency

This section discusses the latency for queue POP operations. From the perspective of a queue manager, the POP operation is basically a read. In a read request, the master (e.g., DSP) needs to wait for a response from the slave (queue manager), which can cause the slave to stall while waiting for the response. The queue POP operation consists of the following actions:

1. A master in the system reads a register for a queue, which generates a POP request to the queue manager.
2. The queue manager reads the head pointer of the queue to get the first descriptor pointer, and the address of the first descriptor is returned to the master.
3. The queue manager reads the linking entry of the first descriptor to find the second descriptor in the queue.
4. The queue manager updates the head pointer to the second descriptor (which is now the first descriptor).

The performance of the POP operation was measured on a C6678 DSP core. An example of the pseudo code that was used to measure the performance of the POP operation is shown in Example 2.

Example 2 Pseudo Code for Push Operation Performance

```
preTSC= TimeStampCount;
for(i=0; i< Number_of_Descriptors; i++)
{
    uiDescriptor[i]= queueRegs->REG_D_Descriptor;
}
AverageCycles= (TimeStampCount - preTSC)/ Number_of_Descriptors;
```

End of Example 2

The average DSP cycles for different POP operations are shown in Table 22.

Table 22 Multicore Navigator POP Latency

No. of Descriptors	VBUSP Space (cycles)	VBUSM Space (cycles)
512	45	87
256	45	87
128	45	87
64	45	87
32	45	87
16	45	87
8	45	87
4	46	87
2	47	88
1	47	88
End of Table 22		

According to the results, a POP operation typically takes at least 45 DSP cycles. The cycles for POP are much higher than PUSH operation since POP is a read operation. The Master must wait until the read data is returned from the Queue Manager.

If there are multiple descriptors to be popped, the accumulator may be used to accumulate multiple descriptors from the Queue Manager into the DSP's local memory. In that case, the DSP can read descriptors from its local memory which only takes about 5 DSP cycles.

10.3 Queue Access Through Different Access Regions

The Queue Manager provides different regions or windows for masters to access a queue.

- Normal Registers via VBUSP configuration bus (only for DSP core)
- Normal Data space via VBUSM data bus

From a master's point of view, all these regions are accessed through different address spaces. The DSP cores can use both VBUSM and VBUSP while all the PKTDMAs use the VBUSM for data movement. PUSH to the VBUSM address is faster than VBUSP address due to the deeper write buffer (and no stalls until the buffer is full), but POP is faster via the VBUSP address due to the higher VBUSM latencies for read operations.

If the DSP core needs to push to a queue which will be popped by the PKTDMA, the DSP core should use the VBUSM region for the pushing to avoid potential race conditions. One possible scenario is as follows:

1. DSP core writes data to packet buffer in DDR3 via VBUSM
2. DSP core pushes descriptor to PKTDMA through VBUSP
3. PKTDMA reads the descriptor
4. PKTDMA reads data from DDR3

Since the descriptors and payload data are going through different VBUS transactions, it is possible that PKTDMA could read the descriptor before the payload data arrives in DDR3. If the descriptor is also pushed via VBUSM, this issue can be avoided.

One more way to avoid a race condition is to use the MFENCE instruction to make sure that the data has been written before pushing the descriptors to the Queue Manager. Refer to the TMS320C66x CPU and Instruction Set Reference Guide for more details about the MFENCE instruction.

10.4 Navigator Linking RAM Performance

The Queue Manager subsystem includes internal linking RAM for 16K entries. If the system has more than 16K descriptors, any memory including LL2 (Local Level 2 memory), SL2 (Shared Level 2 memory), DDR3 (Double Data Rate external memory) in the DSP system can be used as external linking RAM of Queue Manager to store other linking entries.

In general, it takes fewer cycles to access internal linking RAM than external linking RAM, so internal linking RAM should be used whenever possible. For applications that require external linking RAM, the memory that is used can have a large impact on the latency to access the linking RAM. For external linking RAM, L2 memory has the best performance. However, it is relatively small, so the amount of descriptors that can be stored in L2 is limited.

Though slower than L2, MSMC memory also has good performance, and makes a good choice due to its larger size.

Though DDR3 offers much more room than L2 and MSMC memories, DDR3 is typically very inefficient when used as external linking RAM because linking entry access is normally discrete. Thus, when using external linking RAM, MSMC memory is recommended.

Queue Manager access linking entry in external linking RAM consumes more cycles than internal linking RAM.

Table 23 compares the average cycles consumed to PUSH/POP descriptors with external linking RAM (in MSMC) and internal linking RAM.

Table 23 Multicore Navigator PUSH/POP Latency with Linking RAM

No. of Descriptors	PUSH (cycles)		POP (cycles)	
	Internal Linking RAM	External Linking RAM (MSMC)	Internal Linking RAM	External Linking RAM (MSMC)
512	14	14	45	100
256	13	13	45	100
128	11	11	45	99
64	7	7	45	99
32	1	1	45	98
16	1	1	45	96
8	1	1	45	92
4	1	1	46	86
2	1	1	47	74
1	1	1	50	48
End of Table 23				

10.5 Infrastructure PKTDMA Performance

Infrastructure PKTDMA mode in the Queue Manager is used for core-to-core communications. The testcase details for measuring the performance of Infrastructure PKTDMA are as follows:

- Testing is done on a 1GHz TMS320C6670 DSP.
- Use one of the queues assigned to infrastructure PKTDMA (from Queue 800 to Queue 831) as Transmit Queue.
- Use one of the high priority accumulation queues as Receive Queue.
- Different memory endpoints are chosen to store descriptors and data.
- Both Host Descriptor mode and Monolithic Descriptor modes are tested.
- All the packets are enqueued by pushing the descriptors to the Transmit Queue. After enqueing, the PKTDMA *transmit channel* is enabled to initiate transfer.
- For calculating Receive throughput, polling is done on the Receive Queue until all descriptors are received in the RX queue. Time stamp is taken once when RX queue receives the first packet and once after all the packets are received in the RX queue. Time difference is measured.
- Throughput is calculated by dividing the (number of packets×payload size) by the average DSP cycles.

Table 24 represents the Receive Throughput of the Infrastructure PKTDMA.

Table 24 Infrastructure PKTDMA Receive Throughput

Payload Size (Bytes)	No of PKTDMA Channels	No of Packets	Descriptor Mode	TX Memory Endpoint	RX Memory Endpoint	Average DSP Cycles	RX Throughput in Mbps
256	1	100	Host	DDR3	L2	56911	3598.6
256	32	100	Host	DDR3	L2	12169	16829.6
256	1	100	Monolithic	DDR3	L2	50049	4091.9
256	32	100	Monolithic	DDR3	L2	10729	19088.4
End of Table 24							

10.6 Accumulator Interrupt Latency

To save the DSP cycles for POP operation, a small programmable controller (PDSP) is integrated into the QMSS. Depending on the user's configuration, the PDSP monitors one or more specific queues. Once there is a descriptor in the queue, the PDSP pops it from the queue, and writes the pointer of the descriptor to a buffer, which is called accumulation buffer. It usually resides in the L2 memory of the DSP. When the PDSP accumulates a specific number of descriptors to the accumulation buffer, it triggers an interrupt to the DSP core which reads the descriptors in its local L2 instead of the Queue Manager region, which saves about 40 DSP cycles for each descriptor read.

The features of PDSP are different with different firmware. The Acc48 firmware monitors up to 32 queues with high priority, and monitors up to 512 (16×32) queues with low priority. The Acc32 firmware monitors up to 32 queues. The Acc16 firmware monitors up to 512 (16×32) queues.

The delay between queue push operation and accumulation completion interrupt depends on how many queues the PDSP monitors, and how busy these queues are.

The testcase details for measuring Accumulator Interrupt Latency is as follows:

- Initialize the QMSS linking RAM and Memory regions and download the accumulator firmware acc48_le to PDSP0.
- Enable the PKTDMA TX and RX channels and set up the flow registers.
- Program the accumulator channel 0.
- Push the descriptors to high priority queue and before pushing the last descriptor, take the start time stamp.
- Accumulator will trigger an interrupt and once it reaches the water mark in the ISR, record the stop time stamp.

Table 25 shows the latency for different accumulator channels programmed. For low priority accumulation, there are 16 groups of queues, and each group has 32 queues. Software polls these groups periodically. The low priority accumulation may sometimes get faster response depending on the queue position picked to measure the latency since it scans 16 groups instead of 32. It is essentially different polling mechanisms.

Table 25 Multicore Navigator Accumulator Latency (Part 1 of 2)

Firmware	Priority Queue	No of Channels Enabled	Average DSP Cycles
Acc48 using PDSP0	High	1	2581
	High	4	3344
	High	8	4828
	High	16	5332
	High	32	9796
	High	48	10108
Acc32 using PDSP0 and Acc16 using PDSP1	Low	1	1843
	Low	4	2225
	Low	8	4530
	Low	16	5573

Table 25 Multicore Navigator Accumulator Latency (Part 2 of 2)

Firmware	Priority Queue	No of Channels Enabled	Average DSP Cycles
Acc32 using PDSP0 and Acc16 using PDSP1	High	1	1800
	High	4	2225
	High	8	4529
	High	16	5573
End of Table 25			

10.7 Other Performance Considerations for Queue Operation

The following sections discuss a few other important considerations to be taken into account when designing a system around the Multicore Navigator.

10.7.1 Monolithic Descriptors vs. Host Descriptors

One factor that affects the performance of the system is the choice to use Host descriptors or Monolithic descriptors. Although this decision is often based on the architecture of the application, the choice also impacts performance.

When using monolithic descriptors, the descriptors and buffers are allocated in continuous memory, which typically provides good memory access performance. In contrast, host descriptors typically allocate the descriptor memory and buffers separately.

Accessing the descriptors and packet buffers in different portions of memory can introduce extra overhead, especially when the descriptor or packet buffer is allocated in cacheable memory. This effect can be minimized by setting up the host descriptors during initialization, which reduces the number of fields that need to be updated during runtime.

In either case, the designer should carefully consider these facts to choose between host packet and monolithic packet.

10.7.2 Descriptors in Cacheable Memory vs. Non-Cacheable Memory

When choosing memory for descriptor allocation, whether or not the region is cacheable can affect the performance of the application. If the descriptor is in cacheable memory space, software needs to maintain cache coherency. Otherwise, the DSP core or the Multicore Navigator may access incorrect data.

The choice of whether or not to use cacheable memory depends largely on the application. Since host descriptors are separate from the data buffers, and the descriptors are relatively small in size, it may make sense to place these in non-cacheable memory since it may not fully utilize the L1D or L2 cache lines.

Furthermore, any cached updates to the descriptor also must be evicted from the cache before it can be used by the Multicore Navigator. In contrast, if the DSP must access larger blocks of data, such as accessing a monolithic descriptor and its data buffer, then it may make sense to use cacheable memory.

If the descriptors are in cacheable SL2 or DDR3 memory space, the application needs to maintain cache coherency for it. Host packets are relatively smaller (normally 32-64 bytes) and it may not fully utilize the 64-byte L1D cache line or a 128-byte L2 cache line. Therefore, there is more penalty to maintain cache coherency for smaller descriptors. Typically, for small descriptors LL2 is recommended to avoid cache penalty.

Sometimes, The "Return Push Policy" field in a descriptor can be modified to make the descriptor return to the head of the queue (instead of the default tail queue return). This policy makes the recently used descriptor to be re-used more quickly and may utilize cache better in certain use cases.

10.8 PKTDMA Bandwidth

The PKTDMA has a priority-based scheduler to manage the bandwidth among multiple channels. When considering the bandwidth of PKTDMA, the transfer overhead also has to be taken into account.

10.8.1 PKTDMA Transfer Overhead

For every queue transaction, there is an associated PKTDMA transfer overhead. It is defined as the time to transfer a single data element (one word) and is measured as the time between pushing the packet to the Transmit Queue and receiving the packet in the RX queue.

Transfer overhead is a big concern for short transfers and needs to be considered when designing a system. Single-element transfer is overhead dominated. Since the overhead is fixed, the bigger the packet size the more bandwidth can be utilized.

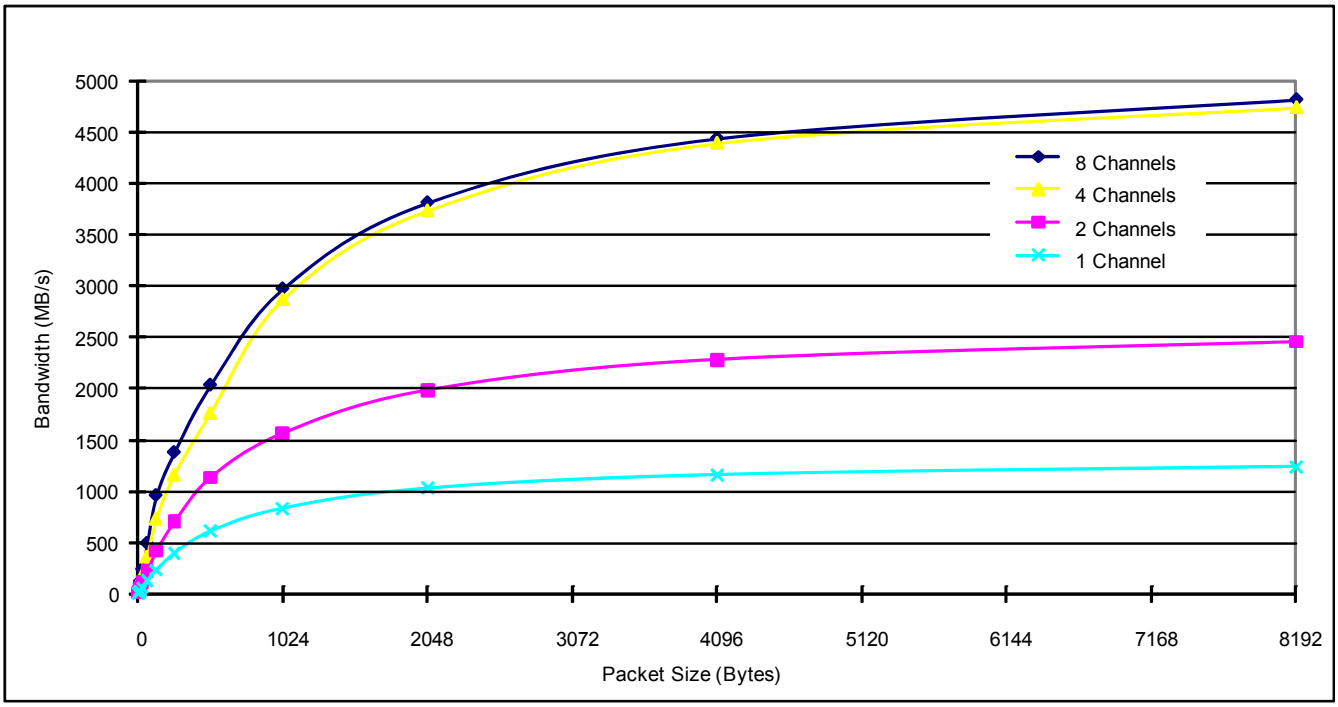
The average overhead for the Packet DMA engine inside the Queue Manager is 550 cycles.

10.8.2 PKTDMA Bandwidth

The PKTDMA has a full-duplex 128-bit interface operating at DSP/3 frequency. The theoretical bandwidth on a 1 GHz DSP is (128bits)/(8bit/byte)×(1000M/3)=5333MB/s. The PKTDMA has a priority-based scheduler to manage the bandwidth among multiple channels. Multiple channels will better utilize the bandwidth capability of PKTDMA and provides more flexibility for software to utilize a simpler design.

Figure 3 shows the total throughput of multiple packet DMA channels.

Figure 3 QMSS PKTDMA Total Throughput with Multiple Channels LL2-to-LL2



There are multiple Packet DMA engines on the Keystone DSP. There are Packet DMA engines in the Queue Manager, FFTC, Serial RapidIO, Packet Accelerator, and the BCP. Packet DMA provides convenient features for packet transfer. Single packet DMA channel does not fully utilize the bandwidth of the memory and bus system. For data transfers requiring high throughput for a single channel, EDMA3 is a better choice.

11 Packet Accelerator (PA)

Applies to All KeyStone Devices

This section discusses the throughput performance of the Packet Accelerator (PA). The PA is one of three components in the network coprocessor (NETCP) and interacts with the Host DSP through the Packet DMA and the queue manager. The Packet DMA inside the NETCP is a bus master, and is responsible for transferring data between the PA and Host memory. For transmitting data to the PA, there are 6 transmit queues in the queue manager, queues 640-645, that each have a dedicated Packet DMA channel responsible for transferring the data to the PA. For receive operations, any of the 32 receive flows available to the NETCP can be used. The receive flows will automatically choose one of the available 24 receive channels that are available to the NETCP.

The PA is able to classify packets at a rate of 1.48 million packets per second for all packet sizes and header types supported by the subsystem. This rate is able to be maintained regardless of where the packet gets stored in system memory after processing. The 1.48 M packet per second processing rate guarantees that minimum size Ethernet packets can be classified at wire rate (1 gigabit per second). For larger packets, the PA is able to classify packets at speeds well above wire rate.

Note—Minimum size packets are assumed to be 84 bytes, which is a 64-byte Ethernet frame plus an extra 20 bytes for Ethernet overhead due to preamble, start of frame delimiter, and interpacket gap.

12 PCIe

Applies to All KeyStone Devices

This section discusses the throughput performance of the Keystone PCIe peripheral. The Keystone PCIe peripheral is connected to the TeraNet through a 128-bit bus. The peripheral operates at a frequency of DSP/3; however, the I/O clock frequency for this module is derived from the PCIe SerDes PLL. The PCIe peripheral supports the PCIe v2 standard, which has a maximum theoretical throughput of 5 gigabits per second per link. Keystone devices support two PCIe links, each capable of transferring data five gigabits per second. Together, these two PCIe links have a maximum theoretical throughput of 10 gigabits per second.

It is important to note that the 10 gigabit per second data rate includes the overhead due to the physical layer, the data link layer (DLL), and the transaction layer packet (TLP). Due to overhead, the raw data that is able to be transferred is less 10 gigabits per second. The remainder of this section discusses:

- Main Factors Affecting PCIe Throughput Performance
- How to Achieve Maximum Throughput with the PCIe Peripheral
- Measured Throughput Performance

12.1 Main Factors Affecting PCIe Throughput Performance

This section discusses the main factors that affect the performance of the PCIe peripheral. To achieve the maximum throughput possible using the Keystone PCIe peripheral, the following factors must be kept in mind:

- Overhead Considerations
- Packet Size Considerations
- EDMA Considerations

12.1.1 Overhead Considerations

This section discusses the overhead associated with the PCIe peripheral. The overhead discussed in this section is related to the PCIe protocol itself, and is not due to the Keystone PCIe implementation. For each PCIe data transfer TLP packets have 20-28 bytes of overhead due to encapsulation. There is also overhead for flow control and acknowledge DLLP packets, which adds 8 bytes of overhead per packet. All data sent over the bus, is encoded using the 8b/10b encoding scheme, which adds an extra 20% overhead. The exact number of bytes of overhead that are required per packet is dependent on the size of the data payload, and the frequency of the DLLP packets.

12.1.2 Packet Size Considerations

This section discusses the effect of packet size on the throughput performance of the Keystone PCIe peripheral. For each packet transferred over the PCIe protocol, the overhead associated with transferring the data should be taken into account. See Section 12.1.1 for more information on PCIe overhead. Since the overhead is added on a per packet basis, the amount of overhead can be reduced by making the data payload as large as possible for all packets transferred.

For outbound transfers, the Keystone PCIe peripheral is able to transfer packets of up to 128 bytes, excluding overhead. Therefore, transferring packets with 128 bytes of data payload will achieve the best throughput. For inbound data transfers, the PCIe peripheral is capable of handling payload sizes larger than 128 bytes, which can further reduce overhead and increase throughput performance.

Note—The packet size is different from the data transfer size. For outbound transfer, the maximum payload size of a packet is 128 bytes. However, data transfers over 128 bytes are supported by the PCIe peripheral. Data transfers over 128 bytes are broken into multiple packets.

12.1.3 EDMA Considerations

This section discusses EDMA considerations when used in conjunction with the Keystone PCIe for outbound transfers. When using EDMA to transmit data to the Keystone PCIe, or when using EDMA to receive data from the Keystone PCIe, the transfer controller (TC) that is used should be taken into consideration. Not all transfer controllers use the same data burst size, and the size of the data burst can have an impact on the performance of the PCIe peripheral. Since the PCIe peripheral supports data payload sizes of up to 128 bytes, if available, a EDMA TC that also supports 128-byte data burst sizes should be chosen. By matching the PCIe data payload size to the EDMA TC data burst size, the PCIe peripheral can send or receive the maximum possible payload size, thereby reducing overhead as much as possible.

When using a 128-byte data payload, the overhead will be introduced every 128 bytes of payload. If using an EDMA TC with a data burst size below 128 bytes, then more overhead will be introduced. For example, if an EDMA TC with a 64-byte data burst size is chosen, then the PCIe will use 64-byte payloads, and packet overhead will be introduced for every 64 bytes of payload data. When transferring 128 bytes, using two 64-byte data payloads will introduce twice as much overhead as a one 128-byte data payload. See Section 12.1.1 for more information on overhead.

12.2 How to Achieve Maximum Throughput with the PCIe Peripheral

This section discusses how to achieve the maximum throughput for the PCIe peripheral. To maximize the throughput of the Keystone PCIe peripheral, the peripheral should be programmed while keeping in mind the "Main Factors Affecting PCIe Throughput Performance". To achieve maximum performance for the Keystone PCIe peripheral, the two main considerations should be to reduce packet overhead (maximize data payload) and to reduce the amount of time related to EDMA transactions.

To reduce overhead, the data payload should be as large as possible. The larger the data payload size, the less packet overhead is introduced, and the more the performance is increased. For outbound transactions, the maximum data payload size is 128 bytes, so payloads as close to 128 bytes as possible are desired. Furthermore, to ensure that the EDMA transfer controller provides 128-byte data payloads to the PCIe peripheral, an EDMA transfer controller that supports 128-byte data burst size should be used.

For inbound transactions, the PCIe v2 specification permits data payloads up to 4096 bytes; however, in KeyStone devices, the maximum PCIe data payload size is limited to 256 bytes for inbound transactions (compared to 128 bytes for outbound. Please refer to the PCIe User Guide for details), so the closer to maximum payload size, the better the throughput performance. Another way to reduce overhead is to reduce packet headers as much as possible. One example may be to remove optional headers, such as the TLP ECRC, if this field is not required by the application.

Lastly, overhead can be reduced by reducing DLLP packets. If not required by the application, DLLP flow control packets can be disabled by programming the FC_DISABLE field in the LANE_SKEW register. IF DLLP acknowledge packets are not required by the application, then the transmission of these packets can be disabled by programming the ACK_DISABLE field in the LANE_SKEW register. If acknowledge packets are required by the application, the frequency of the acknowledge packets can be configured to the lowest rate required by the application by programming the ACK_FREQ register.

12.3 Measured Throughput Performance

This section discusses the measured performance of the Keystone PCIe peripheral. This section also discusses the configuration of the Keystone PCIe peripheral for the measurements specified. The measurements mentioned in this section were collected on a TMS320C6678 device.

Performance data for the PCIe peripheral is captured in Table 26 and Table 27. For all performance tests, both links of the Keystone PCIe module were connected directly with another Keystone PCIe module. One module was configured as the RC and the other device was configured as the EP. For the performance tests, all transactions were outbound transfers initiated by the RC. The tests were measured with L2, MSMC, and DDR memory endpoints, and with 2048, 4096, 8192, 16384, 32768, and 49152-byte data payload transfer sizes.

The performance tests were conducted using 64-byte and 128-byte EDMA data burst sizes. All packets with data payload used PHY, DLLP, and TLP headers. Each TLP packet contained 20-byte overhead of header and footer. An 8-byte DLLP packet was sent after each packet containing a data payload. Performance data with all overhead excluded is shown in Section 12.3.1 and Section 12.3.2.

12.3.1 Read Throughput Performance

This section discusses the measured read performance of the PCIe peripheral. The throughput shown in this section is the data payload only, and does not include any overhead due to PHY, DLLP, or TLP headers, and does not include overhead due to 8b/10b encoding.

For this test, read transactions were done between two Keystone PCIe devices. Device1 was configured as the RC and Device2 was configured as the EP. The read transactions were all outbound reads, where Device1 read data from Device2.

For these transactions, the read request was initiated by Device1. When Device2 received the read request, it used the PCIe master to transfer the read data from memory to the PCIe module. When Device1 received the read data from Device2, the PCIe slave in Device1 provided the data to EDMA to transfer the data from the PCIe module to memory.

Table 26 shows the effect of different EDMA data burst sizes on the PCIe throughput. Data was collected for several combinations of L2, MSMC, and DDR3 endpoints; however, the difference in the measured throughput between the different memory endpoints is negligible. Data was also collected for 2048, 4096, 8192, 16384, 32768, and 49152-byte data payload transfer sizes; however, the difference in measured throughput between different data transfer sizes was also negligible. Table 26 shows the throughput for an outbound read transaction from Device2-L2 memory to Device1-DDR3 memory.

Table 26 Measured PCIe Read Throughput Performance

Data Burst Size	Throughput Performance (Gbps)	Throughput Performance (MBps)
128 bytes	6.45	806.2
64 bytes	5.51	688.8

PCIe performance data was collected for EDMA data burst sizes of 64 bytes and 128 bytes. Table 26 shows that the 128-byte EDMA data burst size allows for better performance than the 64-byte data bursts. The 128-byte bursts perform better, because it results in less PHY/DLLP/TLP overhead.

12.3.2 Write Throughput Performance

This section discusses the measured write performance of the PCIe peripheral. The throughput shown in this section is the data payload only, and does not include any overhead due to PHY, DLLP, or TLP headers, and does not include overhead due to 8b/10b encoding. The write throughput of the PCIe peripheral is shown in Table 27.

Table 27 shows write transactions between two Keystone PCIe devices. Device1 is configured as the RC and Device2 is configured as the EP. The write transactions shown in this figure are all outbound writes, where Device1 writes data to Device2.

For these transactions, the write request was initiated by Device1. Device1 used EDMA to transfer data from memory to the PCIe slave, which was transferred over the PCIe interface to Device2. When Device2 received the write data, the PCIe master was used to transfer the data to memory.

Table 27 shows the effect of different EDMA data burst sizes, and data transfer sizes on the PCIe throughput. Data was collected for several combinations of L2, MSMC, and DDR3 endpoints; however, the difference in the measured throughput between the different memory endpoints is negligible. Data was also collected for 2048, 4096, 8192, 16384, 32768, and 49152-byte data payload transfer sizes; however, the difference in measured throughput between different data transfer sizes was also negligible.

Table 27 shows the throughput for an outbound write transaction from Device1-DDR3 memory to Device2-L2 memory.

Table 27 Measured PCIe Write Throughput Performance

Data Burst Size	Throughput Performance (Gbps)	Throughput Performance (MBps)
128 bytes	5.91	738.8
64 bytes	5.55	693.8

PCIe performance data was collected for EDMA data burst sizes of 64 bytes and 128 bytes. Table 27 shows that the 128-byte EDMA data burst size allows for better performance than the 64-byte data bursts. The 128-byte bursts perform better, because it results in less PHY/DLLP/TLP overhead.

13 RAC Throughput

Applies to C6670

The Receive Accelerator Coprocessor (RAC) performs most of the uplink chip rate receive operations under the UMTS 3GPP FDD (Release 6) standard. RAC accelerates code generation, de-scrambling, de-spreading, coherent accumulation, amplitude and power computation, and non-coherent accumulation. The RAC consists of two shared Generic Correlation Coprocessors (GCCP) which perform various correlation tasks.

Each GCCP is a programmable, highly flexible, vector-based correlation engine supporting a large number of simultaneous correlations. This is achieved via a wide data-path (32 chip-rate samples) and a high processing clock rate (superior to C64x chip rate). The TeraNet switch fabric talks to the front-end interface via a 64-bit-wide data bus and the back-end interface via a 128-bit-wide data bus.

The RAC subsystem is designed to operate at 1/3 the DSP frequency. The RAC is capable of performing 54 uplink streams, meaning that the DMA bus will be transferring up to 54 streams × (8 bits I + 8 bits Q) per sample × 2 samples per chip (2x over sampling) × 3.84 MChips per second = 6.63 Gbps of data into the RAC front end interface.

RAC latency is more critical than throughput. The output of the high priority FIFO shall be drained within 128 × 260 ns. The performance of RAC is highly dependent on the configuration in which it is used. Also, the RAC is programmed using RAC Functional Libraries (RACFL), so the performance is measured in the context of the RACFL.

In general, the RAC supports the following:

- RAC can perform 2×8×32 chip correlations per cycle.
- A cycle is equivalent to 64×3.84MHz = 246 MHz
- Each RAC has 2 GCCPs running independently.
- Each GCCP has 8 branches that run in parallel.
- Each branch operates on 32 chips per RAC cycle.

14 SPI

Applies to All KeyStone Devices

This section discusses the performance of the Keystone SPI peripheral. The SPI module provides an interface between the DSP and other SPI-compliant devices. The primary intent of this interface is to allow for connection to an SPI ROM for boot, but can also be used for data transfers that are not related to boot.

The SPI peripheral connects to the TeraNet through a 32-bit interface. The SPI peripheral is a bus slave, which means that it is not capable of reading and writing memory and must rely on another master, such as EDMA, to provide data to the SPI peripheral. The SPI module connects to the DSP/6 switch fabric. The DSP/6 clock is used is further divided down inside the SPI module itself to derive the SPI I/O clock frequency. The maximum supported SPI I/O clock is 66 MHz.

14.1 Theoretical Throughput Performance

This section discusses the theoretical throughput performance for the SPI peripheral. At the maximum supported SPI I/O clock frequency of 66 MHz, the maximum theoretical data rate of the SPI peripheral is 66 megabits per second. However, due to some idle time between cycles, the maximum achievable data rate is 57 megabits per second on KeyStone devices.

14.2 Measured Throughput Performance

This section discusses the measured throughput of the Keystone SPI peripheral. The throughput discussed in this section was captured on a TMS320C6670 device. For all performance tests, the SPI module was configured as the master in 4-pin mode, with the data looping back internally inside the SPI module. For the performance tests, the SPI I/O clock was operating at the maximum frequency of 66 MHz. The data was transferred over the SPI protocol using 16-bit word length, with the data provided to the SPI module and received from the SPI module by EDMA transfers. For these tests, the performance was measured with L2, MSMC, and DDR memory endpoints, using 8, 32, 128, 256, 512, 1024, and 2048-byte data transfers.

The resulting measurements showed that the SPI peripheral can achieve the maximum 57 megabits per second data rate mentioned in Section 14.1. This data rate was achieved for all memory endpoints and data transfer sizes.

15 SRIO Throughput

Applies to All KeyStone Devices

Serial RapidIO (SRIO) is a high-bandwidth system level interconnect. It is a packet switched interconnect intended for high speed chip-to-chip and board-to-board communication. SRIO supports DirectIO transfers and Message Passing transfers.

The RapidIO specification defines four different bandwidths for each differential pair of I/O signals: 1.25, 2.5, 3.125, and 5 Gbps. Due to 8-bit/10-bit encoding overhead, the effective data bandwidths per differential pair is 1, 2, 2.5, and 4 Gbps respectively. A 1x port is defined as one TX and one RX differential pair. A 4x port is a combination of four of these pairs.

15.1 DirectIO (LSU) Operation:

The DirectIO (Load/Store) module serves as the source of all outgoing directIO packets. In this operation, the RapidIO packet contains the specific address where the data should be stored or read in the destination device. There are 8 LSUs in total.

The DirectIO tests were measured on a TMS320C6670 DSP operating at 1 GHz. L2 memory endpoints were used for both input and output data. The throughput test process includes the following:

1. Separate data transfers are initiated for NWRITE and NREAD transactions.
2. SRIO is configured for 3.125 Gbps and 5 Gbps speeds with different modes 1x, 2x, and 4x modes.
3. The LSUs are set up with default shadow register configuration.
4. Doorbell Registers are used to record the time since the MAU (Memory Access Unit) does not have an interrupt. The total time taken for the packet transfer is calculated by using the doorbell time stamps.
5. 16 DirectIO transactions are sent to better maximize the throughput between the doorbells.
6. Data rate is calculated by dividing the total number of bits by the total time taken.

Table 28 shows the throughput of a DirectIO operation for an NWRITE transaction using a PHY line rate of 3.125 Gbps (SRIO line rate of 2.5 Gbps) in 1x, 2x and 4x modes. For every 256 Bytes there is 20 Bytes of overhead due to packet header and if the packet size is less than 256 Bytes, there will be 18 bytes of overhead.

Table 28 DirectIO Write Throughput with 3.125 Gbps PHY[1]

Payload Size (Bytes)	Total Bits with Overhead	Total Bits without Overhead	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	3328	1024	548.4	168.7	564.2	173.6	547.2	168.4
16	4352	2048	765.5	360.2	763.5	359.3	719.6	338.6
32	6400	4096	1097.4	702.3	1123.4	719.0	1058.2	677.2
64	10496	8192	1765.2	1377.7	1838.8	1435.2	1746.7	1363.3
128	18688	16384	2284.3	2002.7	3159.9	2770.4	3041.7	2666.7
256	35328	32768	2370.4	2198.6	4331.0	4017.2	5411.8	5019.6
512	70656	65536	2403.3	2229.1	4590.4	4257.8	7983.7	7405.2
1024	141312	131072	2422.5	2247.0	4739.5	4396.0	8481.1	7866.5
2048	282624	262144	2427.4	2251.5	4820.3	4471.0	8768.4	8133.0
4096	565248	524288	2433.0	2256.7	4856.0	4504.1	8907.1	8261.7
8192	1130496	1048576	2434.2	2257.8	4877.2	4523.8	8992.4	8340.8
End of Table 28								

1. NWRITE Transaction with 3.125 Gbps PHY Line Rate

Table 29 shows the throughput of a DirectIO operation for an NREAD transaction using a PHY line rate of 3.125 Gbps (SRIO line rate of 2.5Gbps) in 1x, 2x and 4x modes.

Table 29 DirectIO Read Throughput with 3.125 Gbps PHY[1]

Payload Size (Bytes)	Total Bits with Overhead	Total Bits without Overhead	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	3328	1024	179.2	55.2	189.1	58.2	198.9	61.2
16	4352	2048	235.9	111.0	246.9	116.2	259.4	122.1
32	6400	4096	322.2	206.2	349.5	223.7	363.9	232.9
64	10496	8192	464.9	362.9	511.0	398.8	552.9	431.5
128	18688	16384	670.2	587.6	771.8	676.7	854.0	748.7
256	35328	32768	917.4	850.9	1120.7	1039.5	1281.9	1189.0
512	70656	65536	1339.6	1242.6	1844.2	1710.6	2242.2	2079.7
1024	141312	131072	1736.7	1610.9	2680.4	2486.2	3594.6	3334.1
2048	282624	262144	2002.3	1857.2	3477.5	3225.5	5147.4	4774.4
4096	565248	524288	2128.7	1974.5	4054.4	3760.5	6473.6	6004.5
8192	1130496	1048576	2200.9	2041.4	4291.1	3980.1	7093.3	6579.3
End of Table 29								

1. NREAD Transaction with 3.125 Gbps PHY Line Rate

Table 30 shows the throughput of a DirectIO operation for an NWRITE transaction using a PHY line rate of 5 Gbps (SRIO line rate of 4 Gbps) in 1x, 2x and 4x modes.

Table 30 DirectIO Write Throughput with 5 Gbps PHY[1]

Payload Size (Bytes)	Total Bits with Overhead	Total Bits without Overhead	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	3328	1024	551.9	169.8	550.0	169.2	623.8	191.9
16	4352	2048	719.9	338.8	718.6	338.2	721.0	339.3
32	6400	4096	1057.7	676.9	1057.3	676.7	1058.2	677.2
64	10496	8192	1734.6	1353.8	1736.6	1355.4	1735.4	1354.5
128	18688	16384	2984.8	2616.8	3062.1	2684.6	3093.0	2711.7
256	35328	32768	3458.4	3207.8	5501.1	5102.5	5629.1	5221.1
512	70656	65536	3664.5	3399.0	6934.5	6432.0	10903.7	10113.6
1024	141312	131072	3781.6	3507.6	7353.5	6820.6	12947.8	12009.5
2048	282624	262144	3841.1	3562.8	7586	7036.3	13693.0	12700.8
4096	565248	524288	3872.4	3591.8	7713.3	7154.4	14102.6	13080.7
8192	1130496	1048576	3887.4	3605.7	7776.4	7212.9	14311.1	13274.1
End of Table 30								

1. NWRITE Transaction with 5 Gbps PHY Line Rate

Table 31 shows the throughput of a DirectIO operation for an NREAD transaction using a PHY line rate of 5 Gbps (SRIO line rate of 4 Gbps) in 1x, 2x and 4x modes.

Table 31 DirectIO Read Throughput with 5 Gbps PHY[1]

Payload Size (Bytes)	Total Bits with Overhead	Total Bits without Overhead	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	3328	1024	247.6	76.2	262.7	80.8	271.5	83.5
16	4352	2048	316.2	148.8	338.1	159.1	353.7	166.5
32	6400	4096	435.5	278.7	470.0	300.8	495.3	317.0
64	10496	8192	640.1	499.6	707.0	551.8	760.1	593.2
128	18688	16384	939.6	823.7	1085.1	951.3	1186.7	1040.4
256	35328	32768	1315.5	1220.1	1598.6	1482.8	1803.1	1672.4
512	70656	65536	1979.8	1836.4	2663.3	2470.3	3199.6	2967.7
1024	141312	131072	2638.9	2447.6	3986.8	3697.9	5250.7	4870.2
2048	282624	262144	3110.6	2885.2	5316.2	4930.9	7704.9	7146.6
4096	565248	524288	3338.0	3096.1	6338.3	5879.0	10077.7	9347.4
8192	1130496	1048576	3417.1	3169.4	6794.1	6301.8	11177.2	10367.3
End of Table 31								

1. NREAD Transaction with 5 Gbps PHY Line Rate

Figure 4 through Figure 7 show DirectIO throughput in NWRITE and NREAD modes respectively.

Figure 4 DirectIO NWRITE Throughput with 3 Gbps PHY with Overhead

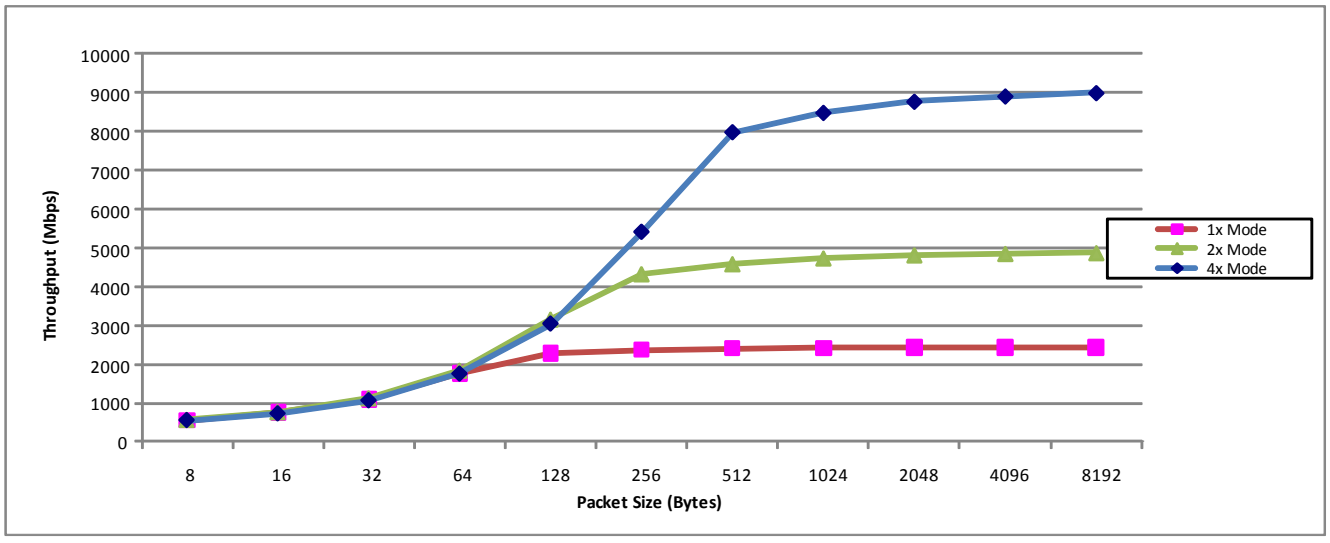


Figure 5 DirectIO NWRITE Throughput with 5 Gbps PHY with Overhead

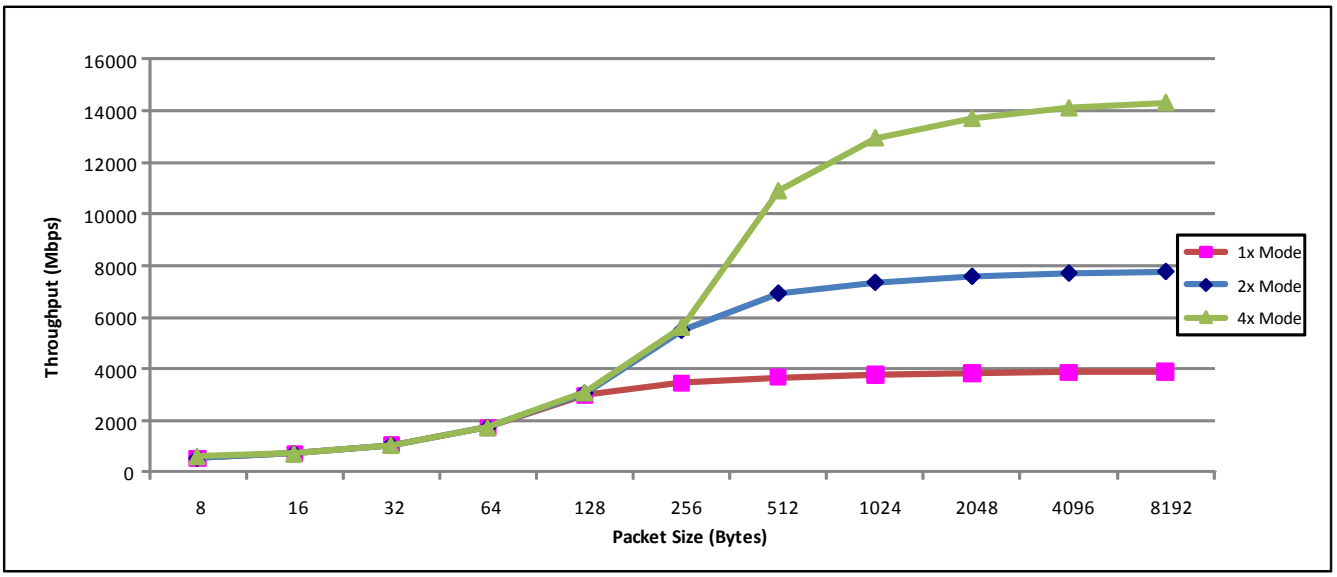


Figure 6 DirectIO NREAD Throughput with 3 Gbps PHY with Overhead

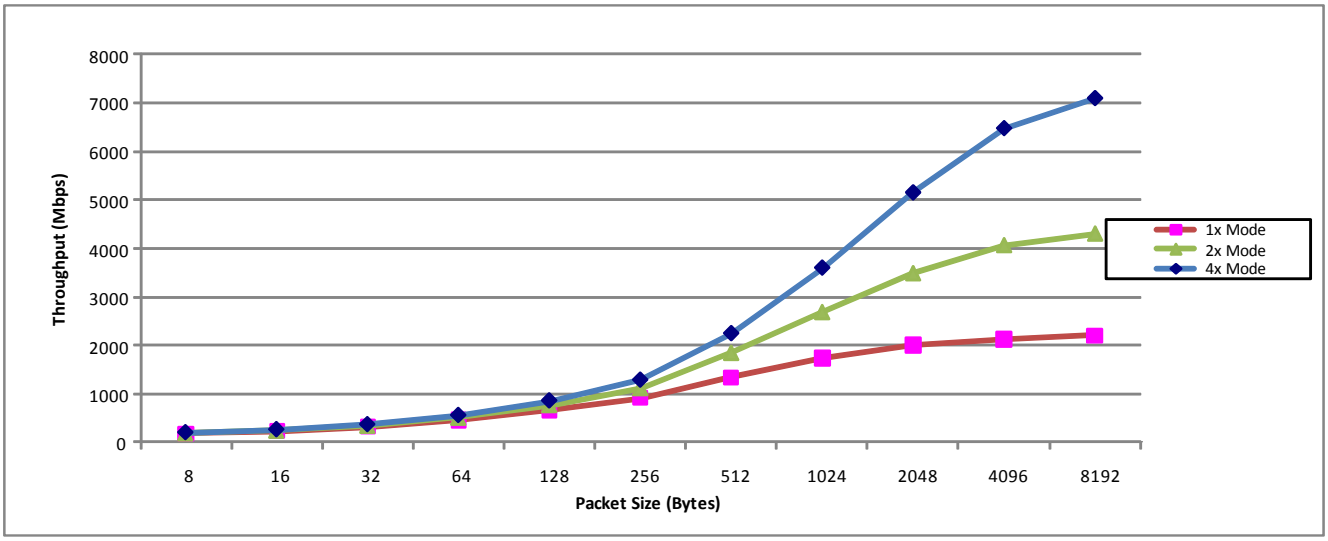
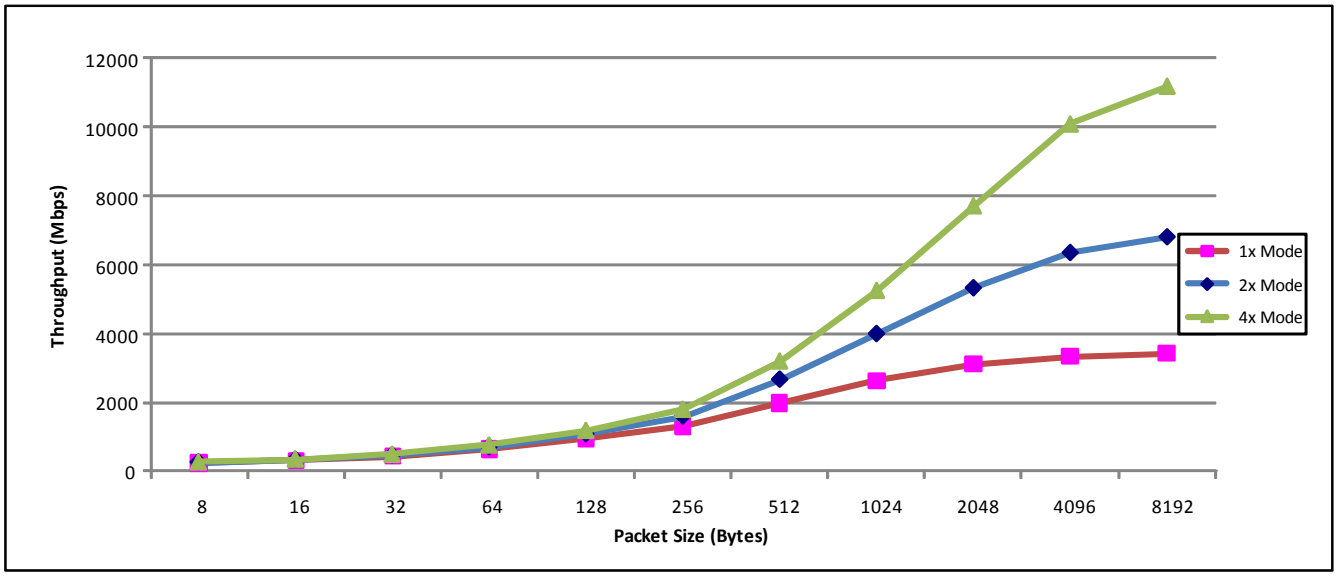


Figure 7 DirectIO NREAD Throughput with 5Gbps PHY with Overhead



15.2 Message Passing Throughput

The Packet DMA module is the incoming and outgoing message passing protocol engine of the SRIO peripheral. With message passing, a mailbox identifier is used instead of a destination address within the RapidIO packet. The mailbox is controlled and mapped to memory by the local (destination) device. For multi-packet messages, four mailbox locations are specified. The Packet DMA transfers messages to the appropriate memory via the DMA bus.

The *ftype* header field of the received RapidIO message packets is decoded by the logical layer of the peripheral. Type 9, Type 11 and Type 13 packets are routed to the Packet DMA.

15.2.1 Type 11 Throughput

The Type 11 message passing tests were measured on a TMS320C6670 DSP operating at 1GHz. L2, MSMC and DDR3 memory endpoints are used for both input and output data. The throughput process includes the following:

1. SRIO is configured for 3.125 Gbps line rate and 5 Gbps line rate with different modes 1x, 2x, and 4x.
2. QMSS Accumulator Firmware is used to monitor the destination queue.
3. All the packets are enqueued by the Host Descriptors. The first few descriptors are programmed for Mailbox 0 and the remaining descriptors are programmed with Mailbox 1 and routed to different RX queues.
4. After enqueuing, the logical transmit channel is enabled to initiate transfer.
5. Multiple packets are sent to get better performance.
6. Time stamps are captured when the interrupts are generated by the QMSS accumulator.
7. 32 transactions are sent for each packet size.
8. 16 Bytes of overhead is added for each packet.

Table 32 and Figure 8 show the throughput of a Type 11 Message Passing operation using a PHY line rate of 3.125 Gbps (SRIO line rate of 2.5 Gbps) in 1x, 2x and 4x modes.

Table 32 Type 11 Message Passing Throughput with 3.125 Gbps PHY[1]

Payload Size (Bytes)	Payload with Overhead (Bytes)	No of Packets TX/RX	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	24	32	877	292	870	290	882	294
16	32	32	1152	576	1153	576	1165	583
32	48	32	1408	939	1731	1154	1719	1146
64	80	32	1711	1369	2866	2293	2921	2337
128	144	32	1897	1687	3164	2812	5230	4649
256	272	32	2030	1911	3596	3385	5367	5051
512	544	32	2183	2055	4100	3859	6665	6273
1024	1088	32	2268	2134	4675	4400	7696	7243
2048	2176	32	2276	2142	4589	4319	7720	7266
4096	4352	32	2280	2146	4577	4308	7744	7288
End of Table 32								

1. Type 11 with 3.125 Gbps PHY Line Rate

Figure 8 Type 11 Throughput with 3 Gbps PHY with Overhead

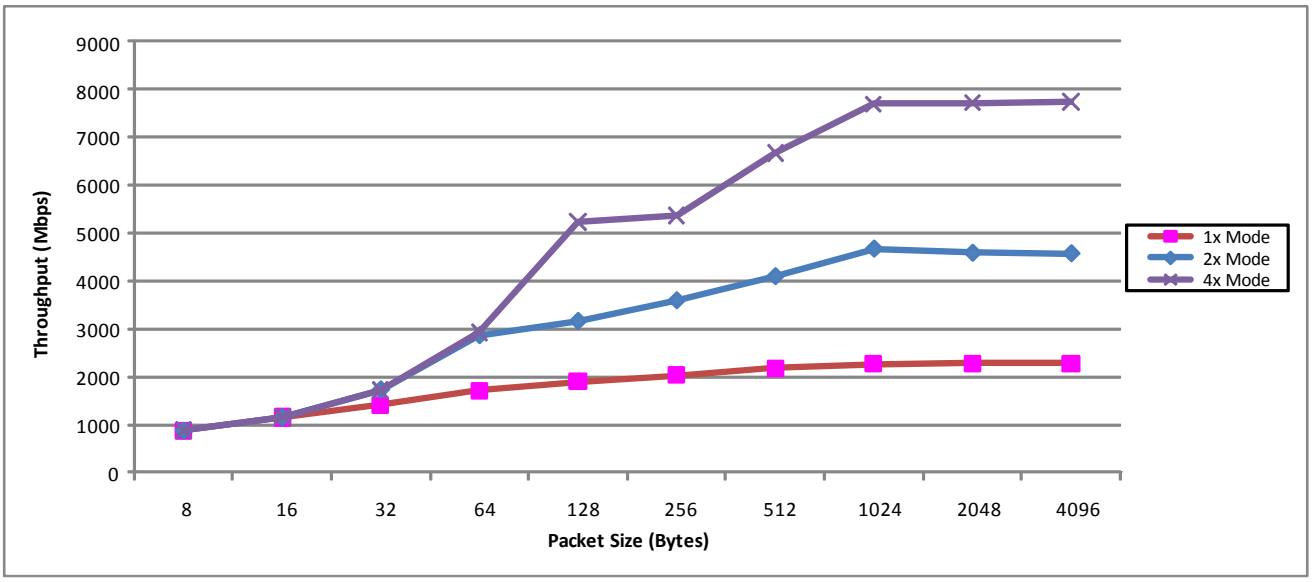


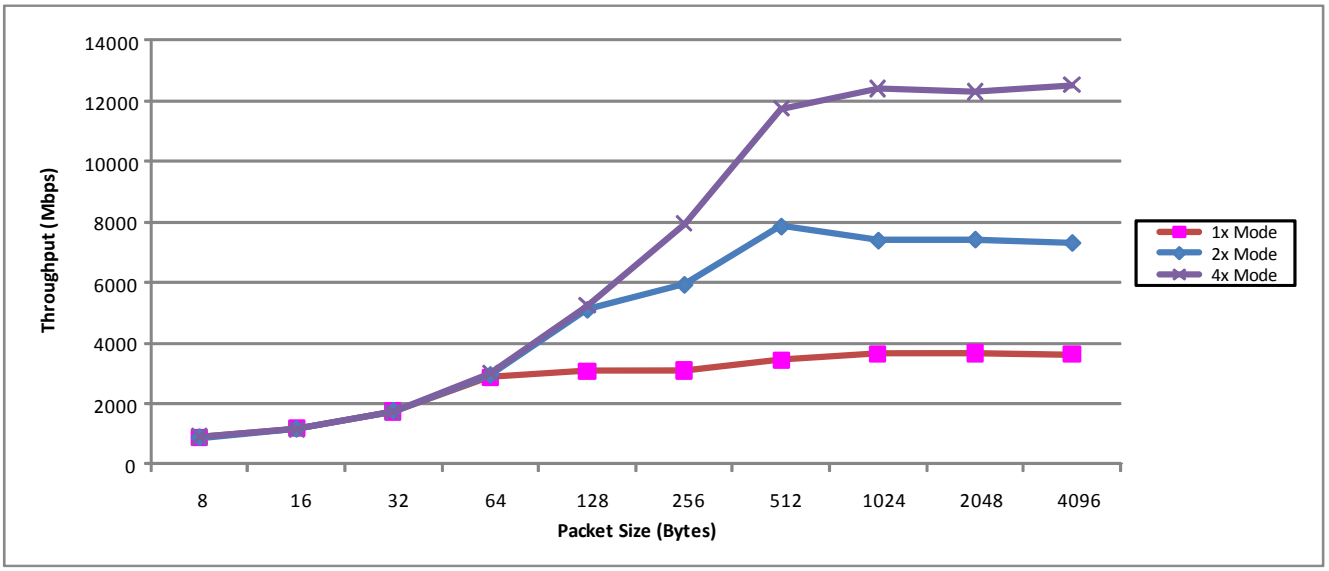
Table 33 and Figure 9 show the throughput of a Type 11 Message Passing operation using a PHY line rate of 5 Gbps (SRIO line rate of 4 Gbps) in 1x, 2x and 4x modes.

Table 33 Type 11 Message Passing Throughput with 5 Gbps PHY[1]

Payload Size (Bytes)	Payload with Overhead (Bytes)	No of Packets TX/RX	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	24	32	891	297	880	293	906	302
16	32	32	1195	598	1158	579	1152	576
32	48	32	1742	1161	1756	1171	1731	1154
64	80	32	2874	2300	2946	2357	3007	2406
128	144	32	3084	2741	5102	4535	5247	4664
256	272	32	3096	2914	5929	5580	7946	7479
512	544	32	3445	3242	7860	7398	11741	11051
1024	1088	32	3646	3432	7395	6960	12407	11677
2048	2176	32	3659	3443	7429	6992	12304	11580
4096	4352	32	3636	3422	7317	6887	12520	11783
End of Table 33								

1. Type 11 with 5 Gbps PHY Line Rate

Figure 9 Type 11 Throughput with 5 Gbps PHY with Overhead



15.2.2 Type 9 Throughput

The Type 9 message passing tests were measured on a TMS320C6670 DSP operating at 1 GHz. L2, MSMC, and DDR3 memory endpoints are used for both input and output data. The throughput process includes the following:

1. SRIO is configured for 3.125 Gbps line rate and 5 Gbps line rate with different modes 1x, 2x, and 4x.
2. QMSS Accumulator Firmware is used to monitor the destination queue.

3. All the packets are enqueued by the Host Descriptors. The first few descriptors are programmed for COS (Class of Service) 41 and the remaining descriptors are programmed with COS 42 and routed to different RX queues.
4. After enqueuing, the logical transmit channel is enabled to initiate transfer.
5. Multiple packets are sent to get better performance.
6. Time stamps are captured when the interrupts are generated by the QMSS accumulator.
7. 32 transactions are sent for each packet size.
8. 16 Bytes of overhead is added for each packet.

Table 34 and Figure 10 show the throughput of a Type 9 Message Passing operation using a PHY line rate of 3.125 Gbps (SRIO line rate of 2.5 Gbps) in 1x, 2x and 4x modes.

Table 34 Type 9 Message Passing Throughput with 3.125 Gbps PHY[1]

Payload Size (Bytes)	Payload with Overhead (Bytes)	No of Packets TX/RX	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	24	32	1057	352	895	298	890	297
16	32	32	1556	778	1528	764	2155	1077
32	48	32	1789	1193	2299	1532	2302	1535
64	80	32	2327	1862	3791	3033	3863	3091
128	144	32	2488	2211	5216	4637	5413	4812
256	272	32	2446	2302	4664	4389	7983	7514
512	544	32	2393	2252	4889	4602	9440	8884
1024	1088	32	2435	2291	4905	4617	9269	8724
2048	2176	32	2429	2287	4869	4583	8925	8400
4096	4352	32	2434	2290	4888	4601	8965	8438
End of Table 34								

1. Type 9 with 3.125 Gbps PHY Line Rate

Figure 10 Type 9 Throughput with 3 Gbps PHY with Overhead

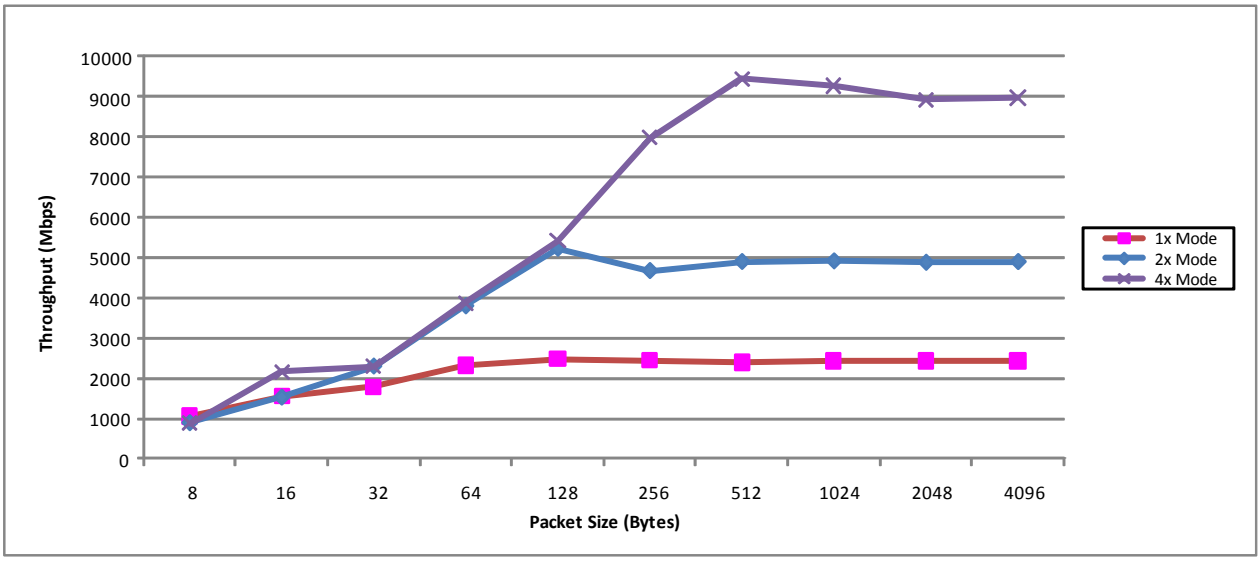


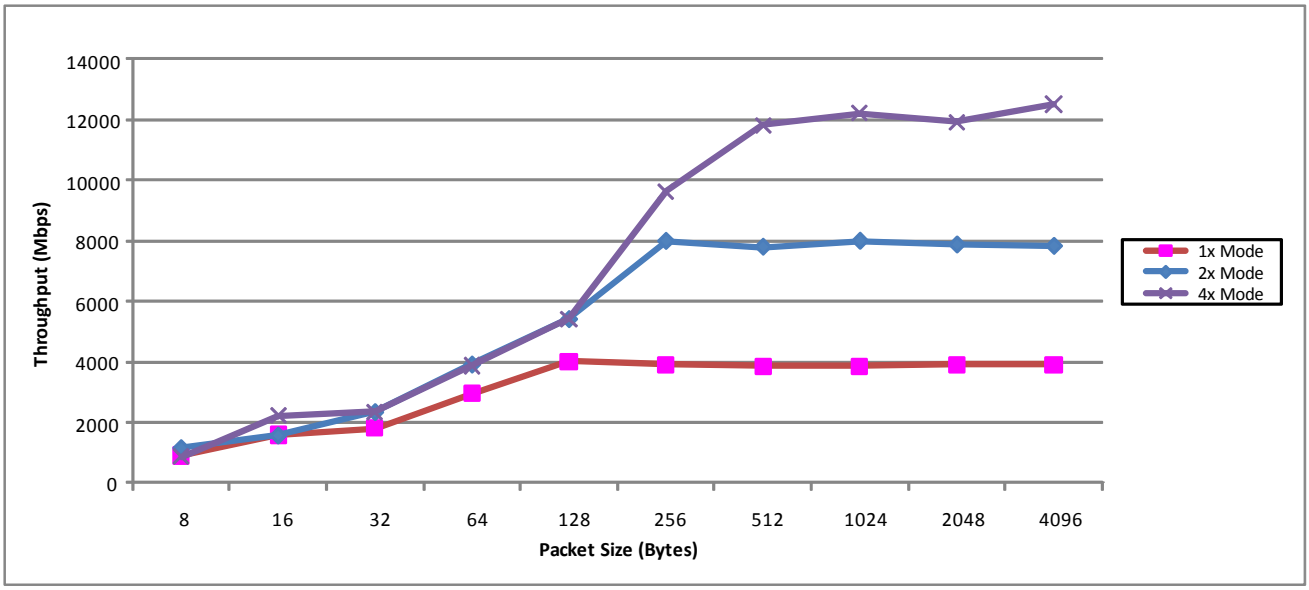
Table 35 and Figure 11 show the throughput of a Type 9 Message Passing operation using a PHY line rate of 5 Gbps (SRIO line rate of 4 Gbps) in 1x, 2x and 4x modes.

Table 35 Type 9 Message Passing Throughput with 5 Gbps PHY[1]

Payload Size (Bytes)	Payload with Overhead (Bytes)	No of Packets TX/RX	1x Lane		2x Lanes		4x Lanes	
			Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)	Data Rate With Overhead (Mbps)	Data Rate Without Overhead (Mbps)
8	24	32	908	303	1166	389	875	292
16	32	32	1577	788	1579	789	2222	1111
32	48	32	1809	1206	2350	1566	2350	1566
64	80	32	2950	2360	3918	3134	3881	3105
128	144	32	4017	3570	5414	4812	5413	4812
256	272	32	3900	3671	7989	7519	9624	9058
512	544	32	3851	3625	7798	7339	11807	11113
1024	1088	32	3851	3624	7994	7524	12207	11488
2048	2176	32	3910	3680	7870	7407	11915	11214
4096	4352	32	3904	3675	7824	7363	12512	11776
End of Table 35								

1. Type 9 with 5 Gbps PHY Line Rate

Figure 11 Type 9 Throughput with 5 Gbps PHY with Overhead



16 TAC Throughput

Applies to C6670

The Transmit Accelerator Coprocessor (TAC) performs most of the WCDMA/HSPA downlink chip-rate transmit operations within the DSP.

The TAC subsystem consists of several components:

- One Front End Interface (FEI)
- Two Spreader Group Co-Processors (SGCP)
- One Back End Interface (BEI)

The TAC is designed to operate at 1/3 DSP frequency. TAC symbol data input is fetched from L2 or other memory to the TAC front end through a 64-bit VBUSM master interface. TAC configuration is input from the TAC functional library. After TAC processing is complete, TAC output data is passed to the Antenna Interface or any device memory location through a 128-bit VBUSP slave interface.

For a DSP running at 1 GHz, the maximum output data rate is 64 streams × 32 bits I/Q per chip × 3.84 MChipsPerSecond = 7.86 Gbps. The TAC's processing latency (time at which input symbol data is supplied to TAC) is more critical than the throughput.

All the tests are measured on a TMS320C6670 DSP operating at 1 GHz. The throughput test details are as follows:

- DDR3 memory endpoints are used for both input and output data.
- 32 cells and 256 Dedicated Physical Channels of Spreading Factor=4 with Transmit Diversity are enabled.
- The dedicated channels are uniformly distributed across all cells.
- Beam Forming is not used for testing.
- All UEs (user equipment) are aligned in time.

The test process includes the following:

1. Code Power Measurements are turned on.
2. In every slot, Uplink TPC, Downlink TPC, and Diversity Messages are supplied to TAC.
3. Symbol Data is supplied to the FEI at the frame boundary with a latency of 24 chips.

Table 36 represents TAC complex throughput (latency).

Table 36 TAC Complex Throughput

Number of Cells	Number of DPCHs	TAC Processing Latency	Time Taken for TAC to process 1 Frame of Data (ns)
32	256	24 Chips	9999937

17 TCP3d Throughput

Applies to C6670

TCP3d is a programmable accelerator on the TMS320C6670 DSP used for decoding of WCDMA, HSUPA, HSUPA+, TD-SCDMA, LTE, and WiMAX turbo codes. The inputs into the TCP3d are channel soft decisions for systematic and parity bits, and the outputs are hard/soft decisions. The TCP3d operates at DSP/2 frequency and has a bus width of 128 bits.

17.1 TCP3d Processing Modes

The TCP3d can be configured in different processing modes as follows:

- **Single Buffered Mode**: In this mode, the TCP3d operates as a single turbo decoder with a single set of input buffers for the code block and a single set of input configuration registers.
- **Double Buffered Mode (Ping/Pong Mode)**: This mode provides a very efficient system interface to the TCP3d by operating as a single turbo decoder with 2 complete sets of input configuration registers and input data buffers. The system can utilize this mode by loading the next code block to be decoded while the TCP3d is decoding the current code block. This reduces the data transfer latency overall. It is generally used for LTE and WiMAX.
- **Split Mode**: This mode configures the TCP3d resources into 2 independent turbo decoders that share a common VBUS interface. In this mode, the TCP3d can decode 2 code blocks in parallel. This mode is only used for WCDMA.

All the tests are measured on a TMS320C6670 DSP operating at 1 GHz. The input data and output data are placed in MSMC SRAM memory.

17.1.1 LTE Throughput

Double Buffered Mode is used for LTE. Sliding window sizes of 32 and 128 are tested and the results are compared. Max-star decoding algorithm is used for TCP3d and a maximum of eight iterations are used to decode the data.

Table 37 shows the TCP3d throughput values for the maximum code block size in LTE.

Table 37 TCP3d Throughput In LTE Mode

Code Block Size	Throughput (Mbps) for Sliding Window size of 32	Throughput (Mbps) for Sliding Window size of 128
6144	118.4	111.6

17.1.2 WCDMA/HSPA Throughput

Split Mode is exclusively used for WCDMA/HSPA. Sliding window sizes of 48 and 128 are tested and the results are compared. A maximum of eight iterations are used to decode the data.

Table 38 shows the TCP3d throughput values for the maximum code block sizes in WCDMA/HSPA mode.

Table 38 TCP3d Throughput In WCDMA/HSPA Mode

Code Block Size	Throughput (Mbps) for Sliding Window size of 48	Throughput (Mbps) for Sliding Window size of 128
5114	86.4	83.7

17.1.3 WiMAX Throughput

Double Buffered Mode is used for WiMAX. Sliding window sizes of 32 and 128 are tested and the results are compared. A maximum of eight iterations are used to decode the data.

Table 39 shows the throughput values for the maximum code block sizes in WiMAX mode.

Table 39 TCP3d Throughput In WiMAX Mode

Code Block Size	Throughput (Mbps) for Sliding Window size of 32	Throughput (Mbps) for Sliding Window size of 128
4800	127.5	118.3

18 TCP3e Throughput

Applies to C6670

TCP3e is a programmable peripheral integrated into the TMS320C6670 DSP used for the encoding of 3GPP, LTE, and WiMAX turbo codes. The inputs into the TCP3e are information bits and the outputs are encoded systematic and parity bits.

The data transfer between the DSP and TCP3e is performed via EDMA transfers. It has a 32 bit wide bus for DMA transfer in and a separate 32 bit wide bus for DMA transfer out of data. TCP3e operates at DSP/3 frequency.

Multiple blocks of the same size and the same offset in memory can be encoded by sending just one set of input configuration parameters where one field, MCNT, corresponds to the number of input code blocks. This reduces the overhead of loading one set of configuration parameters for every code block by the DSP.

All the tests are measured on a TMS320C6670 DSP operating at 1 GHz. The testcase details for measuring the performance are as follows:

- L2 SRAM used for storing input and output data.
- The total number of bits sent for each block size is kept constant and different MCNT values from 1 to 20 are used to compare throughput. For example, for a block length of 6144, a total of 6144×20 bits are sent with different MCNT values.

18.1 LTE Mode Throughput

Table 40 shows the TCP3e throughput values for different block sizes and different MCNT values in LTE mode.

Table 40 TCP3e Throughput For LTE (Part 1 of 2)

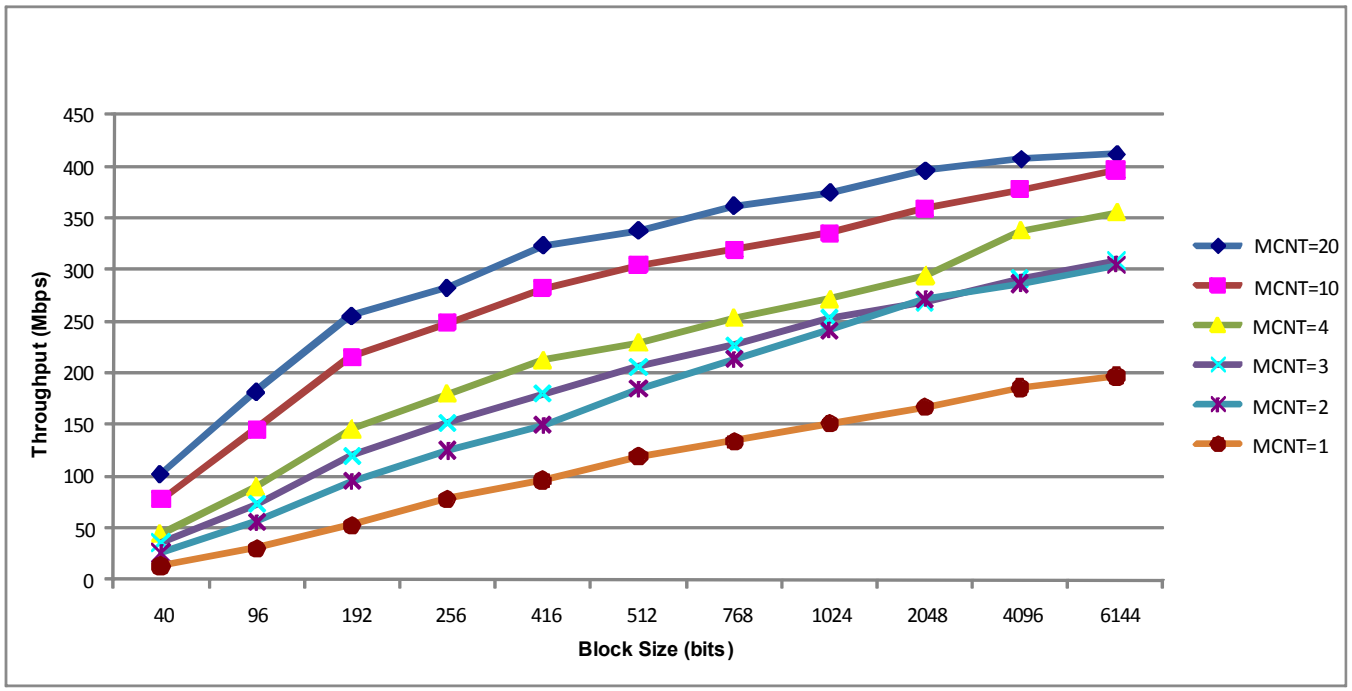
Block Length	Throughput in Mbps					
	MCNT=20	MCNT=10	MCNT=4	MCNT=3	MCNT=2	MCNT=1
40	103	89.5	65.1	56.2	47.3	30.9
96	182	163.1	124.6	107.6	93.4	88.3
192	255.7	234.3	187.7	161.8	145.8	142.1
256	282.6	259.1	212	183.6	167.1	164.3
512	342.2	321	274.1	237.6	225.2	222.1
768	363.8	342	297.2	258.7	248.2	245.9
1024	376.2	358.9	315.8	274.6	266.4	264.6

Table 40 TCP3e Throughput For LTE (Part 2 of 2)

Block Length	Throughput in Mbps					
	MCNT=20	MCNT=10	MCNT=4	MCNT=3	MCNT=2	MCNT=1
2048	396.7	381.4	341.9	297.8	293.5	292.2
4096	408	393.8	356.6	311	309.2	308.6
6144	411.9	398.1	361.8	315.6	314.8	314.2
End of Table 40						

Figure 12 shows the corresponding throughput plot of TCP3e in LTE mode.

Figure 12 TCP3e Throughput for LTE vs. Block Size (DSP @ 1GHz) (Data in L2RAM)



18.2 WCDMA/HSPA Mode Throughput

Table 41 shows the TCP3e throughput values for different block sizes and different MCNT values in WCDMA/HSPA mode.

Table 41 TCP3e Throughput For WCDMA/HSPA (Part 1 of 2)

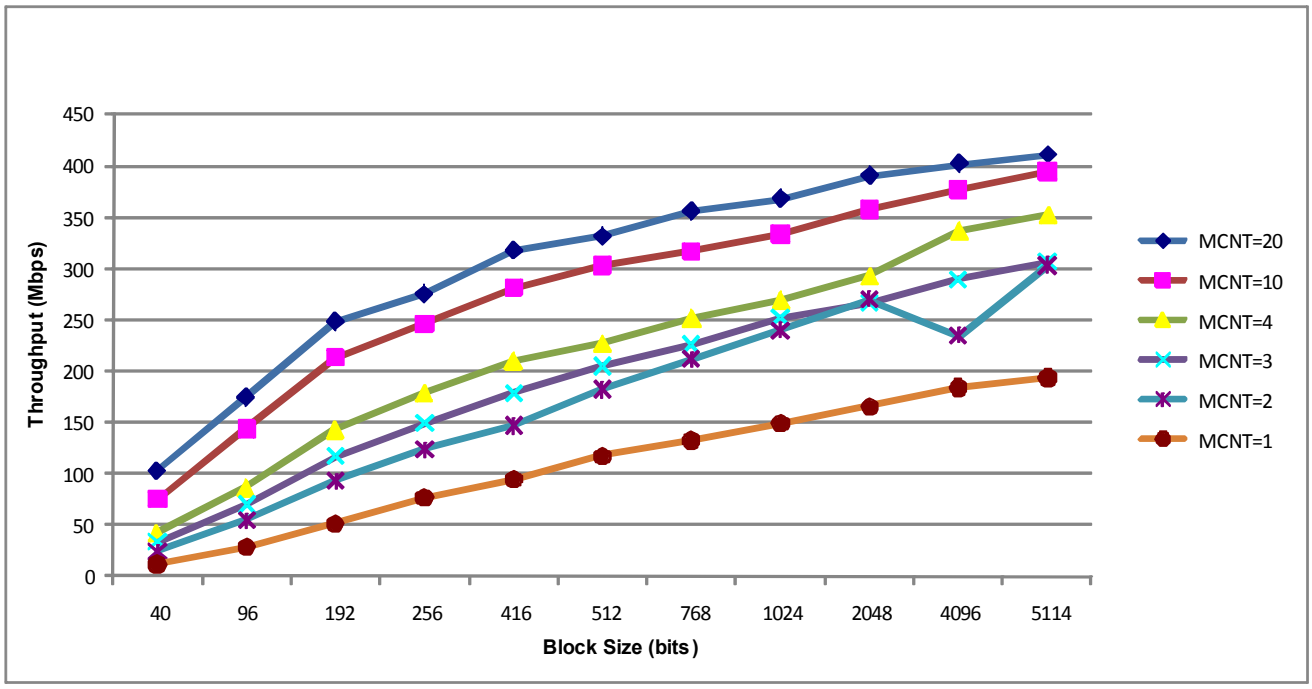
Block Length	Throughput in Mbps					
	MCNT=20	MCNT=10	MCNT=4	MCNT=3	MCNT=2	MCNT=1
40	100.9	87.7	63.8	55	45.6	29.5
96	180.3	161.2	122.7	105.5	91.2	86
192	252	232.7	185	159	143	140
256	280	257.6	210	181.1	165.1	162.1
512	340.2	319.1	272	235.1	223	220.2
768	361.3	340.1	295.8	256.5	246.5	243.9
1024	374.1	356.5	313.6	272.4	264.4	262.2
2048	394.3	379.7	340.1	296.5	291.5	290

Table 41 TCP3e Throughput For WCDMA/HSPA (Part 2 of 2)

Block Length	Throughput in Mbps					
	MCNT=20	MCNT=10	MCNT=4	MCNT=3	MCNT=2	MCNT=1
4096	406	391.2	354	309.6	307.2	306.6
5114	409.7	396.4	359.6	313.8	311.9	311.1
End of Table 41						

Figure 13 shows the throughput plot of TCP3e in WCDMA mode.

Figure 13 TCP3e Throughput for WCDMA vs. Block Size (DSP @ 1GHz) (Data in L2RAM)



18.3 WiMAX Mode Throughput

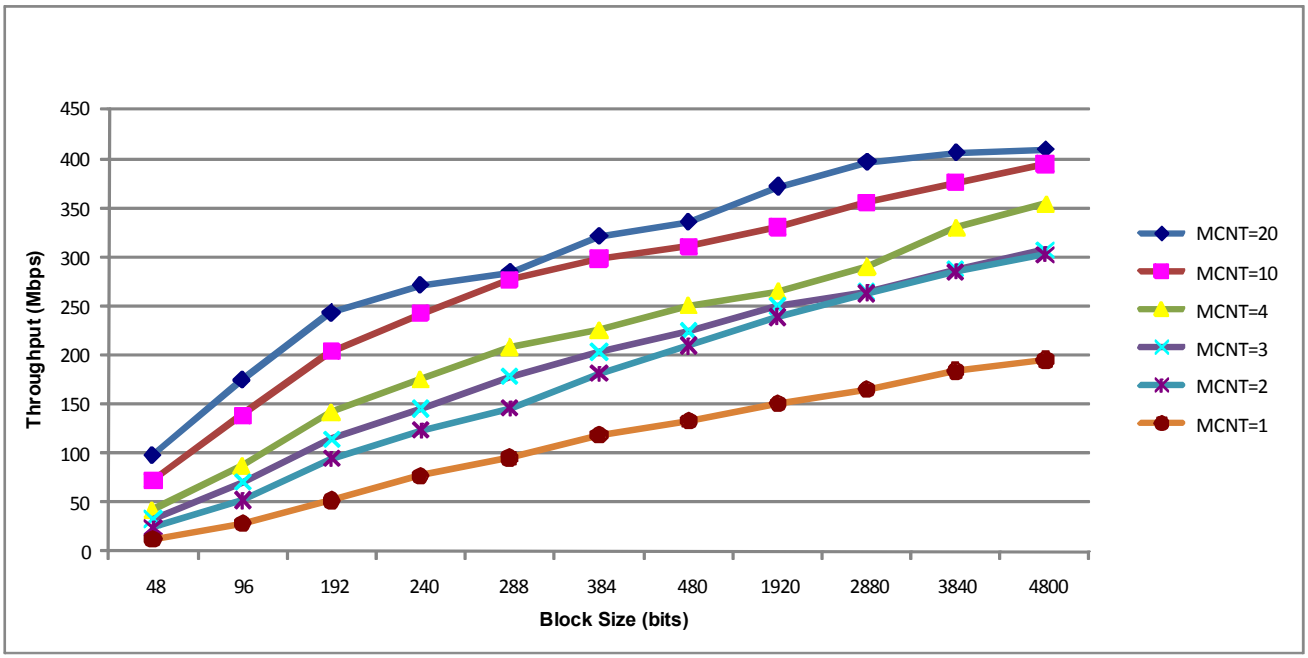
Table 42 shows the TCP3e throughput values for different block sizes and different MCNT values in WiMAX mode.

Table 42 TCP3e Throughput For WiMAX

Block Length	Throughput in Mbps					
	MCNT=20	MCNT=10	MCNT=4	MCNT=3	MCNT=2	MCNT=1
48	102	88.5	64.1	55.2	46.3	30
96	181	162.1	123.2	106.6	92.2	87.2
192	254.8	233.2	186.6	160.1	144.3	141.8
288	281	258	211	182	166	163
384	341	320	272.8	236	224	221
480	362	341	296	257	247	244
1920	375	357	314	273	265	263
2888	395	380	340	296	292	291
3840	407	392	355	310	308	307
4800	410	396.7	360	314	313.5	313
End of Table 42						

Figure 14 shows the throughput plot of TCP3e in WiMAX mode.

Figure 14 TCP3e Throughput for WiMAX vs. Block Size (DSP @ 1GHz) (Data in L2RAM)



19 UART

Applies to All KeyStone Devices

This section discusses the throughput performance of the Keystone universal asynchronous receiver/transmitter (UART) peripheral. The UART module provides an interface between the DSP and UART terminal interface or other UART-based peripheral, and is based on the industry standard TL16C550 asynchronous communications element. The UART peripheral connects to the TeraNet through a 32-bit interface. The UART peripheral is a bus slave, which means that it is not capable of reading and writing memory, and must rely on another master, such as EDMA, to provide data to the UART peripheral.

The operating rate of the UART module is based on DSP/6. The I/O clock frequency for this module is further divided down from the DSP/6 clock inside the UART module itself. The maximum supported UART I/O clock is 128 kilobits per second. The remainder of this section discusses the Theoretical Throughput Performance and the Measured Throughput Performance of the UART peripheral.

19.1 Theoretical Throughput Performance

This section discusses the theoretical throughput performance of the UART peripheral. Table 43 shows the maximum theoretical throughput for several configurations of stop and parity bits when operating at the maximum clock I/O frequency. Although the oversampling mode does influence the throughput, it is not taken into consideration in this table because the impact is negligible. As can be seen in the table, the highest throughput can be achieved when there is 1 start bit, 8 character bits, 0 parity bits, and 1 stop bit.

Table 43 UART Theoretical Throughput Performance (Part 1 of 2)

Start Bits	Character Bits	Parity Bits	Stop Bits	Throughput (Character Bits/Total Bits)
1	5	0	1	91.4
1	5	0	1.5	85.3
1	5	0	2	80.0
1	5	1	1	80.0
1	5	1	1.5	75.3
1	5	1	2	71.1
1	6	0	1	96.0
1	6	0	1.5	90.3
1	6	0	2	85.3
1	6	1	1	85.3
1	6	1	1.5	80.8
1	6	1	2	76.8
1	7	0	1	99.6
1	7	0	1.5	94.3
1	7	0	2	89.6
1	7	1	1	89.6
1	7	1	1.5	85.3
1	7	1	2	81.4
1	8	0	1	102.4
1	8	0	1.5	97.5
1	8	0	2	93.1

Table 43 UART Theoretical Throughput Performance (Part 2 of 2)

Start Bits	Character Bits	Parity Bits	Stop Bits	Throughput (Character Bits/Total Bits)
1	8	1	1	93.1
1	8	1	1.5	89.0
1	8	1	2	85.3
End of Table 43				

19.2 Measured Throughput Performance

This section discusses the measured throughput performance of the Keystone UART peripheral. The throughput shown in Table 44 was measured on a TMS320C6678 device. For this test, the UART I/O clock was configured to operate at the maximum I/O clock frequency, 128 KHz. The UART module was set up in loopback mode, with the source and destination being L2 memory.

Table 44 UART Measured Throughput Performance

Start Bits	Character Bits	Parity Bits	Stop Bits	Theoretical Throughput (Kbaud) (Data Rate * Char Bits / Total Bits)	Measured Throughput (Kbaud)
1	8	1	1	128 kHz × (8 /11) Bits = 93.09	86.2

20 VCP2 Throughput

Applies to C6670

The Viterbi Decoder Coprocessor (VCP2) has been designed to perform decoding of convolutional encoded data for IS2000 and 3GPP wireless standards. It is designed to operate at 1/3 DSP frequency. The inputs are provided as 8-bit branch metrics and the output of the VCP2 can be hard decisions or 8-bit soft decisions. There is a trace-back unit inside the VCP2 which performs the Viterbi backward recursion and generates hard decisions or soft decisions.

If the hard/soft decisions do not fit in the trace-back unit memory, then the *convergent* mode or *mixed* mode is used and the original frame is segmented into sliding windows (SW). Otherwise, the trace-back mode is set to *tailed* mode and no segmentation is required.

All the tests were measured on a TMS320C6670 DSP operating at 1 GHz. The throughput testcase details are as follows:

- L2 memory endpoints are used for both input and output data.
- Different frame lengths are used for each user to test it in tailed/mixed/convergent mode.
- Decoding time is calculated by taking time stamps at start of the decoding, end of decoding and subtracting the interrupt ISR latencies from the overall time taken.
- Throughput in Mbps = (Bytes transferred at output × 8 × 1000)/(VCP2 decode time). For soft decisions throughput, since each output bit is represented by 8 bits of data, the throughput is divided by 8.

Table 45 represents VCP2 throughput for hard decisions on the output.

Table 45 VCP2 Hard Decisions Throughput

Constraint Length	Code Rate	Mode	Frame Length	VCP2 Decode Time	Bytes transferred at output	Throughput in Mbps
9	1/4	Tailed	120	19642	24	9.8
5	1/2	Tailed	1000	28796	136	37.8
7	1/3	Mixed	6000	299956	760	20.3
9	1/2	Convergent	6000	560674	760	10.8
End of Table 45						

Table 46 represents VCP2 throughput for soft decisions on the output.

Table 46 VCP2 Soft Decisions Throughput

Constraint Length	Code Rate	Mode	Frame Length	VCP2 Decode Time	Bytes transferred at output	Throughput in Mbps
7	1/3	Convergent	2500	122465	2508	20.5
9	1/3	Mixed	2500	238044	2508	10.5
End of Table 46						

21 Revision History

Table 47 lists the change history for this document.

Table 47 Document Revision History

Revision	Comments
A1	Added Revision History. Editorial update only; no technical changes.
A	Added the Multicore Navigator Throughput section. (Page 27) Added new graphic - DirectIO NREAD Throughput with 5 Gbps PHY with Overhead (Page 46) Added new graphic - DirectIO NREAD Throughput with 3 Gbps PHY with Overhead (Page 45) Added new graphic - DirectIO NWRITE Throughput with 5 Gbps PHY with Overhead (Page 45) Added new graphic - DirectIO NWRITE Throughput with 3 Gbps PHY with Overhead (Page 44) Added new graphic - Type 9 Throughput with 5 Gbps PHY with Overhead (Page 50) Added new graphic - Type 9 Throughput with 3 Gbps PHY with Overhead (Page 49) Added new graphic - Type 11 Throughput with 5 Gbps PHY with Overhead (Page 48) Added new graphic - Type 11 Throughput with 3 Gbps PHY with Overhead (Page 47) Added BCP HSUPA and HSDPA Throughput Results (Page 19) Added BCP LTE PUSCH and PDSCH Throughput Results (Page 18) Added 1x mode and 2x mode Type 9 Message Passing Throughput Results for 5 Gbps PHY (Page 50) Added 1x mode and 2x mode Type 11 Message Passing Throughput Results for 5 Gbps PHY (Page 48) Added DirectIO Read Throughput Results for 1x and 2x Modes (Page 43) Added DirectIO Write Throughput Results for 1x and 2x Modes (Page 42) Added Terminology Section (Page 3) Added Type 9 Message Passing Throughput Results for 3.125 Gbps PHY (Page 49) Added Type 11 Message Passing Throughput Results for 3.125 Gbps PHY (Page 47)
Initial	Initial release.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265