# TMS320C64x EDMA Performance Data

*Jeffrey Ward*
*Jamon Bowen*

*TMS320C6000 Architecture*

**ABSTRACT**

The enhanced DMA (EDMA) controller of the TMS320C64x™ device is a highly efficient data transfer engine, capable of maintaining transfers at up to 2.4 GB/sec at a 600 MHz CPU clock frequency. This document details measured bandwidth achieved under various operating conditions. For more information on ideal transfer bandwidth and scheduling transfers, please consult *TMS320C64x EDMA Architecture* (SPRA994), and *TMS320C6000 EDMA IO Scheduling and Performance* (SPRAA00).

**Contents**



**List of Figures**

**List of Tables**



# 1 Introduction

The enhanced DMA (EDMA) controller of the TMS320C64x devices is a highly efficient data transfer engine, capable of handling up to 8 bytes per EDMA cycle, resulting in 2.4 GB per second of total data throughput at a CPU rate of 600 MHz (the EDMA frequency being CPU frequency divided by two). The EDMA performs all data movement between the on-chip level-two (L2) memory, external memory (connected to the device through an external memory interface (EMIF)), and the device peripherals. These data transfers include CPU-initiated and event-triggered transfers, master peripheral accesses, cache servicing, and non-cacheable memory accesses. The EDMA architecture has many features designed to facilitate simultaneous multiple high-speed data transfers. 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 EDMA.

This document gives designers a basis for estimating system performance. Most of the tests operate under best-case situations to estimate maximum throughput that can be obtained. Two application notes, *TMS320C64x EDMA Architecture* (SPRA994) and *TMS320C6000 EDMA I/O Scheduling and Performance* (SPRAA00), augment this document by describing ways that interference from other DSP activities can inhibit performance, and methods to mitigate the interference. The transfers described in this document serve as a sample of interesting or typical performance conditions. The performance of other configurations is neglected in the interest of brevity and can generally be inferred from the examples given. When estimating other configurations performance, it is important to remember that the CPU to EMIF clock ratio is more important than the absolute level of these clocks. This document provides a good source of actual performance values to estimate the performance of a system.

# 2 EDMA Performance Data

Transfer performance is based on time. It is necessary to define quantities of time based on transfer events. Figure 1 is a simplified schematic of an example transfer in the EDMA showing the important time intervals. These time intervals are explained in detail in *TMS320C6000 EDMA I/O Scheduling and Performance* (SPRAA00).

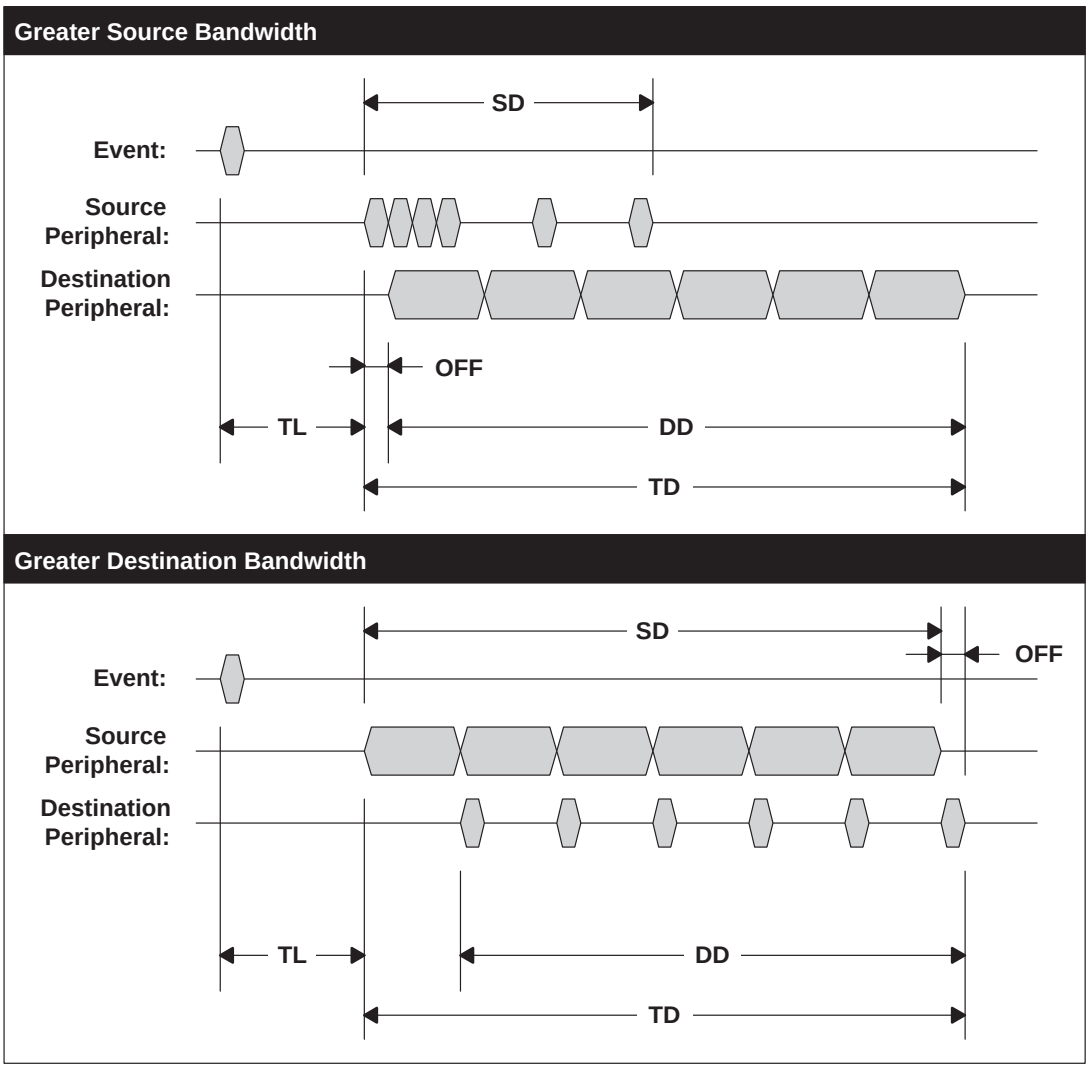


**Figure 1. Transfer Time Intervals**

- **Transfer latency** (TL), is defined as the time from event to the beginning of transfer at the source peripheral.
- **Source duration** (SD), or the interval of time during which the source peripheral reads data.
- **Destination duration** (DD), or the interval of time during which the destination peripheral writes data.
- **Transfer duration** (TD), defined as the time from the first read at the source to the last write at the destination.
- **OFF** is the source/destination offset.

Transfer bandwidth is limited by the lesser of the source or destination port bandwidth. The maximum speed the transfer could achieve is equal to the bandwidth of the limiting port. The transfer bandwidth is found by taking total bytes transferred and dividing it by the greater of the SD or DD. The transfer will never complete within the time defined by the transfer bandwidth, due to the offset and latency in the transmission. The offset and initial latency are variable and depend on a number of factors, so they are not considered when calculating transfer bandwidth. It is important to remember that these two values are generally a bigger concern for short transfers and need to be included when scheduling EDMA traffic in a system. See *TMS320C6000 EDMA I/O Scheduling and Performance* (SPRAA00) for details.

## 2.1 Transfer Latency

The EDMA ideal transfer latency (no interference from other transfers) is based on the following:

- If the source port is EMIF, transfer latency is approximately 30 CPU cycles plus five EMIF cycles. For SDRAM, there may be an additional delay due to SDRAM paging mechanisms, if the current SDRAM page is not open.
- If the source port is L2, transfer latency is approximately 42 CPU cycles.
- If the source port is an on-chip peripheral (such as the UTOPIA or McBSP), the transfer latency is approximately 50 CPU cycles.

These values vary based on the type of source peripheral and readiness of the source and destination ports and peripherals. Also, SDRAM has page open/close overhead which can cause delays in transfers that tend to increase with element count. SBSRAM, which isn't paged, is used for most of the tests in this section to eliminate the uncertainty that page opening and closing brings.

Table 1 shows data measured on a C64x™ DSP. No interfering transfer requests (TRs) are present. In this case, the SDRAM page opening delay appears to be 4 EMIF cycles.

**Table 1. Transfer Latency**

| CPU Frequency (MHz) | EMIF Frequency (MHz) | Source Peripheral | Expected Transfer Latency | | Actual Transfer Latency | |
|---|---|---|---|---|---|---|
| | | | nS | CPU cycles† | nS | CPU cycles† |
| 600 | 100 | EMIF (SBSRAM) | 100 | 60 | 100 | 60 |
| 600 | 133 | EMIF (SBSRAM) | 87.5 | 53 | 100 | 60 |
| 600 | 150 | EMIF (SBSRAM) | 83.3 | 50 | 86.4 | 52 |
| 600 | 100 | EMIF (SDRAM) | 140 | 84 | 131.2 | 79 |
| 600 | 133 | EMIF (SDRAM) | 117.5 | 71 | 128.8 | 77 |
| 600 | 150 | EMIF (SDRAM) | 110 | 66 | 108.8 | 65 |
| 300 | 100 | EMIF (SBSRAM) | 150 | 45 | 148.8 | 44.6 |
| 300 | 75 | EMIF (SBSRAM) | 166.7 | 50 | 160 | 48 |
| 300 | 100 | EMIF (SDRAM) | 190 | 57 | 186.2 | 55.9 |
| 300 | 75 | EMIF (SDRAM) | 220 | 66 | 202.4 | 60.7 |
| Don’t care | Don’t care | L2 Memory | – | 42 | – | 42 |
| Don’t care | Don’t care | McBSP | – | 50 | – | 49 |

† Note that CPU cycle period is defined by the CPU frequency shown.

In addition to transfer latency, there is another factor that influences overall transfer time: the source/destination offset. This value depends on the source and destination, the default burst size of the source/destination, and the transfer size. Calculating this value is discussed in *TMS320C6000 EDMA IO Scheduling and Performance* (SPRAA00).

## 2.2 Bandwidth vs. Transfer Flexibility

QDMA and EDMA channel parameters allow many different transfer configurations. Most typical transfers burst properly, and memory bandwidth is fully utilized. However, in some less common configurations, transfers are unable to burst, 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. These considerations are especially important for long memory transfers. Single-element transfer performance is latency-dominated and is unaffected by these conditions.

### 2.2.1 Element Size Considerations

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. It is best to use 32-bit element size whenever possible. Setting element size in the options register (OPT) to 32 bits allows a transfer to make full use of the available bus width, (whether it is 32 or 64 bits wide) as well as allow the transfer to burst data in multi-element blocks. Setting the element size to 16-bit or 8-bit elements causes the EDMA to mask out the remaining data, wasting bandwidth.

Table 2 shows performance data from a C64x CPU running at 600 MHz, transferring 4, 16, and 128 bytes to/from L2 to/from SBSRAM using an EDMA (or equivalently, a QDMA) channel. The SBSRAM is a 64-bit, 100 MHz memory with an ideal bandwidth of 800 Mb/sec. The transfer is performed with 8, 16, and 32-bit element sizes. Note that the 32-bit element size most efficiently uses available bandwidth.

**Table 2. Bus Width Utilization**

| Element Size | Total Data (bytes) | Read Access | | Write Access | |
|---|---|---|---|---|---|
| | | Transfer Bandwidth (Mb/sec) | SBSRAM Bandwidth Utilization | Transfer Bandwidth (Mb/sec) | SBSRAM Bandwidth Utilization |
| 32-bit | 4† | 400.0 | 50% | 400.0 | 50% |
| 32-bit | 16 | 800.0 | 100% | 800.0 | 100% |
| 32-bit | 128 | 800.0 | 100% | 800.0 | 100% |
| 16-bit | 4 | 143.9 | 18.0% | 200.0 | 25.0% |
| 16-bit | 16 | 85.1 | 10.6% | 105.3 | 13.2% |
| 16-bit | 128 | 74.0 | 9.3% | 100.6 | 12.6% |
| 8-bit | 4 | 50.0 | 7.0% | 55.6 | 7.0% |
| 8-bit | 16 | 42.1 | 5.3% | 51.9 | 6.5% |
| 8-bit | 128 | 38.4 | 4.8% | 50.2 | 6.3% |

† Note that a 4-byte (32 bit) transfer to a 64-bit SBSRAM cannot fully utilize bandwidth.

Non-programmable accesses (cache and master peripheral) are hardware programmed to fully utilize bus width.

### 2.2.2 Update Modes: Increment, Decrement, Index, and Fixed

The source update mode (SUM) and the destination update mode (DUM) help define how the address pointer is updated throughout a transfer. For information on update modes, see *TMS320C6000 DSP EDMA Controller Reference Guide* (SPRU234).

Table 3 shows performance data from a C64x CPU running at 600 MHz, transferring 128 bytes of 32-bit data to/from L2 to/from SBSRAM. The SBSRAM is a 64-bit, 100 MHz memory with an ideal bandwidth of 800 Mb/sec. The source and destination index modes are varied.

**Table 3. Update Mode Effects (L2 to EMIF example)**

| SUM | DUM | Transfer Duration (CPU Cycles) | Transfer Bandwidth (Mb/sec) | SBSRAM Bandwidth Utilization |
|---|---|---|---|---|
| L2: Any | EMIF: Increment | 96 | 800.0 | 100% |
| L2: Any | EMIF: Fixed | 96 | 800.0 | 100% |
| L2: Any | EMIF: Decrement | 144 | 533.3 | 66.7% |
| L2: Any | EMIF: Index | 379 | 202.5 | 25.3% |
| EMIF: Increment | L2: Any | 96 | 800 | 100% |
| EMIF: Fixed | L2: Any | 96 | 800 | 100% |
| EMIF: Decrement | L2: Any | 108 | 715 | 89.4% |
| EMIF: Index | L2: Any | 495 | 155 | 19.4% |

The L2 source has greater bandwidth then the destination EMIF at this clock ration (6:1, CPU:EMIF). This extra bandwidth provides the L2 with ample time to service the EMIF port regardless of its update mode. At lower clock ratios this will not be the case, and decrement and index source update modes will lower performance.

### *2.2.3 2-D Transfers − Element Count*

Two-dimensional transfers offer designers a great deal of flexibility. Two-dimensional transfers allow a complex memory transfer (such as extracting a sub−region from an image in memory) to be submitted to the EDMA easily and accomplished automatically. See the EDMA section of the *TMS320C6000 Peripherals Reference Guide* (SPRU190) for more information on two-dimensional EDMA transfers.

The EDMA and the EMIF are optimized to facilitate large transfers in contiguous bursts. This is optimal for most EDMA traffic, but the performance of two-dimensional EDMA transfers with small element and frame counts suffer as a result. Smaller element counts experience the greatest performance degradation. The following charts summarize this trend, 1 to 128 32-bit words were transferred to/from an external SDRAM running at 133MHz, using various frame/element counts to achieve the total number of words. Note that transfers with only 1 frame complete in one burst and therefore achieve maximum transfer bandwidth, which then degrades for multiple frame transfers. This data is collected from the pin side of the transfer, adding the initial latency for TR submission will increase the transfer time for short transfers significantly.

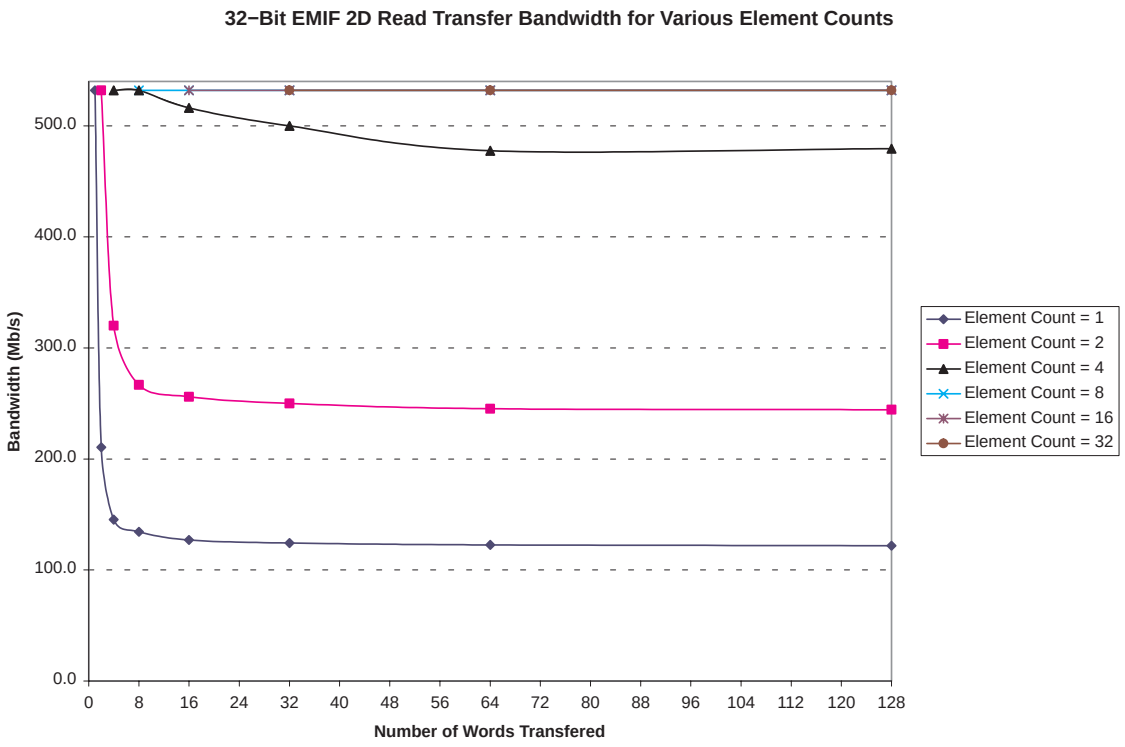


**Figure 2. Transfer Bandwidth for 2-D Reads from a 32-bit SDRAM**

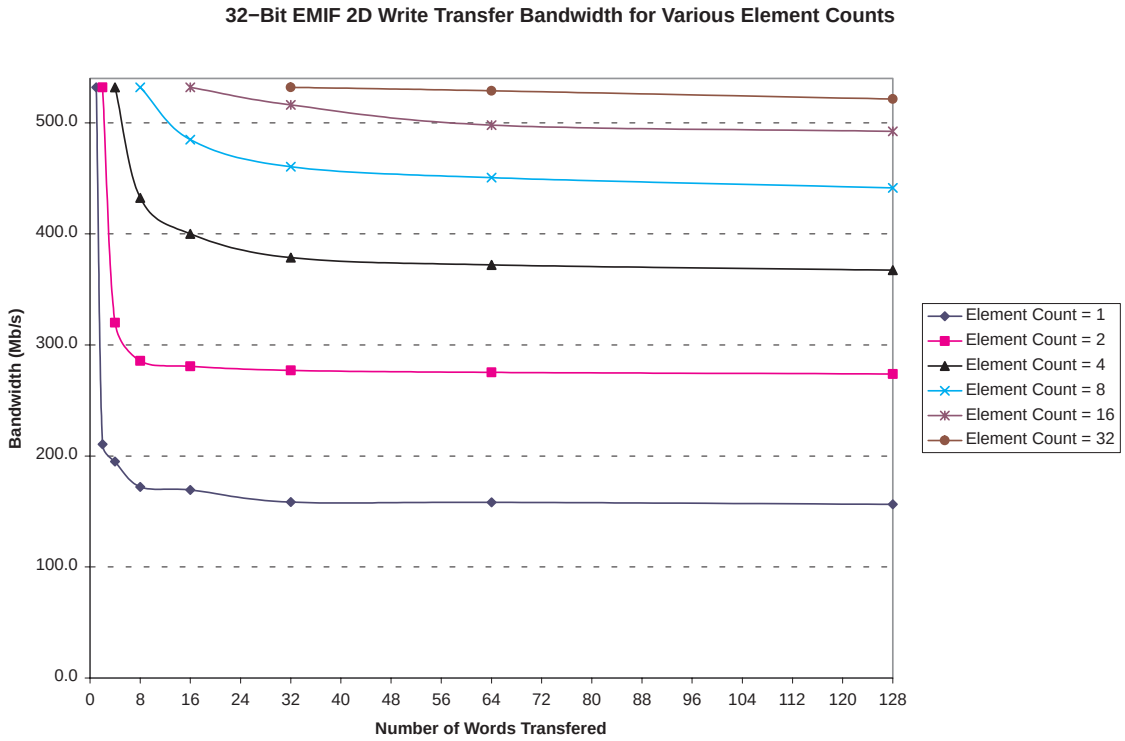


**Figure 3. Transfer Bandwidth for 2-D Writes to 32-bit SDRAM**

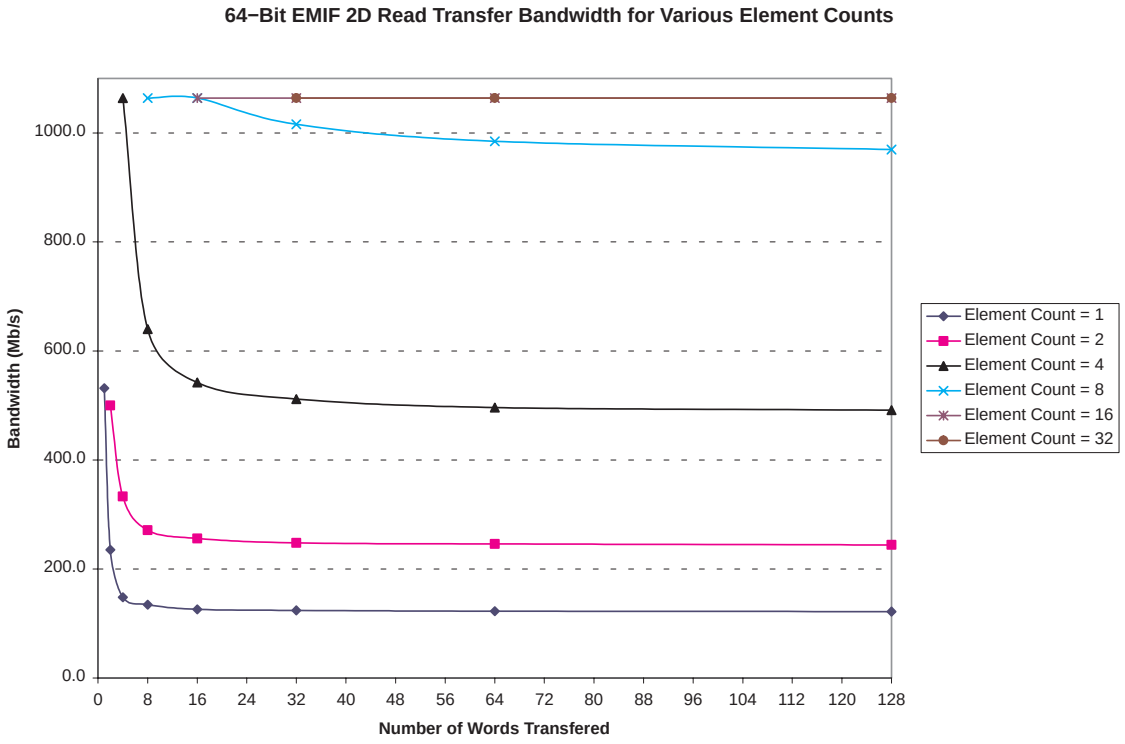


**Figure 4. Transfer Bandwidth for 2-D Reads from a 64-bit SDRAM**

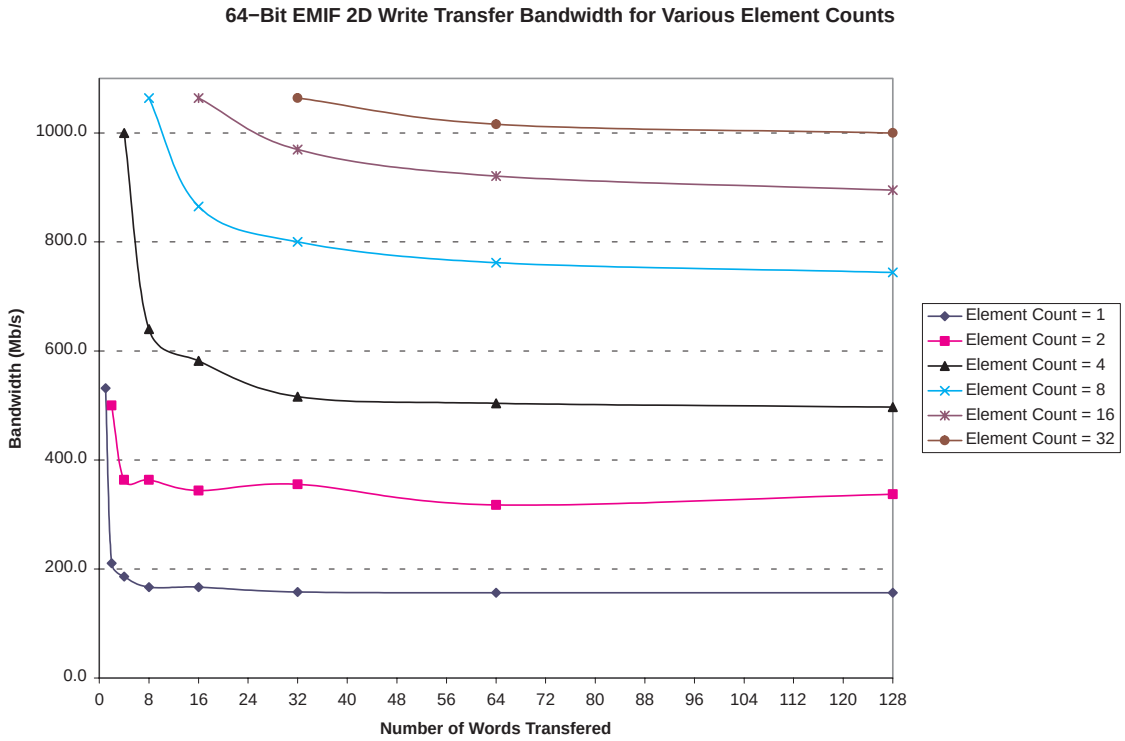


**Figure 5. Bandwidth for 2-D Writes to a 64-bit SDRAM**

### *2.2.4 Simultaneous (2-D) EDMA Transfer Requests*

Two-dimensional transfers offer a great deal of flexibility, and are generally used in systems with multiple transfers competing for resources. The following charts compare total EMIF bandwidth for several 2-D EDMA transfers occurring simultaneously. Various frame and element count combinations were investigated; however, the overall transfer size (element count * frame count) was found to have more bearing than the particular combination settings. Two situations are compared; one where all of the transfers are on the same priority queue, and another where the priority of the transfers is alternated. The results are given in Figure 6 and Figure 7 and represent the expected range of performance for various element/frame count combinations.

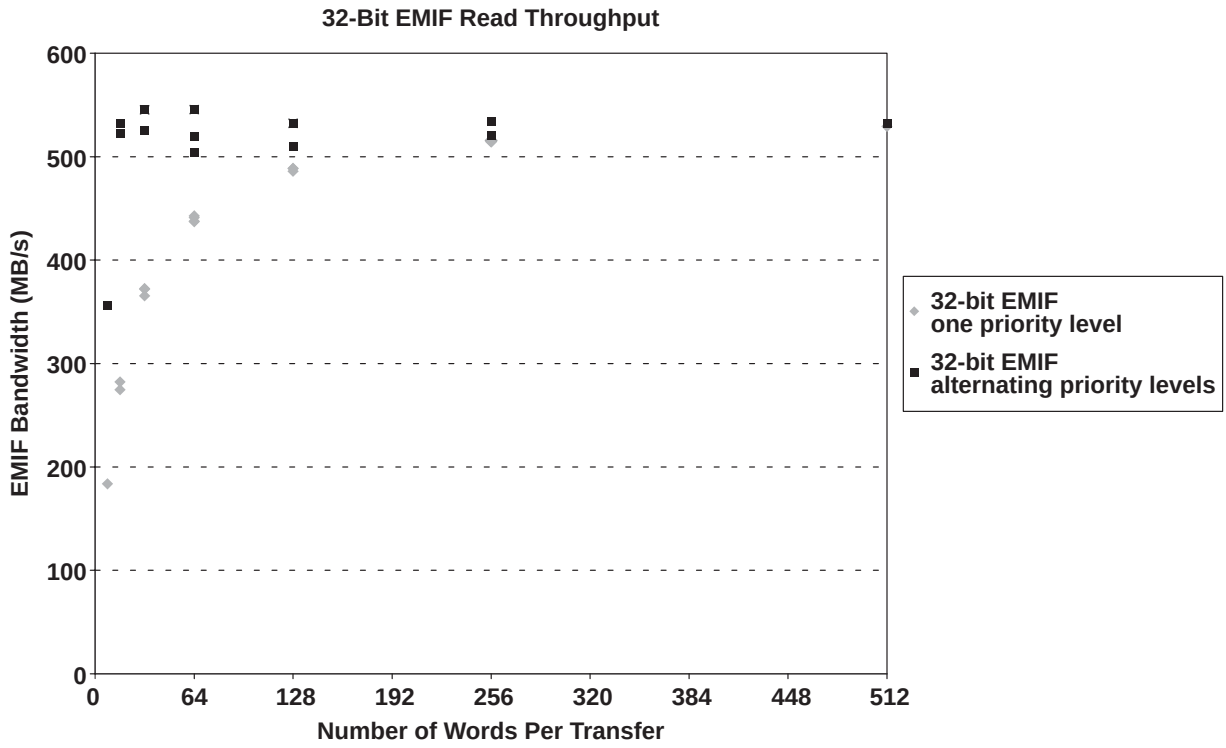


**Figure 6. Read Throughput for Multiple 2-D EDMA Transfers from a 133-MHz, 32-bit EMIF**

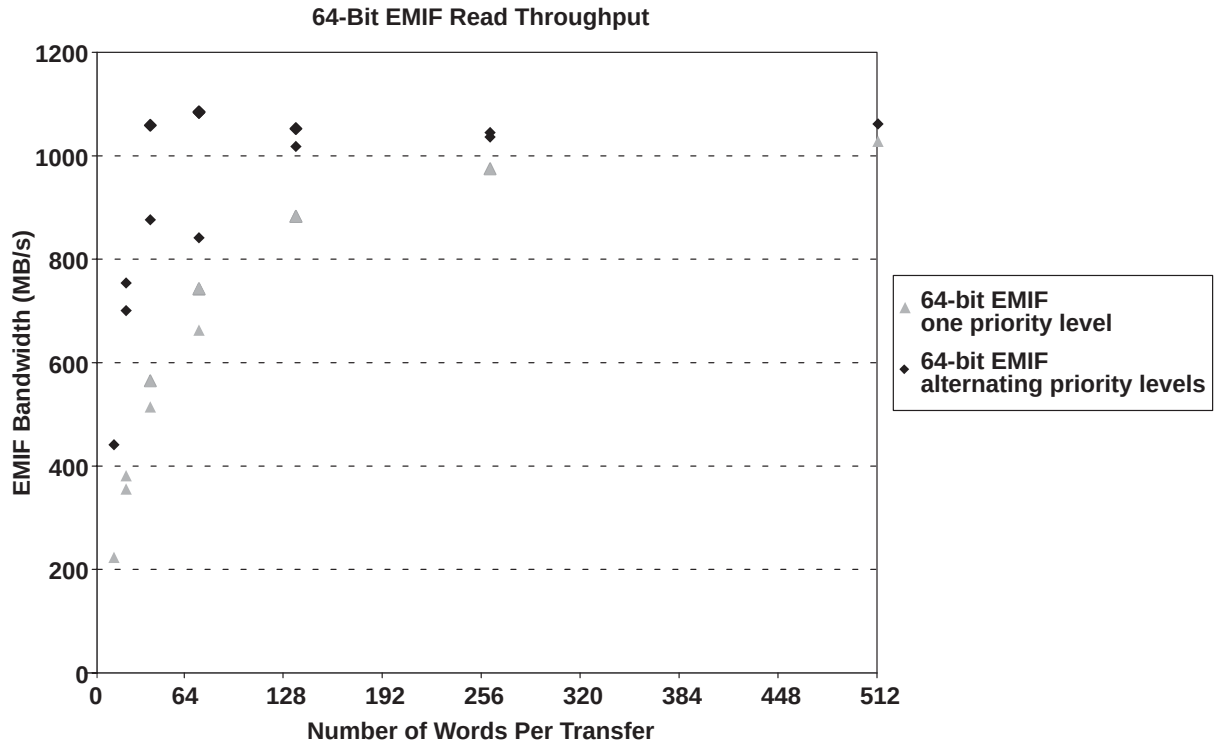


**Figure 7. Read Throughput for Multiple 2-D EDMA Transfers from a 133-MHz, 64-bit EMIF**

The above data clearly show the benefit of using more than one priority level for multiple small transfers. For larger transfers, this is less important; however, keep in mind that transfers on one priority level are processed serially. For writes to the SBSRAM (not presented here), the simultaneous 2-D transfers performed closer to ideal transfer rates due to the write-driven processing of the C64x DSP and the larger write-command buffers of the EMIF. See *TMS320C64x EDMA Architecture* (SPRA994) for more details.

## 2.3 Element Count

Table 4 compares 64-bit, 100 MHz SDRAM accesses to 64-bit, 100 MHz SBSRAM accesses on a 600 MHz C64x CPU. Element counts of 4, 64, 256, and 1024 were used. Note that reads are less affected than writes.

**Table 4. SBSRAM vs. SDRAM**

| Access Type | Memory Type | Total Data (bytes) | EMIF Duration (CPU cycles) | Transfer Bandwidth (Mb/sec) |
|---|---|---|---|---|
| Read | SBSRAM | 16 | 12 | 800.0 |
| Read | SBSRAM | 256 | 192 | 800.0 |
| Read | SBSRAM | 1024 | 768 | 800.0 |
| Read | SBSRAM | 4096 | 3072 | 800.0 |
| Read | SDRAM | 16 | 12 | 800.0 |
| Read | SDRAM | 256 | 192 | 800.0 |
| Read | SDRAM | 1024 | 768 | 800.0 |
| Read | SDRAM | 4096 | 3132 | 784.7 |
| Write | SBSRAM | 16 | 12 | 800.0 |
| Write | SBSRAM | 256 | 192 | 800.0 |
| Write | SBSRAM | 1024 | 768 | 800.0 |
| Write | SBSRAM | 4096 | 3070 | 800.0 |
| Write | SDRAM | 16 | 12 | 800.0 |
| Write | SDRAM | 256 | 204 | 752.9 |
| Write | SDRAM | 1024 | 852 | 721.1 |
| Write | SDRAM | 4096 | 3466 | 709.1 |

## 2.4 EMIF-B

From an EDMA perspective, EMIF-B on the C64x devices is almost identical to EMIF-A. The only difference is the size and number of command buffers. EMIF-B is limited to 8- or 16-bit external memories. The EDMA is optimized to transfer 32-bit elements, however, so transfers by the EDMA to EMIF-B with 16-bit or 8-bit elements will reduce performance and waste bandwidth. Therefore, setting 32-bit transfers in the EDMA options is still the best usage of available bandwidth. EMIF-B will convert these 32-bit words into the appropriate size for the external memory.

Table 5 shows the EMIF-B behavior. A 600-MHz C64x CPU transfers data from/to a 16-bit, 100-MHz SDRAM on EMIF-B to/from internal memory. Note that the SDRAM has an ideal bandwidth of 200 MB/sec and exhibits SDRAM page delays.

**Table 5. SDRAM on EMIF-B**

| Access Type | Data Size (bytes) | EMIF Duration (CPU cycles) | Transfer Bandwidth (Mb/sec) | SDRAM Bandwidth Utilization |
|---|---|---|---|---|
| Read | 64 | 192 | 200.0 | 100% |
| Read | 256 | 768 | 200.0 | 100% |
| Read | 1024 | 3132 | 196.2 | 98% |
| Write | 64 | 204 | 188 | 94% |
| Write | 256 | 851 | 181 | 91% |
| Write | 1024 | 3468 | 177 | 89% |

## 2.5 L2 Bandwidth

The ideal L2 bandwidth of a 600-MHz C64x CPU is 2.4 GB/sec. However, there are many factors that may affect the actual bandwidth attained:

- Ideal bandwidth is determined by CPU frequency – lowering the CPU frequency will lower bandwidth.
- The L2 memory is accessed from the CPU side and from the EDMA side – CPU access to L2 can take L2 bandwidth away from EDMA transfers.
- The L2 memory is cached by the L1 cache – before any data is sent to the EDMA, it must be verified that it is coherent with the L1 cache.

Each of these considerations is examined in the following subsections.

### 2.5.1 CPU and EMIF Frequency

For some CPU and EMIF frequencies, the ideal L2 bandwidth can become the limiting factor in a transfer. The set of tests shown in Table 6 explores this. A 64-bit SBSRAM was used to gather data on a C64x CPU transferring 1024 bytes to/from external memory.

**Table 6. CPU to EMIF Frequency Ratios**

| CPU Frequency (MHz) | EMIF Frequency (MHz) | Ideal Bandwidth | | Actual Transfer Bandwidth (MB/sec) |
|---|---|---|---|---|
| | | EDMA / L2 (MB/sec) | EMIF (MB/sec) | |
| 200 | CPU/4 (50) | 800 | 400 | 400 |
| 300 | CPU/4 (75) | 1200 | 600 | 600 |
| 600 | CPU/4 (150) | 2400 | 1200 | 1200 |
| 200 | 100 | 800 | 800 | 600 |
| 300 | 100 | 1200 | 800 | 800 |
| 600 | 100 | 2400 | 800 | 800 |
| 200 | 133 | 800 | 1064 | 600 |
| 300 | 133 | 1200 | 1064 | 900 |
| 600 | 133 | 2400 | 1064 | 1064 |

### 2.5.2 CPU Activity Affecting L2 Bandwidth

Table 7 summarizes CPU effects on EDMA transfers to/from L2 to/from EMIF. The CPU executes a series of typical (incremental) load or store instructions to L2 memory attempting to reduce the L2 bandwidth available for the EDMA transfer. These tests were performed on a C64x CPU transferring 4 kb of data to/from a 100-MHz, 64-bit SBSRAM (ideal throughput 800 MB/sec).

**Table 7. CPU Effects on L2 EDMA Transfers**

| | | EDMAWEIGHT | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 |
| CPU Speed | CPU Activity | EDMA Transfer Rate (MB/s) | | | |
| 600 | NONE | 800 | 800 | 800 | 800 |
| | LDW | 800 | 800 | 800 | 800 |
| | STW | 328 | 466 | 692 | 800 |
| 300 | NONE | 700 | 700 | 700 | 700 |
| | LDW | 700 | 700 | 700 | 700 |
| | STW | 165 | 235 | 348 | 472 |
| 200 | NONE | 500 | 500 | 500 | 500 |
| | LDW | 500 | 500 | 500 | 500 |
| | STW | 110 | 157 | 232 | 315 |

In most cases, CPU activity did not reduce L2 bandwidth enough to require using the EDMAWEIGHT parameter to guarantee EDMA access to L2. In general, the default value of EDMAWEIGHT (EDMAWEIGHT = 1, EDMA guaranteed to gain access to L2 within 16 CPU accesses) will be adequate for most applications. Some rare types of CPU access to L2 can have negative effects on EDMA transfers to L2. Table 8 shows the results that explore the worst case access pattern. They were obtained on a C64x CPU at 600 MHz transferring 4 kb of data to/from a 100-MHz, 64-bit SBSRAM.

**Table 8. EDMAWEIGHT Effects on EDMA Transfers with Worst-Case CPU Activity (STW on every cycle, accesses spaced 8 words apart)**

| EDMAWEIGHT | Transfer Rate (MB/s) |
|---|---|
| 3 | 800 |
| 2 | 650 |
| 1 | 300 |
| 0 | Blocked |

The CPU performs a store word on every cycle for this worst-case test; adding a few delay slots between the accesses makes a substantial increase in performance. This access pattern is rare and generally is not a concern, but it is something to look into during debugging. Using EDMAWEIGHT to allow access to the L2 prevents this EDMA lockout condition. Other solutions include varying the stride pattern to avoid the 8-word stride, add delay slots, or to write to locations allocated in the L1D cache (recall that the L1D is a read allocated). In nearly every application, however, the default value for EDMAWEIGHT will prevent EDMA lockout from being an issue.

### 2.5.3 Cache Effects on L2 Bandwidth

The L1D cache is always enabled for L2. The status of this cache influences the performance of L2 to EMIF EDMA transfers. L2 data is either invalid (unallocated), clean (allocated and unchanged), or dirty (allocated and changed) in the L1D cache. To test how each of these conditions influences EDMA throughput, a number of test were performed with the entire L1D cache in one of the three states. These tests were performed on a C64x CPU transferring 4 kb of data to/from a 64-bit SBSRAM on EMIF-A.

**Table 9. Cache Effects on L2 EDMA Transfers**

| CPU Frequency | EMIF Frequency | L1D Cache Status | Ideal Bandwidth (MB/sec) | | Actual Transfer Bandwidth (Mb/sec) |
|---|---|---|---|---|---|
| | | | L2 | EMIF | |
| 600 | 100 | Invalid | 2400 | 800 | 800.0 |
| 600 | 100 | Clean | 2400 | 800 | 800.0 |
| 600 | 100 | Dirty | 2400 | 800 | 800.0 |
| 300 | 100 | Invalid | 1200 | 800 | 800.0 |
| 300 | 100 | Clean | 1200 | 800 | 736.6 |
| 300 | 100 | Dirty | 1200 | 800 | 735.3 |
| 200 | 100 | Invalid | 800 | 800 | 610.7 |
| 200 | 100 | Clean | 800 | 800 | 495.7 |
| 200 | 100 | Dirty | 800 | 800 | 495.7 |

In some cases cache status did not reduce L2 bandwidth enough to affect the EDMA transfer.

### 2.5.4 Multiple L2 Transfers

Table 10 examines multiple simultaneous transfers from the L2 memory to different EMIF ports. The CPU and EMIF frequencies are varied to achieve various L2 and EMIF bandwidths. A high priority transfer (A) to a 64-bit memory on EMIF-A competes with a medium priority transfer (B) to a 16-bit memory on EMIF-B.

**Table 10. Prioritized Sharing of L2 Bandwidth**

| | | Ideal Bandwidth (MB/sec) | | | Actual Bandwidth (MB/sec) | |
|---|---|---|---|---|---|---|
| CPU Frequency | EMIF Frequency | L2 | A | B | A | B |
| 600 | 133 | 2400 | 1064 | 266.7 | 1064 | 250 |
| 600 | 100 | 2400 | 800 | 200 | 800 | 190 |
| 399 | 100 | 1596 | 800 | 200 | 800 | 186 |
| 200 | 100 | 800 | 800 | 200 | 624.4 | Delayed |
| 50 | 100 | 200 | 800 | 200 | 162.3 | Delayed |

While the L2 has sufficient bandwidth, both transfers are serviced simultaneously. When the high priority transfer utilizes all the L2 bandwidth, the medium priority transfer is delayed behind it.

Another situation of interest is when two EDMA transfers on different priority queues are competing for the same EMIF port. If the transfers are in the same direction (either both read or both write) then they are arbitrated according to priority. If the transfers are in different directions, however, the transfers are interleaved at the EMIF port and compete for the EMIF bandwidth. If the transfers are smaller than the default burst size (DBS) of the EMIF port (see *TMS320C64x EDMA Architecture*, SPRA994), then this is generally not an issue. For longer simultaneous transfers, the split between the two transfers was found to conform to Table 11.

**Table 11. EMIF Bandwidth for Simultaneous Read and Write Accesses**

| | | % of EMIF BW | |
|---|---|---|---|
| **CPU Frequency** | **EMIF Frequency** | **Read Access** | **Write Access** |
| 600 | 100 | 25 | 75 |
| 400 | 100 | 25 | 75 |
| 300 | 100 | 50 | 50 |
| 200 | 100 | 50 | 50 |

Overall transfer bandwidth was marginally degraded as well, due to transfer switching overhead. Each EDMA transfer would have one to three DBS transfers serviced before switching over to the other transfer. Switching between transfers caused a delay equivalent to the transfer time of 0−5 EMIF words, averaging to around 3 EMIF words per transition (longer for SDRAM due to paging overhead). To avoid this interleaving for time-critical transfers, read and write transfers to the same EMIF port should be placed on the same priority queue.

### *2.5.5 L2-to-L2 Transfers and Cache Effects*

Utilizing the EDMA (or QDMA) to transfer data from the L2 to the L2 isn't the most efficient way to perform internal data transfers. This is because the data is passed from the L2 to the EDMA, and back to the L2. Since the EDMA is a higher level in the memory hierarchy, the L2 controller must verify the L1 cache coherency before sending data to the EDMA, further reducing performance.

The most efficient way to perform L2-to-L2 transfers is to keep the data in the lower memory levels for the duration of the transfer. This can easily be accomplished with the memcpy() function (located in the rts6400.lib library). With this type of transfer, the cache works to increase efficiency, rather then adding overhead as in the EDMA case. There is a trade-off to consider when using the memcpy() function to transfer L2 data, however. The L2 data will be transferred faster, but the CPU is busy transferring data rather than performing computations.

Table 12 contains data from a 1024 byte L2-to-L2 transfer on a 600 MHz C64x CPU with ideal L2 bandwidth of 2400 MB/sec. The cache status column represents the status of the L2 data as cached by L1D.

**Table 12. L2-to-L2 Transfers**

| Transfer type | Cache status | Actual Transfer Bandwidth (MB/sec) |
|---|---|---|
| Memcpy() | Allocated (clean or dirty) in L1D | 2133 |
| Memcpy() | Invalid in L1D | 1422 |
| EDMA / QDMA | Invalid (unallocated) in L1D | 300 |
| EDMA / QDMA | Allocated (clean or dirty) in L1D | 220 |

## 2.6 CPU Accesses and Caching

The following sections examine the access time for various CPU accesses to external memory. It should be noted that using the CPU to access data is generally a bad use of resources and should be avoided. Instead, the EDMA should be given the task of transferring data so the CPU is free to perform actual computation.

When the CPU accesses external memory spaces, a TR may be generated (depending on whether the data is cached). The TR will be for one of the following: a single element – if the memory space is non-cacheable, an L1 cache line – if the memory space is cacheable and the L2 cache is disabled, or an L2 cache line – if the memory space is cacheable and L2 cache is enabled. No TR is generated in the case of an L1 or L2 cache hit.

Incoming and outgoing data to and from the CPU are routed through the L2 controller, and thus utilize the L2 port to submit a TR.

An external memory can be cached by L1 cache, L2 cache, or neither. If the appropriate MAR bit for a memory space is not set, it is not cacheable. If the MAR bit is set and 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 L2 cache size is greater than 0, the external memory space is cached by L2. Remember that L2 is always cached by L1.

The address increment (or memory stride) controls cache utilization. Contiguous LDW or STW accesses (a stride of one word or 4 bytes) 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. Access patterns exploring each of these conditions are covered in the next two sections.

### *2.6.1 CPU Reads (Load Instructions)*

For reads, the CPU will stall, 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 8 and Figure 9 show data collected from a 600 MHz C64x reading from 64-bit, 133-MHz external memories. The time required for 128 LDW (load word) instructions was measured, and the average time for each instruction is reported.

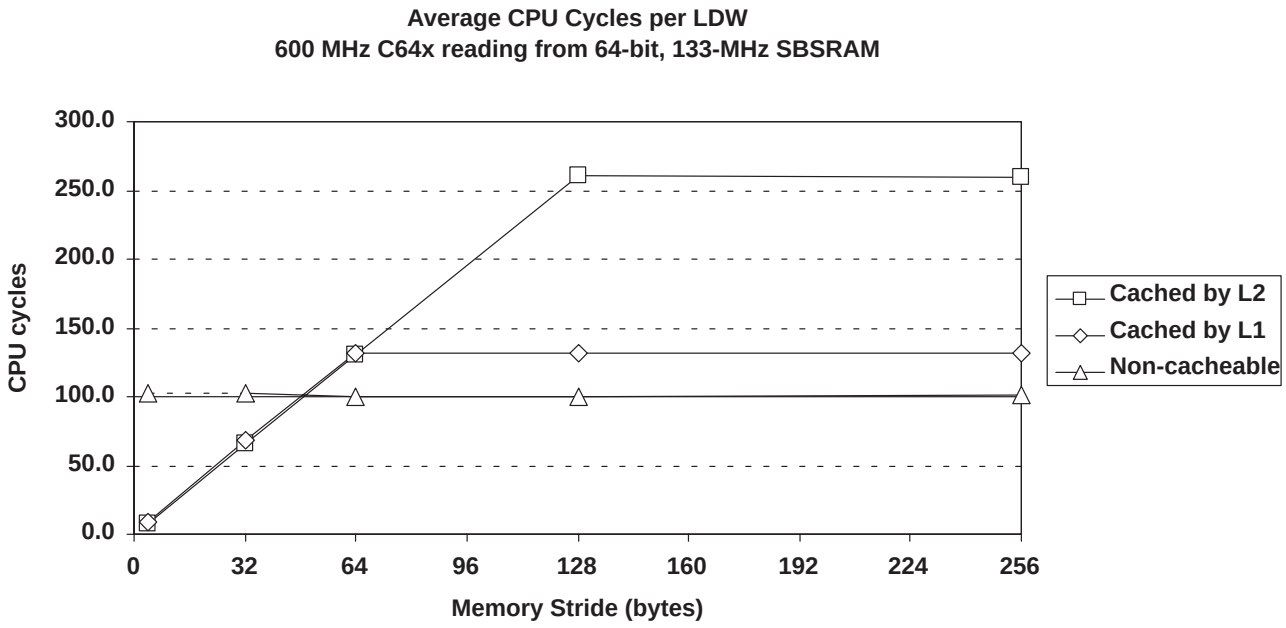


**Figure 8. CPU Reads from SBSRAM**

Delays caused by SDRAM accesses differ from those caused by SBSRAM accesses because of the additional page miss overhead incurred for each TR (cache miss).

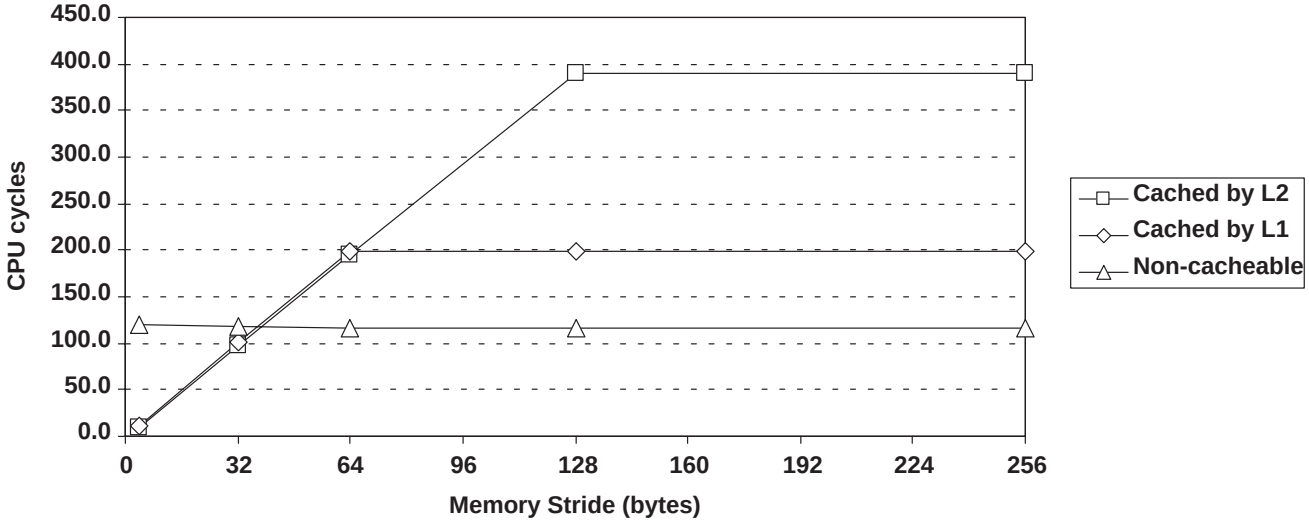


**Figure 9. CPU Reads from SDRAM**

### 2.6.2 CPU Writes (Store Instructions)

For writes, a stall may occur if more than four writes are submitted in quick succession. The length of the stall is equal to the sum of the transfer latency, transfer duration, and some small cache overhead. No data return delay is necessary.

Figure 10 shows data collected from a 600 MHz C64x writing to 64-bit, 133 MHz external memories. The time required for 128 STW (store word) instructions was measured, and the average time for each instruction is reported. Since the L1 is a write-through cache, it has no effect on STW instructions. Therefore, only L2 and non-cached values are shown. Also note that cache writebacks are not included in these measurements.

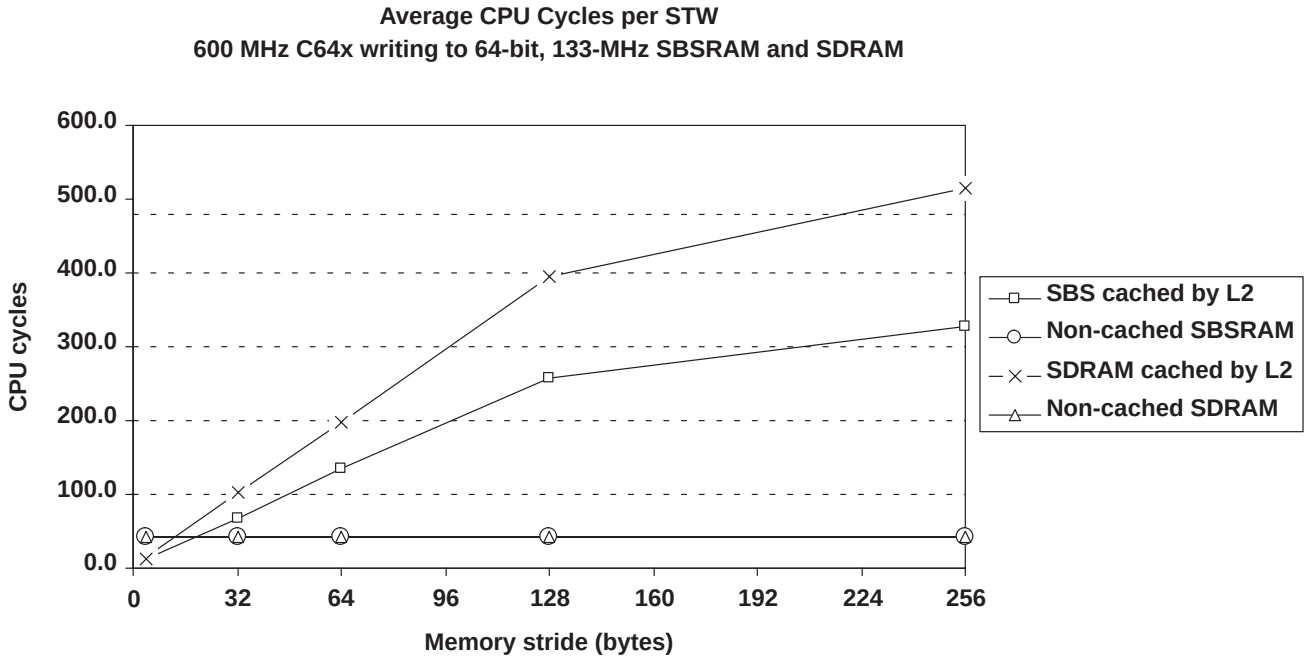


**Figure 10. CPU Writes to EMIF**

## 3 Summary

This document presented data from EDMA throughput trials from a variety of tests and provided insight into the cause of certain performance degradation. The numbers presented represent measured delays under the circumstances described. These should give designers a good starting point for estimating system performance. Actual performance is application-specific and may vary from these numbers, refer to *TMS320C6000 EDMA IO Scheduling and Performance* (SPRAA00), and *TMS320C64x EDMA Architecture* (SPRA994) for more information on creating efficient data transfers in a system.

## 4 References

1. *TMS320C6000 DSP Peripherals Reference Guide* (SPRU190)
2. *TMS320C6000 EDMA Controller Reference Guide* (SPRU234)
3. *TMS320C64x DSP Two-Level Internal Memory Reference Guide* (SPRU610)
4. *TMS320C64x EDMA Architecture* (SPRA994)
5. *TMS320C6000 EDMA IO Scheduling and Performance* (SPRAA00)

## 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 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.

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.

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 |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| | | Telephony | www.ti.com/telephony |
| | | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

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