Application Report
SPRAAY0A–October 2008–Revised October 2009

TMS320C6472/TMS320TCI6486 Throughput

Brighton Feng, Tom Johnson, Yanmin Wu — *Digital Signal Processing Solutions*

ABSTRACT

The TMS320C6472/TMS320TCI6486 has six C64x+ Megamodule cores that run at 500 MHz, 625 MHz, or 700 MHz. This document has been written based on the performance of the C6472/TCI6486 device running at 500 MHz and 625 MHz. Each megamodule has 32KB of L1D SRAM, 32KB of L1P SRAM, and 608KB of LL2 SRAM and all six cores share 768KB of SL2 SRAM. A 32-bit 533-MHz DDR2 SDRAM interface is provided on the DSP to support up to 512MB of external memory. The C6472/TCI6486 also has an Enhanced DMA engine and a high-bandwidth DMA switch fabric capable of connecting all of these entities.

Memory access performance is very critical for software running on the DSP. On the C6472/TCI6486 DSP, all memories can be accessed by DSP cores and by multiple DMA masters. Each DSP core is capable of performing up to 128 bits of load/store operations per cycle. When accessing L1D SRAM, the DSP core can access the memory at up to 8 GBps at a 500-MHz core clock frequency, or 10 GBps at a 625-MHz core clock frequency.

The DMA switch fabric is a 128-bit bus structure operating at the clock rate of CPU/3. It can sustain data throughput up to 2.666 GBps at a 500 MHz core clock rate or 3.333 GBps at a 625-MHz core clock rate per connection. Since this DMA switch fabric supports multiple, concurrent connections, each connection may achieve this theoretical limit as long as there are no conflicts.. The DMA switch fabric interconnects the C64x+ megamodule cores (including their local memories), external DDR2 memory, EDMA controllers and peripheral interfaces.

The C6472/TCI6486 device contains high-bandwidth peripheral interfaces such as Gigabit Ethernet, Serial RapidIO®, TSIP, UTOPIA, and DDR2. Sustaining high throughput through these interfaces is very critical to meeting system and application requirements.

This document provides designers a basis for estimating memory access performance based on throughput measurements under various operating conditions. Some factors affecting memory access performance are discussed. It also addresses throughput to/from the interfaces to memories of the C6472/TCI6486 device. This can be used to estimate transport performance of the C6472/TCI6486 device to facilitate system design.

For a detailed functional description of the megamodule and peripherals, see the corresponding C6472/TCI6486 peripheral user's guides. For AC timings and register offsets, see the *TMS320TCI6486 Communications Infrastructure Digital Signal Processor* data manual (SPRS300) or the *TMS320C6472 Fixed-Point Digital Signal Processor* data manual (SPRS612).

Contents

List of Figures

List of Tables

1 Related Documentation

For details about the C64x+ Megamodule, see the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871).

For configuration of the EMAC interface, see the *TMS320C6472/TMS320TCI6486 DSP Ethernet Media Access Controller (EMAC)/Management Data Input/Output (MDIO) Module User's Guide* (SPRUEF8).

For configuration of the SRIO interface, see the *TMS320C6472/TMS320TCI648x DSP Serial RapidIO (SRIO) User's Guide* (SPRUE13).

For configuration of the TSIP interface, see the *TMS320C6472/TMS320TCI6486 DSP Telecom Serial Interface Port (TSIP) User's Guide* (SPRUEG4).

For configuration of the Utopia interface, see the *TMS320C6472/TMS320TCI6486 DSP Universal Test and Operations PHY Interface for ATM 2 (UTOPIA2) User's Guide* (SPRUEG2).

For configuration of the EDMA3, see the *TMS320C6472/TMS320TCI648x DSP Enhanced DMA (EDMA3) Controller User's Guide* (SPRU727).

For configuration of the HPI, see the *TMS320C6472/TMS320TCI6486 DSP Host Port Interface (HPI) User's Guide* (SPRUEG1).

For more information about the EDMA3 performance on the TCI6482 device, see the *TMS320TCI6482 EDMA3 Performance* application report (SPRAAG8).

2 Abbreviations

ATM — Asynchronous Transfer Mode

CC — EDMA Channel Controller

CODEC — Coder/Decoder

CPU — Central Processing Unit

DMA — Direct Memory Access

DDR — Double-Data Rate

DSP — Digital Signal Processor

EDMA — Enhanced Direct Memory Access

EMIF — External Memory Interface

FIFO — First-In First-Out

GMII — Gigabit Media Independent Interface

HPI — Host Port Interface

IDMA — Internal DMA

L1D — Level-1 Data Memory

L1P — Level-1 Program Memory

LDW — Load Word

LL2 — Local Level-2 Memory

MAC — Media Access Control

MII — Media Independent Interface

MPHY — Multi-PHY

PDMA — Pseudo-Direct Memory Access

PHY — Physical Interface

PLL — Phase-Locked Loop

RGMII — Reduced Gigabit Media Independent Interface

RMII — Reduced Media Independent Interface

Rx — Receive

S3MII — Source-Synchronous Serial Independent Interface

SCR — Switched Central Resource

SDRAM — Synchronous Dynamic Random Access Memory

SL2 — Shared Level-2 Memory

SPHY — Single-PHY

SRAM — Static Random Access Memory

SRIO — Serial RapidIO

STW — Store Word

TC — EDMA Transfer Controller

TR — Transfer Request

TSIP — Telephony Serial Interface Port

Tx — Transmit

UTOPIA — Universal Test and Operations PHY Interface for ATM

3 Introduction

This document presents the expected throughput of the C6472/TCI6486 device between defined endpoints. Endpoints are peripherals, C64x+ megamodules, or memories. This throughput data was compiled by configuring the endpoints or transport mechanisms in various modes and then transferring blocks of data. By measuring the time required to transport this data, the throughput between the endpoints was then determined.

An overview of the TMS320C6472/TMS320TCI6486 device and a detailed discussion of the system interconnect is provided to help explain the throughput results obtained. The EDMA subsystem is also discussed in detail. After these introductory topics, throughput metrics are presented for the primary peripheral interfaces. The remainder of this document addresses throughput from the perspective of the C64x+ processors.

4 TMS320C6472/TMS320TCI6486 Overview

The TMS320C6472/TMS320TCI6486 device is a fixed-point, digital signal processor (DSP) optimized for telephony infrastructure and video transcoding applications. It has six TMS320C64x™ megamodules for data processing and each contains large tightly-coupled SRAM memories to handle large amounts of channel data during CODEC processing. The C6472/TCI6486 device also includes a large shared SRAM memory.

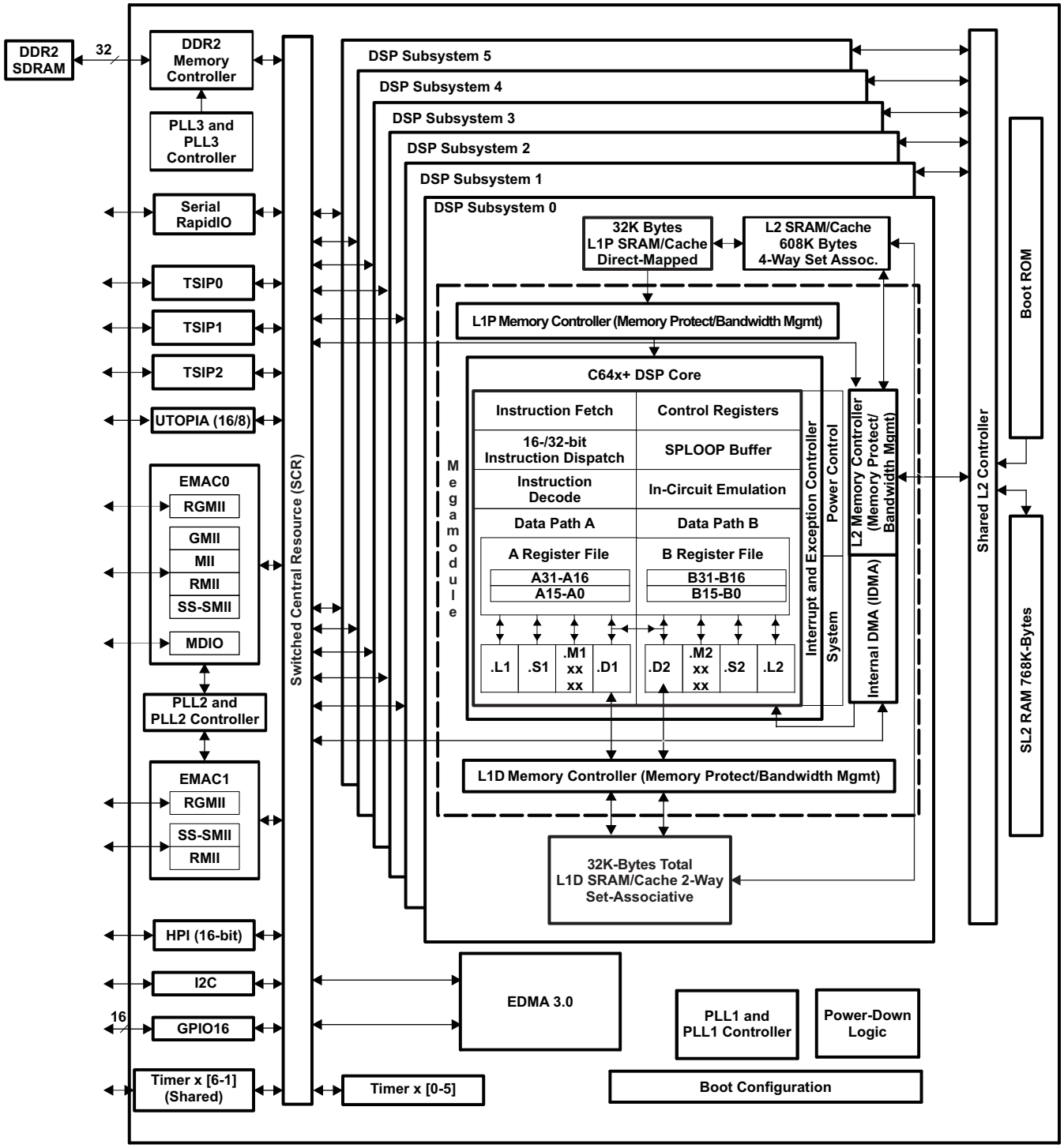
Each of the C64x+ megamodules processes multiple operations each clock cycle. Therefore, the data access bandwidth into these cores is very high and stalls must be minimized. Each megamodule contains a large amount of closely-coupled memory organized as a two-level memory system. The Level-1 program and data memories (L1P and L1D) on each C64x+ megamodule are 32KB each. The Level 2 memory (LL2) on each megamodule is shared between program and data space and is 608KB. Additionally, the C6472/TCI6486 device contains 768KB of on-chip L2 memory that is shared by all six megamodules (SL2).

In addition to the high-performance megamodules and their memories, the C6472/TCI6486 device has multiple high-throughput interfaces. The high-throughput interfaces are:

- Three Telecom Serial Interface Ports (TSIP) each supporting 1024 bi-directional TDM 8-bit timeslots, each operating at 64 Kbps.
- A 16-bit/8-bit UTOPIA slave port supporting up to 800 Mbps bi-directional.
- Two 10/100/1000 Ethernet Media Access Controllers (EMAC) each supporting up to 1 Gbps bi-directional.
- A Serial RapidIO (SRIO) interface with two 1x lanes each operating at 1.25, 2.5 or 3.125 Gbps bi-directional.
- A 32-bit DDR2-533 SDRAM interface with up to 2133 Mbps of throughput.
- A 16-bit multiplexed host-port interface (HPI16) with up to 41.7 Mbps of throughput.

Figure 1 shows the C6472/TCI6486 device functional block diagram.

Figure 1. C6472/TCI6486 Block Diagram



5 System Interconnect

The system interconnect, also referred to as a DMA switch fabric, implemented in the C6472/TCI6486 device is known as a Switched Central Resource (SCR). The SCR combined with bridges and the Enhanced DMA (EDMA) module provides the transport between the endpoints. Bridges at some SCR bus interfaces support interconnect to endpoints in a different clock domain or those with a different bus width.

The modules on the C6472/TCI6486 device are interconnected through two switch fabrics, one for data transfers and one for configuration transfers. Each of the switch fabrics individually allows for low-latency, concurrent data transfers between these modules. The data switch fabric is the only one considered in this document, as it carries all of the high-bandwidth data transfers.

Endpoints can be classified into two categories: masters and slaves. Masters are capable of initiating read and write transfers in the system to slave ports. Slaves, on the other hand, cannot initiate transfers. Masters include the C64x+ megamodules, the EDMA transfer controllers, and the EMAC, TSIP, HPI, UTOPIA, and SRIO peripherals. Slaves include the DDR2 EMIF, the C64x+ megamodule local memories, and the shared memory. The EDMA transfer engine must be used to perform slave-to-slave transfers. The EDMA transfer controllers (TC0 through TC3) connect to the switch fabric as masters (each TC has two master ports connecting with the switch fabric) so that they can transfer between slave endpoints.

The switch fabric provides arbitration between system masters when they attempt to access the same slave simultaneously. As long as two masters are communicating with two separate slaves, the transfers can be concurrent.

The data SCR is a high-throughput interconnect used to move blocks of data across the chip very quickly. The data SCR connects masters to slaves via 128-bit data buses running at the CPU frequency divided by 3 (CPU/3). (CPU/3 is the DSP core clock generated by PLL1 and then divided by 3 in the PLL1 controller.) Endpoints that have a 128-bit data bus interface running at this speed can connect directly to the data SCR; other endpoints require a bridge.

Bridges connected to the data SCR perform bus width conversion when the endpoint has a bus width other than 128 or clock domain arbitration when the endpoint bus operates at a frequency other than CPU/3. For example, TSIP modules require a bridge to convert their 32-bit data bus interface into a 128-bit interface so that they can connect to the data SCR. Bridges also support queuing of transfer requests to allow optimized data transport.

5.1 Data Switch Fabric Connections

Figure 2 shows the connections between slaves and masters through the data SCR. Masters are shown on the left and slaves on the right. The C64x+ megamodules have both slave and master ports on the data SCR. Also note in Figure 2 that the EMAC, UTOPIA, and HPI peripherals all share a single port on the data SCR. This affects the throughput of each when more than one is active.

Not all masters on the C6472/TCI6486 DSP may connect to all slaves directly. This implementation was chosen to manage the complexity of the data SCR routing. Allowed connections are summarized in Table 1. This table shows which master-slave connections are direct (Y) and which are *logical* (C). Logical connections are actually chained connections through the cross-connect bridges, Xconn1, and Xconn2. Direct connections always yield higher throughput rates and less latency.

Note that TC0 and TC1 do not connect directly to the megamodule memories but they must use the cross-connect bridges, whereas TC2 and TC3 connect directly to the megamodule memories. TC2 and TC3, therefore, have higher performance when transferring data blocks into and out from the megamodule memories.

Figure 2. DMA Switched Central Resource Block Diagram

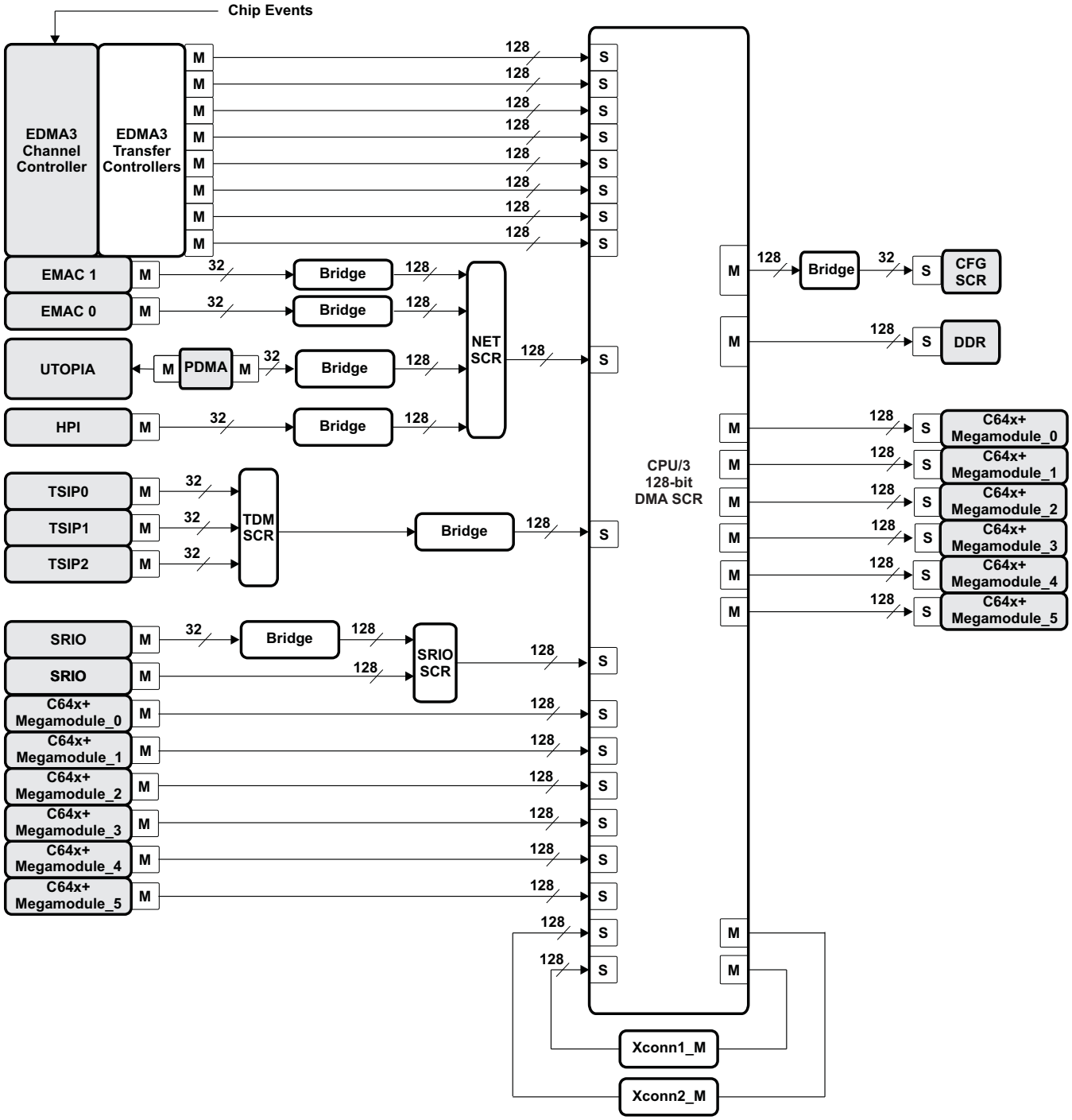


Table 1. DMA SCR Connection Matrix[1] [2] [3]

MASTERS	SLAVES									
	CFGSCR	DDR2	C64x+ Megamodule0	C64x+ Megamodule1	C64x+ Megamodule2	C64x+ Megamodule3	C64x+ Megamodule4	C64x+ Megamodule5	Xconn1_S	Xconn2_S
EDMA3 Transfer Controller0_R	Y	Y	C	C	C	C	C	C	Y	N
EDMA3 Transfer Controller0_W	Y	Y	C	C	C	C	C	C	Y	N
EDMA3 Transfer Controller1_R	Y	Y	C	C	C	C	C	C	N	Y
EDMA3 Transfer Controller1_W	Y	Y	C	C	C	C	C	C	N	Y
EDMA3 Transfer Controller2_R	Y	Y	Y	Y	Y	Y	Y	Y	N	N
EDMA3 Transfer Controller2_W	Y	Y	Y	Y	Y	Y	Y	Y	N	N
EDMA3 Transfer Controller3_R	N	Y	Y	Y	Y	Y	Y	Y	N	N
EDMA3 Transfer Controller3_W	N	Y	Y	Y	Y	Y	Y	Y	N	N
EMAC0	Y	Y	Y	Y	Y	Y	Y	Y	N	N
EMAC1	Y	Y	Y	Y	Y	Y	Y	Y	N	N
UTOPIA	Y	Y	Y	Y	Y	Y	Y	Y	N	N
HPI	Y	Y	Y	Y	Y	Y	Y	Y	N	N
3x TSIP	N	Y	Y	Y	Y	Y	Y	Y	N	N
RapidIO	Y	Y	Y	Y	Y	Y	Y	Y	N	N
C64x+ Megamodule0	N	Y	N	C	C	C	C	C	Y	N
C64x+ Megamodule1	N	Y	C	N	C	C	C	C	Y	N
C64x+ Megamodule2	N	Y	C	C	N	C	C	C	Y	N
C64x+ Megamodule3	N	Y	C	C	C	N	C	C	N	Y
C64x+ Megamodule4	N	Y	C	C	C	C	N	C	N	Y
C64x+ Megamodule5	N	Y	C	C	C	C	C	N	N	Y
Xconn1_M	N	N	Y	Y	Y	Y	Y	Y	N	N
Xconn2_M	N	N	Y	Y	Y	Y	Y	Y	N	N

(1) Y = Direct connection in SCR.
(2) C = Logical connection through Xconn1 or Xconn2 bridges.
(3) N = No physical connection.

6 Transfer Throughput

Theoretically, throughput is simply calculated by dividing the number of bytes being transferred between two endpoints by the amount of time required to complete the transfer. The SCR switch fabric provides the interconnect between the C64x+ megamodules (including their associated on-chip memories), external memory, the EDMA controller, and peripherals. Transfer throughput in a customer's application could be impacted by many elements such as transfer priority, endpoint clock rate, other SCR transfers, endpoint activity, etc. The throughput tests that quantify the peripheral's maximum sustained throughput were conducted under the best-case conditions with the endpoints operating at their maximum rated speed.

6.1 EDMA Operation

The EDMA (also known as EDMA3) is made up of a Channel Controller (CC) and four Transfer Controllers (TCs). The channel controller supports 64 DMA channels. It manages the transfer controllers that perform the DMA transfers. The EDMA is the engine that performs slave-to-slave data transfers. The EDMA is optimized to perform all data movement between the level-one data (L1D) memories, local level-two (LL2) memories, shared level-two (SL2) memory, and external DDR2 memory. The EDMA can also be used to access the level-one program (L1P) memories, but that is not relevant in the context of throughput performance.

The EDMA contains four transfer controllers, TC0 through TC3. Each transfer controller (TC) contains two master ports each with an independent connection to the data SCR. Each TC also contains a buffer so that the data being read by one master port can be written with minimum latency. The buffers are 128 bytes for TC0 and TC1 and 256 bytes for TC2 and TC3. Throughput results are affected by TC choice.

6.2 EDMA Throughput

Since all connections through the SCR are master-slave, the EDMA must exist to allow transfers between slaves. (Each TC has two master ports connecting to the data SCR; one for reads and one for writes.) The channel controller within the EDMA can initiate transfers through the transfer controllers based on event-triggers or when requested by software. The 128-bit data bus connected to the TC runs at the core frequency divided by three (CPU/3), so, in theory, each transfer engine is capable of handling up to 2666 Mbps of data throughput at a core clock rate of 500 MHz (128 / 8 * 500M / 3 = 2666 Mbps). At a core clock rate of 625 MHz, each transfer engine is capable of handling up to 3333 Mbps of data throughput (128 / 8 * 625M / 3 = 3333 Mbps). This establishes the maximum throughput for any transfer across the SCR including transfers between internal memories (LL2 memories, the SL2 memory, and L1D memories).

The maximum throughput for the DDR2 memory is limited by the DDR2 memory interface. For a 16-bit DDR2 memory interface with a 533.3-MHz DDR2 data rate, the theoretical maximum throughput is 533.3M * 16 / 8 = 1066 Mbps. A 32-bit DDR2 memory interface has a theoretical maximum throughput of 533.3M * 32 / 8 = 2133 Mbps.

Reduced throughput occurs when multiple data transactions attempt to access the same SCR endpoint simultaneously. Best-case SCR operation occurs when multiple transfers occur simultaneously where no two transfers use the same SCR port. This is because the SCR supports concurrent transfers between non-conflicting master-slave pairs. If transfers occur such that the source or destination endpoint is the same, then arbitration occurs and the lower priority transaction is stalled until the higher priority transfer completes.

The results shown in Table 2 were taken with the C64x+ megamodules running at 500 MHz and the DDR2 External Memory Interface (EMIF) operating at a data rate of 533.3 MHz (266.6 MHz clock). Throughput rates were computed from test cases where data was transferred between LL2 memory, SL2 memory, L1D memory, and DDR2 memory.

Table 2. EDMA Maximum Measured Throughput

Source and Destination Endpoints	Measured Throughput (MBps)				Maximum Throughput	Throughput Percentage (%)			
	TC0	TC1	TC2	TC3		TC0	TC1	TC2	TC3
Megamodule 0 LL2 to Megamodule 1 LL2	709	709	1765	1765	2666	26.6	26.6	66.2	66.2
Megamodule 0 SL2 to Megamodule 1 SL2	665	665	1682	1682	2666	24.9	24.9	63.1	63.1
32-Bit DDR2 to 32-Bit DDR2	403	403	675	675	2133	18.9	18.9	31.6	31.6
32-Bit DDR2 to Megamodule 0 LL2	786	786	1551	1551	2133	36.8	36.8	72.7	72.7
Megamodule 0 LL2 to 32-Bit DDR2	709	709	1739	1739	2133	33.2	33.2	81.5	81.5
32-Bit DDR2 to Megamodule 0 SL2	784	785	1551	1551	2133	36.8	36.8	72.7	72.7
Megamodule 0 SL2 to 32-Bit DDR2	686	686	1663	1663	2133	32.2	32.2	78	78
32-Bit DDR2 to Megamodule 0 L1D	754	754	1481	1481	2133	35.3	35.3	69.4	69.4
Megamodule 0 L1D to 32-Bit DDR2	786	786	1503	1504	2133	36.9	36.9	70.5	70.5
16-Bit DDR2 to 16-Bit DDR2	321	321	459	459	1066	30.1	30.1	43	43
16-Bit DDR2 to Megamodule 0 LL2	658	658	1033	1034	1066	61.7	61.7	96.9	96.9
Megamodule 0 LL2 to 16-Bit DDR2	708	708	1032	1033	1066	66.4	66.4	96.8	96.8
16-Bit DDR2 to Megamodule 0 SL2	657	657	1032	1033	1066	61.6	61.6	96.8	96.8
Megamodule 0 SL2 to 16 Bit DDR2	686	686	1033	1033	1066	64.3	64.3	96.9	96.9
16-Bit DDR2 to Megamodule 0 L1D	655	654	1035	1035	1066	61.4	61.3	97	97
Megamodule 0 L1D to 16-Bit DDR2	752	752	1037	1036	1066	70.5	70.5	97.2	97.2

The measured throughput values were obtained by initiating transfers between the chosen endpoints on 16 channels at a time on the same command queue that is associated to a particular TC. In this way, the transfer pipelines are optimally filled so that the maximum throughput can be obtained and measured.

The measured throughput values clearly show the throughput limits imposed by the architecture of the data SCR switch fabric. As stated earlier, TC0 and TC1 do not have direct SCR connections to the megamodule memories but must connect *logically* through the cross-connect bridges. They also have smaller data queues. These factors result in lower throughput values for TC0 and TC1. TC2 and TC3 are much closer to the maximum theoretical limits for memory-to-memory transfers. The DDR2 EMIF limits can also be seen in the data collected from the x16 and x32 tests.

The remainder of this document details the throughput behavior under several different memory transfer test conditions and with other endpoints including the peripheral interfaces.

7 Peripheral Interface Throughput

This section discusses the throughput measurements collected while passing data on and off the DSP through the peripheral interfaces.

7.1 EMAC Throughput

7.1.1 Throughput

The C6472/TCI6486 device contains two Gigabit Ethernet MAC (EMAC) interfaces. The Gigabit Ethernet MAC interface was designed to operate at full speed (1000 Mbps) with sustained throughput to either megamodule local memory or DDR2 EMIF. The C6472/TCI6486 device supports RGMII and GMII as Gigabit Ethernet interfaces and RMII, S3MII, and MII as Fast (10/100) Ethernet interfaces. Throughput data was collected in RGMII, GMII, RMII, and S3MII modes.

Each EMAC module has a CPPI RAM which is 8KB. It is accessed through the configuration SCR. The EMAC uses four 32-bit words as one buffer descriptor for each channel that point to data buffers in the DSP memory. The application software uses the buffer descriptors to define ethernet packets to be sent and empty buffers to be filled with incoming packet data. The CPUs create and maintain these buffer descriptors. The EMAC reads from and writes to these buffer descriptors as it transfers data to or from the buffers. The EMAC supports up to 8 channels of transmit and 8 channels of receive. The number of transmit channels is under software control. The number of receive channels typically matches the number of cores with an extra channel allocated for broadcast data. The buffer descriptors can also be located in the L2 memory of any of the cores. For performance reasons it is recommended that descriptors be kept either in the CPPI memory or in L2 memory. Keeping the descriptors in the CPPI memory allows for control access by the EMAC module to be isolated from other real-time data transfers to megamodule memory, though access latencies by the CPU are longer. Keeping the descriptors in L2 memory allows for faster servicing, and less overhead, by the CPU, which may be desirable if a large number of packets are terminated on the device.

7.1.2 RGMII and GMII

The maximum load scenario occurs on the Ethernet MAC when the interface is operating at a Gigabit (1000 Mbps) data rate. The EMAC throughput results shown in Table 3 and Table 4 for the RGMII and GMII interfaces were collected with a core clock rate of 500 MHz. These measurements were collected with the PHY connected in external loopback. The Ethernet packets used in the throughput tests were 1514 bytes long with an 8-byte preamble. The throughput calculation included the Ethernet header and payload but not the preamble.

Table 3. RGMII Throughput

Source	Destination	Buffer Descriptors	Effective Transmit Throughput (Mbps)	Effective Receive Throughput (Mbps)
L2	L2	L2	994.5	994.6
DDR2	DDR2	L2	984.2	985.4
L2	L2	CPPI	994.7	994.8
DDR2	DDR2	CPPI	982.8	984.6

Table 4. GMII Throughput

Source	Destination	Buffer Descriptors	Effective Transmit Throughput (Mbps)	Effective Receive Throughput (Mbps)
L2	L2	L2	999.9	999.8
DDR2	DDR2	L2	999.8	999.9
L2	L2	CPPI	999.7	999.8
DDR2	DDR2	CPPI	999.8	999.7

7.1.3 RMII and S3MII

EMAC throughput results over the RMII and S3MII interfaces at 10 Mbps and 100 Mbps were collected at a core clock rate of 500 MHz. The results are shown in Table 5 and Table 6. These measurements were collected with the PHY connected in external loopback. The Ethernet packets used in the throughput tests were 1514 bytes long with an 8-byte preamble. The throughput calculation included the Ethernet header and payload but not the preamble.

Table 5. 10 Mbps RMII/S3MII Throughput

Source	Destination	Buffer Descriptors	Effective Transmit Throughput (Mbps)	Effective Receive Throughput (Mbps)
L2	L2	L2	9.9	9.9
DDR2	DDR2	L2	9.7	9.6
L2	L2	CPPI	9.8	9.8
DDR2	DDR2	CPPI	9.6	9.7

Table 6. 100 Mbps RMII/S3MII Throughput

Source	Destination	Buffer Descriptors	Effective Transmit Throughput (Mbps)	Effective Receive Throughput (Mbps)
L2	L2	L2	99.9	99.9
DDR2	DDR2	L2	99.8	99.7
L2	L2	CPPI	99.8	99.8
DDR2	DDR2	CPPI	99.7	99.7

7.1.4 Further Considerations

- There are other masters on the data SCR which means that the sustained throughput of the EMAC peripheral may be impacted if multiple masters are accessing the same memory endpoint, whether L2 or DDR2 memory. The priority of the EMAC transactions can be programmed to a higher priority to minimize the impact, if necessary.
- The EMAC peripheral has 24 64-byte transmit buffers (1536 bytes) to enable the largest allowed packet (1518 bytes) to be sent without the possibility of under-run. The EMAC peripheral also has 68 64-byte receive buffers to allow it to hold several packets of the largest allowed packet size. This receive buffer size allows flow control operation without loss of incoming packet data.
- Receive overrun is prevented as long as the average receive memory transfer latency is less than the time required to receive a 64-byte cell on the wire, 512 ns in Gigabit mode. The latency time includes any required buffer descriptor reads for the packet data.

7.2 SRIO Throughput

Serial RapidIO (SRIO) is a non-proprietary, high-bandwidth, system-level interconnect. It is intended to offer multi-gigabit per second performance levels for chip-to-chip and board-to-board communication. The SRIO interface operates at up to a data rate of 3.125 Gbps per differential pair. An 8b/10b encoding scheme is used to ensure ample data transitions for the clock recovery circuits and to improve the bit error rate. Due to the 8b/10b encoding overhead, the effective data bandwidth per differential pair is 1.0, 2.0, and 2.5 Gbps, respectively. Assuming a 10% packet overhead, the maximum data throughput per 1x link equals 2.25 Gbps. Both links operating simultaneously at full load may have a data throughput of 4.5 Gbps data in both the transmit and receive directions.

The SRIO architecture supports I/O memory mapped (DMA) systems. In addition, message passing is also supported. Since the primary application of this interface is to provide both ASIC-to-DSP and DSP-to-DSP communication, both the I/O memory mapped and message passing protocols are

supported. Regardless of the protocol, the peripheral is designed to be an external slave module that is capable of mastering the internal DMA. This means that an external device can push (burst write) data to the DSP as needed, without having to generate an interrupt to the CPU. This has two big benefits: first, it cuts down on the total number of interrupts and second, it reduces handshaking latency associated with read-only peripherals.

The worst-case throughput scenario occurs when the two 1x SRIO ports are running at the highest speed (3.125 Gbps) and the payload size for the packets is small (32 bytes). Each Direct-IO packet has an overhead of 18 bytes and each Message Passing packet has an overhead of 16 bytes. Therefore, the maximum packet rate on any 1x link at 3.125 Gbps is 6.5M packets per second (Mpps) (1 / {[32 + 16] * 8 / 2.5G} = 6.5M). Therefore, if both bi-directional 1x ports are operating at their maximum packet rate, the peripheral must transfer 26M 32-byte packets over the DMA switch fabric each second or one every 38.5 ns.

The SRIO peripheral has sufficient buffering to hold up to 16 full-size packets of 256 bytes each (4KB). It can also hold up to 128 32-byte packets in the same buffer space. Therefore, multiple transfers can be outstanding at a time to the DMA switch fabric.

Only data packet payloads are sent across the data SCR. DOORBELL packets are terminated in the peripheral and their payload is not passed through the data SCR. DOORBELL packets are used for interrupt generation where appropriate bits are set in the Interrupt Condition Status Register (ICSR). Congestion control packets are also terminated in the peripheral and their payload is not passed through the data SCR.

The following are the SRIO transactions that were tested as part of this throughput analysis:

- NREAD - Read operation, data returned is the response
- NWRITE - Write operation, no response
- NWRITE_R - Robust write with response from the target end-point
- SWRITE - Streaming write
- Messages - Message passing

7.2.1 SRIO Throughput Results in Loopback

The SRIO throughput measurements were collected with the DSP running at the core clock rate of 500 MHz. Data was collected at all three link rates: 1.25, 2.5, and 3.125 Gbps. Tests were run in both a loopback and a two-board configuration. The loopback configuration had the SRIO TX output pair externally looped back to the SRIO RX input pair through SMA connectors. The two-board configuration had the SRIO TX and RX pairs cross connected between two boards through SMA connectors. These two configurations yielded nearly identical results, so only the two-board data is shown below. A range of packet payload sizes was included in this data collection to show the effect of small packets on effective throughput. The results are displayed in Table 7 through Table 9 and Figure 3 through Figure 5.

These results show that the destination memory, whether DDR2 or L2, does not affect the throughput results. SWRITE and NWRITE approach theoretical limits with large packets. However, since reads require waiting for a response, their throughput is lower and is highly dependent on implementation. NWRITE_R throughput is also lower since it requires a response. Throughput improves for all transfer modes as the packet size increases.

Table 7. SRIO Two-Board Throughput (Gbps) - 3.125 Gbps

Packet Size in Bytes	Mode: NWRITE_R		Mode: SWRITE		Mode: NREAD		Mode: NWRITE		Mode: MessPass	
	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR
8	118.5	118.2	295	292.2	118.9	122.9	294.1	292.5		
16	154.1	154.1	396.4	392.5	154	153.4	397.2	391.2		
32	218.9	212.2	582.9	495.7	220.1	211.2	582.2	497	320.5	306.4
64	326.9	325.6	954.9	814.4	327.1	324.3	954.9	814.4	318.3	301.2
128	506.8	490.2	1442	1436.7	493.8	493.6	1442	1436.7	573.8	526.2
256	744.6	713.9	2378	2360.9	744.4	715.3	2381.9	2360.9	1086	967.1
512	822.8	801.6	2434.1	2430	822.9	798.3	2435.1	2426	2043.8	1830
1024	866.9	837.3	2462.6	2459.5	867.5	837.2	2462.6	2460	2282	2283
2048	888.2	861.6	2477.6	2475.5	888.2	862.6	2477.1	2475.8	2464.4	2445.9
4096	903.9	874.3	2485.3	2484.7	901.1	874.8	2485.2	2484.4	2457.1	2455.9

Table 8. SRIO Two-Board Throughput (Gbps) - 2.5 Gbps

Packet Size in Bytes	Mode: NWRITE_R		Mode: SWRITE		Mode: NREAD		Mode: NWRITE		Mode: MessPass	
	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR
8	109.4	109.2	296.3	292.9	109.4	108.5	303.8	293.5		
16	144.1	143.1	395.9	391.2	144	142.4	394.6	392.5		
32	199.5	197.9	583.5	496.6	199.3	198.6	583.5	496.6	320.9	306.6
64	291.8	291.8	957	813.6	292.1	291.1	957	812.9	317.9	301.4
128	448.2	444.3	1424.8	1422.2	448.3	440.4	1426.1	1423.5	572.9	542.2
256	640.6	636.7	1915.4	1904.3	640.8	637.9	1912.9	1903	1083.8	994.6
512	698.1	685.1	1953.6	1949	698.1	684.3	1951	1949	1617.8	1620.7
1024	730.7	717	1974.8	1971.5	738	716	1972.2	1970.9	1831.4	1834.5
2048	752.1	735.2	1984.3	1982.8	752.1	734.3	1983.5	1983	1962.5	1963.6
4096	761.3	742	1989	1988.3	762	742	1989.2	1988.4	1963.7	1964

Table 9. SRIO Two-Board Throughput (Gbps) - 1.25 Gbps

Packet Size in Bytes	Mode: NWRITE_R		Mode: SWRITE		Mode: NREAD		Mode: NWRITE		Mode: MessPass	
	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR	Target: L2	Target: DDR
8	75.8	75.8	294.7	292.5	75.8	75.2	294.7	292.2		
16	95.6	95	397.2	394.6	95.5	95.5	394.2	392.9		
32	130.9	129.7	584.2	497	130.2	129.5	581.6	498.4	322	306.4
64	193.6	190.3	932.5	803.9	191.1	190.6	925.6	801.7	309.8	301
128	280.4	279.3	956.6	950.8	280	277.5	958.4	950.2	497.1	498.3
256	389.3	385.1	967.8	964.9	385.7	384.4	968.7	964.9	655.9	657.2
512	410.1	405.1	982	981.7	410.1	401.9	982.6	981.3	810	810.3
1024	420.2	416.2	989.7	988.9	418.8	414	990	988.6	916.6	917.3
2048	425.8	421.4	993.2	993.3	425.6	419.3	993.1	993.4	981.9	982.1
4096	430.9	423.9	995.2	995.1	428.1	421.7	995.2	995.2	982.2	982.1

Figure 3. SRIO Throughput - 3.125 Gbps

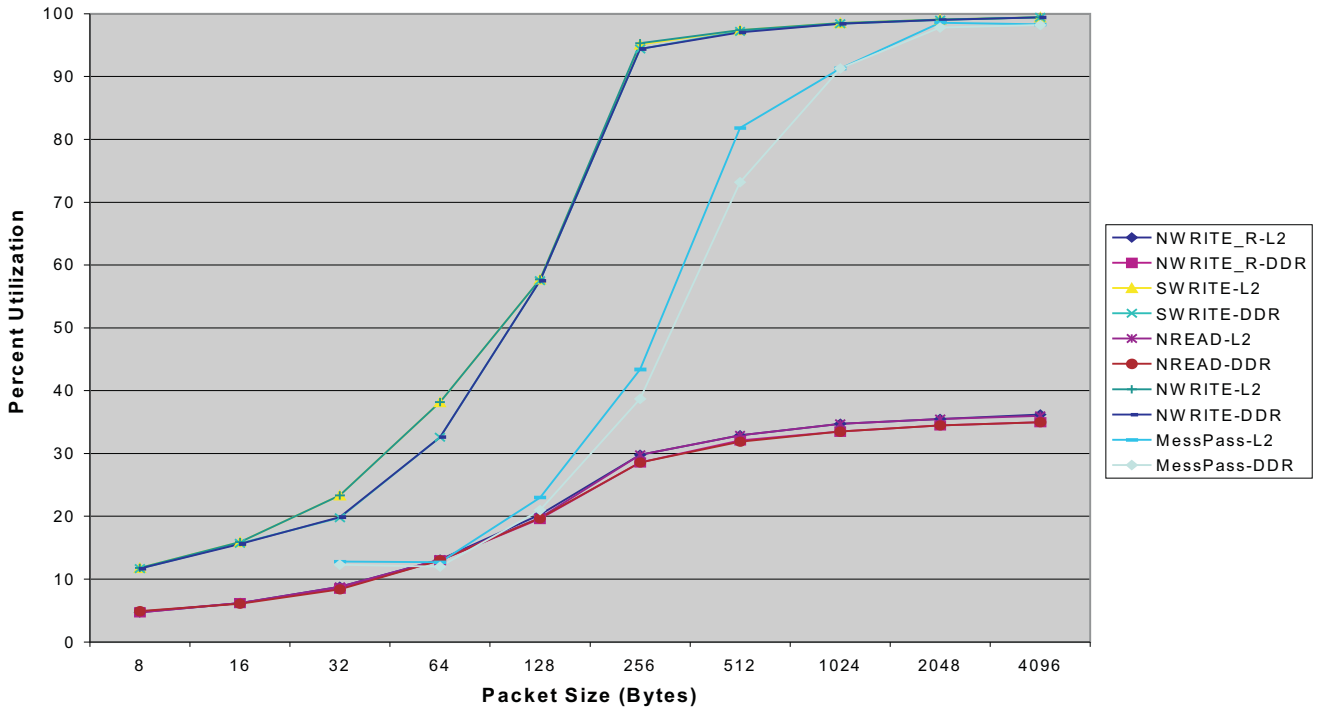


Figure 4. SRIO Throughput - 2.5 Gbps

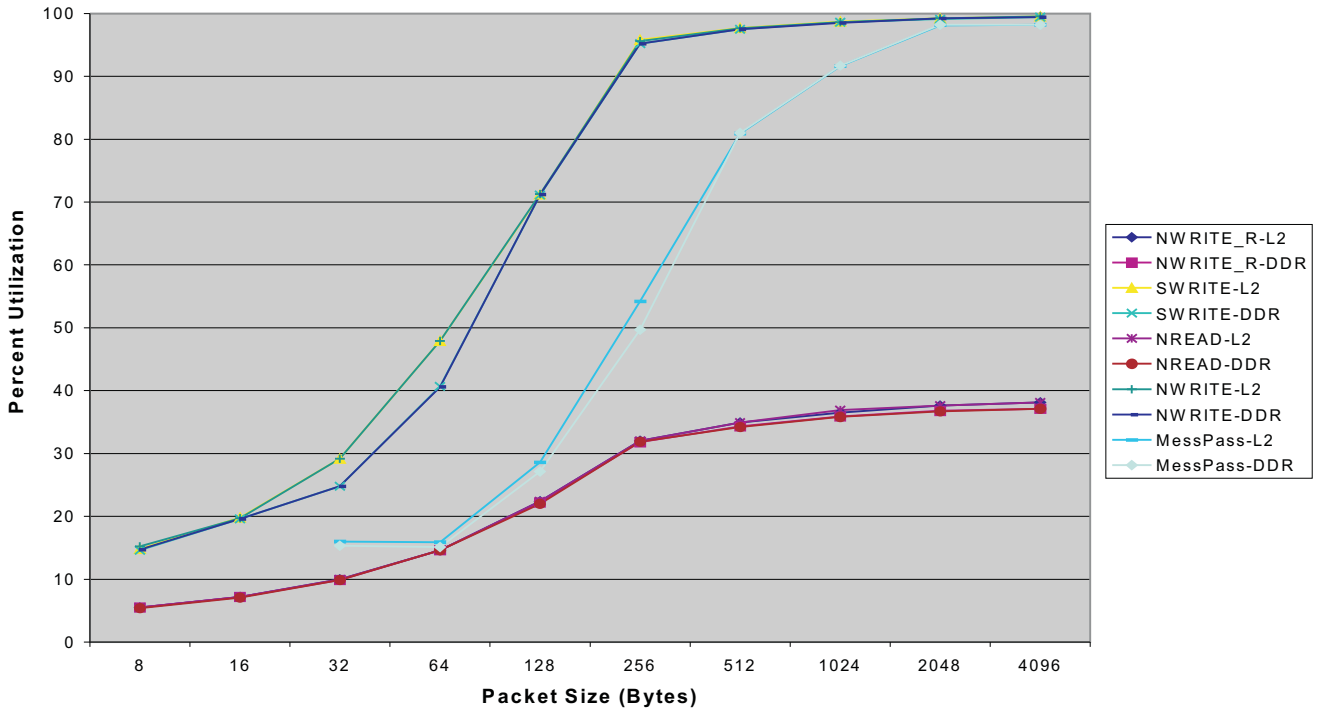
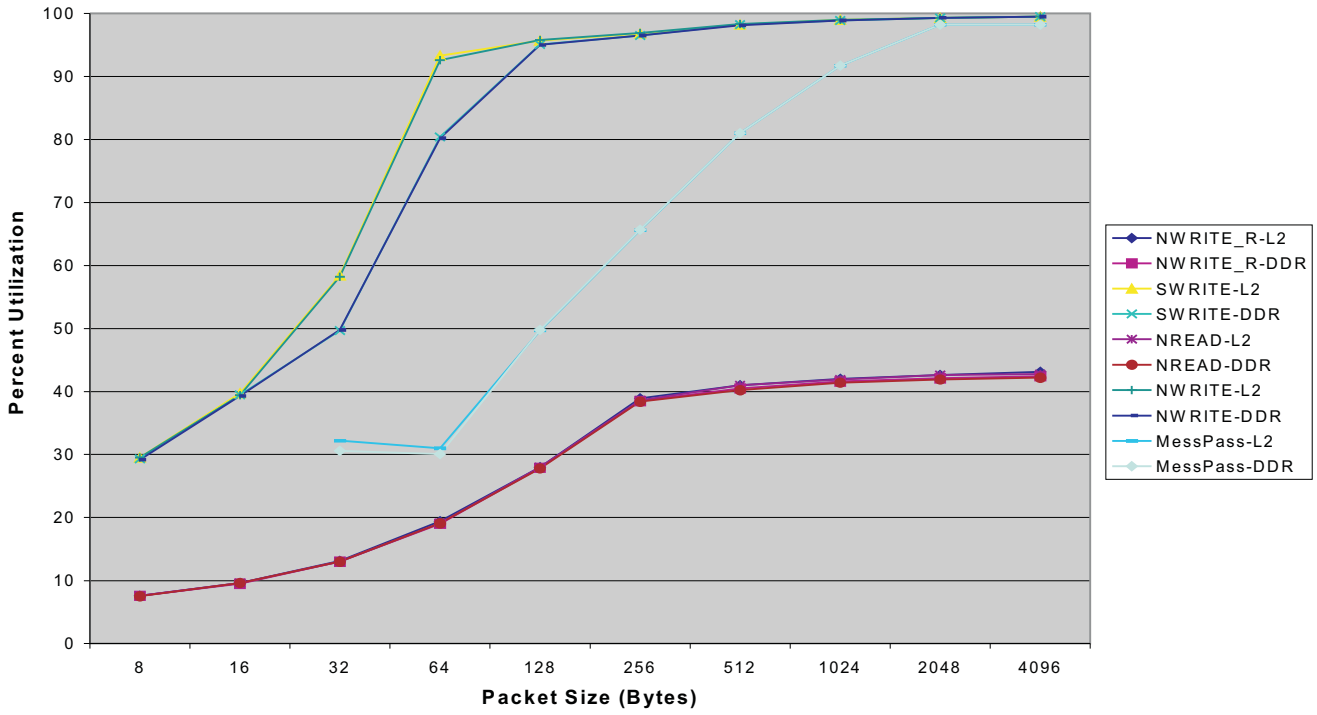


Figure 5. SRIO Throughput - 1.25 Gbps



7.2.2 Further Considerations

- There are other masters on the data SCR, which means that the sustained throughput through the SRIO peripheral may be impacted if multiple masters are accessing the same memory endpoint whether L2 or DDR2 memory. The priority of the SRIO transactions can be programmed to a higher priority to minimize the impact, if necessary.
- There are no traditional DMA FIFOs in the SRIO peripheral. The SRIO module is a master peripheral that initiates its own SCR data transfers between the SRIO peripheral and memory. For example, transmit data is moved into the peripheral only after buffer allocation, handle allocation and flow control is checked. At this point, the peripheral issues a transfer request to the data SCR which then propagates to the memory and completes the transaction by reading the requested block of memory locations. On the RX side, the peripheral accepts data from the SRIO fabric as long as buffer space is available. The peripheral attempts to move received data through the data SCR by issuing transfer requests as frequently as needed to maintain available buffer space. If a stall condition occurs on the data SCR, the receive buffer in the peripheral becomes full. If the condition persists, incoming packets are lost. Physical-layer RETRY responses are then sent to the SRIO fabric to recover the lost packets.

7.3 TSIP Throughput

7.3.1 Throughput

The C6472/TCI6486 device contains three fully-independent TSIP peripherals. TSIP connects to industry standard Time Division Multiplexed (TDM) serial interfaces such as H.110. Each TSIP peripheral has eight transmit data lanes and eight receive data lanes to transport TDM data. Each TSIP also has two frame-sync inputs and two serial clock inputs that establish the frame and channel boundaries for this TDM data.

The TSIP peripheral supports serial data rates of 8.192 Mbps, 16.384 Mbps, and 32.768 Mbps at the RX and TX data lanes so that each can carry 128, 256, and 512 TDM channels, respectively. Regardless of the serial data rate, each TSIP peripheral supports no more than 1024 TDM channels. Each TDM channel transports 64 Kbps of data both in and out.

Internal transfers occur at the frame boundary which repeats every 125 µs (8000 frames per second). Maximum latency occurs when the source and destination buffers are in external DDR2 memory. The maximum throughput demand occurs when all TDM channels are configured for either a-law or u-law CODEC processing - each 8-bit data word at the TDM interface becomes a 16-bit word in the peripheral, compressing on TX and expanding on RX. Data SCR transfers are nearly identical in size for transmit and receive. The only difference is the addition of a 32-bit word that is pre-pended to the data written back to memory.

Throughput measurements were conducted with the three TSIP peripherals as the only masters active on the data SCR. All three TSIP peripherals were configured with 1024 CODEC channels transferring data simultaneously to and from the external DDR2 data buffers. No overflow or underflow conditions were detected. Successful tests were also conducted with the same transport loading with the data buffers placed in either L2 or SL2 memory.

This result is expected since the throughput requirements of the TSIP interface is much lower than the throughput available in the data SCR. The TX throughput load for a single TSIP is 16.384 MBps (8000 * 1024 * 16 / 8). The RX throughput load for a single TSIP is 16.416 MBps (8000 * (1024 + 2) * 16 / 8). All three TSIP interfaces combined only consume 98.4 MBps of data SCR throughput including all RX and TX transactions, which is well within the theoretical limits of the SCR and the memory interfaces.

7.3.2 Further Considerations

- In the event that there is no space in the channel buffers for received data, a buffer overflow occurs. The timeslot samples that have been received since the last frame boundary are lost and an error is logged.
- In the event that the channel buffers are empty and transmit data is needed for the next enabled timeslot, a buffer underflow occurs. The timeslot for that channel is treated as if it was disabled and an error is logged.
- Because of the limited buffering in TSIP, it is recommended that TSIP priority be set to the highest in the system.

7.4 UTOPIA Throughput

7.4.1 Throughput

The UTOPIA peripheral on C6472/TCI6486 device operates only in the slave mode as defined in the UTOPIA Level-2 standard. It supports both the polled (MPHY) and static (SPHY) flow control signaling. The UTOPIA peripheral was implemented to operate at the maximum sustained throughput clock rate of 50 MHz on both transmit and receive channels simultaneously. The C6472/TCI6486 UTOPIA implementation is capable of operating at the maximum clock rate while never overflowing or underflowing the internal buffers while operating in all valid modes as long as no data SCR blocking occurs.

The UTOPIA interface operates with a 50-MHz input clock synchronous with the UTOPIA master. It transports RX and TX data with a sustained throughput of 100 MBps in 16-bit mode and 50 MBps in 8-bit mode. The UTOPIA peripheral and the data SCR connections can sustain this throughput to the L2 or SL2 megamodule memories or to the external DDR2 memory.

UTOPIA throughput test results on the C6472/TCI6486 device operating at a core clock rate of 500 MHz are captured in Table 10. Measurements were collected for all four UTOPIA modes: MPHY 16-bit mode, SPHY 16-bit mode, MPHY 8-bit mode, and SPHY 8-bit mode. RX data was looped back from the TX data in an FPGA that contained a UTOPIA master module. Data was collected while using either the L2 or SL2 megamodule memories or the external DDR2 memory operating at a data rate of 533 MHz.

Table 10. UTOPIA Throughput

SRC-DEST	PHY Mode	Cell Size (Including Header) (Bytes)	Theoretical (Mbps)	Full-Duplex Throughput (Mbps)	Utilization %
L2-L2	MPHY-16	62	800	799.84	99.98
	MPHY-8	62	400	399.82	99.96
	SPHY-16	62	800	799.73	99.97
	SPHY-8	62	400	399.92	99.98
SL2-SL2	MPHY-16	62	800	800.06	100
	MPHY-8	62	400	400.06	100
	SPHY-16	62	800	799.49	99.94
	SPHY-8	62	400	399.89	99.97
DDR2-DDR2	MPHY-16	62	800	799.68	99.96
	MPHY-8	62	400	392.59	98.15
	SPHY-16	62	800	757.52	94.69
	SPHY-8	62	400	391.92	97.98

7.4.2 Further Considerations

UTOPIA data transport across the data SCR may conflict with other transactions. This must be handled by proper software configuration. The priorities for the data SCR transactions from all master ports can be configured so that lower priority transfers do not stall higher priority transfers. Sustained throughput cannot be guaranteed under all conditions. Special care must be taken to prevent stalls if UTOPIA throughput is a critical system requirement. However, short-term stalls do not result in lost data as the UTOPIA interface supports flow-control on a cell-by-cell basis and cell transfers can be paused until the RX transaction queue has space or the TX transmission buffer has data to send.

7.5 HPI Throughput

The HPI is a 16-bit external interface that supports data transfers with data strobe rates up to 20.833 MHz or 41.666 MBps for a C6472/TCI6486 device with a 500-MHz CPU clock. For data blocks of at least 1024 bytes, the overhead associated with reads and writes for all methods of access is less than 5% and may be considered negligible. The results shown below represent an upper limit on throughput for each specific block size. For smaller data block sizes the overhead increases significantly with decreasing block size. Writes have smaller overhead due to the ability to post all writes. Reads incur the additional overhead of the internal read latency for the first read.

7.5.1 Throughput Results

Data transfers using HPI generally follow one of two methods. The first is a memory fill (consecutive writes) or a memory dump (consecutive reads). All that is required, in this case, is a known address, which is written into the HPI address register, and following that a series of writes or reads to fill a block of memory or dump a block of memory. The fill is typical of initial memory image loads while the dump is typical for debug following an application fail. The second method is a messaging protocol. Messages may be written and read. In both cases, accesses to manage the message queue must be completed in addition to the message write or read. The messaging results shown in this report assume that two queue pointers must be read and one must be updated in addition to the actual message transfer.

The UHPI throughput is shown in Figure 6. Writes are always more efficient than reads for the same size transfer owing to fact that all writes can be posted, whereas the internal read latency for the first read is always incurred. Block writes and reads are always more efficient than message writes and reads for the same size data transfer owing to the additional overhead needed for the queue pointers in the message write and read cases. All transfers achieve a nearly ideal throughput for data sizes of 4096 bytes. There is a slight drop off for messages compared to data blocks for data sizes of 1024 bytes. Below that size, both the message and the data efficiency fall off fairly rapidly, but it is more pronounced for messages than for data blocks. Figure 7 shows the results with the maximum data size limited to 256 bytes and serves to

further illustrate the difference between messages and the simpler data block transfers. In particular, it can be seen that writes of as little as 4 bytes are about 50% efficient for simple data writes. For simple data reads, the data size must be greater than 16 bytes to reach 50% efficiency. For messages, 50% efficiency requires approximately 40-byte messages for writes and nearly 50-byte messages for reads. Table 11 and Table 12 have the data rates along with the utilization percentage for the various sized transfers.

Table 11. HPI Bandwidth and Utilization for Simple Block Accesses

Payload (Bytes)	Address Write With Data Read			Address Write With Data Write		
	Cycles (CPU)	Throughput (MBps)	Utilization %	Cycles (CPU)	Throughput (MBps)	Utilization %
4	268	7.46	17.91	96	20.83	50.00
8	316	12.66	30.38	144	27.78	66.67
16	412	19.42	46.60	240	33.33	80.00
64	988	32.39	77.73	816	39.22	94.12
256	3292	38.88	93.32	3120	41.03	98.46
1024	12508	40.93	98.24	12336	41.50	99.61
4096	49372	41.48	99.55	49200	41.63	99.90

Table 12. HPI Bandwidth and Utilization for Message Protocol Accesses

Payload (Bytes)	Message Read			Message Write		
	Cycles (CPU)	Throughput (MBps)	Utilization %	Cycles (CPU)	Throughput (MBps)	Utilization %
4	680	2.94	7.06	508	3.94	9.45
8	728	5.49	13.19	556	7.19	17.27
16	824	9.71	23.30	652	12.27	29.45
64	1400	22.86	54.86	1228	26.06	62.54
256	3704	34.56	82.94	3532	36.24	86.98
1024	12920	39.63	95.11	12748	40.16	96.39
4096	49784	41.14	98.73	49612	41.28	99.07

Figure 6. HPI Throughput for Data Blocks Up to 4096 Bytes

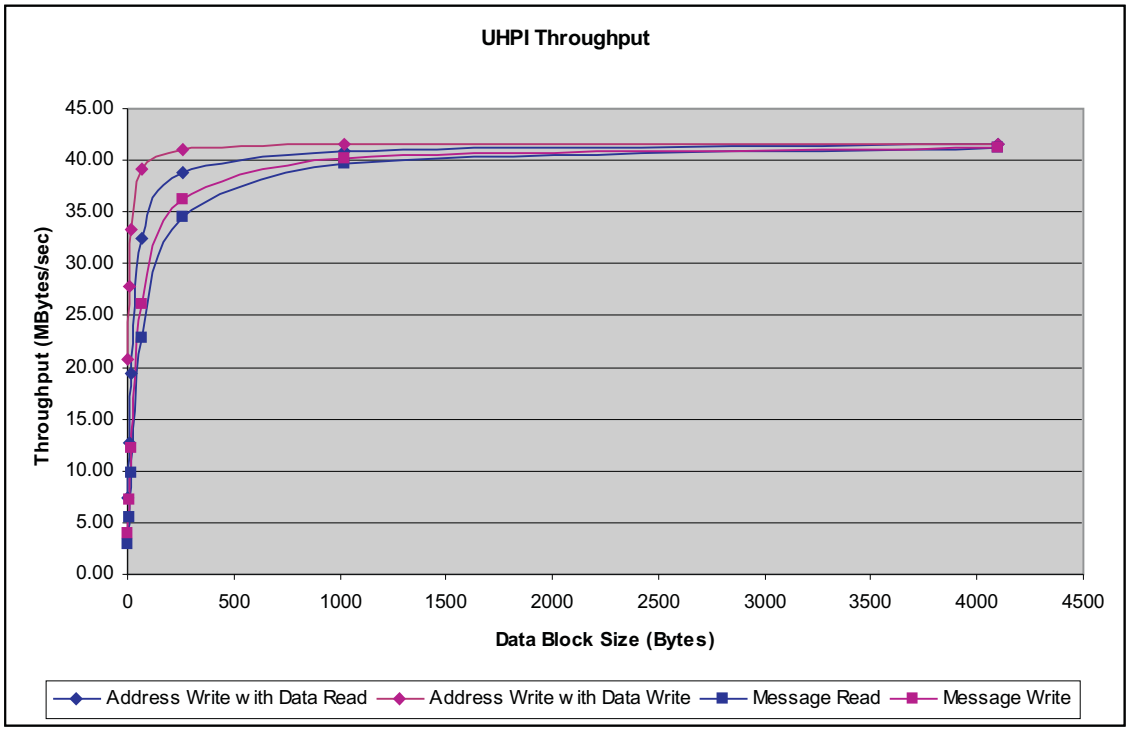
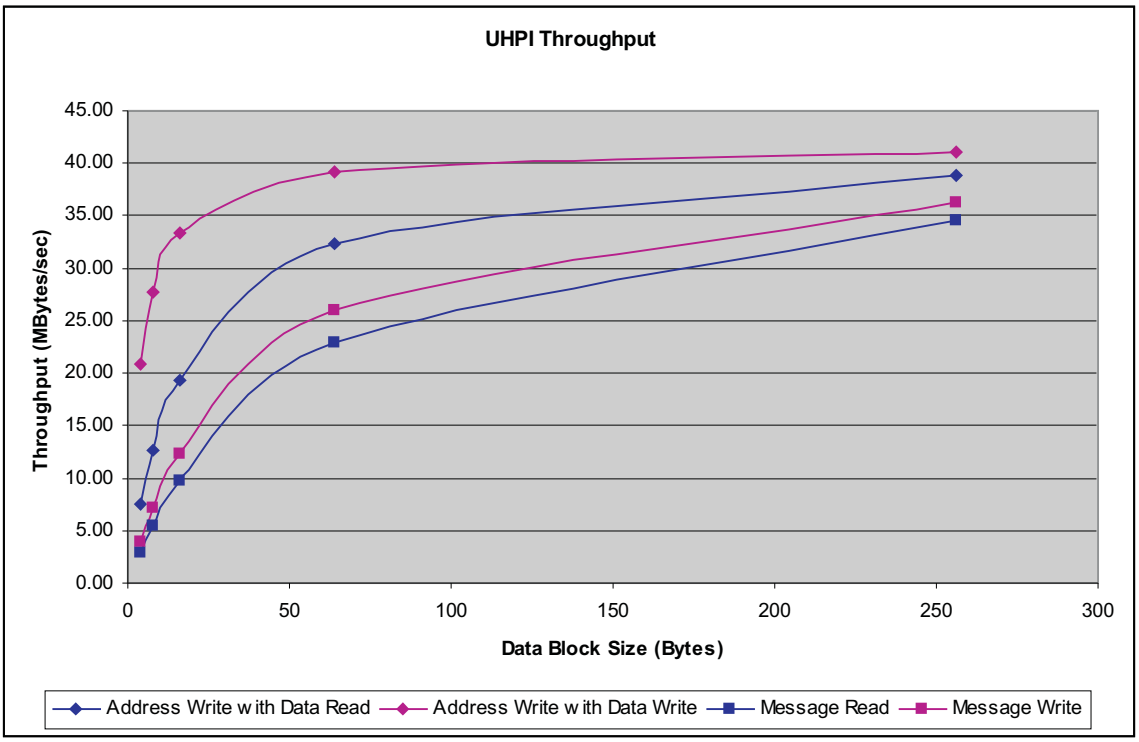


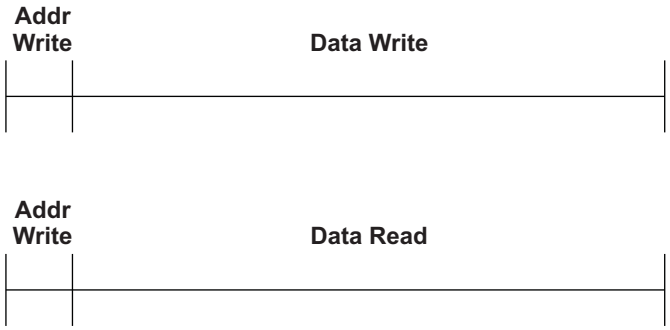
Figure 7. HPI Throughput for Data Blocks Up to 256



7.5.2 Block Write and Read Transfers

Block writes and reads involve setting an address in memory to be accessed and then performing a series of writes or reads. The address is the first (lowest addressed) element in the memory block to be accessed. Figure 8 shows the timelines for block write and read transfers. For stand-alone usage the block write is frequently used to initialize a memory region. As one example, this might be done as part of a boot procedure. Similarly, the stand-alone usage of block reads is frequently to dump a memory region. As one example, this might be done in cases where the CPU is no longer communicating with the host and the host wants to snapshot the memory as a way to examine possible causes.

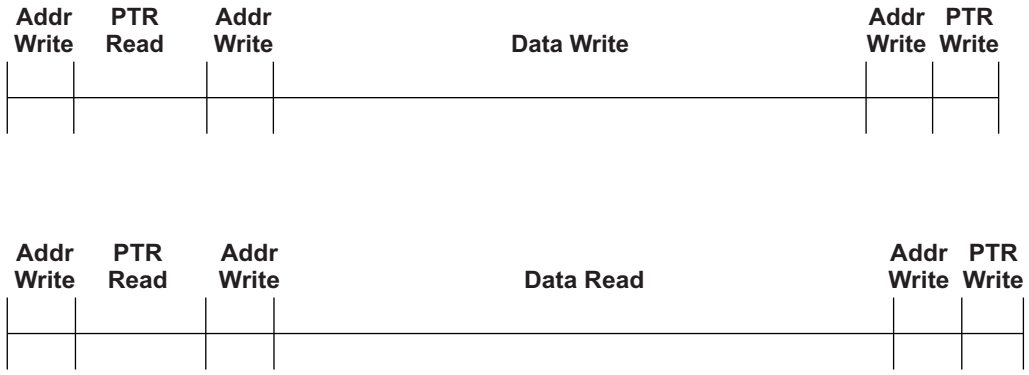
Figure 8. Block Write and Read Timelines



7.5.3 Message Write and Read Transfers

Message writes and reads may follow a variety of protocols. For this throughput analysis, a simple message protocol has been assumed. This message protocol consists of two sets of message buffers; one for messages from the host to the CPU and a second for messages from the CPU to the host. Messages from the host to the CPU use data writes while messages from the CPU to the host used data reads. Each consists of a single block of message buffers and a pair of pointers; a write pointer and a read pointer. For messages written from the host to the CPU, the host must read the pointers, decide whether there is space for the message that needs to be written, write the message if there is space, and then update the write pointer to indicate the presence of the new message. For messages read by the host from the CPU, the host must read the pointers, decide whether there is at least one message that can be read, read the message if it is present, and then update the read pointer to indicate that the message has been read. The CPU updates the read pointer for the message buffer used by the host to write messages to the CPU and updates the write pointer for the message buffer used by the host to read messages from the CPU. Figure 9 shows the timelines for message write and read transfers. In these timelines, the pointer read times include the decision time. The throughput results for messages in this report have idealized the decision time to 0.

Figure 9. Message Protocol Write and Read Timelines



7.5.4 Further Considerations

The HPI throughput results provided represent an upper limit on throughput for each specific block size. Consideration for each of the following items may be necessary depending on the system and application.

- If a message protocol is used for reads and writes then some additional time must be added to the sequence to account for the decision process. An additional 12-24 CPU cycles may be reasonable. The actual number is the number of host clock cycles required for a compare and response and the ratio of the host clock to the CPU clock.
- If the HSTROBE rate is less than the rate used in this report (CPU/24), then the throughput results need to be derated by the ratio of that frequency to CPU/24. This report uses a 500-MHz CPU clock frequency and an HSTROBE rate of 20.83 MHz.
- The HPI peripheral supports two configurations for using the address registers. One configuration uses separate address registers for reads and writes while the other uses a common address register for both reads and writes. The operation sequences in this report assume that a common address register is being used. In some systems and with some access sequences it may be beneficial to configure HPI to use separate address registers for reads and writes. However, in most cases a common address register is more efficient because making use of separate address registers generally requires a write to the HPIC to be performed before the write to initialize the appropriate address register each time it must be written. Using a common address register eliminates the extra step of writing to the HPIC.
- The HPI peripheral has a read FIFO and a write FIFO. Each FIFO has eight 32-bit locations. For sequences of reads or writes, the HPI generates internal read or write requests for four 32-bit words at a time. For host writes to the HPI that are less than four 32-bit words, the HPI delays making the write request to memory until its state machine reaches a transaction timeout or the host begins a new write sequence by modifying the address register. For the case where the host writes a new value to the address register, the write to the address register may be stalled. The stall corresponds to the time that it takes for the HPI to generate the internal write request and empty the write FIFO. The duration of this stall is less than the read latency time for the first data read of a sequence.
- Internal read and write requests may be delayed due to access conflicts at the memory or points in between. These delays are determined by regular arbitration and generally have a small impact or no impact because the FIFO elements within the HPI allow the arbitration delays to remain hidden. The measurements taken for this report were made in a device that did not include requests by other masters and, therefore, the results do not include any arbitration delay. The priority of the HPI requests can be changed to a higher priority to minimize the impact of arbitration delays if this is critical in a given system.

8 CPU Memory Access Performance

8.1 Introduction

As stated previously, each of the six C64x+ megamodules contains:

- 32KB L1D memory that runs at the CPU frequency and can be used as normal data memory or cache.
- 32KB L1P memory that runs at the CPU frequency and can be used as normal program memory or cache.
- 608KB LL2 memory that runs at the CPU frequency divided by 2 (CPU/2) and can be used as normal RAM or cache for both data and program.

Additionally, all of the megamodules share 768KB SL2 memory that runs at CPU/2 and can be used as data or program memory. A 32-bit 533-MHz DDR2 SDRAM interface is provided on the DSP to support up to 512MB of external memory that can be used as data or program memory. The DDR2 interface can also be configured to only use a 16-bit data bus.

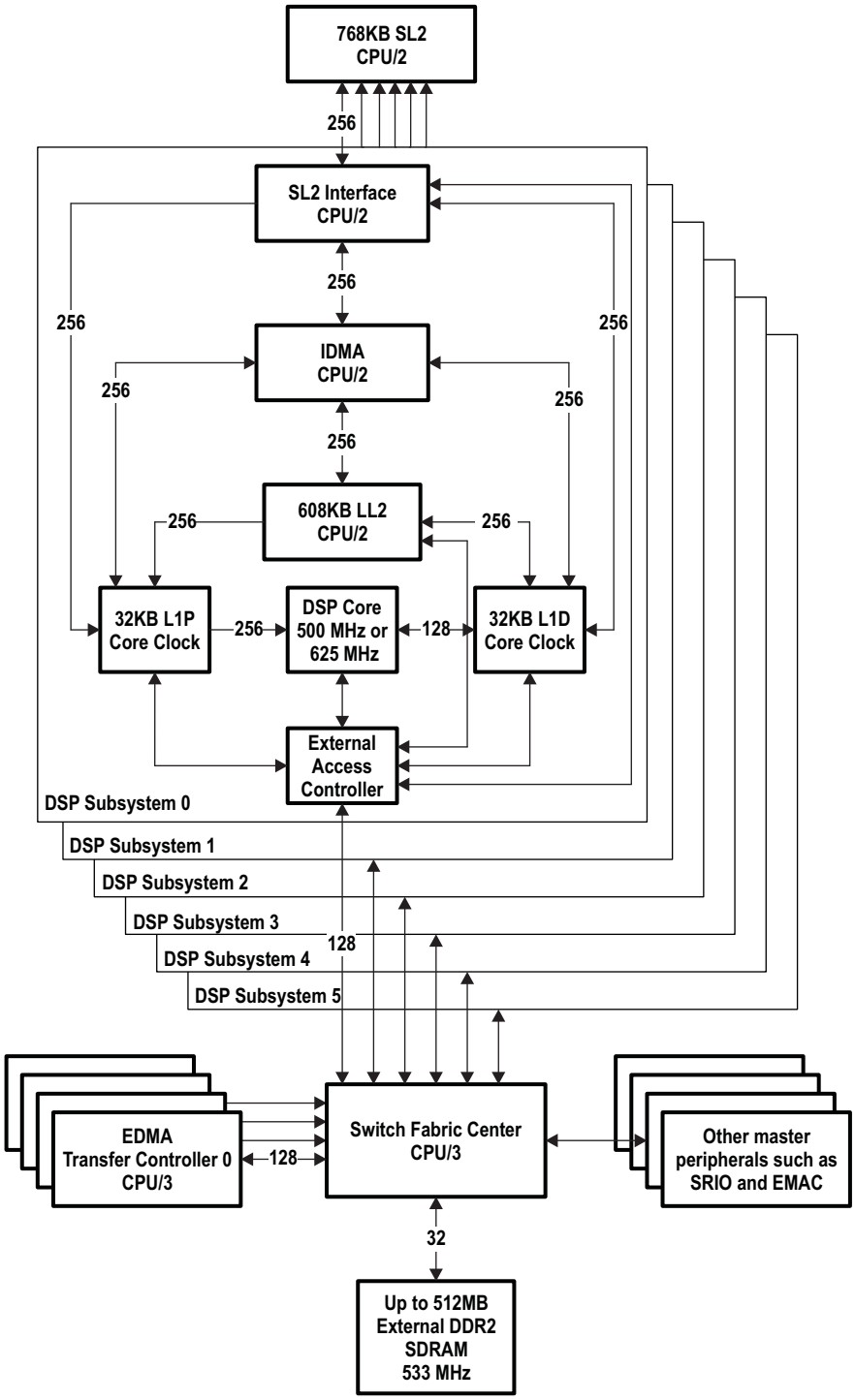
Memory access performance is very critical for software running on the DSP core. Each C64x+ core has the capability of sustaining up to 128 bits of load or store operations per cycle to the level-one data memory (L1D) which equates to 8 GBps at a 500-MHz core clock frequency and 10 GBps at a 625-MHz core clock frequency. When accessing data in the level-two memories or the external memory, the access rate depends on the memory access pattern and cache configuration.

Within the C64x+ megamodule, there is an Internal DMA (IDMA) engine that can move data at a rate of the CPU/2 with a data width of 256 bits. It is capable of transferring up to 8 GBps at a 500-MHz core clock frequency and 10 GBps at a 625-MHz core clock frequency. It operates in the background of the DSP core activity (i.e., data can be brought in to buffer A while the DSP core is accessing buffer B). The IDMA can only transfer data between the L1 and L2 memories and peripheral configuration ports; it cannot access external memory.

The EDMA is used to transfer data between the megamodule memories and the external memory through the DMA Switch Fabric or SCR. Each of the four EDMA TCs can transfer data simultaneously in the background of the DSP core activity. With a working knowledge of this architecture and the way in which data transfers interact and are performed, it is possible to create an efficient system and maximize the bandwidth utilization of the DMA switch fabric.

Figure 10 shows the TMS320C6472/TMS320TCI6486 memory system from the perspective of the CPU cores.

Figure 10. TMS320C6472/TMS320TCI6486 Memory System



This section gives designers a basis for estimating memory access performance and provides measured performance data achieved under various operating conditions. Most of the tests operate under best-case situations to estimate maximum throughput that can be obtained under the conditions described. The transfers described serve as a sample of performance conditions relevant to the software architect.

Multiple factors affecting memory access performance are discussed in this section such as access stride, index, and conflict. This section should be helpful for analyzing the following common questions:

1. Should I use DSP core or DMA for data copy?
2. How many cycles are consumed for my function with many memory accesses?
3. How much degradation is caused by multiple masters sharing memory?

Most of the performance data was captured on the C6472/TCI6486 EVM with 32-bit 533-MHz DDR2 memory.

8.2 DSP Core or EDMA and IDMA For Memory Copy

The bandwidth of memory copy is limited by the worst of the following three factors:

1. Bus Bandwidth
2. Source Throughput
3. Destination Throughput

Table 13 and Table 14 summarize the maximum theoretical bandwidth of the C64x+ CPU core, IDMA, and EDMA on the C6472/TCI6486 device.

Table 13. Theoretical Bus Bandwidth at 500 MHz

Master	Maximum Bandwidth (MBps)	Comments
C64x+ core	8000	(128 bits) / (8 bits/byte) * 500M = 8000 MBps
IDMA	8000	(256 bits) / (8 bits/byte) * (500M / 2) = 8000 MBps
EDMA	2666	(128 bits) / (8 bits/byte) * (500M / 3) = 2666 MBps

Table 14. Theoretical Bus Bandwidth at 625 MHz

Master	Maximum Bandwidth (MBps)	Comments
C64x+ core	10000	(128 bits) / (8 bits/byte) * 625M = 10000 MBps
IDMA	10000	(256 bits) / (8 bits/byte) * (625M / 2) = 10000 MBps
EDMA	3333	(128 bits) / (8 bits/byte) * (625M / 3) = 3333 MBps

Table 15 and Table 16 summarize the maximum theoretical throughput of the different memories in the C6472/TCI6486 device and also the external DDR2 memory.

Table 15. Maximum Throughput of Memory Endpoints at 500 MHz

Memory	Maximum Bandwidth (MBps)	Comments
L1D	16000	(256 bits) / (8 bits/byte) * 500M = 16000 MBps
L1P	16000	(256 bits) / (8 bits/byte) * 500M = 16000 MBps
LL2	8000	(256 bits) / (8 bits/byte) * (500M / 2) = 8000 MBps
SL2	8000	(256 bits) / (8 bits/byte) * (500M / 2) = 8000 MBps
DDR2	2133	(32 bits) / (8 bits/byte) * 533M = 2133 MBps

Table 16. Maximum Throughput of Memory Endpoints at 625 MHz

Memory	Maximum Bandwidth (MBps)	Comments
L1D	20000	(256 bits) / (8 bits/byte) * 625M = 20000 MBps
L1P	20000	(256 bits) / (8 bits/byte) * 625M = 20000 MBps
LL2	10000	(256 bits) / (8 bits/byte) * (625M / 2) = 10000 MBps
SL2	10000	(256 bits) / (8 bits/byte) * (625M / 2) = 10000 MBps
DDR2	2133	(32 bits) / (8 bits/byte) * 533M = 2133 MBps

Table 17 shows the memory-to-memory transfer bandwidth using either the CPU core, the EDMA, or the IDMA. It was measured using a linear memory block copy for each scenario. The bandwidth is measured by taking total bytes transferred and dividing it by the time required. After presenting this performance overview, some of the interesting cases are explored further.

Table 17. DSP Core, EDMA, and IDMA Transfer Bandwidth Comparison

	500-MHz DSP			625-MHz DSP		
Bandwidth (MBps) for Src → Dst	**DSP Core**	**EDMA**	**IDMA**	**DSP core**	**EDMA**	**IDMA**
L1D → L1D (16KB L1D cache)	3813	1603	1888	4767	2076	2360
LL2 → L1D (16KB L1D cache)	1581	1652	3537	1977	2076	4421
L1D → LL2 (16KB L1D cache)	3904	1795	4586	4880	2275	5733
LL2 → L1P (16KB L1P cache)	N/A	1624	3537	N/A	2076	4421
LL2 → LL2 (32KB L1D cache)	1141	1769	1705	1427	2212	2132
LL2 → SL2 (32KB L1D cache, non-prefetchable)	904	1769	992	1052	2212	1240
LL2 → SL2 (32KB L1D cache, prefetchable)	904	1769	992	1052	2212	1240
SL2 → LL2 (32KB L1D cache, non-prefetchable)	999	1697	992	1249	2125	1240
SL2 → LL2 (32KB L1D cache, prefetchable)	1229	1843	1577	1537	2311	1971
SL2 → SL2 (32KB L1D cache, non-prefetchable)	666	1557	990	832	1950	1238
SL2 → SL2 (32KB L1D cache, prefetchable)	761	1697	1572	951	2125	1965
LL2 → LL2 of another core (non-cacheable)	222	1769	N/A	277	2212	N/A
LL2 → LL2 of another core (32KB L1D cache)	222	1769	N/A	277	2212	N/A
LL2 → LL2 of another core (32KB L1D, 256KB L2 cache)	272	1769	N/A	340	2212	N/A
LL2 of another core → LL2 (non-cacheable)	50	1769	N/A	62	2212	N/A
LL2 of another core → LL2 (32KB L1D cache)	333	1769	N/A	416	2212	N/A
LL2 of another core → LL2 (32KB L1D, 256KB L2 cache)	463	1769	N/A	579	2212	N/A
LL2 → 32-bit DDR2 (non-cacheable)	218	1732	N/A	271	2067	N/A
LL2 → 32-bit DDR2 (32KB L1D cache)	218	1732	N/A	271	2067	N/A
LL2 → 32-bit DDR2 (32KB L1D, 256KB L2 cache)	279	1732	N/A	319	2067	N/A
32-bit DDR2 → LL2 (non-cacheable)	61	1542	N/A	69	1771	N/A
32-bit DDR2 → LL2 (32KB L1D cache)	350	1542	N/A	405	1771	N/A
32-bit DDR2 → LL2 (32KB L1D, 256KB L2 cache)	477	1542	N/A	549	1771	N/A
32-bit DDR2 → 32-bit DDR2 (non-cacheable, src/dst in same bank)	43	674	N/A	49	706	N/A

Table 17. DSP Core, EDMA, and IDMA Transfer Bandwidth Comparison (continued)

	500-MHz DSP			625-MHz DSP		
Bandwidth (MBps) for Src → Dst	**DSP Core**	**EDMA**	**IDMA**	**DSP core**	**EDMA**	**IDMA**
32-bit DDR2 → 32-bit DDR2 (32KB L1D cache, src/dst in same bank)	122	674	N/A	142	706	N/A
32-bit DDR2 → 32-bit DDR2 (32KB L1D, 256KB L2 cache, same bank)	157	674	N/A	183	706	N/A
32-bit DDR2 → 32-bit DDR2 (non-cacheable, different bank)	46	840	N/A	53	808	N/A
32-bit DDR2 → 32-bit DDR2 (32KB L1D cache, different bank)	133	840	N/A	156	808	N/A
32-bit DDR2 → 32-bit DDR2 (32KB L1D, 256KB L2 cache, diff bank)	169	840	N/A	197	808	N/A
LL2 → 16-bit DDR2 (non-cacheable)	218	1029	N/A	271	1030	N/A
LL2 → 16-bit DDR2 (32KB L1D cache)	218	1029	N/A	271	1030	N/A
LL2 → 16-bit DDR2 (32KB L1D, 256KB L2 cache)	240	1029	N/A	277	1030	N/A
16-bit DDR2 → LL2 (non-cacheable)	58	1031	N/A	67	1028	N/A
16-bit DDR2 → LL2 (32KB L1D cache)	299	1031	N/A	336	1028	N/A
16-bit DDR2 → LL2 (32KB L1D, 256KB L2 cache)	393	1031	N/A	436	1028	N/A
16-bit DDR2 → 16-bit DDR2 (non-cacheable, src/dst in same bank)	42	456	N/A	48	452	N/A
16-bit DDR2 → 16-bit DDR2 (32KB L1D cache, src/dst in same bank)	116	456	N/A	133	452	N/A
16-bit DDR2 → 16-bit DDR2 (32KB L1D, 256KB L2 cache, same bank)	137	456	N/A	155	452	N/A
16-bit DDR2 → 16-bit DDR2 (non-cacheable, different bank)	45	442	N/A	52	397	N/A
16-bit DDR2 → 16-bit DDR2 (32KB L1D cache, different bank)	126	442	N/A	147	397	N/A
16-bit DDR2 → 16-bit DDR2 (32KB L1D, 256KB L2 cache, diff bank)	146	442	N/A	166	397	N/A

Generally speaking:

- DSP cores access internal memory most efficiently, but using the DSP core to access external data is not the best use of resources and it should be avoided.
- The IDMA is good at moving linear blocks of data between the memories internal to the C6472/TCI6486 device (L1D, L1P, LL2, and SL2), but it cannot access external memory. It is also useful for writing masked blocks to peripheral configuration space, but that is beyond the scope of this document.
- The EDMA is best for transferring data to and from external memory and should be used for this task.
- The cache configurations and contents dramatically affect the DSP core performance, but they do not affect EDMA and IDMA performance. Note that all test data for the DSP core in this application report are based on a *cold cache* (all of the caches are flushed before the test begins).
- The SL2 prefetch buffer improves the performance of reads from the SL2.
- The DDR2 bank architecture affects the performance slightly for the DDR2 → DDR2 case. If the source and destination are in the same bank, the DDR2 row switch happens frequently (every row switch introduces extra delay cycles). If the source and destination are in different banks, the frequency of row switching is reduced, thus improving the performance.
- The EDMA throughput on 32-bit DDR2 is about 2x the 16-bit DDR2 because EDMA throughput is much higher than the bandwidth of either DDR2 configuration. The bottleneck is at the DDR2 interface. However, the DSP core throughput on 16-bit DDR2 is almost same as the 32-bit DDR2 because the bottleneck is at the DSP core.

The above EDMA throughput data was measured using Transfer Controller 2 (TC2). TC3 has equivalent performance, while the throughput of TC0 and TC1 are not as good as TC2 due to their smaller buffers, as discussed in Section 6.

8.3 DSP Core Memory Access Performance

L1D memory runs at the same rate as the DSP core, so DSP core can access L1D memory without stalling once each clock cycle. For some applications which require use of a small data block very quickly, part of the L1D can be used as normal RAM to store this small data block for very fast processing.

Normally, L1D is used as data cache. If a cache hit occurs, the DSP core can access data in one clock cycle. However, whenever a cache miss occurs, the DSP core stalls until the data arrives in the cache.

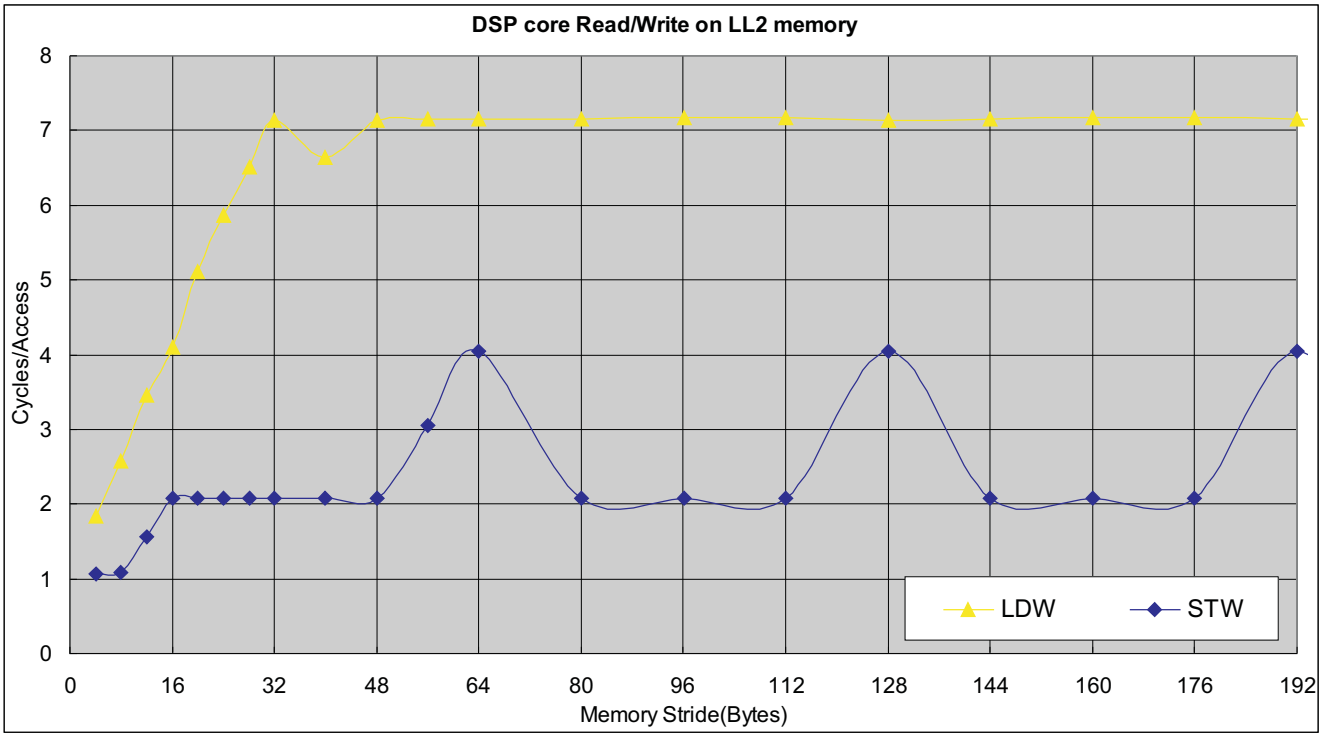
The following subsections examine the performance of DSP core accesses to L2 and external DDR2 memory through L1D cache. The pseudo code for these tests is similar to the following:

```
flushCache(); preCycle = getTimeStampCount(); for(I=0; i<accessCount; I++) { Access Memory at
address; address += stride; } cycles = getTimeStampCount()-preCycle; cycles/Access =
cycles/accessCount;
```

8.3.1 DSP Core LL2 Memory Access Performance

Figure 11 shows performance data for DSP core access to LL2. The time required for 1024 consecutive LoaD Word (LDW) or STore Word (STW) instructions was measured and the average time for each instruction is reported. The cycles for LDB/STB, and LDDW/STDW are same as LDW/STW.

Figure 11. LL2 Memory Access Performance



Since the L1D is a read-allocate cache, DSP core reads from LL2 should always go through L1D cache. So, DSP core accesses to LL2 highly depend on the cache. The address increment (or memory stride) affects cache utilization. Contiguous accesses utilize cache to the fullest. A memory stride of 64 bytes or more causes every access to miss the cache because the L1D cache line size is 64 bytes.

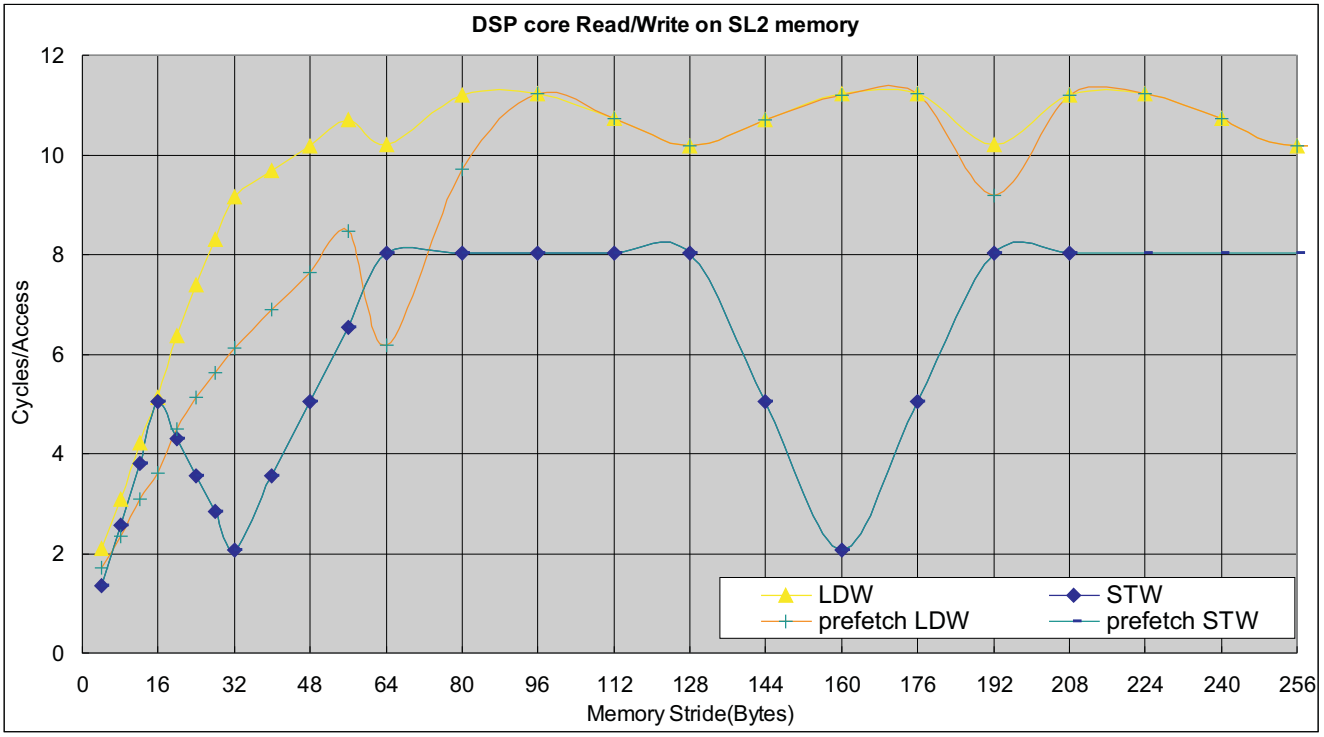
Since the L1D is not a write-allocate cache and the cache is flushed before the test, any write to the LL2 goes through the L1D write buffer (4x16 bytes). For write operations, if the stride is less than 16 bytes,

several writes may be merged into one write to the LL2 in the L1D write buffer, thus it achieves the efficiency close to 1 cycle/write. When the stride is a multiple of 64 bytes, every write always accesses the same bank of LL2 (because the LL2 is organized as 4x16 byte banks), which requires 4 cycles. For other strides, the consecutive writes access to different banks of LL2, they can be overlapped with pipeline, which only requires 2 cycles.

8.3.2 DSP Core SL2 Memory Access Performance

Figure 12 shows performance data for DSP core access to SL2. The time required for 1024 consecutive LDW or STW instructions was measured and the average time for each instruction is reported. The cycles for LDB/STB, and LDDW/STDW are the same as LDW/STW.

Figure 12. SL2 Memory Access Performance



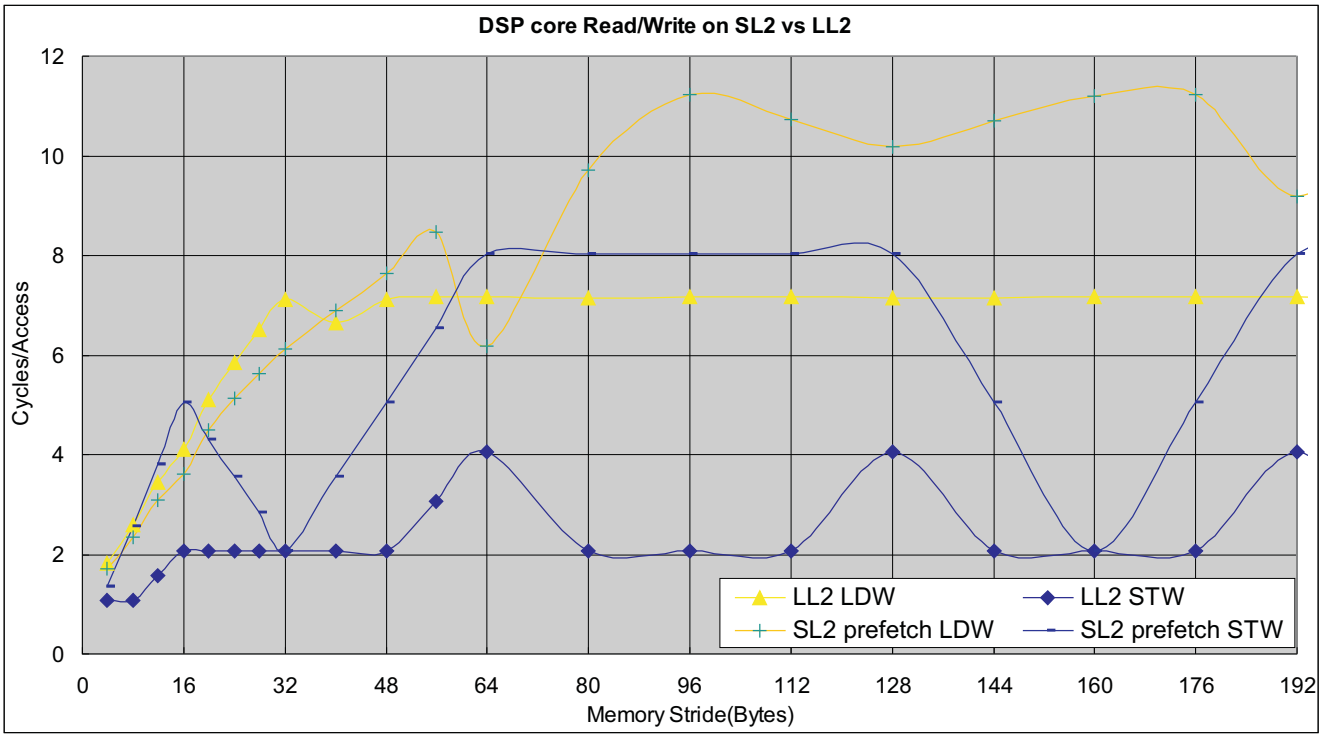
DSP core reads from SL2 should always go through L1D cache, so DSP core accesses to SL2 are highly dependent on the cache, as it is with LL2. There is an additional prefetch buffer (4x32bytes) inside SL2. The functionality is like cache and it is programmable by software. Enabling it benefits multicore read and write accesses. It also benefits single core read operations.

Since the L1D is not a write-allocate cache, any write goes through the L1D write buffer (4x16 bytes) to the SL2. For write operations, if the stride is less than 16 bytes, multiple writes may be merged into one write burst to the SL2 in the L1D write buffer, thus achieving better efficiency. When the stride is (4N+1)*32 bytes, every write always accesses a different bank of SL2 (because the SL2 is organized as 4x32-byte banks), it can be overlapped with pipeline, which requires only 2 cycles. For other strides, the consecutive writes may access the same bank of SL2, which may result in up to 8 cycles per access.

The curve for STW and prefetch STW is overlapped because the prefetch buffer does not help write operation.

Figure 13 shows a performance comparison of DSP core access of SL2 versus LL2.

Figure 13. SL2 Access Performance Compared to LL2



For memory strides less than 32 bytes, the performance of prefetchable SL2 reads is slightly better than reads from LL2. For reads with a larger memory stride and for write operations, the performance of LL2 is much better than SL2. So, SL2 is very suitable for contiguously accessed data read or write data or for read only data; otherwise, LL2 performance is best.

8.3.3 DSP Core DDR2 Memory Access Performance

DSP core accesses to external DDR2 memory are also highly dependent on the cache. When the DSP core accesses external memory space, an an external access controller transfer request (TR) may be generated (depending on whether the data is cached) to the DMA switch fabric. The TR is for one of the following:

- A single element - if the memory space is non-cacheable.
- An L1D cache line - if the memory space is cacheable and the L2 cache is disabled
- An L2 cache line - if the memory space is cacheable and L2 cache is enabled.

No transfer request is generated in the case of an L1 or L2 cache hit.

External memory can be cached by L1 cache, L2 cache, or neither. If the appropriate MAR bit for an external memory space is not set, it is not cacheable. If the MAR bit is set and the L2 cache size is zero (all L2 is defined as SRAM), the external memory space is cached by L1. If the MAR bit is set and the L2 cache size is greater than 0, then the external memory space is cached by L2 and L1.

The address increment (or memory stride) affects cache utilization. Contiguous accesses utilize cache memory to the fullest. A memory stride of 64 bytes or more causes every access to miss in the L1 cache because the L1 line size is 64 bytes. A memory stride of 128 bytes causes every access to miss in L2 because the L2 line size is 128 bytes.

If a cache miss happens, the DSP core stalls, waiting for the return data. The length of the stall is equal to the sum of the transfer latency, transfer duration, data return time, and some small cache request overhead.

Figure 14 and Figure 15 show data collected from a 500-MHz C6472/TCI6486 device with 32-bit DDR2 at 533 MHz. The time required for 1024 LDW instructions was measured and the average time for each instruction is reported. The cycles for LDB and LDDW are the same as LDW.

Figure 14. DDR2 Memory Read Access Performance - Large Strides

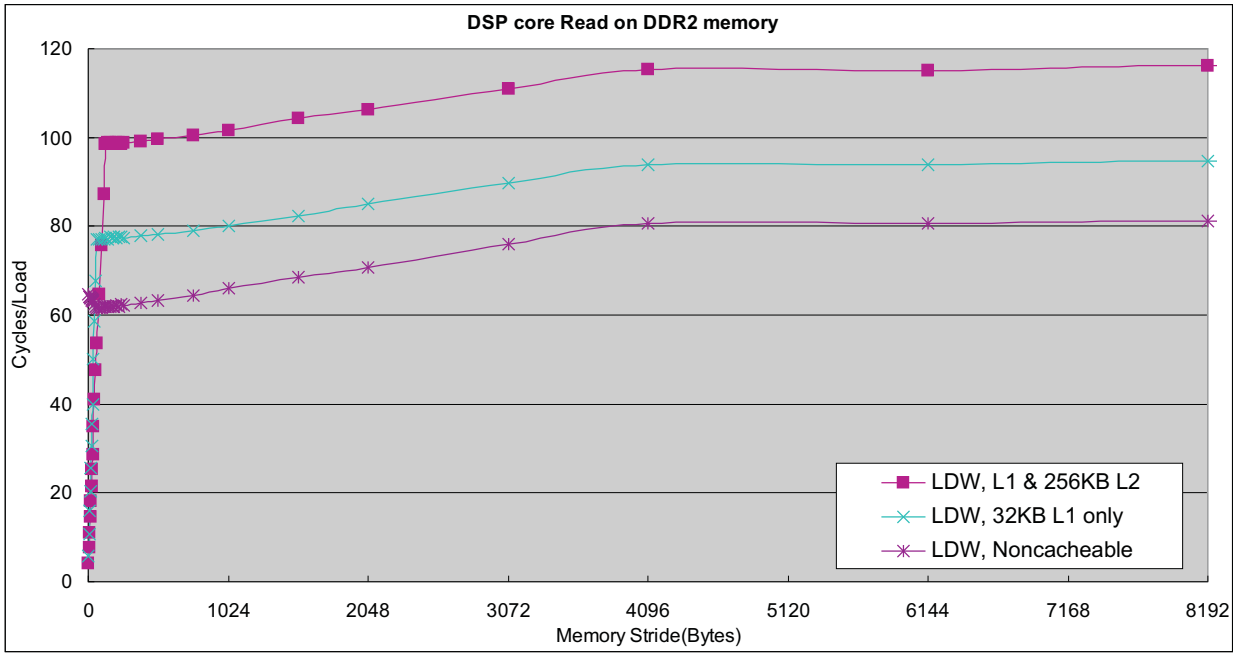
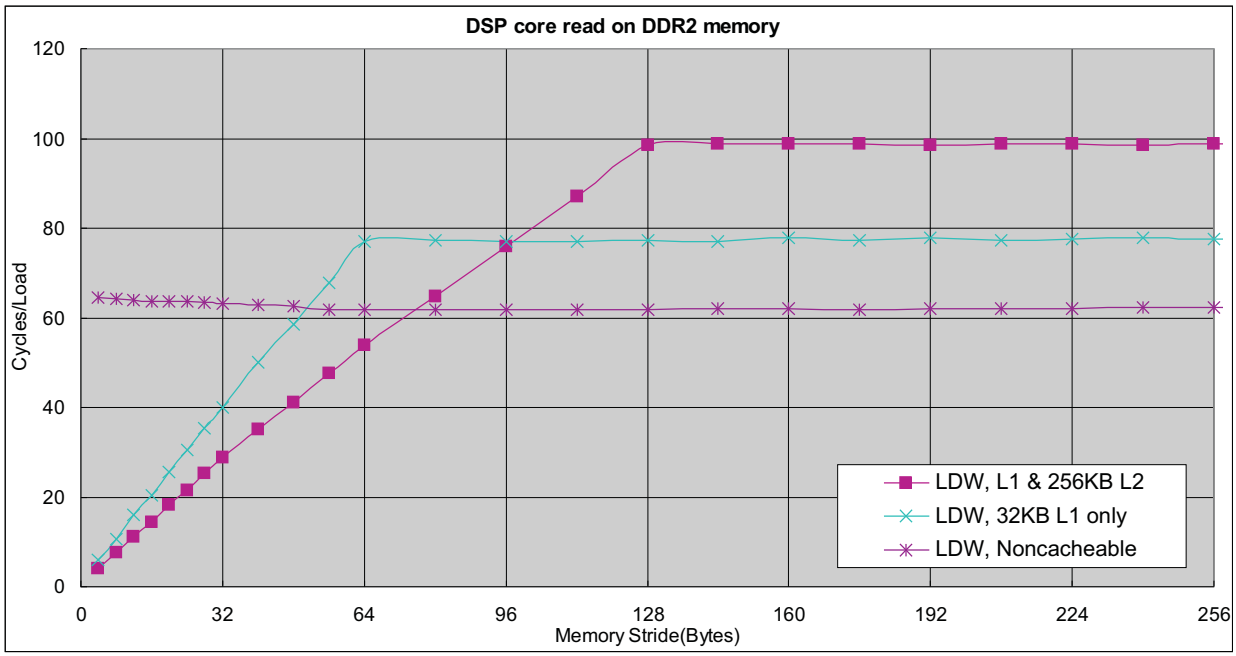


Figure 15. DDR2 Memory Read Access Performance - Small Strides



For memory strides less than 128 bytes, the performance is dominated by cache as with other memories discussed above. For memory strides larger than 128 bytes, the performance becomes worse because of DDR2 SDRAM row switching. The row size on the C6472/TCI6486 device during this test was 4096 bytes. So, for a stride larger than 4096, every read access is to a new row, which results in about 20 additional cycles. Note that the DDR2 SDRAM row size may be different for other DDR2 SDRAM memory configurations.

Figure 16 and Figure 17 show data collected from a 500-MHz DSP core writing to a 32-bit wide, 533-MHz external DDR2 memory bank. The time required for 1024 STW instructions was measured and the average time for each instruction is reported. The cycles for STB and STDW are the same as STW.

Figure 16. DDR2 Memory Write Access Performance - Large Strides

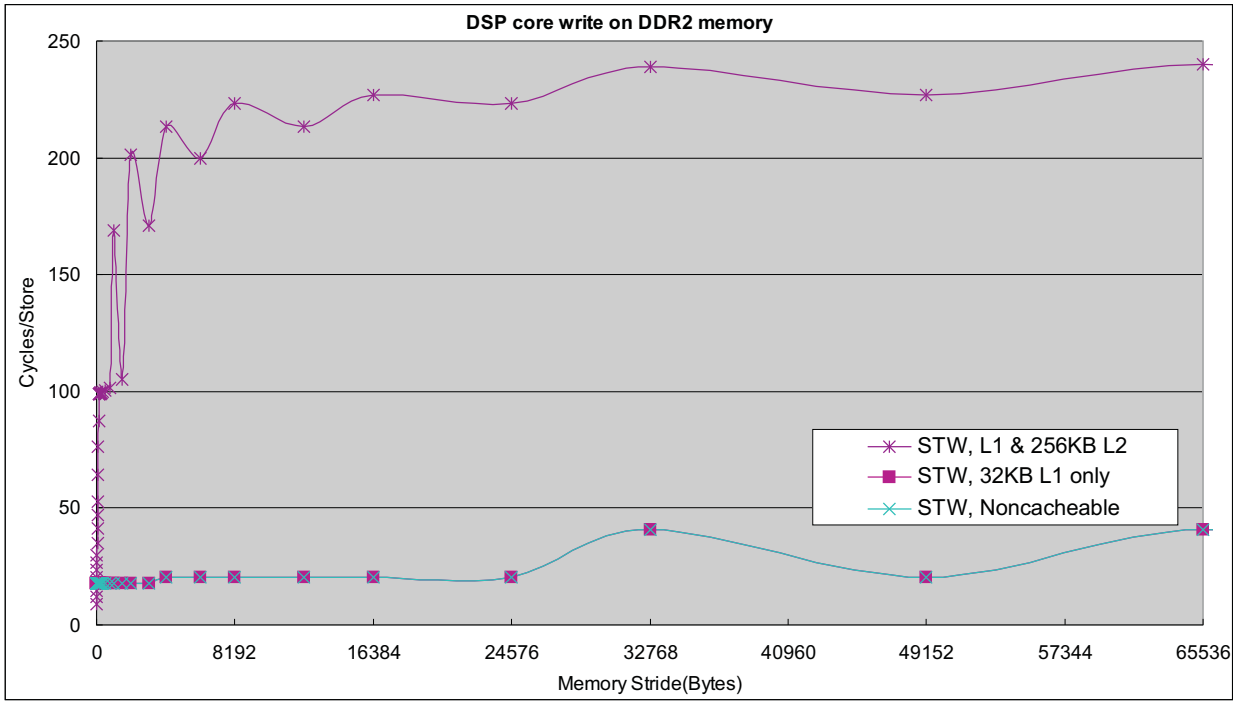
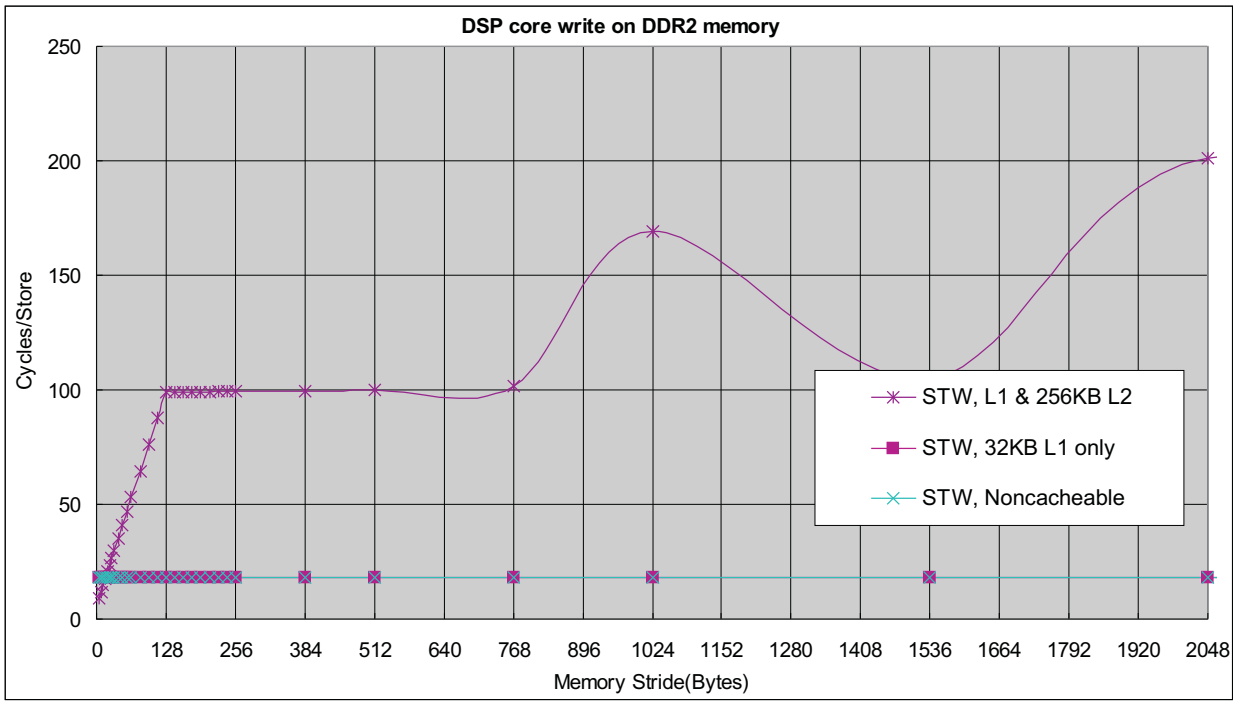


Figure 17. DDR2 Memory Write Access Performance - Small Strides



Since the L1D cache is not a write-allocate cache and the cache is flushed before this test, it has no effect on write operation. Therefore, the 32KB L1-only case and non-cacheable case are overlapped in Figure 17.

For non-cacheable access, cycles for write are about 1/4 of the cycles for read, because four write operations are combined into one transfer request in the L1D write buffer; while four read operations submit four transfer requests to DMA switch fabric.

Similar to cacheable read operations, the performance of cacheable write operations with memory strides less than 128 bytes is dominated by cache, as discussed above. For memory strides larger than 4096 bytes, the performance becomes worse because of DDR2 SDRAM row switching.

L2 cache is a write-allocate cache, for any write operation, it always reads 128 bytes containing the accessed data into a cache line and then it modifies the data in the L2 cache. This data is later written back to real external memory when a cache conflict occurs or if a cache writeback is invoked by the software. When the memory stride is equal to or larger than 1024 bytes, the cycle count for write operations increases to about 2x that of the read operations because cache conflicts happen frequently. For very large memory strides, every write operation results in a cache line write back (for conflict) and a cache line read (for write-allocate).

9 EDMA Access Performance

The EDMA architecture has many features designed to facilitate multiple high-speed data transfers simultaneously; this was introduced previously in Section 6. Its performance is affected by the memory type and other factors discussed in the following sections.

9.1 DMA Transfer Overhead

Initial latency is defined as the time between DMA event initiation to real data transfer start. Since initial latency is hard to measure, we measured transfer overhead instead. It is defined as the sum of the latency and the time to transfer the smallest element. The values vary based on the type of source and destination memory and whether another master is accessing either memory. Table 18 and Table 19 show the average cycles measured on a 500-MHz C6472/TCI6486 device for the smallest transfer (1 word) between different endpoints.

Table 18. EDMA Transfer Overhead (Cycles)

	Destination		
Source	**L1D**	**LL2**	**DDR2**
L1D	170	170	170
LL2	193	193	170
DDR2	193	193	204

Table 19. IDMA Transfer Overhead (Cycles)

	Destination	
Source	**L1D**	**LL2**
L1D	85	85
LL2	86	88

Transfer overhead is a big concern for short transfers and it needs to be included when scheduling DMA traffic in a system. Single-element transfer performance is latency-dominated. So, for small transfers, the overhead incurred by the DMA versus the DSP core must be considered.

9.2 EDMA Performance of the Four Transfer Controllers

The EDMA transfer engine on the C6472/TCI6486 device includes four Transfer Controllers (TCs). The four TCs are not exactly same. Table 20 contains a summary of the key attributes.

Table 20. Transfer Controller Attributes

Name	TC0	TC1	TC2	TC3
FIFO Size	128 bytes	128 bytes	256 bytes	256 bytes
Bus Width	128 bits	128 bits	128 bits	128 bits
Destination FIFO Entries	2 entries	4 entries	4 entries	4 entries
Default Burst Size	64 bytes	64 bytes	64 bytes	64 bytes

For more information about TC differences, see the *TMS320C6472/TMS320TCI648x DSP Enhanced DMA (EDMA3) Controller User's Guide* (SPRU727).

Table 21 and Table 22 compare the maximum throughput measured for different TCs on the C6472/TCI6486 device with a 32-bit 533-MHz DDR2 interface.

Table 21. Throughput Comparison Between TCs on a 500-MHz C6472/TCI6486 Device

Bandwidth (MBps)	TC0	TC1	TC2	TC3
L2 → L1D	688	688	1652	1652
L2 → L2	709	709	1769	1769
L2 → DDR2	710	710	1738	1738
DDR2 → L2	785	785	1560	1560
DDR2 ↔ DDR2	402	402	674	674

Table 22. Throughput Comparison Between TCs on a 625-MHz C6472/TCI6486 Device

Bandwidth (MBps)	TC0	TC1	TC2	TC3
L2 → L1D	868	871	2076	2077
L2 → L2	888	888	2212	2212
L2 → DDR2	886	885	2067	2067
DDR2 → L2	892	895	1767	1764
DDR2 ↔ DDR2	425	425	706	706

In conclusion, TC2 and TC3 achieve about two times the bandwidth than that of TC0 and TC1 due to their larger FIFO buffer size. TC1 also performs slightly better than TC0 for small transfers because it can queue additional transfer requests. Without special note, all performance data in this application report is measured on TC2 or TC3.

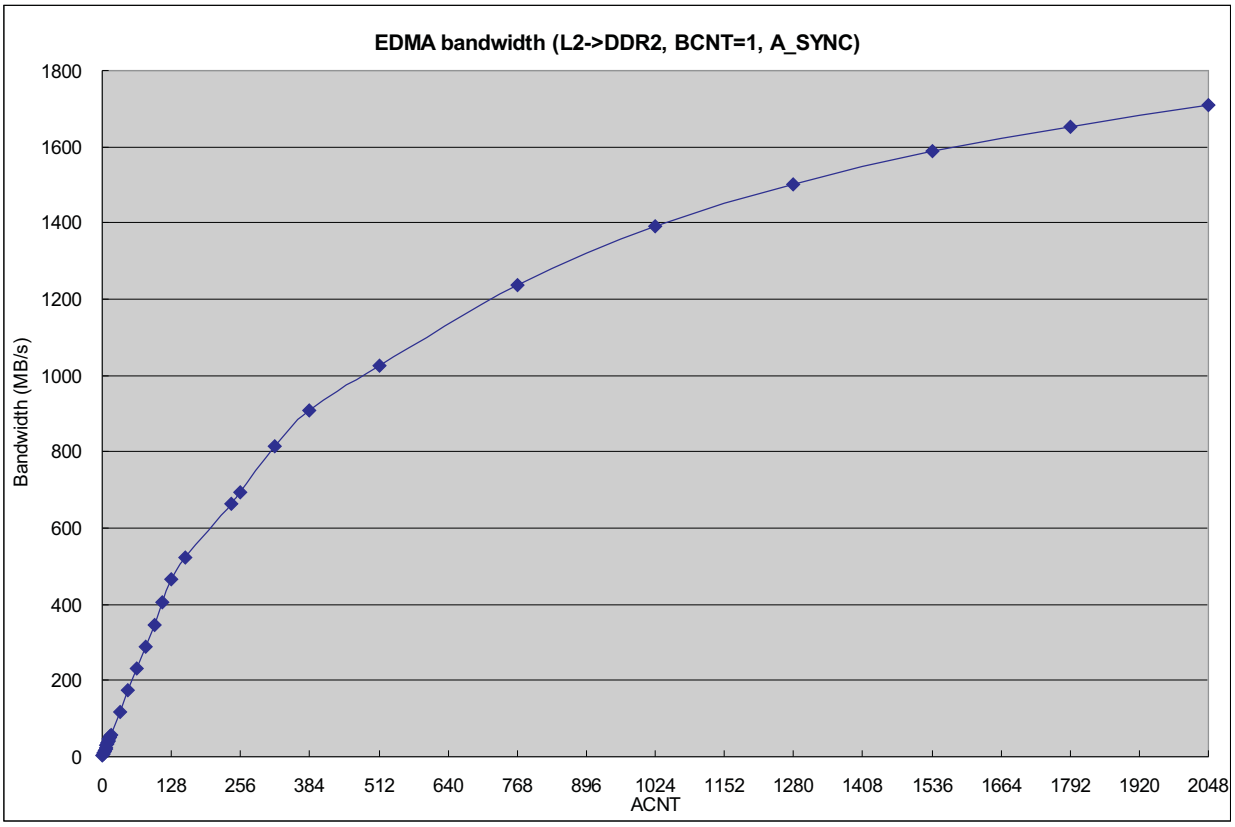
9.3 EDMA Bandwidth versus Transfer Flexibility

EDMA channel parameters allow many different transfer configurations. Most typical transfers progress properly and memory bandwidth is fully utilized. However, in some less common configurations, transfers are unable to burst efficiently, thus reducing performance. To properly design a system, it is important to know which configurations offer the best performance for high-speed operations and which must trade throughput for flexibility.

9.3.1 First-Dimension Size (ACNT) Considerations - Burst Width

To make full utilization of bandwidth in the transfer engine, it is important to fully utilize the bus width available and allow for data bursting. ACNT size should be a multiple of 16 bytes to fully utilize the 128-bit bus width. ACNT should also be a multiple of 64 bytes to fully utilize the 64-byte default burst width. Lastly, ACNT should be a multiple of 256 bytes to fully utilize the 256-byte FIFO. Since these ideal conditions are not always met, performance is less than optimal. Figure 18 shows performance data from a 625-MHz C6472/TCI6486 device with 32-bit 533-MHz DDR2, transferring data blocks of different sizes from L2 to DDR2 using an EDMA channel.

Figure 18. Effect of ACNT on EDMA Bandwidth



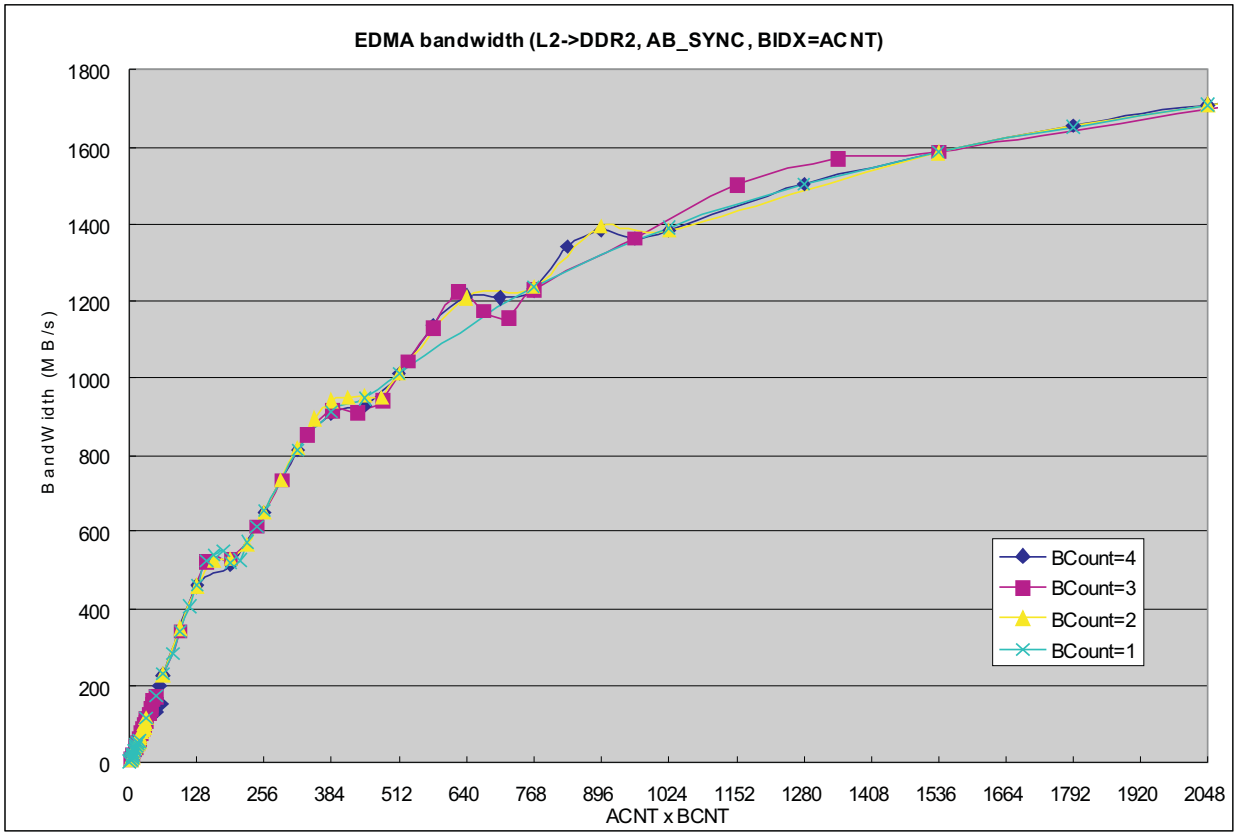
In conclusion, as ACNT is increased, the transfer bandwidth also increases.

9.3.2 Two-Dimensional Considerations, Transfer Optimization

If two-dimensional (2D) transfer (AB_Sync) is linear (BIDX=ACNT), the 2D transfer is optimized as a one-dimensional (1D) transfer. Linear indicates that the two-dimensional data has continuous addresses. The two dimensions in a transfer consists of BCNT arrays of ACNT bytes. If BIDX=ACNT, then these arrays are together and look like a one-dimensional block. For more details, see the *TMS320C6472/TMS320TCI648x DSP Enhanced DMA (EDMA3) Controller User's Guide* (SPRU727).

Various ACNT and BCNT combinations were investigated. However, the overall transfer size (ACNT × BCNT) was proven to have the most bearing on bandwidth. Figure 19 contains linear 2D transfer test results. It shows, no matter what value of BCNT, the resulting bandwidth is the same.

Figure 19. Linear 2D Transfers



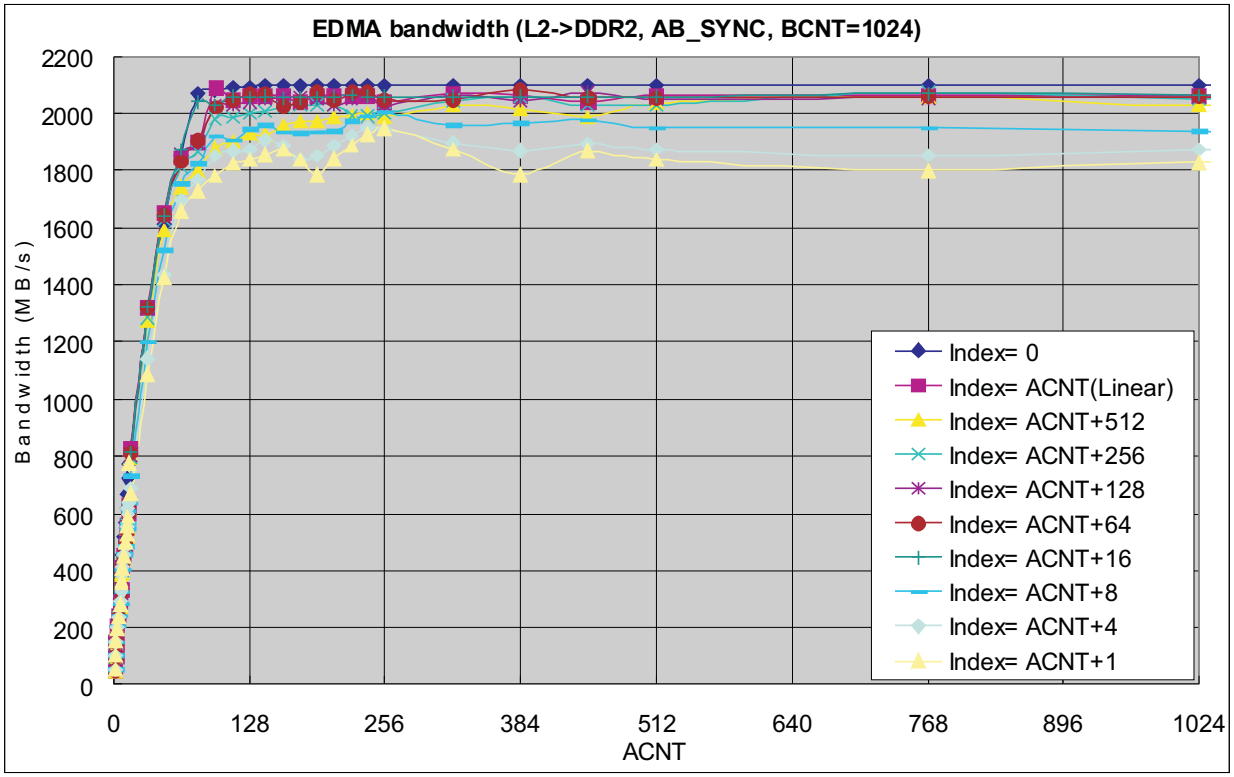
If 2D transfer is not linear, the bandwidth utilization is determined by the ACNT, as shown previously in Figure 18.

9.3.3 Index Consideration

Index dramatically affects the EDMA throughput. Linear transfers (Index=ACNT) fully utilize bandwidth. Fixed Index transfers (Index=0) can also fully utilize the bandwidth, like the linear transfers. Other Index modes lower the EDMA performance. Odd Index values have the worst performance. If the Index is a power of 2 and it is larger than 8, the performance degradation is very small.

Figure 20 shows the effect of Index on EDMA throughput. This was captured while transferring 1024 rows (BCNT=1024) of 2D data from L2 to DDR2, with different Index values.

Figure 20. Effect of Index on EDMA Transfers



Unless otherwise noted, all other performance data in this application report is measured with Index=0 or Index=ACNT.

9.3.4 Address Alignment

Address alignment may slightly impact the performance. The default burst size of EDMA is 64 bytes. Therefore, the EDMA TC breaks the ACNT array at the 64-byte boundaries resulting in additional transfers for non-aligned block moves. This results in an extra burst being generated to handle the unaligned head and tail data. For big transfers this overhead may be ignored. All resulting data presented in this document was collected using address aligned transfers.

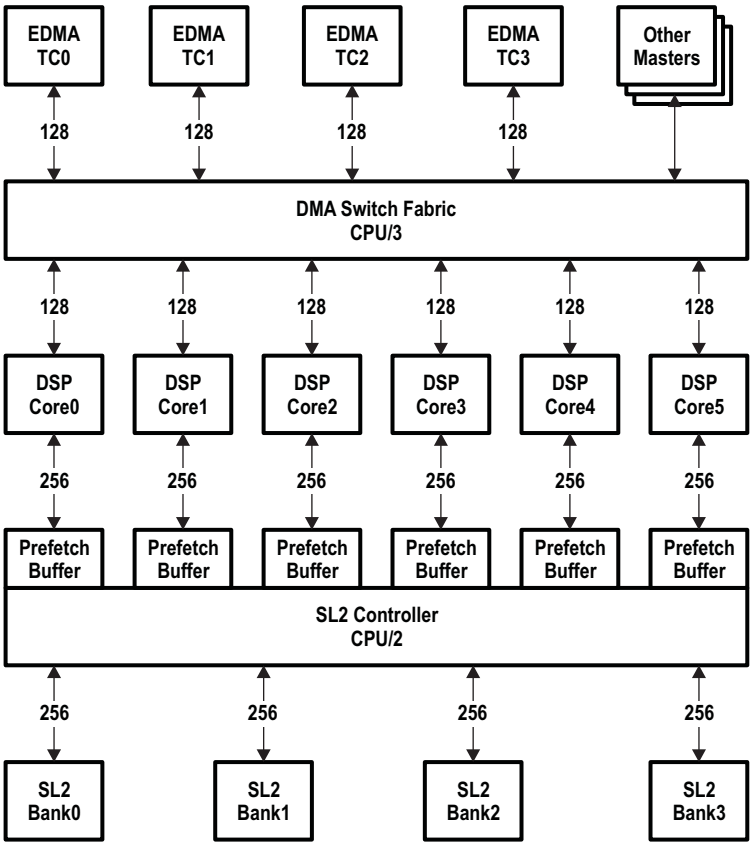
10 Performance of Multiple Masters Sharing Memory

Since the C6472/TCI6486 device includes six CPU cores and multiple other DMA masters; these masters sometimes attempt to access the same memory endpoints simultaneously. This section discusses the performance of multiple masters accessing the same memory.

10.1 Performance of Multiple Masters Sharing SL2

Figure 21 shows the SL2 architecture and access paths.

Figure 21. SL2 Memory Architecture and Access Paths



The memory in the SL2 is broken into four banks. The SL2 memory is interleaved among these banks at 32-byte (256-bit) boundaries as shown in Table 23.

Table 23. Data Organization in SL2 Banks

Bank 0			Bank 1			Bank 2			Bank 3		
byte 0	…	byte 31	byte 32	…	byte 63	byte 64	…	byte 95	byte 96	…	byte 127
byte 128	…	byte 159	byte 160	…	byte 191	byte 192	…	byte 223	byte 224	…	byte 255
byte 256	…	byte 287	byte 288	…	byte 319	byte 320	…	byte 351	byte 352	…	byte 383
…	…	…	…	…	…	…	…	…	…	…	…

All six cores and other system masters access the four SL2 banks through the SL2 controller, which is a cross-bar switch. Multiple masters can access different banks in parallel. However, if multiple masters access the same bank, the accesses are arbitrated with a round-robin mechanism. The priority of the masters does not affect the arbitration.

The SL2 controller provides a prefetch buffer for every core to improve read performance and reduce the potential competition of multiple masters on the same bank.

DMA masters have no direct access path to the SL2. They must access the SL2 through one of the core's access paths via the DMA switch fabric.

Table 24 through Table 27 show the performance data measured when multiple masters access SL2 simultaneously. Each master accesses a different data buffer on the SL2 simultaneously. L1D cache size is set to 32KB for each core.

Table 24. Performance of Multiple DSP Cores Sharing SL2 at 500 MHz

LL2 → SL2, Prefetchable							LL2 → SL2, Non-Prefetchable						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	902	902	902	902	899	899	Core 0	902	902	902	902	899	899
Core 1		895	895	895	893	892	Core 1		895	895	895	893	892
Core 2			904	903	901	901	Core 2			904	903	901	901
Core 3				895	893	893	Core 3				895	893	893
Core 4					902	901	Core 4					902	901
Core 5						893	Core 5						893
Total Bandwidth	902	1797	2701	3595	4488	5379	Total Bandwidth	902	1797	2701	3595	4488	5379
SL2 → LL2, Prefetchable							**SL2 → LL2, Non-Prefetchable**						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	1111	1111	1111	1111	1111	1111	Core 0	912	912	912	912	912	911
Core 1		1103	1103	1103	1103	1103	Core 1		912	912	912	912	911
Core 2			1111	1111	1111	1111	Core 2			921	921	921	919
Core 3				1103	1103	1103	Core 3				912	913	911
Core 4					1111	1111	Core 4					921	920
Core 5						1103	Core 5						912
Total Bandwidth	1111	2214	3325	4428	5539	6642	Total Bandwidth	912	1824	2745	3657	4579	5484

Table 25. Performance of Multiple DSP Cores Sharing SL2 at 625 MHz

LL2 → SL2, Prefetchable							LL2 → SL2, Non-Prefetchable						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	1128	1127	1127	1127	1123	1124	Core 0	1128	1127	1127	1127	1123	1124
Core 1		1119	1119	1119	1116	1115	Core 1		1119	1119	1119	1116	1116
Core 2			1130	1129	1127	1126	Core 2			1130	1129	1127	1126
Core 3				1119	1117	1116	Core 3				1119	1117	1116
Core 4					1127	1127	Core 4					1127	1127
Core 5						1116	Core 5						1116
Total Bandwidth	1128	2246	3376	4494	5610	6724	Total Bandwidth	1128	2246	3376	4494	5610	6725
SL2 → LL2, Prefetchable							**SL2 → LL2, Non-Prefetchable**						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	1156	1156	1156	1156	1156	1156	Core 0	986	986	986	986	986	986
Core 1		1147	1147	1147	1147	1147	Core 1		978	978	978	978	978
Core 2			1156	1156	1156	1156	Core 2			986	986	987	986
Core 3				1147	1147	1147	Core 3				978	978	978
Core 4					1156	1156	Core 4					987	987
Core 5						1147	Core 5						978
Total Bandwidth	1156	2303	3459	4606	5762	6909	Total Bandwidth	986	1964	2950	3928	4916	5893

Table 26. Performance of Multiple DMA Masters Sharing SL2 at 500 MHz

LL2 → SL2, Prefetchable					LL2 → SL2, Non-Prefetchable				
Master	**Bandwidth (MBps)**				**Master**	**Bandwidth (MBps)**			
EDMA TC0	711	711	711	711	EDMA TC0	711	711	711	711
EDMA TC1		711	711	711	EDMA TC1		711	711	711
EDMA TC2			1599	1599	EDMA TC2			1599	1599
EDMA TC3				1599	EDMA TC3				1598
Total Bandwidth	711	1422	3021	4620	Total Bandwidth	711	1422	3021	4619
SL2 → LL2, Prefetchable					**SL2 → LL2, Non-Prefetchable**				
Master	**Bandwidth (MBps)**				**Master**	**Bandwidth (MBps)**			
EDMA TC0	709	709	709	709	EDMA TC0	666	666	666	666
EDMA TC1		711	711	711	EDMA TC1		666	666	666
EDMA TC2			1767	1767	EDMA TC2			1598	1590
EDMA TC3				1777	EDMA TC3				1579
Total Bandwidth	709	1420	3187	4964	Total Bandwidth	666	1332	2930	4501

Table 27. Performance of Multiple DMA Masters Sharing SL2 at 625 MHz

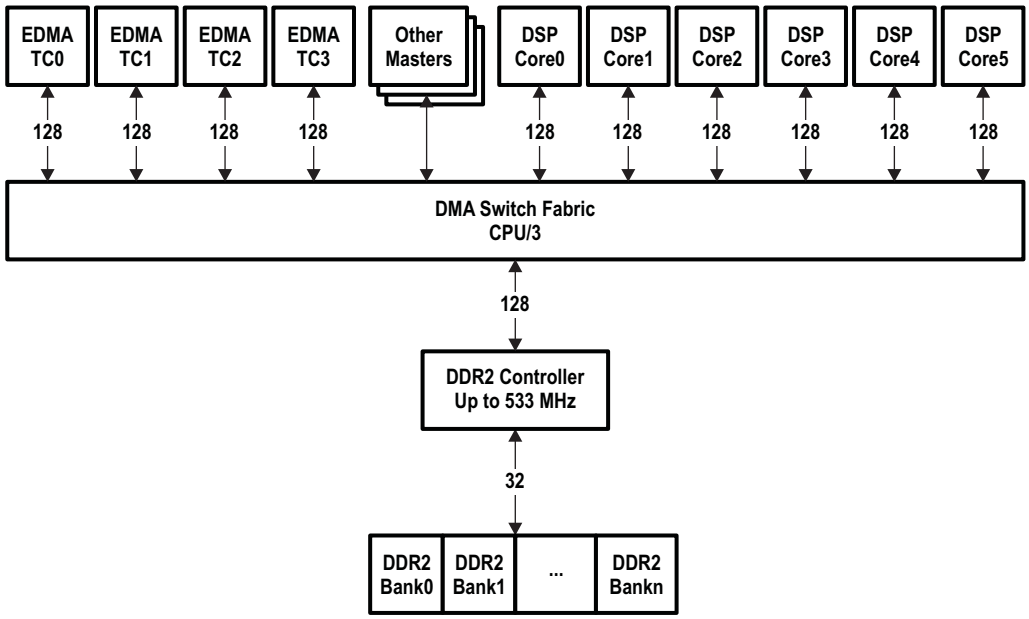
LL2 → SL2, Prefetchable					LL2 → SL2, Non-Prefetchable				
Master	**Bandwidth (MBps)**				**Master**	**Bandwidth (MBps)**			
EDMA TC0	888	888	888	888	EDMA TC0	888	888	888	888
EDMA TC1		888	888	888	EDMA TC1		888	888	888
EDMA TC2			1999	1999	EDMA TC2			1999	1999
EDMA TC3				1999	EDMA TC3				1999
Total Bandwidth	888	1776	3775	5774	Total Bandwidth	888	1776	3775	5774
SL2 → LL2, Prefetchable					**SL2 → LL2, Non-Prefetchable**				
Master	**Bandwidth (MBps)**				**Master**	**Bandwidth (MBps)**			
EDMA TC0	886	886	886	886	EDMA TC0	834	833	833	833
EDMA TC1		888	888	888	EDMA TC1		833	833	833
EDMA TC2			2209	2209	EDMA TC2			1999	1991
EDMA TC3				2221	EDMA TC3				1975
Total Bandwidth	886	1774	3983	6204	Total Bandwidth	834	1666	3665	5632

The above data proves that the SL2 access is not a bottleneck for multiple masters accessing SL2 simultaneously. The SL2 has enough bandwidth to support multiple masters. The throughput limitation is either at the master itself or the transport mechanism between the SL2 and the individual DMA masters. Additionally, read throughput is a little bit better than write throughput. The prefetch buffer improves the read performance.

10.2 Performance of Multiple Masters Sharing DDR2

Figure 22 shows the DDR2 access paths.

Figure 22. DDR2 Memory Architecture and Access Paths



Multiple masters access the DDR2 SDRAM through the DMA switch fabric. If multiple masters access it at the same time, it is arbitrated at the DDR2 controller based on the priority of the masters.

The DDR2 memory on the C6472/TCI6486 EVM used for these tests has eight banks. Most DDR2 memory configurations have either four or eight banks although the DDR2 controller does support memory configurations with one and two banks. This must be verified on your system, as it affects performance. Also note that the row size may be different for other memory configurations, as discussed previously.

The DDR2 memory on the C6472/TCI6486 EVM contains eight banks organized as shown in Table 28.

Table 28. Memory Organization in DDR2 Banks

	Bank 0	Bank 1	...	Bank 7
Row 0	byte 0 to 4095	byte 4096 to 8191	…	byte 28672 to 32767
Row 1	byte 32768 to 36863	byte 36864 to 40959	…	byte 61440 to 65535
...	…	…	…	…

Even though DDR2 has multiple banks, there are not multiple buses connected to it like SL2. Therefore, the number of banks does not directly improve the throughput.

The DDR2 SDRAM is accessed in rows (or pages). Before the master can access the data in a row, the controller must first open the row and then it can randomly access any memory in the row. If the master wants to access data in a different row in the same bank, the previous row must be first be closed and then new row must be opened. The row switch (row close and then row open) operations introduce extra delay cycles. This is referred to as row switch overhead.

Every bank can have an open row, so the DDR2 with eight banks can have eight open rows at the same time. Since they are interleaved as shown in Table 28, this dramatically reduces the probability of row switch. For example, after a master opens row 0 in bank 0 for access, it can then open row 1 in bank 1 without closing the row 0 in bank 0. Then the master can access both row 0 in bank 0 and row 1 in bank 1 randomly without any row switch overhead.

Two data structures for test are defined to verify the effect of DDR2 row switch overhead.

Table 29 is the worst case with maximum row switching overhead. Every master access results in row switch if the previous access is from a different master.

Table 29. Worst-Case Multiple Master Access to DDR2

	Bank 0	Bank 1	Bank2	…	Bank n	…
Row 0	Master 0 Access Range					
Row 1	Master 1 Access Range					
Row 2	Master 2 Access Range					
…	…					
Row n	Master n Access Range					
…	…					

Table 30 is the best case without any row switching because each master always access an open row dedicated to it.

Table 30. Best-Case Multiple Master Access to DDR2

	Bank 0	Bank 1	Bank2	…	Bank n	…
Row 0	Master 0 Access Range					
Row 1		Master 1 Access Range				
Row 2			Master 2 Access Range			
…				…		
Row n					Master n Access Range	
…	…					…

10.2.1 Performance of Multiple DSP Cores Sharing the DDR2 Memory

Table 31 and Table 32 show the performance of multiple DSP cores sharing the 32-bit 533-MHz DDR2 under different scenarios. The DDR2 is configured as cacheable, the L1D cache is set to 32KB and the L2 cache size is set to 256KB. The non-cacheable case is not measured because it demands less bandwidth than the cacheable case. The priority for all DSP cores is equivalent for this test.

Table 31. Performance of Multiple DSP Cores Sharing DDR2 at 500 MHz

LL2 → DDR2, Different Row on Different DDR2 Bank							LL2 → DDR2, Different Row on Same DDR2 Bank						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	247	247	248	248	253	245	Core 0	247	235	234	237	221	209
Core 1		251	250	248	250	243	Core 1		239	233	235	221	209
Core 2			250	249	251	245	Core 2			234	235	223	213
Core 3				250	251	246	Core 3				235	224	215
Core 4					251	247	Core 4					226	216
Core 5						247	Core 5						217
Total Bandwidth	247	498	748	995	1256	1473	Total Bandwidth	247	474	701	942	1115	1279
DDR2 → LL2, Different Row on Different DDR2 Bank							**DDR2 → LL2, Different Row on Same DDR2 Bank**						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	368	361	349	345	331	311	Core 0	359	334	323	272	257	199
Core 1		364	355	346	329	306	Core 1		331	321	272	256	199
Core 2			357	350	332	310	Core 2			323	280	263	206
Core 3				353	334	312	Core 3				287	269	211
Core 4					335	314	Core 4					273	215
Core 5						315	Core 5						218
Total Bandwidth	368	725	1061	1394	1661	1868	Total Bandwidth	359	665	967	1111	1318	1248

Table 32. Performance of Multiple DSP Cores Sharing DDR2 at 625 MHz

LL2 → DDR2, Different Row on Different DDR2 Bank							LL2 → DDR2, Different Row on Same DDR2 Bank						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	299	298	297	296	296	285	Core 0	299	281	272	264	251	220
Core 1		299	297	296	295	282	Core 1		281	273	263	248	212
Core 2			298	297	296	284	Core 2			273	265	250	217
Core 3				297	296	286	Core 3				266	252	221
Core 4					297	287	Core 4					253	223
Core 5						287	Core 5						225
Total Bandwidth	299	597	892	1186	1480	1711	Total Bandwidth	299	562	818	1058	1254	1318
DDR2 → LL2, Different Row on Different DDR2 Bank							**DDR2 → LL2, Different Row on Same DDR2 Bank**						
Master	**Bandwidth (MBps)**						**Master**	**Bandwidth (MBps)**					
Core 0	407	406	403	390	360	340	Core 0	407	374	350	313	265	210
Core 1		407	402	387	353	330	Core 1		374	349	307	254	199
Core 2			404	390	359	337	Core 2			352	314	262	207
Core 3				395	363	340	Core 3				320	267	211
Core 4					366	345	Core 4					270	215
Core 5						347	Core 5						218
Total Bandwidth	407	813	1209	1562	1801	2039	Total Bandwidth	407	748	1051	1254	1318	1260

Table 31 and Table 32 show that the DDR2 interface has just enough bandwidth to support multiple DSP cores accessing the DDR2 memory simultaneously. When all six cores access the DDR2 simultaneously, total bandwidth required is close to the theoretical bandwidth of DDR2 (2133 MBps). The throughput limitation is at either the DSP core or the transport mechanism between rather than at the DDR2 interface.

The performance of multiple cores accessing different rows on the same DDR2 bank is worse than the performance of multiple cores accessing different rows on different DDR2 banks due to the DDR2 row switch overhead. Actual memory use by an application is between these two extreme cases based upon memory allocation and usage.

10.2.2 Performance of Multiple EDMA Masters Sharing the DDR2 Memory

Table 33 and Table 34 show the performance of multiple EDMA TCs sharing a 32-bit 533-MHz DDR2 memory under different conditions. The priority for all EDMA TCs is the same for these tests.

Table 33. Performance of Multiple EDMA Masters Sharing DDR2 (DDR2 at 533 MHz and CPU at 500 MHz)

LL2 → DDR2, Different Row on Different DDR2 Bank, Same Priority								LL2 → DDR2, Different Row on Same DDR2 Bank, Same Priority							
Master	**Bandwidth (MBps)**							**Master**	**Bandwidth (MBps)**						
EDMA TC0	710	703	695				521	EDMA TC0	710	321	214				160
EDMA TC1		703	695			695	521	EDMA TC1		321	214			214	160
EDMA TC2			695		1043	695	521	EDMA TC2			214		321	214	160
EDMA TC3				1738	1043	695	521	EDMA TC3				1738	321	214	160
Total Bandwidth	710	1406	2085	1738	2086	2085	2084	Total Bandwidth	710	642	642	1738	642	642	640
DDR2 → LL2, Different Row on Different DDR2 Bank, Same Priority								**DDR2 → LL2, Different Row on Same DDR2 Bank, Same Priority**							
Master	**Bandwidth (MBps)**							**Master**	**Bandwidth (MBps)**						
EDMA TC0	785	781	470				325	EDMA TC0	785	619	312				231
EDMA TC1		781	470			376	324	EDMA TC1		619	312			278	231
EDMA TC2			940		940	752	648	EDMA TC2			692		698	557	465
EDMA TC3				1560	940	752	627	EDMA TC3				1560	698	557	463
Total Bandwidth	785	1562	1880	1560	1880	1880	1924	Total Bandwidth	785	1238	1316	1560	1396	1392	1390
TC0 and TC2 Read DDR2 → LL2, TC1 and TC3 Write LL2 → DDR2, Different DDR2 Bank								**TC0 and TC2 Read DDR2 → LL2, TC1 and TC3 Write LL2 → DDR2, Same DDR2 Bank**							
Master	**Bandwidth**							**Master**	**Bandwidth**						
EDMA TC0	784	693	578				578	EDMA TC0	782	472	270				210
EDMA TC1		703	46			162	26	EDMA TC1		698	494			218	176
EDMA TC2			1156		1310	1309	1156	EDMA TC2			549		297	440	408
EDMA TC3				1738	321	162	26	EDMA TC3				1736	1327	218	189
Total Bandwidth	784	1396	1780	1738	1631	1633	1786	Total Bandwidth	782	1170	1313	1736	1624	876	983

Table 34. Performance of Multiple EDMA Masters Sharing DDR2 (DDR2 at 533 MHz and CPU at 625 MHz)

LL2 → DDR2, Different Row on Different DDR2 Bank, Same Priority								LL2 → DDR2, Different Row on Same DDR2 Bank, Same Priority							
Master	**Bandwidth (MBps)**							**Master**	**Bandwidth (MBps)**						
EDMA TC0	887	865	691				521	EDMA TC0	887	310	210				160
EDMA TC1		872	692			692	521	EDMA TC1		321	211			211	160
EDMA TC2			695		1035	692	521	EDMA TC2			214		314	211	160
EDMA TC3				2070	1036	693	521	EDMA TC3				2013	315	212	160
Total Bandwidth	887	1737	2078	2070	2071	2077	2084	Total Bandwidth	887	631	635	2013	629	634	640
DDR2 → LL2, Different Row on Different DDR2 Bank, Same Priority								**DDR2 → LL2, Different Row on Same DDR2 Bank, Same Priority**							
Master	**Bandwidth (MBps)**							**Master**	**Bandwidth (MBps)**						
EDMA TC0	896	890	487				354	EDMA TC0	896	615	324				233
EDMA TC1		890	487			393	347	EDMA TC1		617	324			276	233
EDMA TC2			975		972	778	690	EDMA TC2			699		748	547	466
EDMA TC3				1780	973	779	653	EDMA TC3				1741	749	550	460
Total Bandwidth	896	1780	1949	1780	1945	1950	2044	Total Bandwidth	896	1232	1347	1741	1497	1373	1392
TC0 and TC2 Read DDR2 → LL2, TC1 and TC3 Write LL2 → DDR2, Different DDR2 Bank								**TC0 and TC2 Read DDR2 → LL2, TC1 and TC3 Write LL2 → DDR2, Same DDR2 Bank**							
Master	**Bandwidth**							**Master**	**Bandwidth**						
EDMA TC0	896	764	598				598	EDMA TC0	896	429	202				189
EDMA TC1		875	54			68	27	EDMA TC1		871	668			224	189
EDMA TC2			1197		1531	1529	1197	EDMA TC2			426		316	456	379
EDMA TC3				2086	135	68	27	EDMA TC3				2086	1484	224	189
Total Bandwidth	896	1639	1849	2086	1666	1665	1849	Total Bandwidth	896	1300	1296	2086	1800	904	946

Table 33 and Table 34 show that the DDR2 interface does not have enough bandwidth to support EDMA accesses from all four TCs to it simultaneously. Additionally, for write operations, the available bandwidth is equally split between the TCs. However, for read operations, the allocated bandwidth for TC0 and TC1 is equal and the allocated bandwidth for TC2 and TC3 is equal, while the TC2 and TC3 allocated bandwidth is twice that of TC0 and TC1. This is the result of the increased FIFO size in the TC2 and TC3 controllers. Also note that when the EDMA masters were accessing rows in different banks, the performance was much improved as seen for the DSP core accesses as expected. This is addressed further in the next section. Normally, reads achieve more bandwidth when competing against writes because the read command is performed before the write command if the Read FIFO inside the DDR EMIF is not full.

10.2.3 Effect of DDR2 Bank and Row Switching on Multiple Masters Sharing DDR2

The performance of multiple EDMA TCs accessing different rows on the same DDR2 bank is much worse than the performance of multiple EDMA TCs accessing different rows on different DDR2 banks. The reason is the DDR2 row switch overhead. The result becomes worse when DDR2 load becomes heavy. The worst case is when multiple EDMA TCs write to different rows on the same DDR2 bank, which is almost dominated by the row switch overhead because every write burst results in row switch.

The probability of multiple masters accessing the same DDR2 bank depends on the number of active DMA masters and number of DDR2 banks. For example, if four EDMA TCs randomly access DDR2 memory, the probability of at least two TCs accessing the same DDR2 bank is:

$$1 - C_8^4 P_4^4 / 8^4 = 59\%$$

Table 35 lists the probability for different combinations of masters and banks.

Table 35. Probability of Multiple Masters Accessing the Same DDR2 Bank

	2 Masters	4 Masters	6 Masters	8 Masters	10 Masters
4 Banks	25%	90.6%	100%	100%	100%
8 Banks	12.5%	59%	92.3%	99.7%	100%

According to the above data, to minimize the row switch overhead, DDR2 memory with eight banks is strongly recommended. The DDR2 controller on the C6472/TCI6486 device is optimized to alleviate the row switch overhead.

The DDR2 memory controller makes use of following FIFOs:

- Write Command FIFO, stores up to 7 write commands from masters
- Write Data FIFO, stores up to 176 bytes of data from masters
- Read Command FIFO, stores up to 22 read commands from masters
- Read Data FIFO, stores up to 272 bytes of data to masters

The DDR2 memory controller performs command re-ordering and scheduling in an attempt to achieve efficient transfers with maximum throughput. The goal is to maximize the utilization of open rows, while hiding the overhead of opening and closing DDR2 SDRAM rows. Command re-ordering takes place within the command FIFOs.

The DDR2 memory controller examines each of the commands in the FIFOs and performs the following reordering:

- Among all pending reads, the controller processes reads to rows already open. Similarly for pending writes, the controller selects writes to rows already open.
- Then it selects the highest priority command from the pending reads and writes to open new rows (causing row switching delays). If multiple commands have the highest priority, then the DDR2 memory controller selects the oldest command.
- If the Read FIFO is not full, then a read command can be performed; otherwise, the controller processes to the next write command.

As the above test data showed, the read performance degradation caused by row switching is much less than the write performance degradation caused by row switching. The reason is the read FIFOs are much bigger than the write FIFOs.

Another method may alleviate the row switch overhead is to organized data for different masters, as shown in Table 36.

Table 36. Data Buffer Organization to Minimize Row Switch Overhead

	Bank 0	Bank 1	Bank 2	…	Bank n	…
Row 0	Data buffer for master 0	Data buffer for master 1	Data buffer for master 2	…	Data buffer for master n	…
Row 1						
Row 2						
…						
Row n						
Row n+1	…	…	…	…	…	
…	…	…	…	…	…	

A special EDMA configuration [BIDX = (row size) * (bank number)] can be utilized to transfer data between the L2 memory and the DDR2 memory while keeping all of the DDR2 memory accesses in a single bank.

10.2.4 Effect of Priority on Multiple Masters Sharing DDR2

If there are multiple commands pending to access open rows or multiple commands pending to access rows currently not open, the DDR2 controller arbitrates these requests according the priority of the masters. All of the results shown above were measured with equal priority for all masters.

Table 37 shows the effect of priority when multiple EDMA TCs access the 32-bit DDR2 memory simultaneously. This data was collected with the core clock operating at 625 MHz.

Table 37. Effect of Priority When Multiple TCs Access DDR2

DDR2 → LL2, different row on different DDR2 bank														
Master	**Priority**	**Bandwidth (MBps)**			**Priority**	**Bandwidth (MBps)**			**Priority**	**Bandwidth (MBps)**				
EDMA TC0	0	897	870	578	7			28	0	890	487			354
EDMA TC1	1	885	870	558	6		57	28	0	890	487		393	347
EDMA TC2	2		325	756	5	217	344	754	0		975	972	778	680
EDMA TC3	3			205	4	1741	1553	1134	0			973	779	663
Total Bandwidth		1782	2065	2097		1958	1954	1944		1780	1949	1945	1950	2044

The data in Table 37 proves that the priority dramatically affects the bandwidth allocation between multiple read operations when the DDR2 is heavily loaded.

Additional tests show that the priority does not affect bandwidth allocation for EDMA write operations since they are dominated by row switch overhead. When multiple DSP cores access the DDR2, the priority does not affect the bandwidth allocation because the six DSP cores cannot heavily load the DDR2 interface.

The re-ordering and scheduling rules listed above may lead to command starvation, which is the prevention of certain commands from being processed by the DDR2 memory controller. A continuous stream of DDR2 SDRAM commands to a row in an open bank can block commands to the closed row in the same bank. To avoid command starvation, the DDR2 memory controller can momentarily raise the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PRIO_RAISE field in the Burst Priority Register (BPRIO) sets the number of the transfers that must be made before the DDR2 memory controller raises the priority of the oldest command. Leaving the PRIO_RAISE bits at their default value (0xFF) disables this feature in the DDR2 memory controller. This means commands can stay in the command FIFO indefinitely. Therefore, these bits should be set appropriately when configuring the DDR2 controller to enable this feature according application requirements.

All above data are measured with BPRIO = 0x7F. Table 38 shows the effect of BPRIO register on a 500-MHz C6472/TCI6486 device with 32-bit DDR2 interface. When collecting this data, the BPRIO register contains the default value (0xFF).

Table 38. Effect of BPRIO When Multiple TCs Access DDR2

DDR2 → LL2, Different Row on Different DDR2 Bank										
Master	**Priority**	**BPRIO=0xFF**			**BPRIO=0x7F**			**BPRIO=0**		
EDMA TC0	0	783	753	557	783	753	547	783	626	472
EDMA TC1	1	778	752	539	778	752	529	775	470	462
EDMA TC2	2		375	784		375	748		783	472
EDMA TC3	3			***0***			55			472
Total Bandwidth		1561	1880	1880	1561	1880	1879	1558	1879	1878
DDR2 → LL2, Different Row on Same DDR2 Bank										
Master	**Priority**	**BPRIO=0xFF**			**BPRIO=0x7F**			**BPRIO=0**		
EDMA TC0	0	619	619	356	619	542	311	613	280	178
EDMA TC1	1	619	619	304	619	540	266	613	280	160
EDMA TC2	2		***0***	517		172	474		559	321
EDMA TC3	3			671			155			321
Total Bandwidth		1238	1238	1848	1238	1254	1206	1226	1119	980

The data in Table 38 shows that a master may be starved completely (bandwidth = 0) when BPRIO = 0xFF, even though the total bandwidth is maximized. The bandwidth allocation is very fair with BPRIO = 0, but the overall bandwidth is reduced. BPRIO = 0x7F is a possible balance between the two extreme cases.

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**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

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