Technical notes on using Analog Devices DSPs, processors and development tools
Visit our Web resources http://www.analog.com/ee-notes and http://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

ADSP-SC59x SHARC+ Processor System Optimization Techniques

Contributed by Naga Snigdha Kondepati and Mitesh Moonat *Rev 01 – December 6, 2022*

Introduction

The ADSP-SC59x SHARC+ processors (both ADSP-2159x/ADSP-SC591/SC592/SC594 and ADSP-SC595/SC596/SC598 processor families) provide an optimized architecture that supports high-system bandwidth and advanced peripherals. The EE445 note discusses key architectural features of the processor that contributes to the overall system bandwidth and available bandwidth optimization techniques.

Most of the theoretical content of this EE Note is the same as in *ADSP-2156x SHARC+ Processor System Optimization Techniques (EE-412)*[1]. The EE445 note includes the figures, tables, and data specific to the ADSP-SC59x family of processors. Throughout this EE Note, references to **ADSP-SC594** processors refer to ADSP-2159x/ADSP-SC591/SC592/SC594 processors, and the **ADSP-SC598** processors refer to ADSP-SC595/SC596/SC598 processors. In these EE Note sections, the discussions refer to data measured on the ADSP-SC598 processor at 1 GHz CCLK (core clock) and 900 MHz DCLK (ddr clock) as an example. However, the concepts discussed also apply to the ADSP-SC594 processors at 1 GHz CCLK and 800 MHz DCLK. See the Appendix on page 34 for the data corresponding to these modes of operation.

ADSP-SC59x Processor Architecture

This section describes the ADSP-SC59x processor's key architectural features that play a crucial role in system bandwidth and performance. For detailed information, refer to the *ADSP-2159x/ADSP-SC591/SC592/SC594 SHARC+ Processor Hardware Reference* [3] *or ADSP-SC595/SC596/SC598 SHARC+ Processor Hardware Reference* [4].

The overall architecture of the ADSP-SC59x processors consists of three main system components: system bus completers, system bus requesters, and system crossbars. Figure 1 through Figure 4 show how these components are interconnected to form the complete system in the ADSP-2159x/ ADSP-SC591/SC592/SC594 and in the ADSP-SC595/SC596/SC598 processors.

System Bus Completers

As shown in Figure 1 (right) and Figure 3 (right), system bus completers include on-chip and off-chip memory devices and controllers, such as L1 SRAM, L2 SRAM, the Dynamic Memory Controller (DMC) for DDR3/DDR3L SDRAM devices, memory-mapped peripherals such as SPI FLASH, and the System Memory Mapped Registers (MMRs). Each system bus completer has its own latency characteristics, operating in each clock domain. For example, L1 SRAM runs at CCLK, L2 SRAM runs at SYSCLK, the DMC interface runs at DCLK, and so forth.

Figure 1 ADSP-2159x/ADSP-SC591/SC592/SC594 System Cross Bar (SCB) Block Diagram

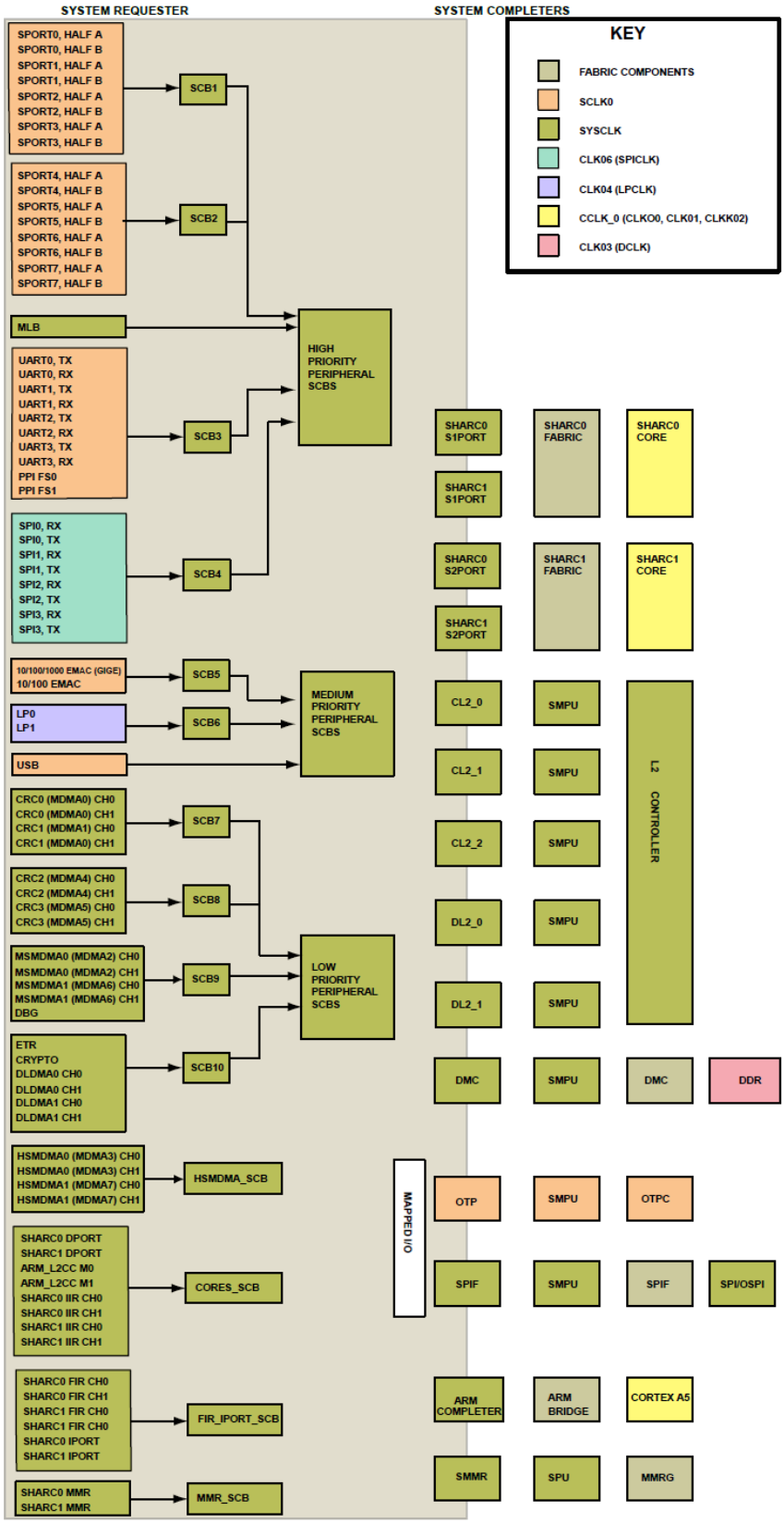


Figure 2 ADSP-2159x/ADSP-SC591/SC592/SC594 SCB Interconnections

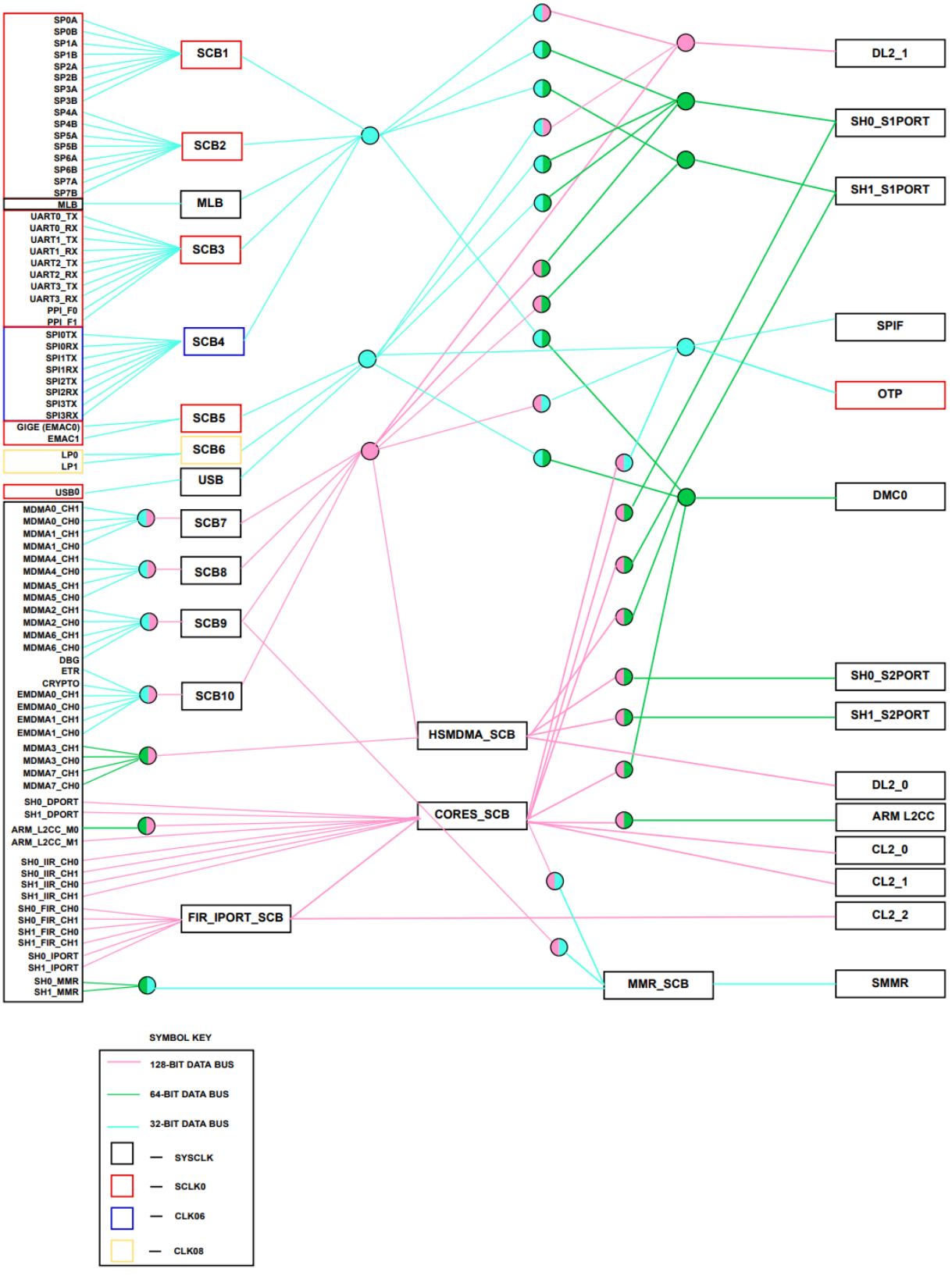


Figure 3 ADSP-SC595/SC596/SC598 System Cross Bar (SCB) Block Diagram

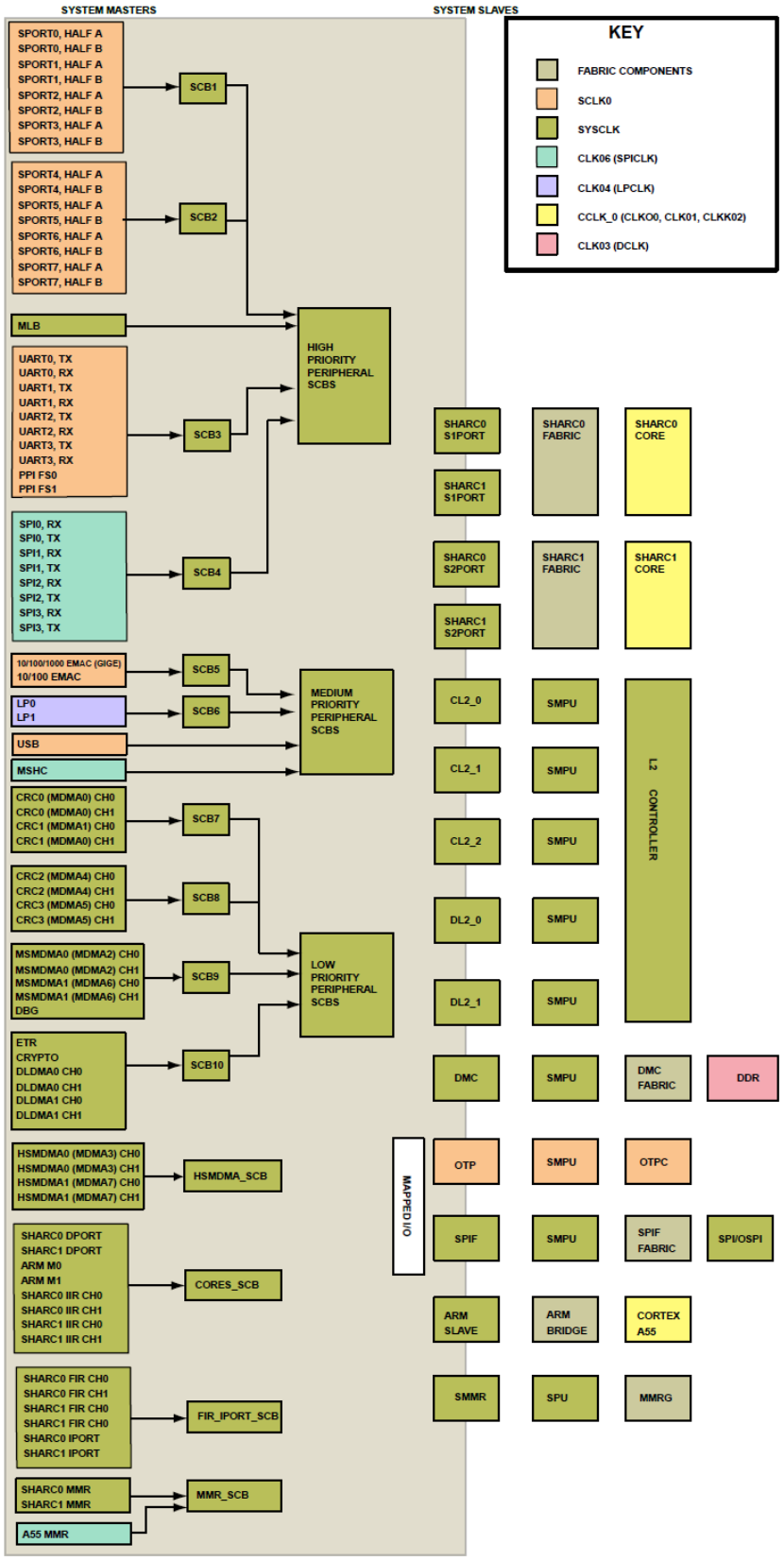
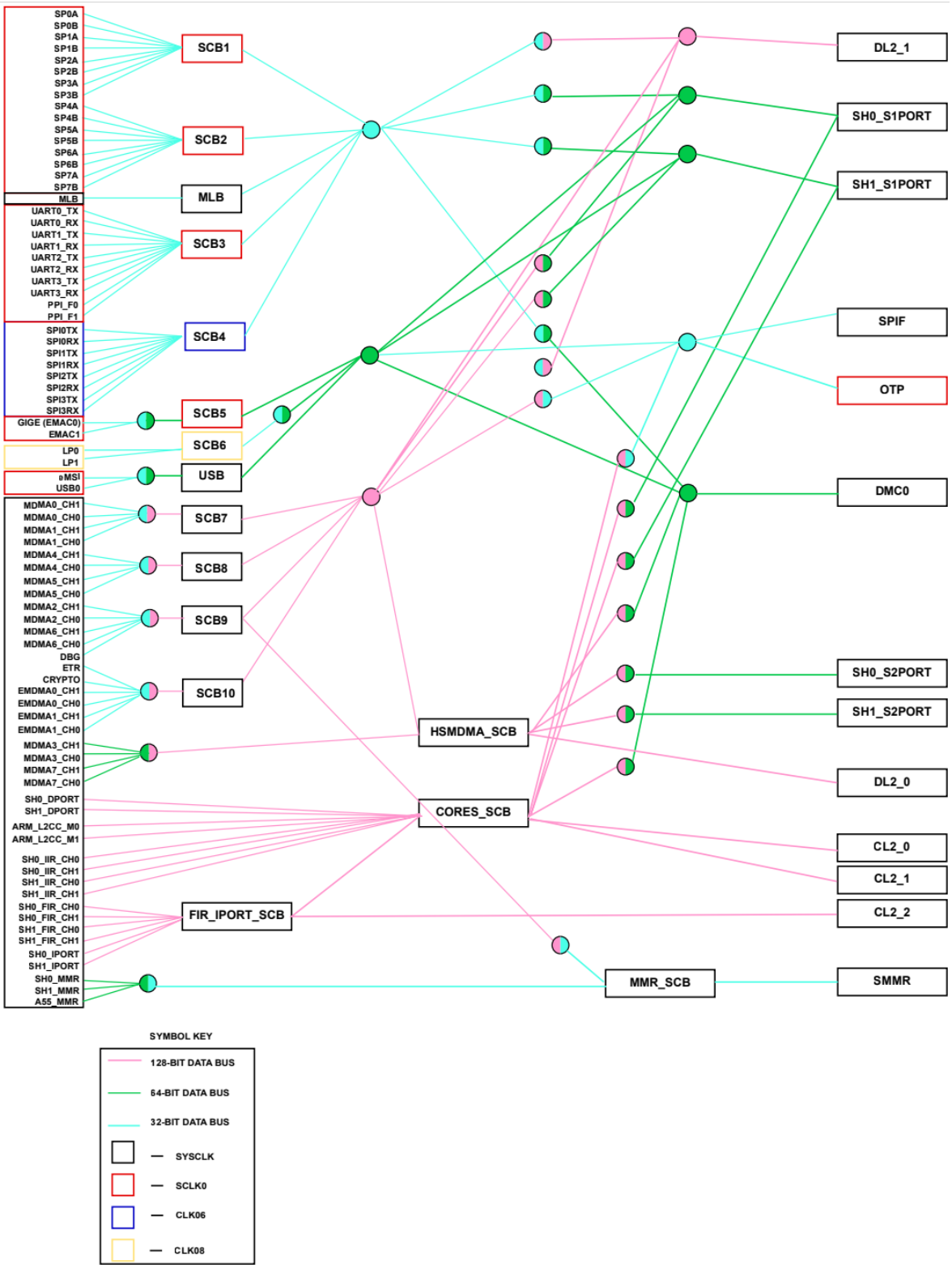


Figure 4 ADSP-SC595/SC596/SC598 SCB Interconnections



System Bus Requesters

The system bus requesters are shown in Figure 1 (left) and Figure 3 (left). They include peripheral Direct Memory Access (DMA) channels such as the Serial Port (SPORT) and Serial Peripheral Interface (SPI). Also included are the Memory-to-Memory DMA channels (MDMA) and the core. Each peripheral runs at a different clock speed and has individual bandwidth requirements. For example, high speed peripherals such as the Link Port require higher bandwidth than slower peripherals such as the SPORT or UART.

System Crossbars

The System Crossbars (SCB) are the fundamental building blocks of the system bus interconnect. As shown in Figure 2 and Figure 4, the SCB interconnect is built from multiple SCBs in a hierarchical model connecting system bus requesters to system bus completers. They provide concurrent data transfer between multiple bus requesters and multiple bus completers, providing flexibility and full-duplex operation. The SCBs also provide a programmable arbitration model for bandwidth and latency management. SCBs run on different clock domains (SCLK0, SYSCLK, SPI clock, and LP clock) that introduce their own latencies to the system.

System Latencies, Throughput, and Optimization Techniques

The following sections describe aspects related to latencies and throughput of system bus requesters, system bus completers, and the system cross bars. The sections also discuss optimization techniques to reduce system latencies and improve throughput.

Understanding the System Requesters

DMA Parameters

Each DMA channel has two buses: one that connects to the SCB, which in turn is connected to the SCB completer (for example, memories), and another bus that connects to either a peripheral or another DMA channel. The SCB/memory bus width can vary among 8, 16, 32, or 64 bits and is defined by the `DMA_STAT.MBWID` bit field. The peripheral bus width can vary among 8, 16, 32, 64, or 128 bits and is defined by the `DMA_STAT.PBWID` bit field. For ADSP-SC59x processors, the memory and peripheral bus widths for most of the DMA channels is 32 bits (4 bytes). However, for some channels, it is 64 bits (8 bytes).

The DMA parameter `DMA_CFG.PSIZE` determines the width of the peripheral bus in use. It can be configured to 1, 2, 4, or 8 bytes. However, it cannot be greater than the maximum possible bus width defined by the `DMA_STAT.PBWID` bit field. This restriction exists because burst transactions are not supported on the peripheral bus.

The DMA parameter `DMA_CFG.MSIZE` determines the actual size of the SCB bus in use. It also determines the minimum number of bytes that are transferred from and to memory corresponding to a single DMA request or grant. It can be configured to 1, 2, 4, 8, 16, or 32 bytes. When the `MSIZE` value is greater than `DMA_STAT.MBWID`, the SCB performs burst transfers to transfer the data equal to the `MSIZE` value.

It is important to choose the appropriate `MSIZE` value, both from a functionality and a performance perspective. When choosing the `MSIZE` value, consider the following factors:

- The start address of the work unit must align to the `MSIZE` value. Failing to do so generates a DMA error interrupt.

- From a performance perspective, use the highest possible `MSIZE` value (32 bytes) for improved average throughput. This results in a higher likelihood of uninterrupted sequential accesses to the completer (memory), which is the most efficient for typical memory designs.
- From a performance perspective, the minimum `MSIZE` value is determined by the burst length supported by the memory device in some cases. For example, for DDR3 accesses, the minimum `MSIZE` value is limited by the DDR3 burst length (16 bytes). Any `MSIZE` value below this length leads to a significant throughput loss. For details, refer to the L3/External Memory Throughput section on page 21.

Memory to Memory DMA (MDMA)

The ADSP-SC59x processors support multiple MDMA streams (MDMA0/1/2/3/4/5/6/7) to transfer data from one memory to another (L1/L2/L3/memory-mapped peripherals such as SPI FLASH). Different MDMA streams can transfer the data at different bandwidths, as they run at different clock speeds and support different data bus widths. Table 1 shows the MDMA streams and the corresponding maximum theoretical bandwidth supported by the ADSP-SC59x processors. Table 2 and Table 3 show the memory completers and their maximum theoretical bandwidth supported by the ADSP-SC59x processors.

Table 1 MDMA Streams and Maximum Theoretical Bandwidth

Maximum CCLK/SYSCLK/SCLKx Speed (MHz): 1000 / 500 / 125						
MDMA Stream No.	**MDMA Type**	**MDMA Source Channel**	**MDMA Destination Channel**	**Clock Domain**	**Bus Width (bits)**	**Maximum Bandwidth (MB/s)**
0	CRC0	8	9	SYSCLK 500 MHz Maximum	32	2000
1	CRC1	18	19		32	2000
2	Enhanced Bandwidth or Medium Speed MDMA (MSMDMA)	39	40		32	2000
3	Maximum Bandwidth or High Speed MDMA (HSMDMA)	43	44		64	4000
4	CRC2	45	46		32	2000
5	CRC3	47	48		32	2000
6	Enhanced Bandwidth or Medium Speed MDMA (MSMDMA1)	49	50		32	2000
7	Maximum Bandwidth or High Speed MDMA (HSMDMA1)	51	52		64	4000

Table 2 Memory Completers and Maximum Theoretical Bandwidth (SC594 Processors)

Maximum CCLK / SYSCLK /SCLKx / DCLK Frequency (MHz): 1000 / 500 / 125 / 800				
Memory Type (L1 / L2 / L3)	**Clock Domain**	**Bus Width (Bits)**	**Data Rate Clock Rate**	**Maximum Theoretical Bandwidth (MB/s)**
L1	CCLK	32	1	4000
L2	SYSCLK	128	1	8000
L3	DCLK	16	2	3200

Table 3 Memory Completers and Maximum Theoretical Bandwidth (SC598 Processors)

Maximum CCLK / SYSCLK /SCLKx / DCLK Frequency (MHz): 1000 / 500 / 125 / 900				
Memory Type (L1 / L2 / L3)	**Clock Domain**	**Bus Width (Bits)**	**Data Rate Clock Rate**	**Maximum Theoretical Bandwidth (MB/s)**
L1	CCLK	32	1	4000
L2	SYSCLK	128	1	8000
L3	DCLK	16	2	3600

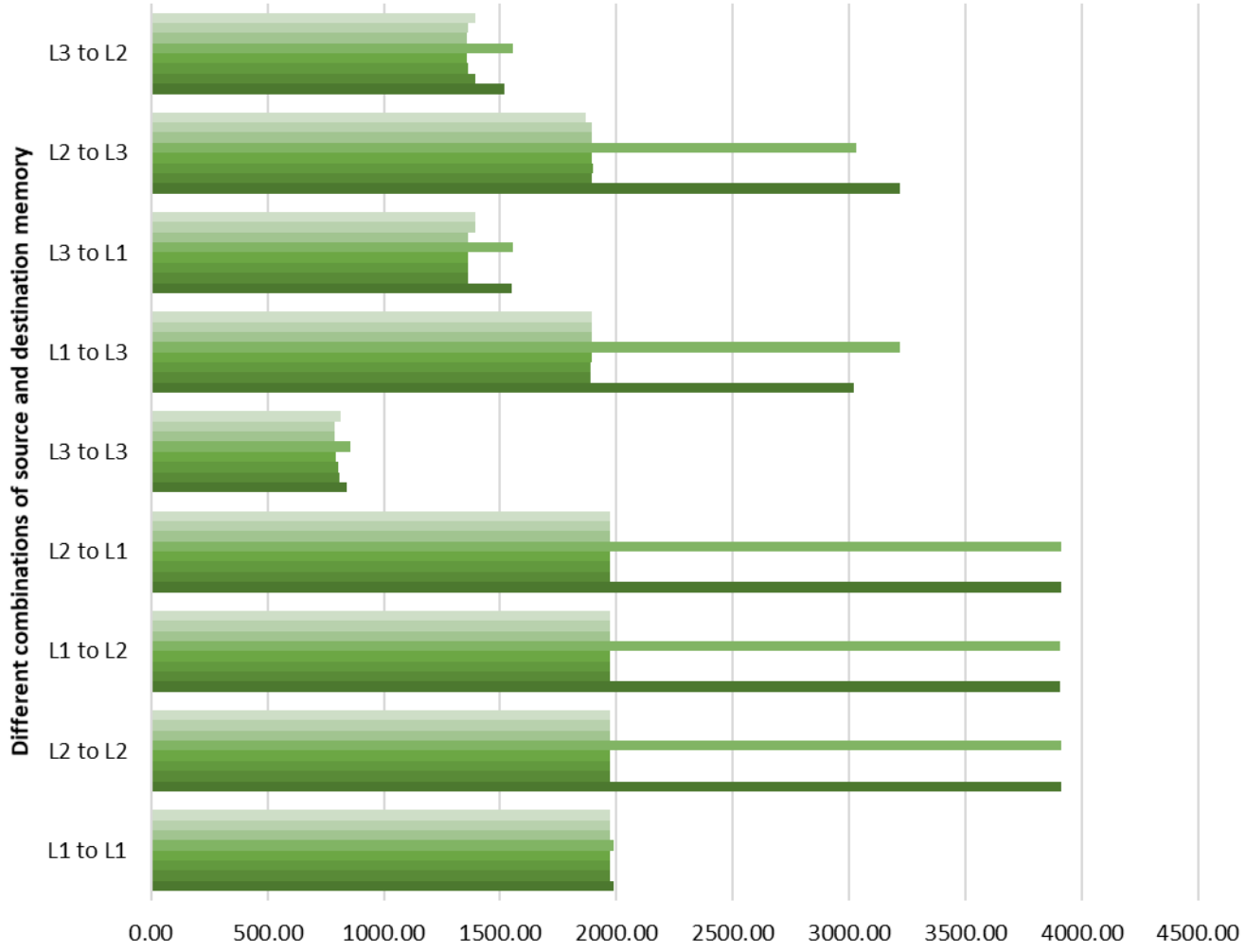
The actual (measured) MDMA throughput is always less than or equal to the minimum of the maximum theoretical throughput supported by the MDMA, source memory, or destination memory. For example, the measured throughput of MDMA0 between L1 and L2 is less than or equal to 2000 MB/s, which is limited by the maximum bandwidth of MDMA0. Similarly, the measured throughput of MDMA3 between L1 and L3 is less than or equal to 3200 MB/s for the ADSP-SC594 processors and 3600 MB/s for the ADSP-SC598 processors, which is limited by the maximum bandwidth of L3.

Figure 5 shows the actual throughput measured on the bench for various MDMA streams with different combinations of source and destination memories. The measurements were taken with:

- `MSIZE` = 32 bytes
- DMA count = 16,384 bytes at CCLK = 1 GHz
- SYSCLK = 500 MHz
- DCLK = 900 MHz

The source code found in the `Test1_MDMA_Throughput` subfolder of the Zip file, supplied with the application note[5], can be used to measure MDMA throughput.

Figure 5 MDMA Throughput on ADSP-SC598 Processor



	L1 to L1	L2 to L2	L1 to L2	L2 to L1	L3 to L3	L1 to L3	L3 to L1	L2 to L3	L3 to L2
MDMA0	1972.08	1973.50	1973.03	1973.50	813.75	1895.86	1393.91	1869.89	1393.20
MDMA1	1973.03	1973.50	1973.03	1973.50	790.28	1896.30	1394.15	1896.30	1360.80
MDMA2	1973.03	1973.50	1973.03	1973.50	786.03	1895.86	1361.93	1895.86	1358.99
MDMA3	1985.46	3914.00	3908.40	3912.13	854.49	3217.60	1554.75	3031.83	1553.28
MDMA4	1973.03	1973.50	1973.03	1973.50	795.34	1895.86	1359.67	1894.54	1359.44
MDMA5	1973.03	1973.50	1973.03	1973.50	804.79	1889.73	1361.48	1896.74	1359.89
MDMA6	1973.03	1973.50	1973.03	1973.50	807.89	1888.43	1361.93	1895.42	1394.15
MDMA7	1985.46	3914.00	3908.40	3912.13	838.06	3021.76	1550.93	3217.60	1517.04

Optimizing Non-32 Byte-Aligned MDMA Transfers

In many cases, the start address and count of a MDMA transfer may not be aligned to a 32-byte address boundary. In such cases, the `MSIZE` value may need to be configured to less than 32 bytes. This configuration can affect the MDMA performance. One option to get better throughput for such cases is to split the single MDMA transfer into more than one transfer using a descriptor-based DMA. The first and last (if needed) MDMA transfers can use `MSIZE` < 32 bytes for non-32 byte-aligned address and count values. The second transfer can use `MSIZE` = 32 bytes for 32-byte-aligned address and count values.

The MDMA service available with CCES provides an additional API called `adi_mdma_Copy1DAuto`. It is compatible with the standard 1D-transfer API `adi_mdma_Copy1D` that is used for single-shot 1D transfers. In Table 4, the MDMA performance of `adi_mdma_Copy1DAuto` is approximately 1.25 to 2.16 times better than `adi_mdma_Copy1D` for non 32-byte-aligned start addresses. The example source code found in the `Test2_MDMA_1DAuto` subfolder of the Zip file with the EE445 note, can be used to measure MDMA performance for both APIs using the ADSP-SC598 processor.

Table 4 adi__mdma_Copy1D vs. adi_mdma_Copy1DAuto Performance on ADSP-SC598 Processor

S. No.	Source Memory Address	Destination Memory Address	DMA Count	MSIZE (Bytes)	Copy1D MDMA Cycles	Copy1DAuto MDMA Cycles	Additional API Overhead	Effective Improvement Factor
1	0x2C0001	0x300000	256	1	1800	1330	114	1.25
2	0x2C0000	0x300001	256	1	1810	1251	110	1.33
3	0x2C0001	0x300000	1024	1	5362	2866	92	1.81
4	0x2C0000	0x300001	1024	1	5352	2787	110	1.85
5	0x2C0001	0x300000	4096	1	19560	9010	92	2.15
6	0x2C0000	0x300001	4096	1	19570	8931	110	2.16

Bandwidth Limiting and Monitoring

MDMAs are equipped with a bandwidth limit and monitor mechanism. The bandwidth limit feature can be used to reduce the number of DMA requests being sent by the corresponding requesters to the SCB.

The `DMA_BWLCNT` register can be programmed to configure the number of SYSCLK cycles between two DMA requests. This configuration can be used to ensure that such DMA channels' requests do not occur more frequently than required. Programming a value of `0x0000` allows the DMA to request as often as possible. A value of `0xFFFF` represents a special case and causes all requests to stop. The maximum throughput, in MB/s, is determined by the `DMA_BWLCNT` register and the `MSIZE` value and is calculated as follows:

*Bandwidth = min (SYSCLK frequency in MHz*DMA bus width in bytes,*
*SYSCLK frequency in MHz*MSIZE in bytes / DMA_BWLCNT)*

The API `adi_mdma_BWLimit` can be used to program the `DMA_BWLCNT` register for a target bandwidth and `MSIZE` value. The example source code found in the `Test3_MDMA_BWLimit` subfolder of the Zip file with the EE445 Note shows how to use this API. Figure 6 shows a sample result of this code with the target and measured bandwidth for different MDMA use cases.

Figure 6 MDMA Bandwidth Limit Results on ADSP-SC598 Processor

```
Testing combination 1
MDMA Stream no = 0
MSIZE value = 32
Source channel = 1
Target BW = 400.000000 MB/s

Measured BW = 389.835350 MB/s

Test combination 1 passed


Testing combination 2
MDMA Stream no = 1
MSIZE value = 32
Source channel = 1
Target BW = 300.000000 MB/s

Measured BW = 296.060700 MB/s

Test combination 2 passed


Testing combination 3
MDMA Stream no = 2
MSIZE value = 32
Source channel = 1
Target BW = 700.000000 MB/s

Measured BW = 693.825750 MB/s

Test combination 3 passed


Testing combination 4
MDMA Stream no = 3
MSIZE value = 32
Source channel = 1
Target BW = 600.000000 MB/s

Measured BW = 591.864750 MB/s

Test combination 4 passed
```

The bandwidth monitor feature can be used to check if such channels are starving for resources. The `DMA_BMCNT` register can be programmed to the number of SYSCLK cycles within which the corresponding DMA should finish. Each time the `DMA_CFG` register is written (MMR access only), a work unit ends, or an autobuffer wraps, the DMA loads the value in the `DMA_BWMCNT` register into the `DMA_BWMCNT_CUR` register. The DMA decrements `DMA_BWMCNT_CUR` every SYSCLK that a work unit is active. When the `DMA_BWMCNT_CUR` value reaches `0x00000000` before the work unit finishes, the `DMA_STAT.IRQERR` bit is set, and the `DMA_STAT.ERRC` bit is set to `0x6`. The `DMA_BWMCNT_CUR` value remains at `0x00000000` until it is reloaded when the work unit completes. Unlike other error sources, a bandwidth monitor error does not stop work unit processing. Programming `0x00000000` disables bandwidth monitor functionality. This feature can also be used to measure the actual throughput.

The API `adi_mdma_BWMonitor` can be used to program the `DMA_BWMCNT` register for a given target bandwidth and `MSIZE` value. The example source code found in the `Test4_MDMA_BWMonitor` subfolder of the Zip file with the EE445 Note, shows how to use this API. Figure 7 shows a sample result of this code with a target bandwidth and bandwidth monitor expiration message for a given MDMA use case. The API `adi_mdma_BWMeasure` uses the `DMA_BMCNT` and `DMA_BWMCNT_CUR` registers to measure the MDMA bandwidth as shown in the example code `MDMA_BWLimit`.

Figure 7 MDMA Bandwidth Monitor Results on ADSP-SC598 Processor

```
Testing combination 1
MDMA Stream no = 0
MSIZE value = 32
Source channel = 1
Target BW = 400.000000 MB/s
BW Monitor Expired!! Test combination 1 passed


Testing combination 2
MDMA Stream no = 1
MSIZE value = 32
Source channel = 1
Target BW = 300.000000 MB/s
BW Monitor Expired!! Test combination 2 passed


Testing combination 3
MDMA Stream no = 2
MSIZE value = 32
Source channel = 1
Target BW = 700.000000 MB/s
BW Monitor Expired!! Test combination 3 passed


Testing combination 4
MDMA Stream no = 3
MSIZE value = 32
Source channel = 1
Target BW = 600.000000 MB/s
BW Monitor Expired!! Test combination 4 passed
```

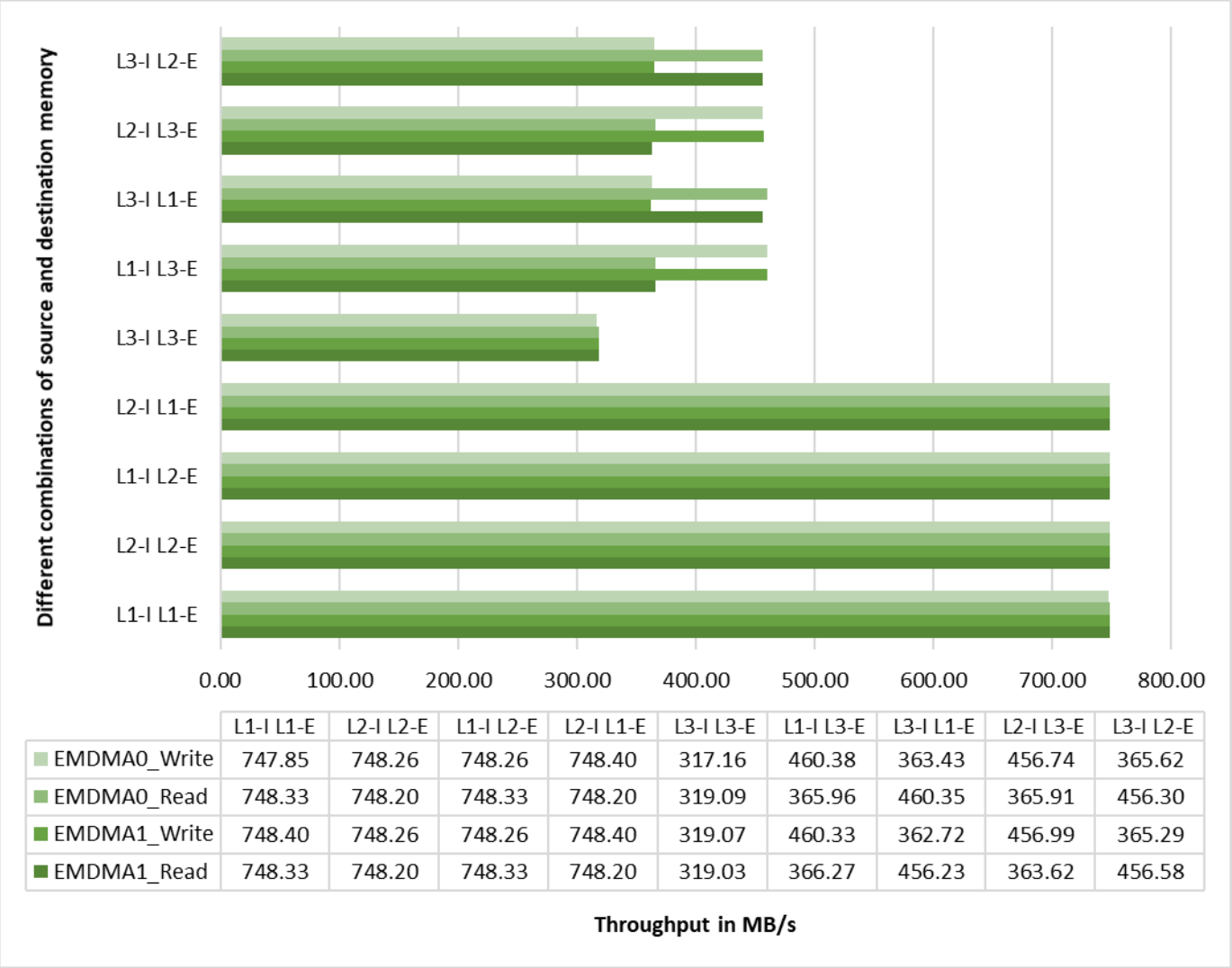
Extended Memory DMA (EMDMA)

The ADSP-SC59x processors also support Extended Memory DMA (EMDMA). The EMDMA engine is mainly used to transfer the data from one memory type to another in a non-sequential manner (such as circular, delay line, and scatter/gather). For details regarding EMDMA, refer to the *ADSP-2159x/ADSP-SC591/SC592/SC594 SHARC+ Processor Hardware Reference* [3] *or ADSP-SC595/SC596/SC598 SHARC+ Processor Hardware Reference* [4]. The EMDMA on the ADSP-SC59x processors has been enhanced to run at the SYSCLK speed instead of the SCLK speed. This enhancement results in improved EMDMA throughput.

Figure 8 shows throughput measured on the bench for EMDMA0/EMDMA1 streams with different combinations of source and destination memories for sequential transfers of 4096 32-bit words at CCLK = 1 GHz, SYSCLK = 500 MHz, and DCLK = 900 MHz .

The source code found in the `Test5_EMDMA_Throughput` subfolder of the Zip file supplied with the application note[5], can be used to measure EMDMA throughput.

Figure 8 Measured EMDMA Throughput on ADSP-SC598 Processors



	L1-I L1-E	L2-I L2-E	L1-I L2-E	L2-I L1-E	L3-I L3-E	L1-I L3-E	L3-I L1-E	L2-I L3-E	L3-I L2-E
EMDMA0_Write	747.85	748.26	748.26	748.40	317.16	460.38	363.43	456.74	365.62
EMDMA0_Read	748.33	748.20	748.33	748.20	319.09	365.96	460.35	365.91	456.30
EMDMA1_Write	748.40	748.26	748.26	748.40	319.07	460.33	362.72	456.99	365.29
EMDMA1_Read	748.33	748.20	748.33	748.20	319.03	366.27	456.23	363.62	456.58

Optimizing Non-Sequential EMDMA Transfers with MDMA

In some cases, the non-sequential transfer modes supported by EMDMA can be replaced by descriptor-based MDMA for better performance.

The example source code found in the `Test6_MDMA_Circular_Buffer` subfolder of the Zip file with the EE445 Note, illustrates how a MDMA descriptor-based mode can be used to emulate a circular buffer memory-to-memory DMA transfer mode.

The example code compares the core cycles measured (Table 5) to write and read 4096 32-bit words to and from the DDR3 memory in circular buffer mode. The example uses a starting address offset of 1024 words for these situations:

- EMDMA
- MDMA with MSIZE = 4 bytes (for 4-byte-aligned address and count)
- MDMA with MSIZE = 32 bytes (for 32-byte-aligned address and count)

Table 5 MDMA Emulated Circular Buffer vs. EMDMA on ADSP-SC598 Processor

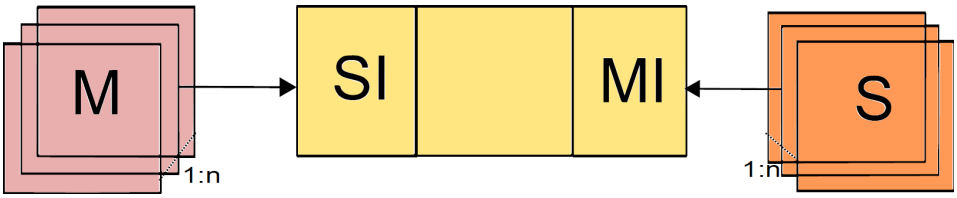
Write/Read	Core Cycles		
	EMDMA	MDMA3 MSIZE = 4 bytes	MDMA3 MSIZE = 32 bytes
Write	36308	30137	6259
Read	47642	36312	11266

As shown in Table 5, the MDMA emulated circular buffer (`MSIZE` = 4 bytes) is faster than EMDMA. The performance is further improved with `MSIZE` = 32 bytes when the addresses and counts are 32-byte aligned.

Understanding the System Crossbars

As shown in Figure 2 on page 3 and Figure 4 on page 5, the SCB interconnect consists of a hierarchical model connecting multiple SCB units. Figure 9 shows the block diagram for a single SCB unit. It connects the System Bus Requesters (M) to the System Bus Completers (S) by using a Completer Interface (SI) and Requester Interface (MI). On each SCB unit, each S is connected to a fixed MI. Similarly, each M is connected to a fixed SI.

Figure 9 Single SCB Block Diagram



The completer interface of the crossbar (where requesters such as DDE are connected) perform these two functions, arbitration and clock domain conversion.

Arbitration

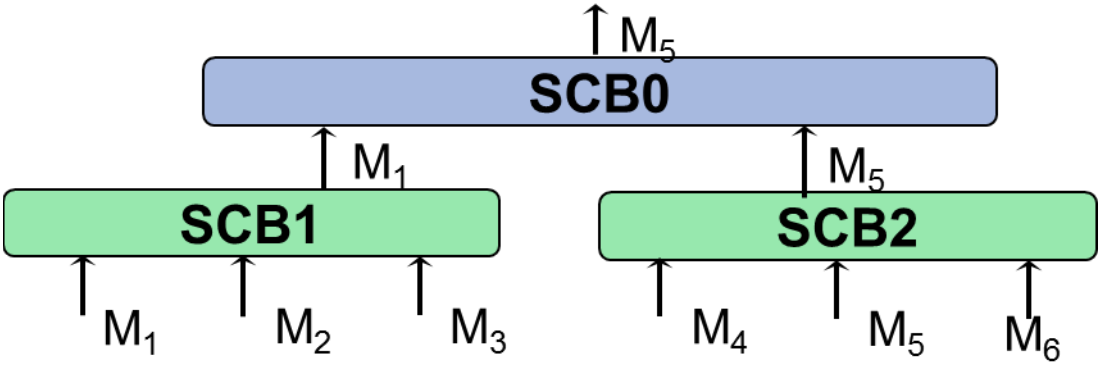
The programmable Quality of Service (QoS) registers can be viewed as being associated with SCBx. For example, the programmable QoS registers for SPORT0-3 and MDMA0 can be viewed as residing in SCB1. Whenever a transaction is received at SPORT0 half A, the programmed QoS value is associated with that transaction and is arbitrated with the rest of the requesters at SCB1.

Programming the SCB QOS Registers

Consider a scenario where:

- At SCB1, requesters 1, 2, and 3 have RQOS values of 6, 4, and 2, respectively.
- At SCB2, requesters 4, 5, and 6 have RQOS values of 12, 13, and 1, respectively.

Figure 10 Arbitration Among Various Requesters



As shown in Figure 10, in this case:

- Requester 1 wins the arbitration at SCB1, and Requester 5 wins the arbitration at SCB2.
- In a perfect competition at SCB0, however, requesters 4 and 5 had the highest overall RQOS values. So, the requesters would have fought for arbitration directly at SCB0. Because of the mini-SCBs, however, Requester 1, at a much lower RQOS value, wins against Requester 4 and makes it all the way to SCB0.

Clock Domain Conversion

Table 6 and Table 7 shows the various clock domain crossings that are available on the ADSP-SC59x processors. The clocking relationships are defined with respect to SYSCLK. The CLKO3 to SYSCLK, CLKO4 to SYSCLK and CLKO8 to SYSCLK clock domains are programmable in the ADSP-SC594 processors. The CLKO2 to SYSCLK, CLKO3 to SYSCLK, CLKO4 to SYSCLK and CLKO8 to SYSCLK clock domains are programmable in the ADSP-SC598 processors. These should be programmed depending upon the clock ratios of the two clock domains for optimum performance.

Table 6 Clock Domain Crossing Options for ADSP-SC594 processors

		Clocking Relationships with respect to SYSCLK					
Functional Clocks	**Possible Targets**	**System Fabric**	**MMRG Fabric**	**DMC Fabric**	**SHARC Fabric**	**ARM Fabric**	**SPIF Fabric**
SYSCLK	System Fabric and Infrastructure Modules	1:1	1:1	*1	*	*	*
SCLK0	SCLK0 Clock Domain	Synch (m:1)	Synch (m:1)	*	*	*	*
CLKO0	SHARC0 and its Accelerators	*	*	*	Synch (m:1)	*	*
CLKO1	SHARC1 and its Accelerators	*	*	*	Synch (m:1)	*	*
CLKO2	ARM	*	*	*	*	Synch (m:1)	
CLKO3	DMC	*	Prog	Prog	*	*	*
CLKO4	CAN	*	Prog	*	*	*	*
CLKO6	SPI	Synch (m:n) (except SPIF slave)	Synch (m:n)	*	*	*	Synch (m:n)
CLKO8	LP	Prog	Prog	*	*	*	*

*1 * Either the interface is not present, or CDC is not required

Prog designates programmable

Synch (m:1) designates Synchronous (m:1)

Synch (m:n) designates Synchronous (m:n)

Table 7 Clock Domain Crossing Options for ADSP-SC598 processors

		Clocking Relationships with respect to SYSCLK					
Functional Clocks	**Possible Targets**	**System Fabric**	**MMRG Fabric**	**DMC Fabric**	**SHARC Fabric**	**ARM Fabric**	**SPIF Fabric**
SYSCLK	System Fabric and Infrastructure Modules	1:1	1:1	*1	*	*	*
SCLK0	SCLK0 Clock Domain	Synch (m:1)	Synch (m:1)	*	*	*	*
CLKO0	SHARC0 and its Accelerators	*	*	*	Synch (m:1)	*	*
CLKO1	SHARC1 and its Accelerators	*	*	*	Synch (m:1)	*	*
CLKO2	ARM	*	*	*	*	Prog	
CLKO3	DMC	*	Prog	Prog	*	*	*
CLKO4	CAN	*	Prog	*	*	*	*
CLKO6	SPI	Synch (m:n) (except SPIF slave)	Synch (m:n)	*	*	*	Synch (m:n)
CLKO8	LP	Prog	Prog	*	*	*	*

*1 * Either the interface is not present, or CDC is not required

Prog designates programmable

Synch (m:1) designates Synchronous (m:1)

Synch (m:n) designates Synchronous (m:n)

Programming Sync Mode in the IBx Registers

Programming the SYNC mode in the `IBx` registers improves the DMC MDMA throughput. For most of the work unit size values the throughput is better when the CDC is programmed in SYNC mode as compared to the ASYNC mode.

Understanding the System Completers

Memory Hierarchy

As shown in Table 2 on page 8 and Table 3 on page 8. ADSP-SC59x processors have a hierarchical memory model (L1/L2/L3). The following sections discuss the access latencies and achievable throughput associated with the different memory levels.

L1 Memory Throughput

L1 memory runs at CCLK and is the fastest accessible memory in the hierarchy. SHARC+ L1 memory is accessible by both the core and DMA (system). For system (DMA) accesses, L1 memory supports two ports: the S1 port and the S2 port. Two different banks of L1 memory can be accessed in parallel with these ports. Like the ADSP-2156x processors, the system (DMA) accesses are hardwired to the L1 memory of the SHARC+ processer using the S1 port. One exception is the High-Speed MDMA (HSMDMA or MDMA3 and MDMA7), which are hardwired using the S2 port. From a programming perspective, when accessing the L1 memory of the SHARC+ processer, use a multiprocessor memory offset of 0x28000000 for all DMA accesses (including HSMDMA).

The maximum theoretical throughput of L1 memory (for system/DMA accesses) is 1000 * 4 = 4000 MB/s for 1 GHz CCLK operation. As shown in Figure 5 on page 9, the maximum measured L1 throughput using MDMA3 is approximately 3912 MB/s.

L2 Memory Throughput

L2 memory access times are longer than L1 memory because the maximum L2 clock frequency (SYSCLK) is half the CCLK. The L2 memory controller contains five ports to connect to the system crossbar. Port 0, port 1, and port 2 are 128-bit interfaces dedicated to core and MDMA3 traffic. Port 3 and port 4 are 128-bit interfaces that connect through DMA. Each port has a read and a write channel.

To ensure complete bandwidth utilization and optimal performance of the L2 ports, the controllers are allocated as shown in Table 8.

Table 8 L2 Port Requester Allocation

Requesters	CL2_0	CL2_1	CL2_2	DL2_0	DL2_1
SHARC0 (Core1)	✓	✓			
SHARC1 (Core2)	✓	✓			
ARM (Core0)	✓	✓			
SHARC0 IPORT			✓		
SHARC1 IPORT			✓		
SHARC0_FIR_CH0			✓		
SHARC0_FIR_CH1			✓		
SHARC0_IIR_CH0	✓	✓			
SHARC0_IIR_CH1	✓	✓			
SHARC1_FIR_CH0			✓		
SHARC1_FIR_CH1			✓		
SHARC1_IIR_CH0	✓	✓			
SHARC1_IIR_CH1	✓	✓			
HSMDMA0				✓	
HSMDMA1				✓	
Peripherals					✓

Note the following about the L2 requestor ports:

- To reduce the probability of port conflict in a multicore system, the cores are connected on the CL2_0 and CL2_1 port of L2. CL_0 has access only to the first 1 MB of L2 memory. CL2_1 has access to second 1 MB of L2 memory.
- To avoid conflict during multicore boot, the booting of the Arm® core happens through the CL2_0 port and the booting of the SHARC cores happens through the CL2_1 port. The respective bootROM addresses are routed accordingly.

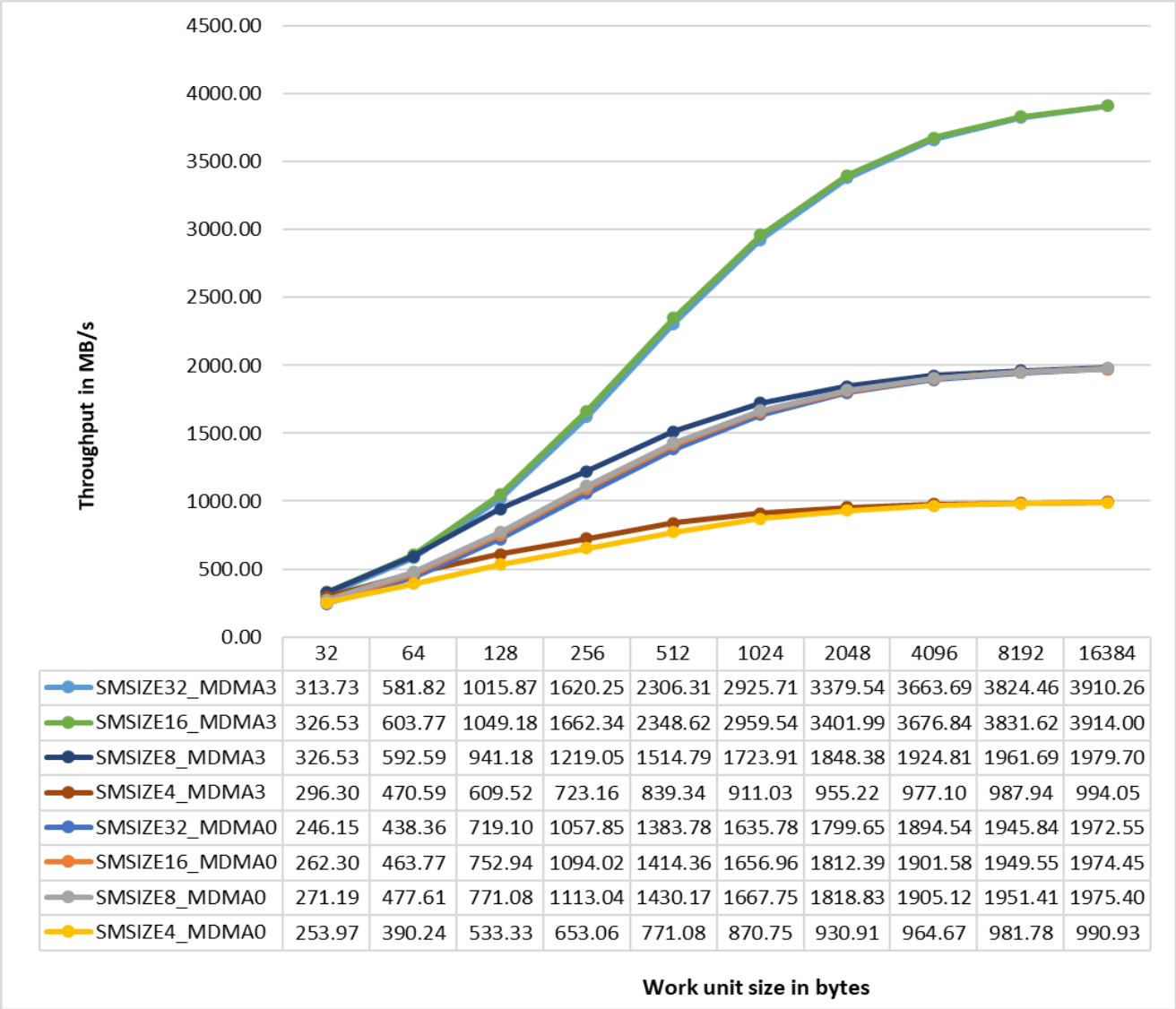
For additional details, refer to the *ADSP-2159x/ADSP-SC591/SC592/SC594 SHARC+ Processor Hardware Reference [3] or ADSP- SC595/SC596/SC598 SHARC+ Processor Hardware Reference [4]*
.

Consider the following important points regarding L2 memory throughput:

- Because L2 memory runs at the SYSCLK speed, it can provide a maximum theoretical throughput of 500 MHz * 16 = 8000 MB/s in one direction (for HSMDMA accesses, it can reach 4000 MB/s). Because separate read and write channels exist, the total throughput in both directions equals 16000 MB/s. To operate L2 SRAM memory at its optimum throughput, use both the core and DMA ports and separate read and write channels in parallel. All of them should access different banks of L2.
- All accesses to L2 memory are converted to 64-bit accesses (8-byte) by the L2 memory controller. To achieve optimum throughput for DMA access to L2 memory, configure the DMA channel `MSIZE` to 8 bytes or more.
- Unlike L3 (DMC) memory accesses, L2 memory throughput for sequential and non-sequential accesses is the same.
- L2 SRAM is ECC-protected and organized into eight banks. A single 8- or 16-bit access, or a non-32-bit address-aligned 8-bit or 16-bit burst access, to an ECC-enabled bank creates an additional latency of two SYSCLK cycles. This latency is due to the ECC implementation. The implementation is in terms of 32-bit accesses. Any write that is less than 32-bit to an ECC-enabled SRAM bank is implemented as a read-followed-by-write and requires three cycles to complete (two cycles for the read, one cycle for the write).
- When performing simultaneous core and DMA accesses to the same L2 memory bank, read and write priority control registers can be used to increase DMA throughput. When both the core and the DMA engine access the same bank, the best access rate that DMA can achieve is one 64-bit access every three SYSCLK cycles during the conflict period. This throughput is achieved by programming the read and write priority count bits (`L2CTL_RPCR.RPC0` and `L2CTL_WPCR.WPC0`) to zero, while programming the `L2CTL_RPCR.RPC1` and `L2CTL_WPCR.WPC1` bits to one.

Figure 11 shows the measured MDMA throughput at CCLK = 1 GHz and SYSCLK = 500 MHz for a case where both source and destination buffers are in different L2 memory banks. As an example, for MDMA3, the maximum throughput is very close to 3910 MB/sec in one direction (7820 MB/s in both directions) for `MSIZE` = 32 bytes and drops significantly for smaller `MSIZE` values.

Figure 11 L2 MDMA Throughput for Different MSIZE and Work Unit Sizes on an ADSP-SC598 Processor



	32	64	128	256	512	1024	2048	4096	8192	16384
SMSIZE32_MDMA3	313.73	581.82	1015.87	1620.25	2306.31	2925.71	3379.54	3663.69	3824.46	3910.26
SMSIZE16_MDMA3	326.53	603.77	1049.18	1662.34	2348.62	2959.54	3401.99	3676.84	3831.62	3914.00
SMSIZE8_MDMA3	326.53	592.59	941.18	1219.05	1514.79	1723.91	1848.38	1924.81	1961.69	1979.70
SMSIZE4_MDMA3	296.30	470.59	609.52	723.16	839.34	911.03	955.22	977.10	987.94	994.05
SMSIZE32_MDMA0	246.15	438.36	719.10	1057.85	1383.78	1635.78	1799.65	1894.54	1945.84	1972.55
SMSIZE16_MDMA0	262.30	463.77	752.94	1094.02	1414.36	1656.96	1812.39	1901.58	1949.55	1974.45
SMSIZE8_MDMA0	271.19	477.61	771.08	1113.04	1430.17	1667.75	1818.83	1905.12	1951.41	1975.40
SMSIZE4_MDMA0	253.97	390.24	533.33	653.06	771.08	870.75	930.91	964.67	981.78	990.93

L3 Memory and External Memory Throughput

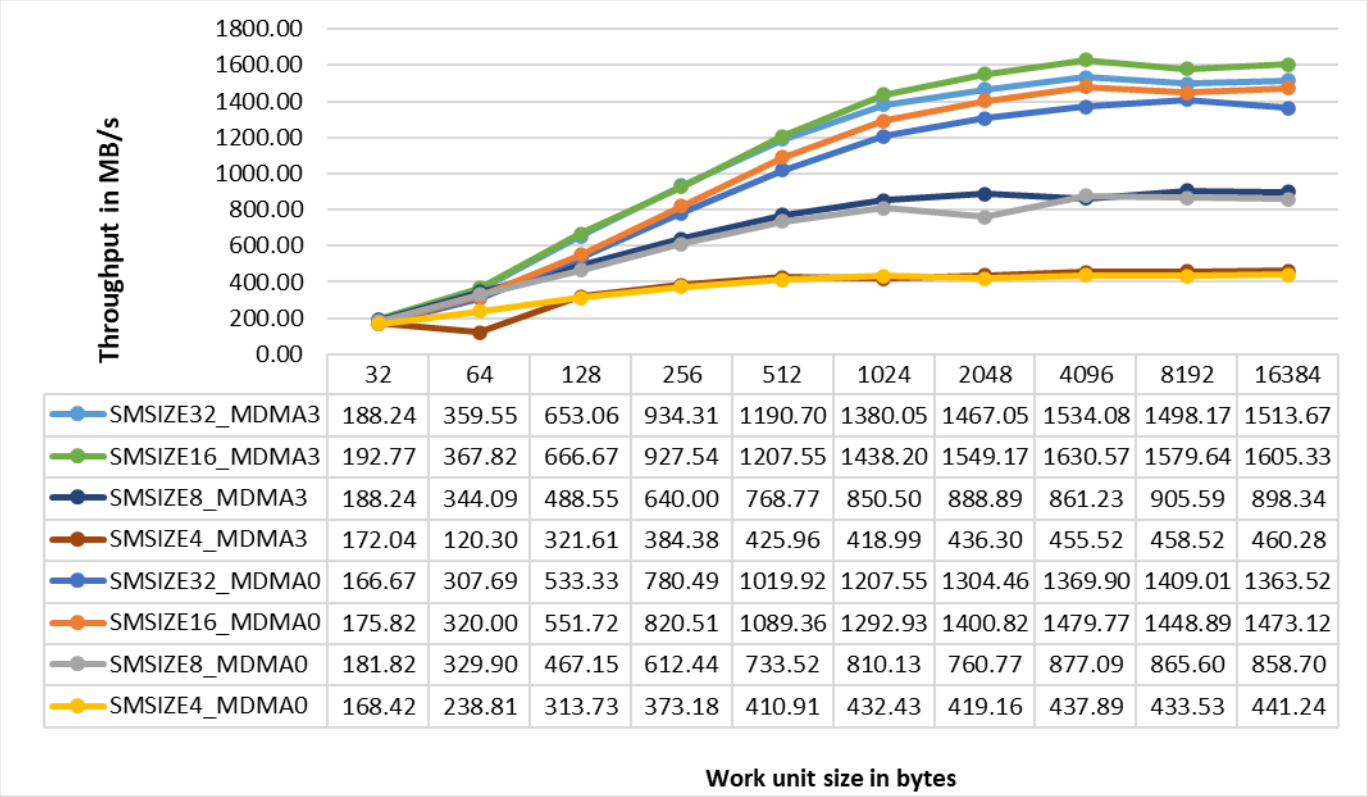
The ADSP-SC59x processors provide interfaces for connecting to DDR3/DDR3L memory devices. The DMC interface operates at speeds of up to 900 MHz. For the 16bit DDR3 interface, the maximum theoretical throughput that the DMC can deliver equals 3600 MB/s. However, the practical maximum DMC throughput is less because of the latencies introduced by the internal system interconnects, as well as the latencies derived from the DRAM technology itself (access patterns, page hit to page miss ratio, and so forth).

Although most of the throughput optimization concepts are illustrated using MDMA as an example, the same can be applied to other system requesters as well.

The MDMA3 stream (HSMDMA) can request DMC accesses faster than any other requesters (for example, 4000 MB/s). The practical DMC throughput possible using MDMA depends upon factors such as whether the accesses are sequential or non-sequential, the block size of the transfer, and DMA parameters (for example, `MSIZE`).

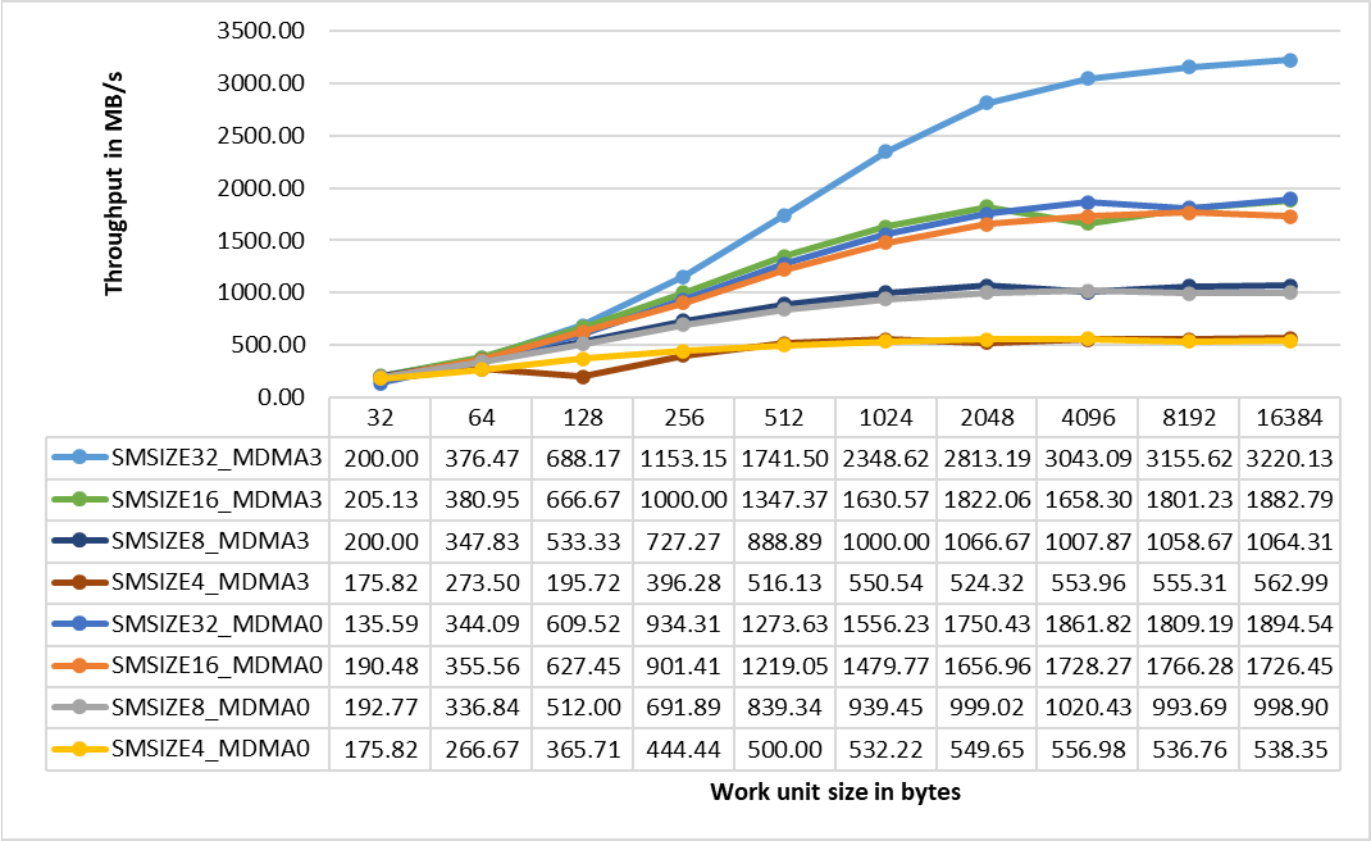
Figure 12 and Figure 13 provide the DMC measured throughput at CCLK = 1 GHz and DCLK = 900 MHz using MDMA0 (channels 8 and 9) and MDMA3 (channels 43 and44) streams for various MSIZE values and buffer sizes.

Figure 12 DMC Measured Throughput for Sequential MDMA Reads on an ADSP-SC598 Processor



	32	64	128	256	512	1024	2048	4096	8192	16384
SMSIZE32_MDMA3	188.24	359.55	653.06	934.31	1190.70	1380.05	1467.05	1534.08	1498.17	1513.67
SMSIZE16_MDMA3	192.77	367.82	666.67	927.54	1207.55	1438.20	1549.17	1630.57	1579.64	1605.33
SMSIZE8_MDMA3	188.24	344.09	488.55	640.00	768.77	850.50	888.89	861.23	905.59	898.34
SMSIZE4_MDMA3	172.04	120.30	321.61	384.38	425.96	418.99	436.30	455.52	458.52	460.28
SMSIZE32_MDMA0	166.67	307.69	533.33	780.49	1019.92	1207.55	1304.46	1369.90	1409.01	1363.52
SMSIZE16_MDMA0	175.82	320.00	551.72	820.51	1089.36	1292.93	1400.82	1479.77	1448.89	1473.12
SMSIZE8_MDMA0	181.82	329.90	467.15	612.44	733.52	810.13	760.77	877.09	865.60	858.70
SMSIZE4_MDMA0	168.42	238.81	313.73	373.18	410.91	432.43	419.16	437.89	433.53	441.24

Figure 13 DMC Measured Throughput for Sequential MDMA Writes on an ADSP-SC598 Processor



	32	64	128	256	512	1024	2048	4096	8192	16384
SMSIZE32_MDMA3	200.00	376.47	688.17	1153.15	1741.50	2348.62	2813.19	3043.09	3155.62	3220.13
SMSIZE16_MDMA3	205.13	380.95	666.67	1000.00	1347.37	1630.57	1822.06	1658.30	1801.23	1882.79
SMSIZE8_MDMA3	200.00	347.83	533.33	727.27	888.89	1000.00	1066.67	1007.87	1058.67	1064.31
SMSIZE4_MDMA3	175.82	273.50	195.72	396.28	516.13	550.54	524.32	553.96	555.31	562.99
SMSIZE32_MDMA0	135.59	344.09	609.52	934.31	1273.63	1556.23	1750.43	1861.82	1809.19	1894.54
SMSIZE16_MDMA0	190.48	355.56	627.45	901.41	1219.05	1479.77	1656.96	1728.27	1766.28	1726.45
SMSIZE8_MDMA0	192.77	336.84	512.00	691.89	839.34	939.45	999.02	1020.43	993.69	998.90
SMSIZE4_MDMA0	175.82	266.67	365.71	444.44	500.00	532.22	549.65	556.98	536.76	538.35

The following important observations can be made for the ADSP-SC598 processor:

- The throughput trends are similar for reads and writes with regards to `MSIZE` and buffer size values.
- The peak measured read throughput is 1630 MB/s for MDMA3 with `MSIZE` = 16 bytes. The peak measured write throughput is 3220 MB/s for MDMA3 with `MSIZE` = 32 bytes.
- The throughput depends largely upon the DMA buffer size. For smaller buffer sizes, the throughput is significantly lower. The throughput increases significantly with a larger buffer size. For example, for MDMA3 with `MSIZE` = 32 bytes and a buffer size of 32 bytes, the read throughput is 188 MB/s, whereas it reaches 1514 MB/s for a 16 KB buffer size. This difference is affected by the overhead incurred when programming the DMA registers, as well as the system latencies when sending the initial request from the DMA engine to the DMC controller.

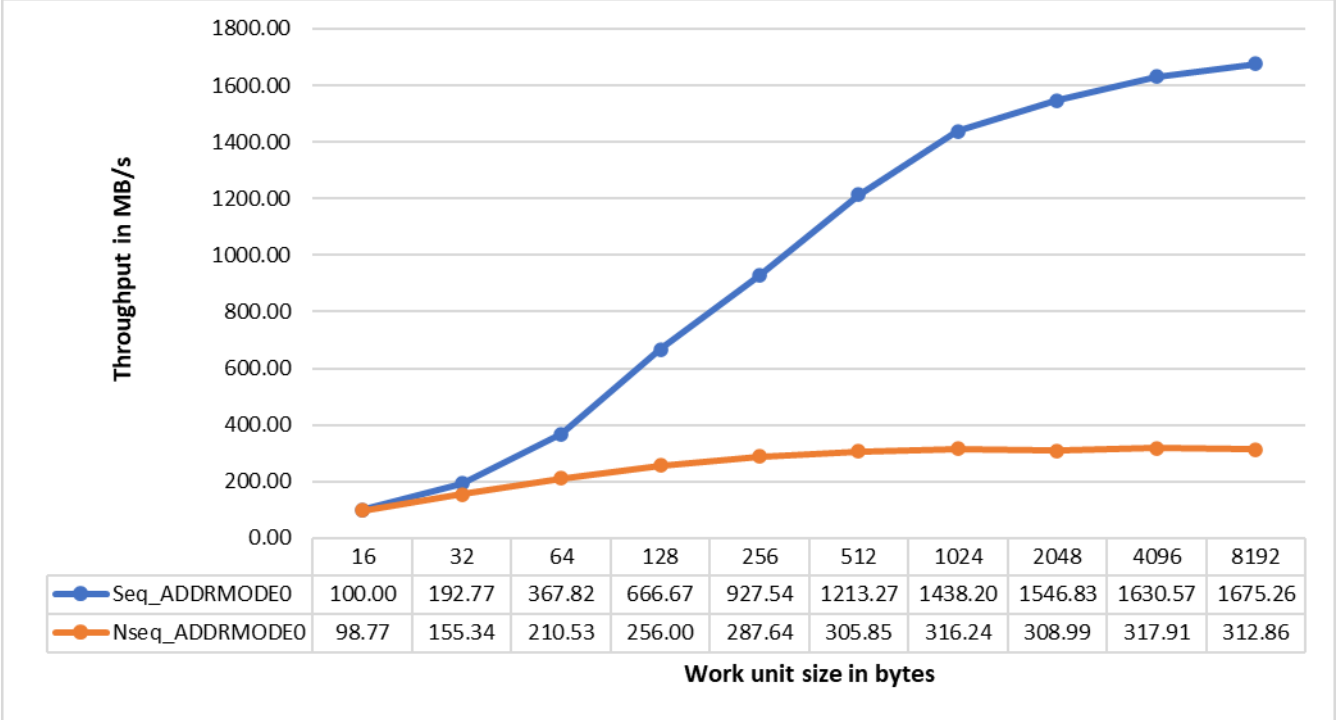
Try to rearrange the DMC accesses such that the DMA count is as large as possible. Better sustained throughput is obtained for continuous transfers over time.

- To some extent, throughput also depends upon the MSIZE value of the source MDMA channel for reads and destination MDMA channel for writes. For Figure 12 and Figure 13 , in most cases, greater MSIZE values provide better results. Ideally, the MSIZE value should be at least equal to the DDR memory burst length. For MDMA3 with a buffer size of 16384 bytes, the read throughput is 1514 MB/s for MSIZE = 32 bytes, while it reduces significantly to 460 MB/s for MSIZE = 4 bytes. For MSIZE = 4 bytes, although all accesses are still sequential, the full DDR3 memory burst length of 16 bytes (eight 16-bit words) is not used.

For sequential reads, it is easy to achieve optimum throughput, particularly for larger buffer sizes. The DRAM memory page hit ratio is high, and the DMC controller does not need to close and open DDR device rows frequently. However, in case of non-sequential accesses, throughput can drop slightly or significantly depending upon the page hit-to-miss ratio.

Figure 14 provides a comparison of the DMC throughput numbers measured for sequential MDMA read accesses for MSIZE = 16 bytes (equals DDR3 burst length) and ADDRMODE set to 0 (bank interleaving) versus non-sequential accesses with a modifier of 2048 bytes (equals DDR3 page size, thus leading to a worst-case scenario with maximum possible page misses). As shown, for a buffer size of 8192 bytes, throughput drops significantly from 1675 to 312 MB/s.

Figure 14 DMC Throughput for Sequential vs. Non-Sequential Read Accesses on an ADSP-SC598 Processor



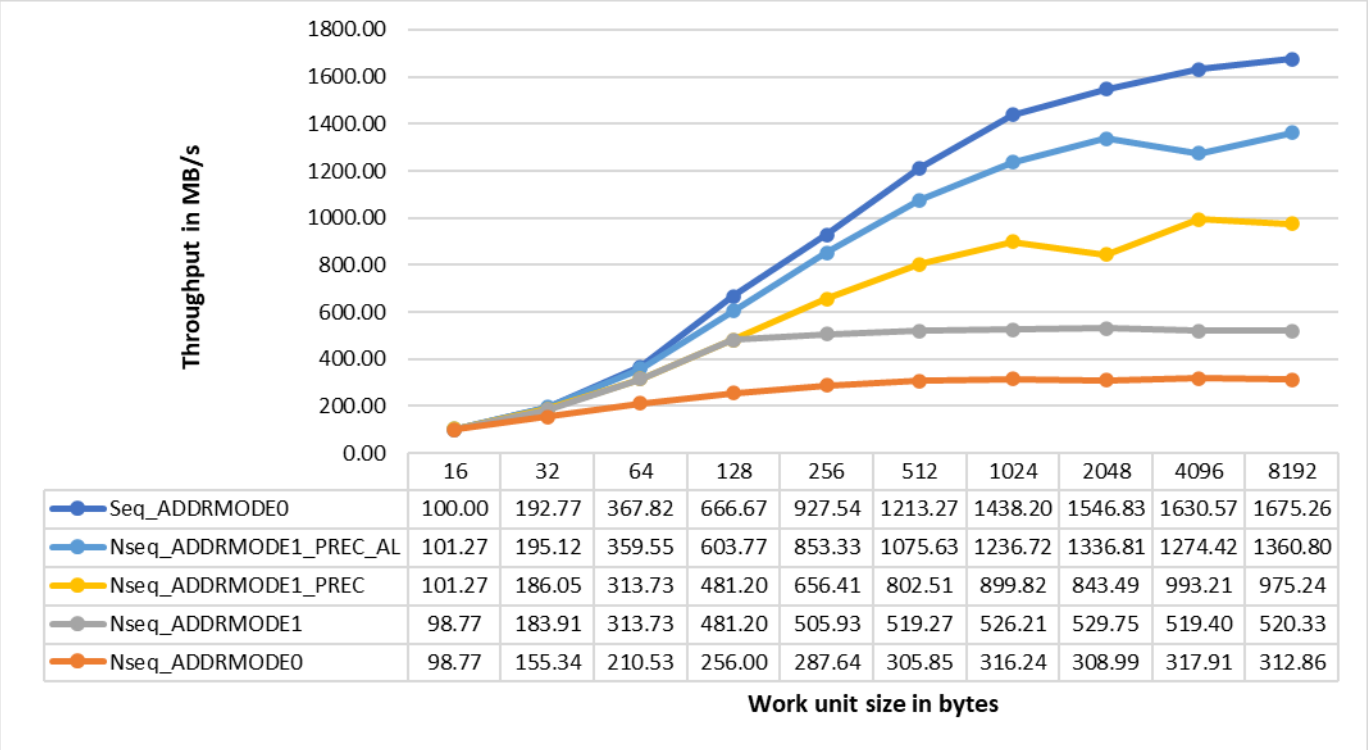
	16	32	64	128	256	512	1024	2048	4096	8192
Seq_ADDRMODE0	100.00	192.77	367.82	666.67	927.54	1213.27	1438.20	1546.83	1630.57	1675.26
Nseq_ADDRMODE0	98.77	155.34	210.53	256.00	287.64	305.85	316.24	308.99	317.91	312.86

DDR memory devices support concurrent bank operations that provide the DMC controller the feature to activate a row in another bank without pre-charging the row of a bank. This feature is extremely helpful in cases where DDR access patterns incur page misses. By setting the `DMC_CTL.ADDRMODE` bit, throughput can be improved by ensuring that such accesses fall into different banks. Figure 15 shows in gray color how the DMC throughput increases from 312.86 MB/s to 520.33 MB/s by setting this bit for the non-sequential access pattern shown in Figure 14.

The throughput can be further improved using the `DMC_CTL.PREC` bit, which forces the DMC to close the row automatically as soon as a DDR read burst is complete with the help of the `Read with Auto Precharge` command. This configuration allows the row of a bank to proactively pre-charge after it has been accessed. It improves the throughput by saving the latency involved in pre-charging the row at the time when the next row of the same bank must be activated.

The yellow line in Figure 15 shows the increase in throughput. Setting the `DMC_CTL.PREC` bit results in an increase from 520 MB/s to 975 MB/s. The same result can be achieved by setting the `DMC_EFFCTL.PRECBANK[7-0]` bits. This feature can be used on a per bank basis. However, note that setting the `DMC_CTL.PREC` bit overrides the `DMC_EFFCTL_PRECBANK[7-0]` bits. Also, setting the `DMC_CTL.PREC` bit results in pre-charging of the rows after every read burst, while setting the `DMC_EFFCTL_PRECBANK[7-0]` bits pre-charge the row after the last burst corresponding to the respective `MSIZE` settings. This configuration can provide an added throughput advantage for cases where `MSIZE` (for example, 32 bytes) is greater than the DDR3 burst length (16 bytes).

Figure 15 Optimizing Throughput for Non-Sequential Accesses on an ADSP-SC598 Processor



	16	32	64	128	256	512	1024	2048	4096	8192
Seq_ADDRMODE0	100.00	192.77	367.82	666.67	927.54	1213.27	1438.20	1546.83	1630.57	1675.26
Nseq_ADDRMODE1_PREC_AL	101.27	195.12	359.55	603.77	853.33	1075.63	1236.72	1336.81	1274.42	1360.80
Nseq_ADDRMODE1_PREC	101.27	186.05	313.73	481.20	656.41	802.51	899.82	843.49	993.21	975.24
Nseq_ADDRMODE1	98.77	183.91	313.73	481.20	505.93	519.27	526.21	529.75	519.40	520.33
Nseq_ADDRMODE0	98.77	155.34	210.53	256.00	287.64	305.85	316.24	308.99	317.91	312.86

For reads, the throughput can be further improved by using the additive latency feature. Figure 16 and Figure 17 illustrate how this feature helps to avoid gaps in a burst of data when accessing different banks for CAS Latency = 4 and Additive Latency= 3. Note that these figures were used to explain the concept from the reference TN4702[7] which refers to DDR2. ADSP-SC59x processors support DDR3/DDR3L devices only, but the same concept applies here.

Figure 16 Reads Without Additive Latency

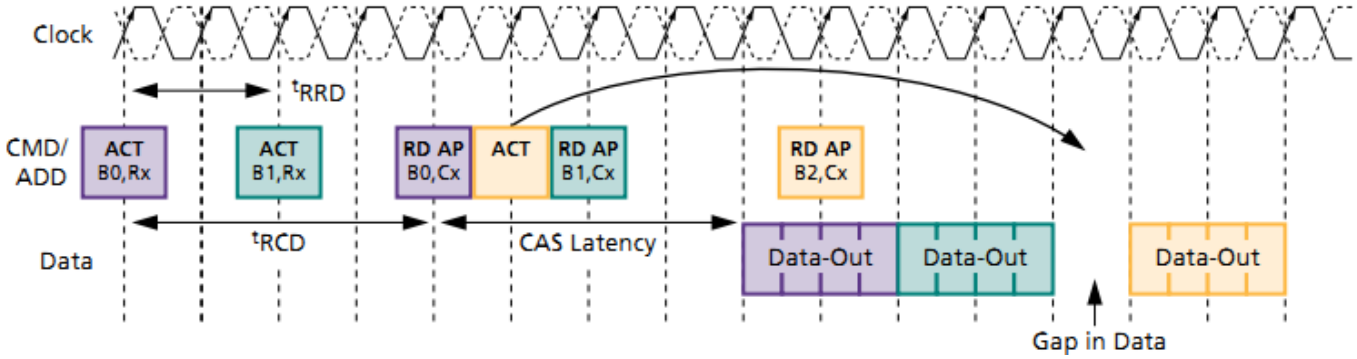
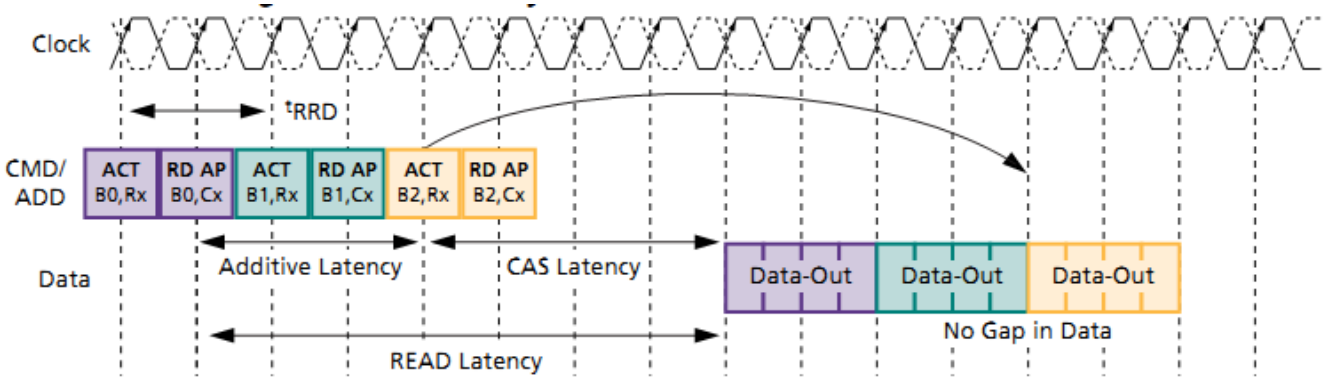


Figure 17 Reads with Additive Latency



Programming the additive latency to `tRCD-1` helps the DMC to send the `Read with Autoprecharge` command right after the `Activate` command, before `tRCD` completes. This sequence enables the controller to schedule the `Activate` and `Read` commands for other banks, eliminating gaps in the data stream. Figure 15 on page 24 shows how the throughput improves from 975 MB/s to 1360 MB/s by programming the additive latency (`AL`) in the `DMC_EMR1` register to `tRCD-1` (equals 8 or CL-1 in this case).

The DMC also provides elevating the priority of the accesses requested by a SCB Requester using the DMC_PRIO and DMC_PRIOMSK registers. The example source code found in the Test10_DMC_SCB_PRIO subfolder of the associated Zip file with the EE445 Note, describes how two MDMA DMC read channels (8 and 18) run in parallel. Table 9 summarizes the measured throughout. As shown, programming the DMC SCB priority for a MDMA channel results in an increased throughput of the higher priority MDMA when compared with the other MDMA running in parallel.

Table 9 DMC Measured Throughput for Different DMC_PRIO Settings on ADSP-SC598 Processor

Test Case Number	Priority Channel (SCB ID)	MDMA0 (Ch. 8) Throughput (MB/s)	MDMA1 (Ch. 18) Throughput (MB/s)
1	None	892	908
2	MDMA0 (0x0031)	881	769
3	MDMA1 (0x0011)	752	888

The Postpone Autorefresh command can be used to ensure that auto-refreshes do not interfere with any critical data transfers. Up to eight Autorefresh commands can be accumulated in the DMC. The exact number of Autorefresh commands can be programmed using the NUM_REF bit in the DMC_EFFCTL register.

After the first refresh command is accumulated, the DMC constantly looks for an opportunity to schedule a refresh command. When the SCB read and write command buffers become empty (which implies no outstanding accesses) for the programmed number of clock cycles (IDLE_CYCLES) in the DMC_EFFCTL register, the accumulated number of refresh commands are sent sequentially to the DRAM memory.

After every refresh, the SCB command buffers are checked to ensure that they stay empty. However, if the SCB command buffers are always full, once the programmed number of refresh commands accumulates the refresh operation is elevated to urgent priority and one refresh command is sent immediately. After this, the DMC continues to wait for an opportunity to send out refresh commands. If self-refresh is enabled, all pending refresh commands are given out only after the DMC enters self-refresh mode.

System MMR Latencies

Table 10 shows the measured MMR latency in core cycles for different peripherals on the ADSP-SC598 processor. The measurement was taken with CCLK = 1 GHz, SYSCLK = 500 MHz, and SCLK = 250 MHz. These numbers can be used to approximate the MMR access latency of the SHARC+ core for different peripherals.

Table 10 MMR Access Latency for ADSP-SC598 Processor Peripherals in CCLK cycles (approximate)

S. No.	Register	Peripheral	Write Latency (Core Cycles)	Read Latency (Core Cycles)
1	FIR0_INIDX	FIR	41	41
2	IIR0_INIDX	IIR	41	41
3	CRC0_DCNT	CRC	56	56
4	CRC1_DCNT		56	56
5	EMDMA0_INDX1	EMDMA	56	56
6	EMDMA1_INDX1		56	56
7	MEC0_PERR_IMASK0	MEC	56	54
8	MISCREG_PLL2_CONTROL	MISCREG	56	54
9	MLB0_MDAT0	MLB	56	54
10	TAPC_SDBGKEY0	TAPC	56	54
11	TSGENWR0_CNTCR	TSGENWR	56	56
12	CDU0_CLKINSEL	CDU	58	56
13	CGU0_OSCWDCTL	CGU	58	56
14	CSTSGENWR0_CNTCR	CSTSGENWR	58	58
15	DPM0_PER_DIS0	DPM	58	56
16	L2CTL0_RPCR0	L2CTL	58	56
17	RCU0_MSG	RCU	58	56
18	SEC0_RAISE	SEC	58	58
19	SMPU2_RADDR0	SMPU	58	56
20	SPU0_SECUREP10	SPU	58	58
21	SWU1_LA0	SWU	58	56
22	SWU2_LA0		58	56
23	TRU0_SSR0	TRU	58	58
24	PKTE0_SA_ADDR	PKTE	60	60
25	PKA0_APTR	PKA	64	64
26	PKIC0_ACK	PKIC	64	64
27	TRNG0_OUTPUT0	TRNG0	64	64
28	DDRPFB0_CTL0	DDRPFB	68	68
29	DDRPFB0_CTL1		68	68

S. No.	Register	Peripheral	Write Latency (Core Cycles)	Read Latency (Core Cycles)
30	DMC0_PRIO	DMC	70	68
31	EMAC0_MAC_CFG	EMAC	86	96
32	EMAC1_MAC_IEN		86	96
33	OTPC0_PMC_MODE0	OTPC	86	88
34	PINT1_ASSIGN	PINT1	86	88
35	PORTB_DATA_SET	PORTB	86	88
36	UART0_CLK	UART	86	88
37	UART1_CLK		86	88
38	DMA1_XCNT	DMA	88	88
39	EMAC1_MAC_CFG	EMAC	88	96
40	EPPI0_CTL	EPPI	88	88
41	SPI1_CLK	SPI	88	96
42	SPORT0_DIV_A	SPORT	88	96
43	SPORT1_DIV_A		88	96
44	WDOG0_WIN	WDOG	88	88
45	CNT0_CNTR	CNT	90	88
46	EMAC0_MAC_IEN	EMAC	90	96
47	PINT0_ASSIGN	PINT0	90	88
48	PORTA_DATA_SET	PORTA	90	88
49	SPI0_CLK	SPI0	90	96
50	TIMER0_TMR0_WID	TIMER	90	96
51	ASRC0_MUTE	ASRC	92	96
52	DMA0_XCNT	DMA	92	88
53	OSPI0_FCA	OSPI	92	88
54	PADS0_PORTA_PDE	PADS	92	88
55	USBC0_CFG_H	USBC	92	96
56	USBC0_OTG_CTL		92	96
57	WDOG1_WIN	WDOG	92	88
58	HADC0_CHAN_MSK	HADC	94	96
59	EMSI0_CTL1	EMSI	96	88
60	PDM1_CTL0	PDM	96	96
61	SPDIF1_TX_UBUFF_A0	SPDIF	96	96
62	TMU0_FLT_LIM_HI	TMU	96	96
63	DAI0_IMSK_FE	DAI	98	96
64	DAI1_IMSK_FE		98	96
65	PCG0_PW1	PCG	98	96
66	PDM0_CTL0	PDM	98	96
67	SPDIF0_TX_UBUFF_A0	SPDIF	98	96

S. No.	Register	Peripheral	Write Latency (Core Cycles)	Read Latency (Core Cycles)
68	ASRC1_MUTE	ASRC	106	96
69	EMSI0_CMD	EMSI	106	96
70	TWI0_CLKDIV	TWI	128	136
71	TWI1_CLKDIV		132	136
72	LP0_DIV	LP	148	160
73	LP1_DIV		148	160
74	SCB0_SP0A_READ_QOS	SCB	495	496

The MMR latency numbers are measured with the "sync" instruction after the write. This ensures that the write has taken affect. The SHARC+ core supports posted writes, which means that the core does not necessarily wait until the actual write is complete. This helps in avoiding unnecessary core stalls

The MMR access latencies can vary based on the following factors:

- **Clock ratios–**All MMR accesses are through SCB0, which is in the SYSCLK domain, while peripherals are in the SCLK0/1, SYSCLK, and DCLK domains.
- **Number of concurrent MMR access requests in the system–**Although a single write incurs half the system latency when compared to back-to-back writes, the latency observed on the core is shorter. Similarly, the system latency incurred by a read followed by a write, or vice versa, is different than a latency observed on the core.
- Memory type (L1/L2/L3) from where the code is executed.

System Bandwidth Optimization Procedure

Although the optimization techniques can vary from one application to another, the general procedure for bandwidth optimization remains the same. Figure 18 provides a flow chart of typical steps used in a system bandwidth optimization procedure for ADSP-SC59x processor-based applications.

A typical procedure for an SCB completer includes the following steps:

1. Identify the individual and total throughput requirements for all the requesters accessing the corresponding SCB completer in the system. Consider the total throughput requirement as *X*.
2. Calculate the observed throughput the corresponding SCB completer(s) can supply under the specific conditions. Refer to this calculated value as *Y*.
3. For $X < Y$, bandwidth requirements are met. However, for $X > Y$, one or more peripherals are likely to hit reduced throughput or an underflow condition.

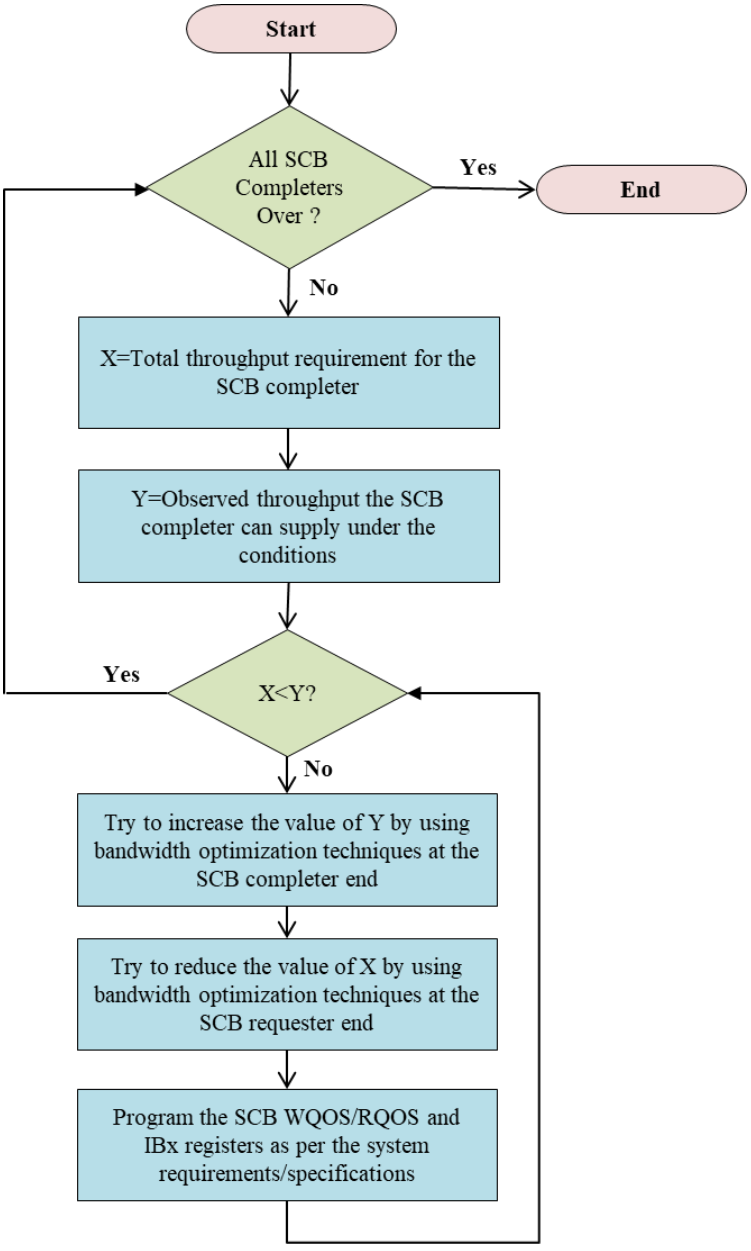
In this case, apply the bandwidth optimization techniques to:

- *Increase* the value of *Y* by applying the completer-specific optimization techniques (for example, using the DMC efficiency controller features).
- *Decrease* the value of *X*:
  - Reanalyze whether a peripheral needs to run that fast. If not, then slow down the peripheral to reduce the bandwidth requested by the peripheral.
  - Reanalyze whether an MDMA can be slowed down. The corresponding `DMAx_BWLCNT` register can be used to limit the bandwidth of that DMA channel.

Application Example

Figure 18 shows a block diagram of the example application code found in the `Test12_Multiple_DMAs` subfolder of the Zip file supplied with the EE445 Note.

Figure 18 Typical System Bandwidth Optimization Procedure

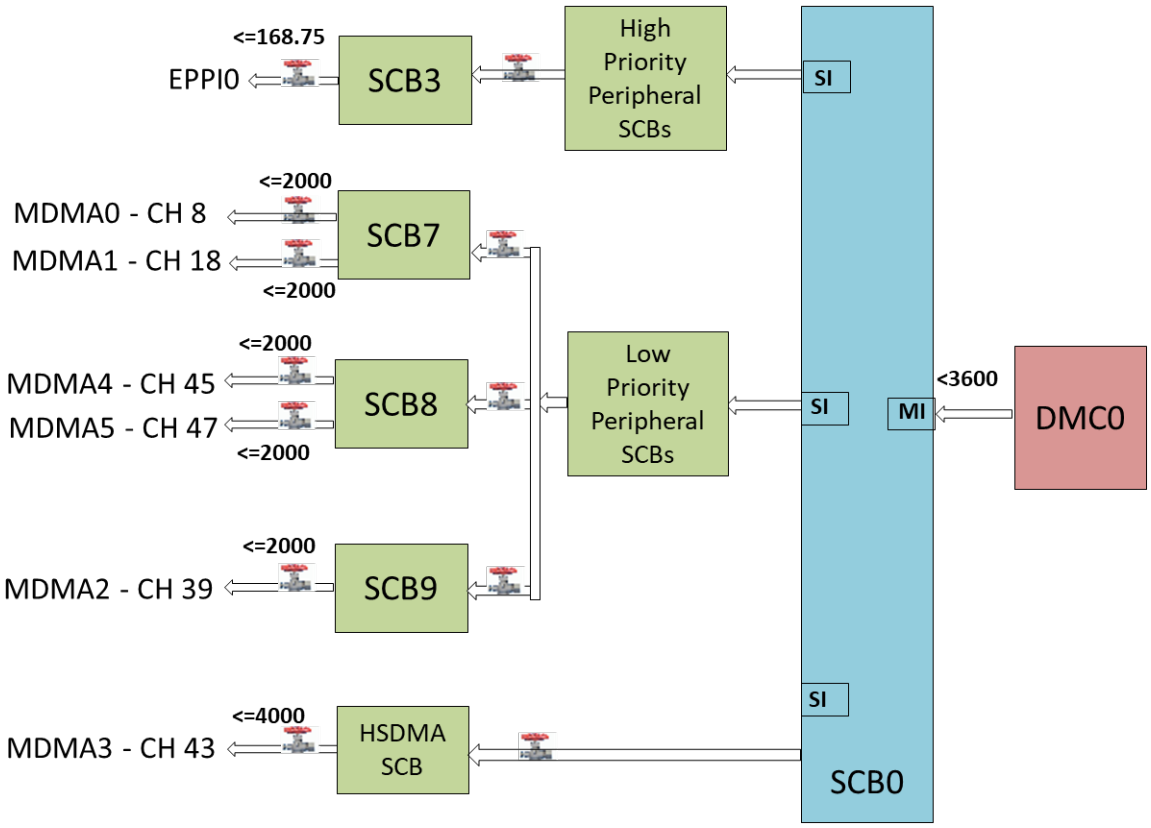


The procedures in Figure 18 generate the following throughput for the Figure 19 example application:

- CCLK = 1 GHz, DCLK = 900 MHz, SYSCLK = 500 MHz, and SCLK0 = 125 MHz.
- EPPI0(SCB3): 24-bit transmit, internal clock and frame sync mode at EPPICLK = 56.25 MHz. Required throughput = 56.25 MHz * 3 = 168.75 MB/s.
- MDMA0 channel 8: required throughput = 500 MHz * 4 ≤ 2000 MB/s.
- MDMA1 channel 18: required throughput = 500 MHz * 4 ≤ 2000 MB/s.
- MDMA2 channel 39: required throughput = 500 MHz * 4 ≤ 2000 MB/s.
- MDMA3 channel 43: required throughput = 500 MHz * 8 ≤ 4000 MB/s.
- MDMA4 channel 45, required throughput = 500 MHz * 4 <= 2000 MB/s.
- MDMA5 channel 47, required throughput = 500 MHz * 4 <= 2000 MB/s.

Throughput requirements for MDMA channels depend on the corresponding `DMAx_BWLCNT` register values. If not programmed, the MDMA channels request is for the bandwidth with full throttle.

Figure 19 Example Application–DMC Throughput Distribution Across SCBs on ADSP-SC598 Processor



As shown in Table 11 and Figure 19, the total required throughput from the DMC controller equals 14168.75 MB/s. Theoretically, with DCLK = 900 MHz and a bus width of 64 bits, the maximum possible throughput is only 14400 MB/s. For this reason, there is a possibility that one or more requesters may not get the required bandwidth. For requesters such as MDMA, this can result in decreased throughput.

Table 11 Example Application - System Bandwidth Optimization Steps on ADSP-SC598 Processor

S. No.	Condition	SCB3		SCB7				SCB9		HSDMA SCB		SCB8				Total throughput (X)	Measured throughput (Y)	PPI Underflow ?
		PPI0		MDMA0		MDMA1		MDMA2		MDMA3		MDMA4		MDMA5				
		Required	Measured	Required	Measured	Required	Measured	Required	Measured	Required	Measured	Required	Measured	Required	Measured			
1	No optimization	168.75	188.73	2000	56.83	2000	56.83	2000	113.27	4000	223.46	2000	56.84	2000	56.83	14168.75	752.78	YES
2	Optimization at the slave - T1	168.75	188.53	2000	111.07	2000	111.02	2000	221.45	4000	399.03	2000	111.04	2000	111.05	14168.75	1253.20	NO
3	Optimization at the master - T2	168.75	188.53	200	190.19	100	98.99	200	196.69	300	292.92	150	148.85	100	99.09	1218.75	1215.25	NO
4	Optimization at the master - T3	168.75	188.48	150	148.98	200	195.84	100	99.15	200	196.85	200	193.56	200	193.85	1218.75	1216.71	NO
5	Optimization at the master - T4	168.75	188.63	100	99.25	75	74.69	300	295.15	400	387.93	100	99.25	75	74.70	1218.75	1219.61	NO

All units are expressed in MB/s.

Table 11 shows the expected and measured throughput for all DMA channels and the corresponding SCBs at various steps of bandwidth optimization.

All DMA channels run without applying any optimization techniques. To replicate the worst-case scenario, the source buffers of all DMA channels are placed in a single DDR3 SDRAM bank. The first row corresponding to the *No optimization* condition in Table 11 shows the measured throughput numbers under this condition. As illustrated, the individual measured throughput of almost all channels is significantly less than expected.

- Total expected throughput from the DMC (***X***) = 14168.75 MB/s
- Effective DMC throughput (***Y***) = 752.78 MB/s

The reality is that *X* is an order of magnitude greater than *Y*, showing a definite need for implementing the corresponding bandwidth optimization techniques.

Table Condition 2

When there are frequent DDR3 SDRAM page misses within the same bank, throughput significantly drops. Although DMA channel accesses are sequential, multiple channels trying to access the DMC concurrently result in page misses. To work around this, move the source buffers for each DMA channel to different DDR3 SDRAM banks. This configuration allows parallel accesses to multiple pages of different banks, helping improve the calculated *Y* value. The *Optimization at the slave–T1* row in Table 11 provides the measured throughput numbers under this condition. Both individual and overall throughput numbers increase significantly. The maximum throughput delivered by the DMC (*Y*) increases to 1253.20 MB/sec. However, since *X* is still greater than *Y*, the measured throughput is still lower than expected.

Table Conditions 3, 4 and 5

Now, not much can be done to significantly increase the calculated value of *Y*. Alternatively, optimization techniques can be employed at the Requester end. Depending on the application requirements, the bandwidth limit feature of the MDMAs can be used to reduce the overall bandwidth requirement and get a predictable bandwidth at various MDMA channels. *Optimization at the master–T2*, *Optimization at the master–T3,* and *Optimization at the master–T4* rows in Table 11 show the measured throughput under two such conditions with different bandwidth limit values for the various MDMA channels. As shown in Table 11, the individual, overall, and expected throughput values are approximately the same.

System Optimization Techniques Checklist

This checklist summarizes the bandwidth optimization techniques discussed in this E445 Note, plus more:

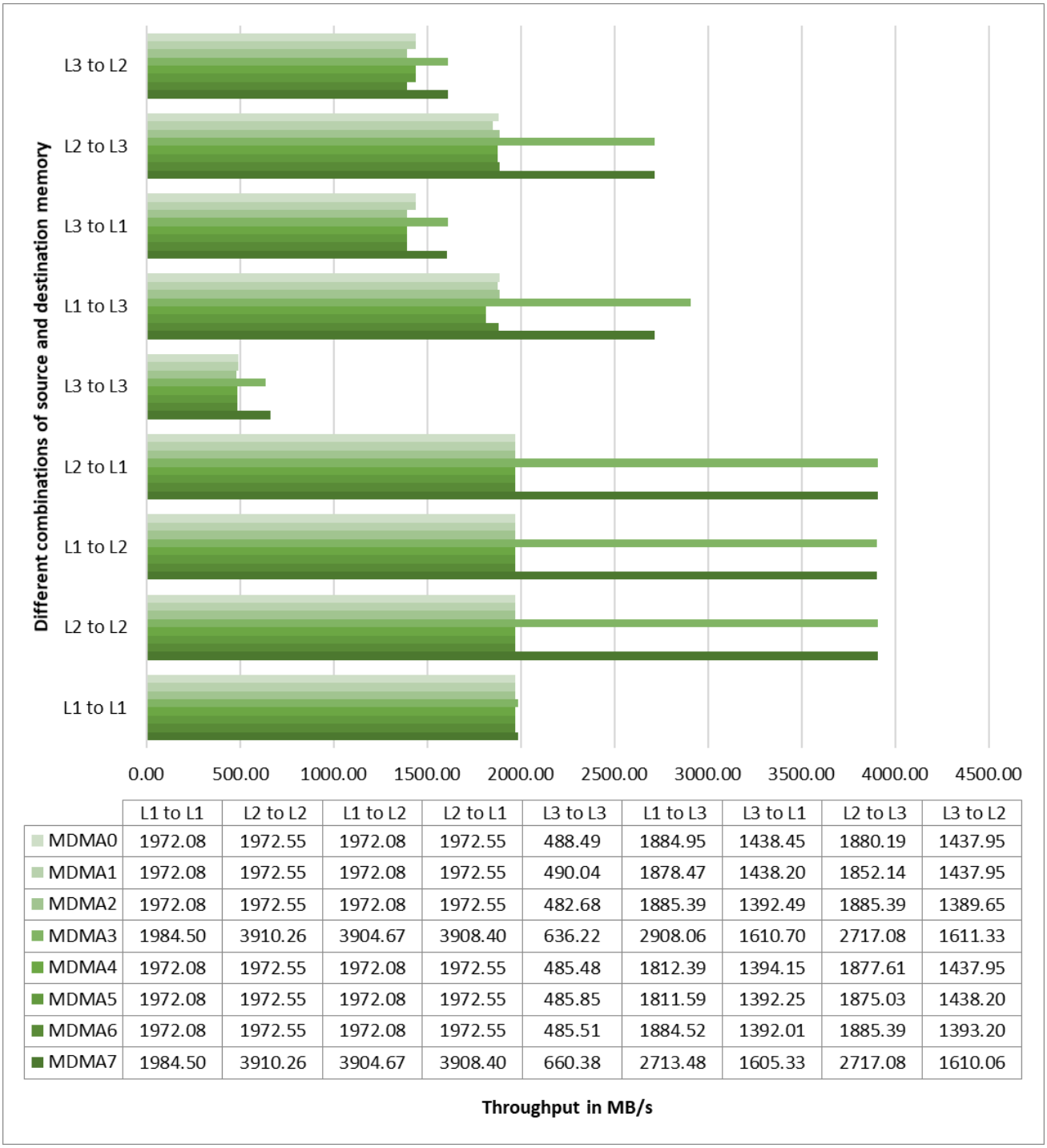
- Analyze the overall bandwidth requirements and use the bandwidth limit feature for memory pipe DMA channels to regulate the overall DMA traffic.
- Program the DMA channels' `MSIZE` parameters to optimal values to maximize throughput and avoid any potential underflow/overflow conditions.
- Split single MDMA (when required or possible) of a smaller `MSIZE` value into multiple descriptor-based MDMA transfers to maximize the usage of larger `MSIZE` values for better performance.
- Use MDMA instead of EMDMA for sequential data transfers to improve performance. When possible, emulate EMDMA non-sequential transfer modes with MDMA.
- Program SCB `RQOS` and `WQOS` registers to allocate priorities to requesters (system requirements).
- Program the clock domain crossing (`IBx`) registers depending upon the clock ratios across SCBs.
- Use optimization techniques at the SCB completer end, such as:
  - Efficient usage of the DMC controller
  - Usage of multiple L2/L1 sub-banks to avoid access conflicts
  - Usage of instruction/data caches
- Maintain the optimum clock ratios across different clock domains.
- Use Trigger Routing Unit (TRU)–Because MMR latencies affect the interrupt service latency, ADSP-SC59x processors provide the TRU for bandwidth optimization and system synchronization. The TRU provides for synchronizing system events without processor core intervention. It maps the trigger requesters to trigger completers (triggers receivers), offloading processing from the core. For a detailed discussion on this topic, refer to application note *Utilizing the Trigger Routing Unit for System Level Synchronization (EE-360)*[6]. The EE360 Note is written for the ADSP-BF60x processor and the concepts also apply to the ADSP-SC59x processors.

Appendix

This appendix provides optimization data figures and tables for the ADSP-SC594 Processors:

- Figure 20–*MDMA Throughput on ADSP-SC594*
- Table 12–*adi_mdma_Copy1D vs. adi_mdma_Copy1DAuto Performance on ADSP-SC594*
- Figure 21–*Measured EMDMA Throughput on ADSP-SC594*
- Table 13–*MDMA Emulated Circular Buffer vs. EMDMA on ADSP-SC594*
- Figure 22–*L2 MDMA Throughput for Different MSIZE and Work Unit Sizes on ADSP-SC594*
- Figure 23–*DMC Measured Throughput for Sequential MDMA Reads on ADSP-SC594*
- Figure 24–*DMC Measured Throughput for Sequential MDMA Writes on ADSP-SC594*
- Figure 25–*DMC Throughput for Sequential vs. Non-Sequential Read Accesses on ADSP-SC594*
- Figure 26–*Optimizing Throughput for Non-Sequential Accesses on ADSP-SC594*
- Table 14–*DMC Measured Throughput for Different DMC_PRIO Settings on ADSP-SC594*
- Table 15–*MMR Access Latency for ADSP-SC594 Processors in CCLK cycles (approximate)*
- Figure 27–*Example Application—DMC Throughput Distribution Across SCBs on ADSP-SC594*
- Table 16–Example Application—System Bandwidth Optimization Steps on ADSP-SC594

Figure 20 MDMA Throughput on ADSP-SC594 Processor



	L1 to L1	L2 to L2	L1 to L2	L2 to L1	L3 to L3	L1 to L3	L3 to L1	L2 to L3	L3 to L2
MDMA0	1972.08	1972.55	1972.08	1972.55	488.49	1884.95	1438.45	1880.19	1437.95
MDMA1	1972.08	1972.55	1972.08	1972.55	490.04	1878.47	1438.20	1852.14	1437.95
MDMA2	1972.08	1972.55	1972.08	1972.55	482.68	1885.39	1392.49	1885.39	1389.65
MDMA3	1984.50	3910.26	3904.67	3908.40	636.22	2908.06	1610.70	2717.08	1611.33
MDMA4	1972.08	1972.55	1972.08	1972.55	485.48	1812.39	1394.15	1877.61	1437.95
MDMA5	1972.08	1972.55	1972.08	1972.55	485.85	1811.59	1392.25	1875.03	1438.20
MDMA6	1972.08	1972.55	1972.08	1972.55	485.51	1884.52	1392.01	1885.39	1393.20
MDMA7	1984.50	3910.26	3904.67	3908.40	660.38	2713.48	1605.33	2717.08	1610.06

Table 12 adi_mdma_Copy1D vs. adi_mdma_Copy1DAuto Performance on ADSP-SC594 Processor

S. No.	Source Memory Address	Destination Memory Address	DMA Count	MSIZE (Bytes)	Copy1D MDMA Cycles	Copy1DAuto MDMA Cycles	Additional API Overhead	Effective Improvement Factor
1	0x2C0001	0x300000	256	1	1825	1317	111	1.28
2	0x2C0000	0x300001	256	1	1825	1241	99	1.36
3	0x2C0001	0x300000	1024	1	5377	2853	89	1.83
4	0x2C0000	0x300001	1024	1	5377	2777	99	1.87
5	0x2C0001	0x300000	4096	1	19585	8997	89	2.16
6	0x2C0000	0x300001	4096	1	19585	8921	99	2.17

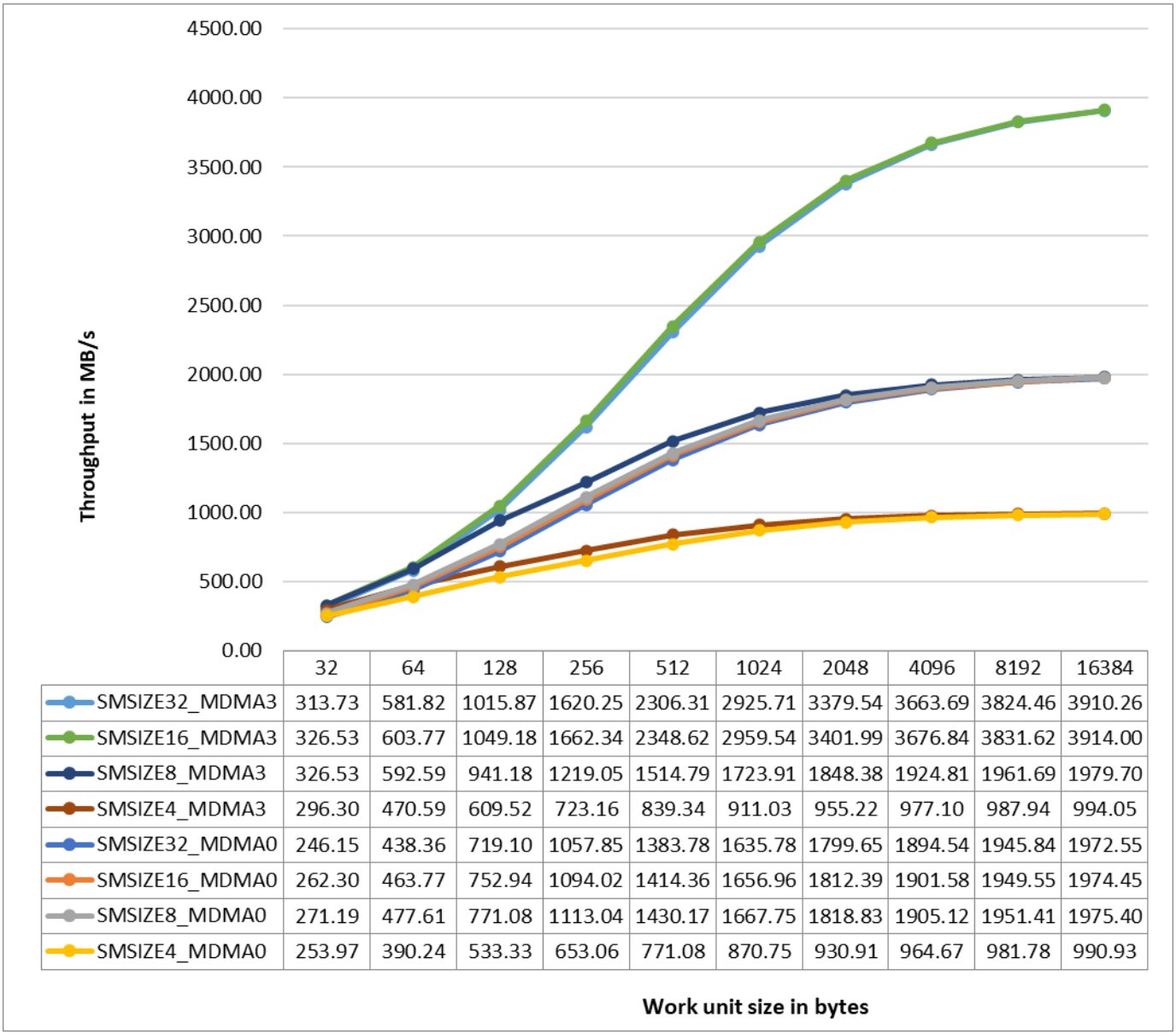
Figure 21 Measured EMDMA Throughput on ADSP-SC594 Processor

	L1-I L1-E	L2-I L2-E	L1-I L2-E	L2-I L1-E	L3-I L3-E	L1-I L3-E	L3-I L1-E	L2-I L3-E	L3-I L2-E
EMDMA0_Write	747.68	748.37	748.30	748.44	251.49	470.33	375.10	465.49	378.29
EMDMA0_Read	748.47	748.33	748.37	748.33	251.73	375.01	469.83	374.99	465.16
EMDMA1_Write	748.54	748.30	748.30	748.54	253.40	470.02	375.69	470.13	377.92
EMDMA1_Read	748.47	748.33	748.47	748.23	251.85	374.83	464.52	375.19	464.99

Table 13 MDMA Emulated Circular Buffer vs. EMDMA on ADSP-SC594 Processor

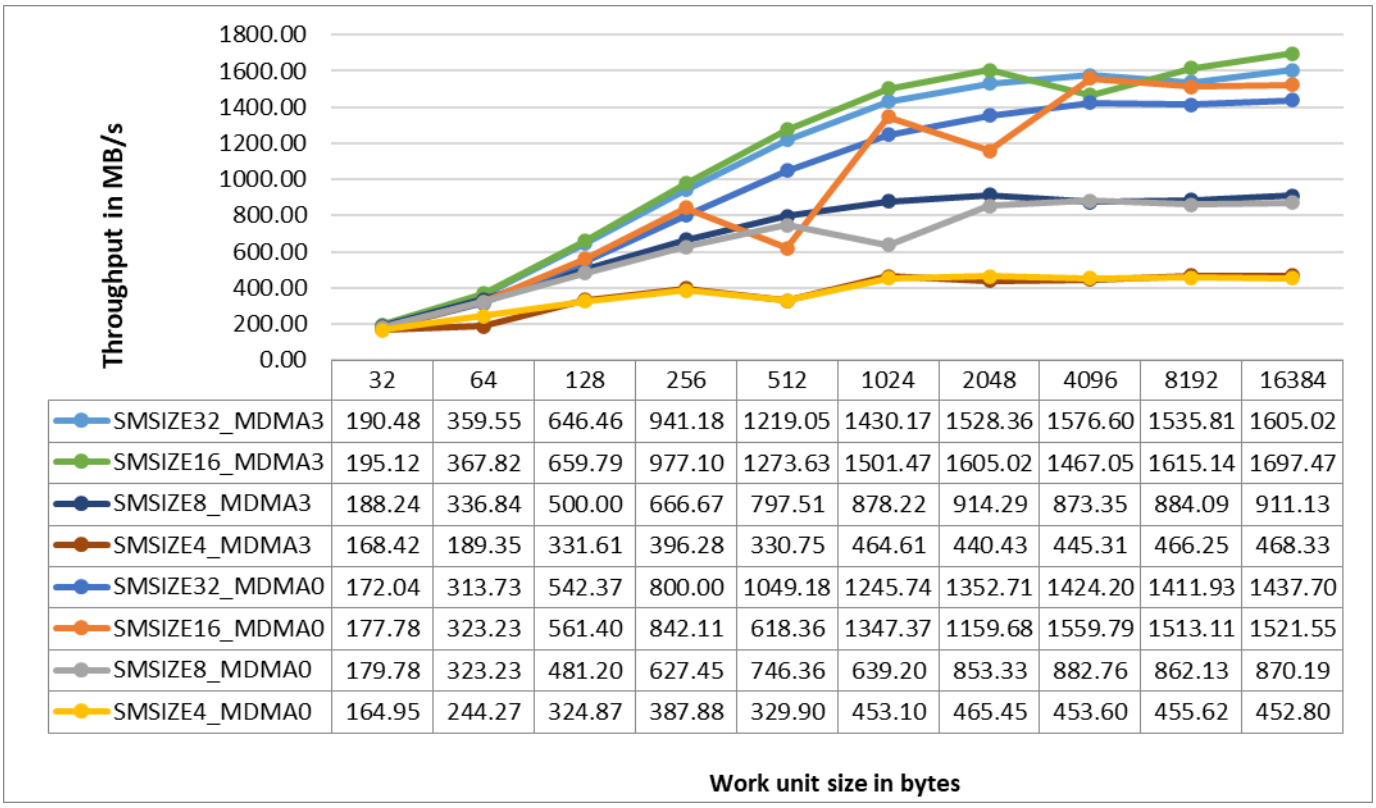
Write/Read	Core Cycles		
	EMDMA	MDMA3 MSIZE = 4 bytes	MDMA3 MSIZE = 32 bytes
Write	35656	29329	6607
Read	46390	35693	11233

Figure 22 L2 MDMA Throughput for Different MSIZE and Work Unit Sizes on ADSP-SC594 Processor



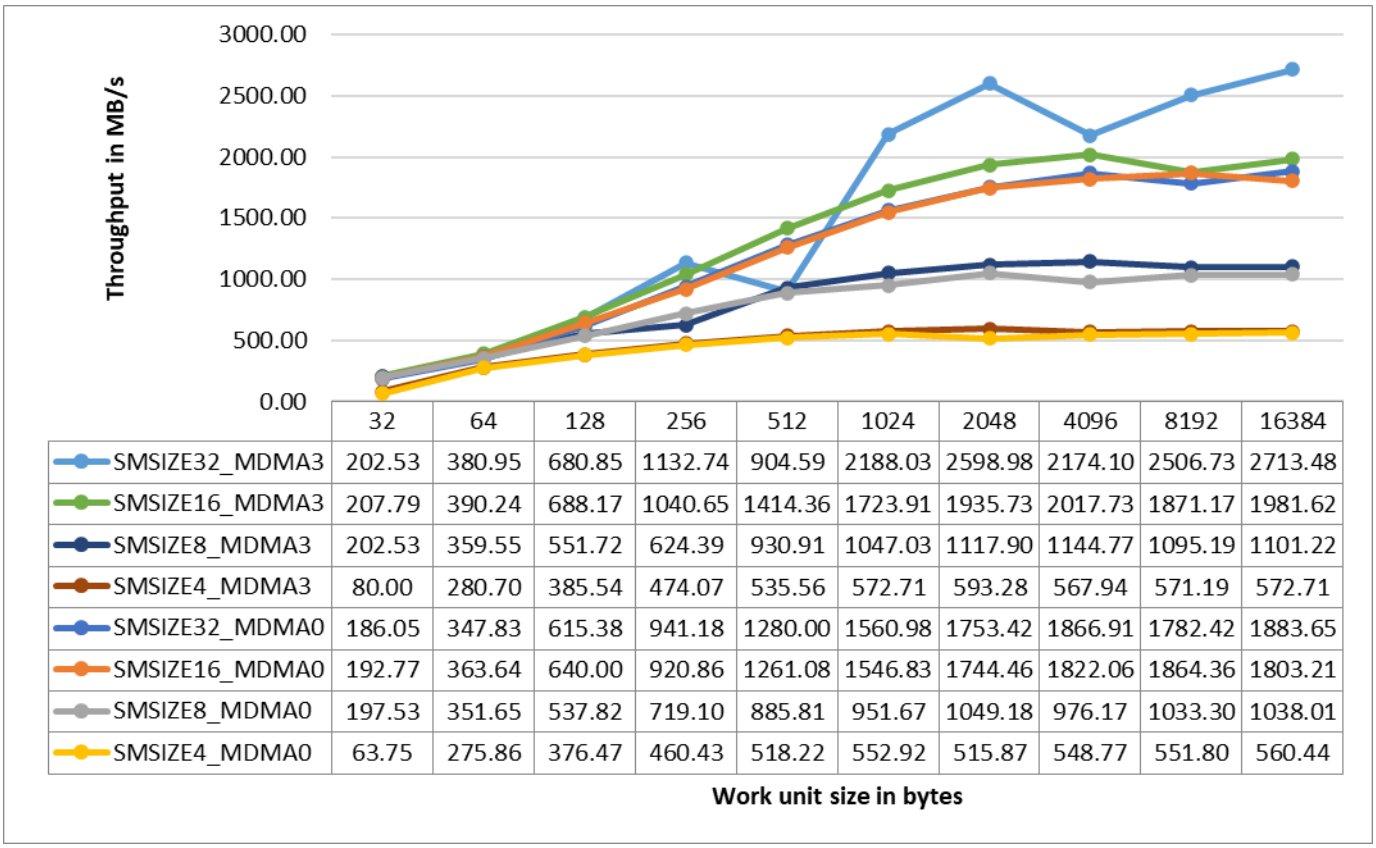
	32	64	128	256	512	1024	2048	4096	8192	16384
SMSIZE32_MDMA3	313.73	581.82	1015.87	1620.25	2306.31	2925.71	3379.54	3663.69	3824.46	3910.26
SMSIZE16_MDMA3	326.53	603.77	1049.18	1662.34	2348.62	2959.54	3401.99	3676.84	3831.62	3914.00
SMSIZE8_MDMA3	326.53	592.59	941.18	1219.05	1514.79	1723.91	1848.38	1924.81	1961.69	1979.70
SMSIZE4_MDMA3	296.30	470.59	609.52	723.16	839.34	911.03	955.22	977.10	987.94	994.05
SMSIZE32_MDMA0	246.15	438.36	719.10	1057.85	1383.78	1635.78	1799.65	1894.54	1945.84	1972.55
SMSIZE16_MDMA0	262.30	463.77	752.94	1094.02	1414.36	1656.96	1812.39	1901.58	1949.55	1974.45
SMSIZE8_MDMA0	271.19	477.61	771.08	1113.04	1430.17	1667.75	1818.83	1905.12	1951.41	1975.40
SMSIZE4_MDMA0	253.97	390.24	533.33	653.06	771.08	870.75	930.91	964.67	981.78	990.93

Figure 23 DMC Measured Throughput for Sequential MDMA Reads on ADSP-SC594 Processor



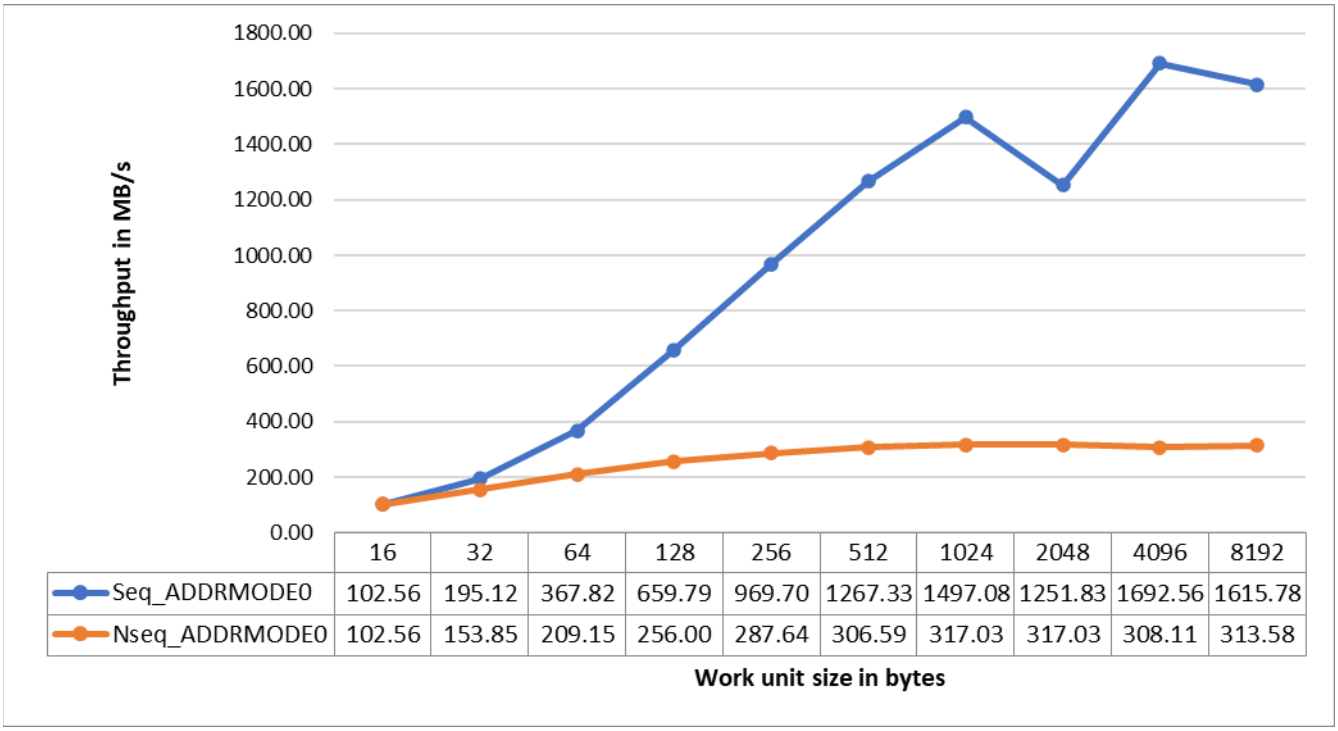
	32	64	128	256	512	1024	2048	4096	8192	16384
SMSIZE32_MDMA3	190.48	359.55	646.46	941.18	1219.05	1430.17	1528.36	1576.60	1535.81	1605.02
SMSIZE16_MDMA3	195.12	367.82	659.79	977.10	1273.63	1501.47	1605.02	1467.05	1615.14	1697.47
SMSIZE8_MDMA3	188.24	336.84	500.00	666.67	797.51	878.22	914.29	873.35	884.09	911.13
SMSIZE4_MDMA3	168.42	189.35	331.61	396.28	330.75	464.61	440.43	445.31	466.25	468.33
SMSIZE32_MDMA0	172.04	313.73	542.37	800.00	1049.18	1245.74	1352.71	1424.20	1411.93	1437.70
SMSIZE16_MDMA0	177.78	323.23	561.40	842.11	618.36	1347.37	1159.68	1559.79	1513.11	1521.55
SMSIZE8_MDMA0	179.78	323.23	481.20	627.45	746.36	639.20	853.33	882.76	862.13	870.19
SMSIZE4_MDMA0	164.95	244.27	324.87	387.88	329.90	453.10	465.45	453.60	455.62	452.80

Figure 24 DMC Measured Throughput for Sequential MDMA Writes on ADSP-SC594 Processor



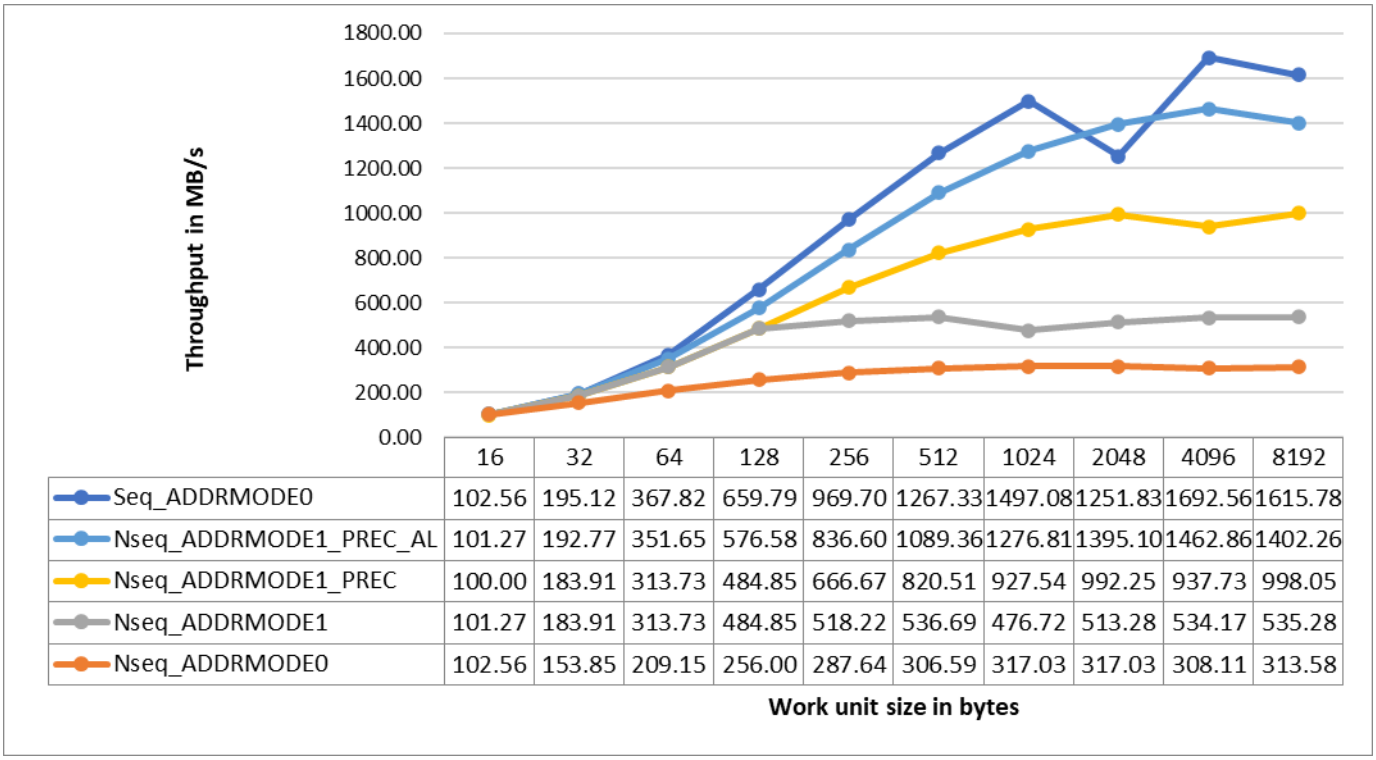
	32	64	128	256	512	1024	2048	4096	8192	16384
SMSIZE32_MDMA3	202.53	380.95	680.85	1132.74	904.59	2188.03	2598.98	2174.10	2506.73	2713.48
SMSIZE16_MDMA3	207.79	390.24	688.17	1040.65	1414.36	1723.91	1935.73	2017.73	1871.17	1981.62
SMSIZE8_MDMA3	202.53	359.55	551.72	624.39	930.91	1047.03	1117.90	1144.77	1095.19	1101.22
SMSIZE4_MDMA3	80.00	280.70	385.54	474.07	535.56	572.71	593.28	567.94	571.19	572.71
SMSIZE32_MDMA0	186.05	347.83	615.38	941.18	1280.00	1560.98	1753.42	1866.91	1782.42	1883.65
SMSIZE16_MDMA0	192.77	363.64	640.00	920.86	1261.08	1546.83	1744.46	1822.06	1864.36	1803.21
SMSIZE8_MDMA0	197.53	351.65	537.82	719.10	885.81	951.67	1049.18	976.17	1033.30	1038.01
SMSIZE4_MDMA0	63.75	275.86	376.47	460.43	518.22	552.92	515.87	548.77	551.80	560.44

Figure 25 DMC Throughput for Sequential vs. Non-Sequential Read Accesses on ADSP-SC594 Processor



	16	32	64	128	256	512	1024	2048	4096	8192
Seq_ADDRMODE0	102.56	195.12	367.82	659.79	969.70	1267.33	1497.08	1251.83	1692.56	1615.78
Nseq_ADDRMODE0	102.56	153.85	209.15	256.00	287.64	306.59	317.03	317.03	308.11	313.58

Figure 26 Optimizing Throughput for Non-Sequential Accesses on ADSP-SC594 Processor



	16	32	64	128	256	512	1024	2048	4096	8192
Seq_ADDRMODE0	102.56	195.12	367.82	659.79	969.70	1267.33	1497.08	1251.83	1692.56	1615.78
Nseq_ADDRMODE1_PREC_AL	101.27	192.77	351.65	576.58	836.60	1089.36	1276.81	1395.10	1462.86	1402.26
Nseq_ADDRMODE1_PREC	100.00	183.91	313.73	484.85	666.67	820.51	927.54	992.25	937.73	998.05
Nseq_ADDRMODE1	101.27	183.91	313.73	484.85	518.22	536.69	476.72	513.28	534.17	535.28
Nseq_ADDRMODE0	102.56	153.85	209.15	256.00	287.64	306.59	317.03	317.03	308.11	313.58

Table 14 DMC Measured Throughput for Different DMC_PRIO Settings on ADSP-SC594 Processor

Test Case Number	Priority Channel (SCB ID)	MDMA0 (Ch. 8) Throughput (MB/s)	MDMA1 (Ch. 18) Throughput (MB/s)
1	None	726	737
2	MDMA0 (0x0031)	859	725
3	MDMA1 (0x0011)	729	916

Table 15 MMR Access Latency for ADSP-SC594 Processors in CCLK cycles (approximate)

S. No.	Register	Peripheral	Write Latency (Core Cycles)	Read Latency (Core Cycles)
1	FIR0_INIDX	FIR	41	41
2	IIR0_INIDX	IIR	41	41
3	MLB0_MDAT0	MLB	55	54
4	MEC0_PERR_IMASK0	MEC	55	54
5	CRC0_DCNT	CRC	55	56
6	CRC1_DCNT		55	56
7	EMDMA0_INDX1	EMDMA	55	56
8	EMDMA1_INDX1		55	56
9	TAPC_SDBGKEY0	TAPC	55	54
10	MISCREG_CAN_SYSCTL	MISCREG	55	54
11	SMPU2_RADDR0	SMPU	57	56
12	SWU1_LA0	SWU	57	56
13	SWU2_LA0		57	56
14	L2CTL0_RPCR0	L2CTL	57	56
15	SEC0_RAISE	SEC	57	58
16	TRU0_SSR0	TRU	57	58
17	SPU0_SECUREP10	SPU	57	58
18	RCU0_MSG	RCU	57	56
19	CGU0_OSCWDCTL	CGU	57	56
20	CDU0_CLKINSEL	CDU	57	56
21	DPM0_PER_DIS0	DPM	57	56
22	GICDST0_EN	GICD	57	60
23	GICDST0_SPI_EN_SET0	GICD	57	60
24	GICCPU0_CTL	GICCPU	57	58
25	GICCPU0_EOI	GICCPU	57	58
26	PKTE0_SA_ADDR	PKTE	59	60
27	TRNG0_OUTPUT0	TRNG	63	64
28	PKA0_APTR	PKA	63	64

S. No.	Register	Peripheral	Write Latency (Core Cycles)	Read Latency (Core Cycles)
29	PKIC0_ACK	PKIC	63	64
30	DMC0_PRIO	DMC	71	70
31	PORTA_DATA_SET	PORTA	85	88
32	PADS0_PORTA_PDE	PADS	85	88
33	OTPC0_PMC_MODE0	OTPC	85	88
34	TIMER0_TMR0_WID	TIMER0	85	96
35	OSPI0_FCA	OSPI	85	88
36	EMAC1_MAC_IEN	EMAC	85	88
37	SPORT1_DIV_A	SPORT	87	96
38	UART0_CLK	UART	87	88
39	EMAC0_MAC_IEN	EMAC	87	88
40	EMAC1_MAC_CFG		87	88
41	DMA1_XCNT	DMA	89	88
42	SPORT0_DIV_A	SPORT	89	96
43	UART1_CLK	UART	89	88
44	PORTB_DATA_SET	PORTB	89	88
45	PINT0_ASSIGN	PINT	89	88
46	SPI0_CLK	SPI	89	96
47	SPI1_CLK		89	96
48	DMA0_XCNT	DMA	91	88
49	PINT1_ASSIGN	PINT	91	88
50	WDOG0_WIN	WDOG	91	88
51	WDOG1_WIN		91	88
52	CNT0_CNTR	CNT	91	88
53	SPDIF0_TX_UBUFF_A0	SPDIF	91	96
54	DAI0_IMSK_FE	DAI	91	96
55	EPPI0_CTL	EPPI	91	88
56	EMAC0_MAC_CFG	EMAC	91	88
57	TMU0_FLT_LIM_HI	TMU	93	96
58	SPDIF1_TX_UBUFF_A0	SPDIF	93	96
59	DAI1_IMSK_FE	DAI	93	96
60	ASRC0_MUTE	ASRC	95	96
61	USBC0_CFG_H	USBC	95	96
62	PDM0_CTL0	PDM	95	96
63	PDM1_CTL0		95	96
64	ARMPMU0_PMCCNTR	ARMPMU	95	94
65	ARMPMU0_PMCR		95	94
66	HADC0_CHAN_MSK	HADC	97	96
67	PCG0_PW1	PCG	97	96
68	USBC0_OTG_CTL	USBC	97	96
69	ASRC1_MUTE	ASRC	99	96

S. No.	Register	Peripheral	Write Latency (Core Cycles)	Read Latency (Core Cycles)
70	TWI1_CLKDIV	TWI	125	136
71	TWI0_CLKDIV		131	136
72	LP0_DIV	LP	147	160
73	LP1_DIV		147	160
74	SCB0_SP0A_READ_QOS	SCB	495	496

Figure 27 Example Application—DMC Throughput Distribution Across SCBs on ADSP-SC594 Processor

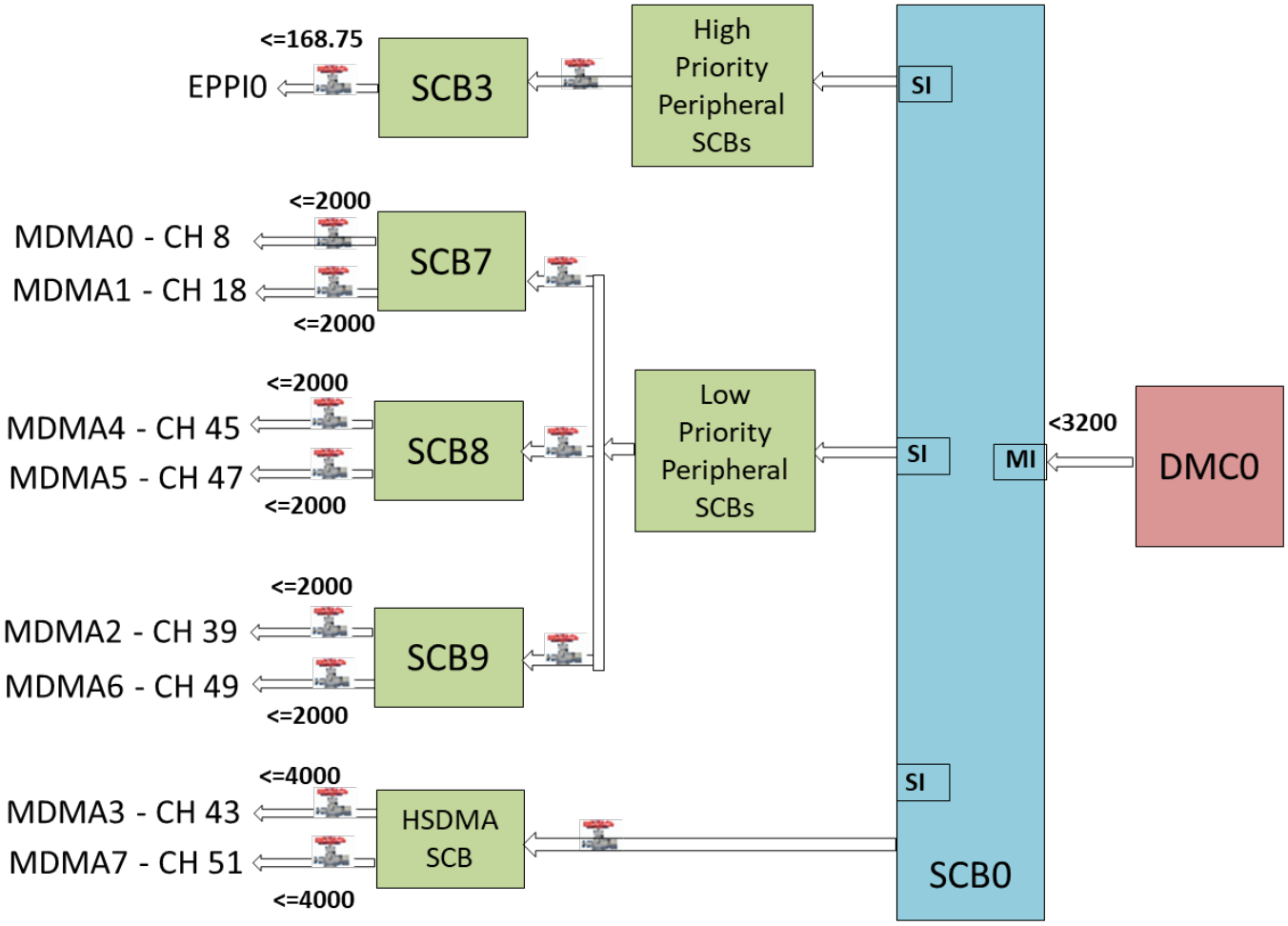


Table 16 Example Application—System Bandwidth Optimization Steps on ADSP-SC594 Processor

S. No.	Condition	SCB3		SCB7				SCB9		HSDMA SCB		SCB8				Total throughput (X)	Measured throughput (Y)	PPI Underflow ?
		PPI0		MDMA0		MDMA1		MDMA2		MDMA3		MDMA4		MDMA5				
		Required	Measured	Required	Measured	Required	Measured	Required	Measured	Required	Measured	Required	Measured	Required	Measured			
1	No optimization	168.75	188.83	2000	56.14	2000	56.14	2000	111.89	4000	220.62	2000	56.15	2000	56.14	14168.75	745.91	YES
2	Optimization at the slave - T1	168.75	188.72	2000	69.06	2000	69.05	2000	137.61	4000	264.69	2000	69.06	2000	69.06	14168.75	867.25	YES
3	Optimization at the master - T2	168.75	188.69	100	99.20	100	99.20	100	99.20	100	99.21	100	99.21	100	99.21	768.75	783.92	NO
4	Optimization at the master - T3	168.75	188.78	75	74.68	75	74.67	100	99.23	200	197.16	75	74.67	75	74.67	768.75	783.86	NO
5	Optimization at the master - T4	168.75	188.89	50	49.80	50	49.80	150	149.26	250	245.67	50	49.80	50	49.80	768.75	783.03	NO

References

[1] *ADSP-2156x SHARC+ Processor System Optimization Techniques (EE-412)*, Rev 2, September 2020, Analog Devices, Inc.

[2] *ADSP-SC5xx/215xx SHARC+ Processor System Optimization Techniques (EE-401)*, Rev 1, February 2018.Analog Devices, Inc.

[3] *ADSP-2159x/ADSP-SC591/SC592/SC594 SHARC+ Processor Hardware Reference,* Rev 0.4, April 2022. Analog Devices, Inc.

[4] *ADSP- SC595/SC596/SC598 SHARC+ Processor Hardware Reference,* Rev 0.1, May 2022. Analog Devices, Inc.

[5] *ADSP-21591/21593/21594/ADSP-SC591/SC592/SC594 SHARC+ Dual-Core DSP with Arm Cortex-A5 Data Sheet.* Rev A, April 2022. Analog Devices, Inc.

[6] *ADSP-SC595/SC596/SC598 SHARC+ Dual-Core DSP with Arm Cortex-A55 Data Sheet*. Rev PrE, July 2022. Analog Devices, Inc.

[7] *TN-47-02: DDR2 Offers New Features/Functionality Introduction. Rev A, June 2006. Micron Technology, Inc.*

[8] *ADSP-SC59x SSDD API Reference Manual for SHARC+ Core*, Version 2.0, CrossCore® Embedded Studio Help.

[9] *Associated ZIP file for EE-445: ADSP-SC59x SHARC+ Processors System Optimization Techniques*, December 2022. Analog Devices, Inc.

Document History

Revision	Description
Rev 1 – December 6, 2022 *by Naga Snigdha Kondepati*	Initial release.