Application Report
SPRAA00 − March 2003

TMS320C6000 EDMA IO Scheduling and Performance

Jeffrey Ward
Jamon Bowen

TMS320C6000 Architecture

ABSTRACT

The enhanced DMA (EDMA) is a highly efficient and parallel data transfer engine. To make the best use of its resources, it is necessary to understand the architecture and schedule transfers intelligently. This document details how to summarize, analyze, and schedule system traffic to produce efficient designs. An example audio/video system is presented and analyzed in full. Finally, EDMA performance is discussed in terms of the bandwidth of the EDMA and of the various peripherals.

Contents

List of Figures

List of Tables

1 Introduction

To design a highly efficient, bandwidth-maximizing system using the TMS320C6000™ devices, it is important to schedule system traffic in an intelligent manner. This document will walk you through the process of proper IO scheduling and provide some examples.

1.1 EDMA Architecture

Prior to setting up the data movement of a system, it is important to have a working knowledge of the architecture of the enhanced DMA (EDMA) transfer engine. The EDMA architecture controls how multiple transfers (from multiple transfer requestors) interact with one another, impacting the overall system performance. For details about the EDMA architecture of the TMS320C64x™ DSP, see *TMS320C64x EDMA Architecture* (SPRA994). For details about the EDMA architecture of the C621x™/C671x™ DSP, see *TMS320C621x/TMS320C671x EDMA Architecture* (SPRA996).

This document assumes that you have read the aforementioned EDMA architecture document pertaining to your device, as architectural features will be referenced without extensive explanation.

2 Proper IO Scheduling

Proper IO scheduling within a system involves setting up data transfers in an efficient way, on defined priority levels, and at certain times or intervals. The goal is to maximize bandwidth utilization, minimize transfer blocking, and to create a system that effectively utilizes the highly parallel and powerful architecture of the TMS320C6000 DSPs.

It is typical to consider the worst-case scenario when performing IO scheduling. From a system traffic perspective, the worst-case scenario is when the most sensitive transfers experience the longest delays due to interference from other transfer requests (TRs). The amount of interference can be affected by scheduling the priority and order of the interfering TRs. To ensure that the real-time deadlines of a system are met, it is necessary to predict the interference caused by the other transfers in different scheduling scenarios.

The TMS320C6000 DSP has many programmable features to alter transfer scheduling:

- Cache transfers to EMIF and have programmable priority (varies by device).
- Master peripheral (HPI, PCI, EMAC) accesses have programmable priority (varies by device).
- EDMA channels are extremely flexible. They are able to transfer data on user defined schedules, priorities, and in various ways.
- Quick DMA (QDMA, which is used for direct CPU DMA requests) transfers have programmable priority, and transfer size flexibility.

All transfer traffic is handled by the EDMA transfer controller. Inside the transfer controller there are several interaction points for transfers. Understanding how the transfers interact at these points, and ways to alter the interaction, assist proper transfer scheduling. While the EDMA is a very efficient transfer engine, improper use of its resources can unnecessarily inhibit performance.

There are four basic steps in proper IO scheduling: defining, summarizing, analyzing, and scheduling system traffic. Each of these steps is examined in the following sections.

2.1 Defining System Traffic

Data transfers are implemented in the DSP in the following way:

- System events cause transfer requestors to submit one or more TRs to the EDMA priority queues.
- TRs await processing in the queues, and become active when they enter the EDMA queue register sets.
- An active TR submits read commands (to the source peripheral port) and write commands (to the destination peripheral port) for small bursts of data to be transferred.

From the perspective of the EDMA transfer controller, system traffic is comprised of TRs. TRs are generated by system-level requests (cache and host/master servicing), user-programmed requests (QDMA, EDMA), and cache update requests.

For the purposes of scheduling system traffic, each TR can be defined by its source port, destination port, transfer size, periodicity, deadline, ideal transfer latency, ideal transfer duration, and the stage(s) of operation in which it occurs.

2.1.1 TR Generation

To define system traffic, it is important to know when and how TRs are generated in the DSP. There are several transfer requestors that submit TRs to the EDMA:

- The L2 controller submits TRs for all cache, CPU, and QDMA transfers.
- The master peripheral submits TRs based on its internal activity (such as host driven accesses for PCI and HPI, or data packet arrival for EMAC).
- The EDMA channel controller submits TRs based on internal and external events and the programming of the EDMA registers and parameter RAM.

Full details on TR submission (including when they are submitted and how many elements are transferred per TR) can be found in the appropriate EDMA architecture document for each device.

2.1.2 Stages of Operation

In any system there may be different stages of operation, such as boot-up, stand-by, power-down, and any number of stages in between. A data transfer (and its corresponding TR) is generally confined to one or more stage(s) in which it occurs. The purpose of these stages is to realize that some transfers will not interfere with each other because they do not occur in the same stage.

The worst-case scenario may have to be considered in each stage. It is common to first schedule transfers that occur in the stages with the most traffic, then to schedule the other stages based on any overlapping transfers from the busiest stage.

If a recurring transfer exists in multiple stages of operation some properties, such as priority, can be changed from one stage to the next. It is possible to change a recurring transfer's priority between different stages through the following:

- For EDMA/QDMA, priority can be modified on the fly via the priority bits in the options parameter.
- The priority of master peripheral transfers can be modified (C64x only) by changing the TRCTL register value. Special care must be taken when changing the value of this register, however, see the reference guide for the peripheral for details.
- For cache/CPU accesses, priority can be changed between stages of operation (when all cache operations have flushed) via the CCFG register (C64x only).

The TR priority level cannot be changed mid-transfer. The changes above will only place subsequent TRs on the new priority level. Changing priority in these ways makes it is possible to schedule transfers differently among the stages of operation.

2.1.3 Source and Destination Ports

As of this writing, there are up to seven peripheral ports that handle data transfers directed by the EDMA's transfer controller: the L2 memory controller, McBSP/McASP peripherals, UTOPIA peripheral, master peripheral, video peripheral, EMIF A, and EMIF B (peripheral availability varies by device family). The memory-mapped peripheral architecture of the TMS320C6000 DSP translates different memory addresses to ports inside the EDMA. Data transfers are defined by their source and destination address, which correspond to different source/destination ports within the DSP. The ports interface with the EDMA and contain internal buffers to isolate transfer data on the peripheral side. The port architecture allows the EDMA to service multiple transfers at the fast clock rate of the EDMA, while the ports buffer data transfers to the peripherals, which generally operate at some lower clock rate. The appropriate EDMA architecture document explains peripheral ports in detail for each device.

If a peripheral does not have a port (such as the I2C peripheral), it is because it is not considered to be a high-speed peripheral. Transfers to these peripherals are routed through the L2 controller to the peripheral configuration bus.

2.1.4 Transfer Duration and Latency

Actual transfer duration and latency cannot be calculated exactly prior to IO scheduling. This is because transfers interact with each other in a variety of ways, changing duration and latency. Transfer duration and latency can vary depending on other traffic in the EDMA transfer controller. For scheduling purposes, the ideal, uninterrupted values are used in interference calculations to determine the actual transfer duration and latency; later these values are verified in the system.

Figure 1 details the important time intervals associated with EDMA transfers, including the transfer latency and duration. TL is transfer latency, defined as the time from event to the beginning of transfer at the source peripheral. SD is source duration, or the interval of time during which the source peripheral reads data. DD is destination duration, or the interval of time during which the destination peripheral writes data. TD is transfer duration, defined as the time from the first read at the source to the last write at the destination. OFF is the source/destination offset.

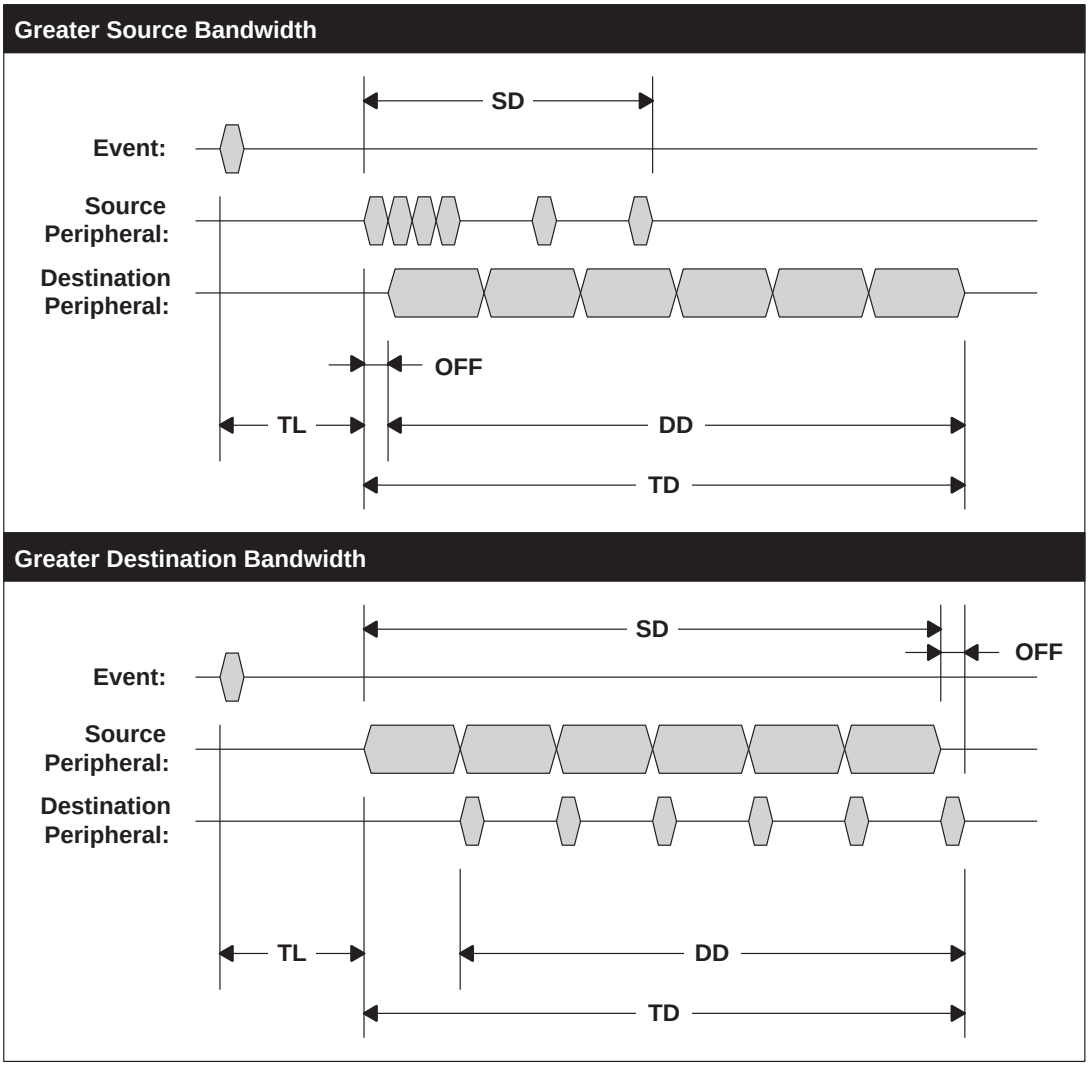


Figure 1. Important Transfer Time Intervals

2.1.4.1 Transfer Duration: Greater Source Bandwidth

When transferring from a source with greater bandwidth, transfer duration is equal to the destination duration plus the offset. In this case, the destination duration is equal to the transfer size divided by the destination bandwidth (the limiting bandwidth factor). The offset is equal to the initial burst size divided by the source bandwidth, where the initial burst size has an upper limit of the source read command buffer size.

For example, suppose a 600-MHz C64x DSP is transferring 64 bytes from the L2 to a 64-bit, 100-MHz SBSRAM on EMIF-A. The SBSRAM has a bandwidth of 800 MB/sec while the L2 has a faster bandwidth of 2.4 GB/sec. The initial burst size is 2 words (L2 read command buffer size), so the offset is 3.33 nS. The destination duration is 80 nS. Thus, the transfer duration is 83.33 nS in the ideal case.

NOTE:

Bandwidth is calculated by: $\frac{64\ bits}{cycle} \cdot \frac{1\ Byte}{8\ bits} \cdot 100\frac{Mcycles}{\text{sec}} = \frac{800\ MBytes}{\text{sec}}$, other bandwidth calculations are done in a similar manor.

Offset time calculation: $2\ Words \cdot \frac{4\ Bytes}{Word} \cdot \frac{\text{sec}}{2.4\ GBytes} = 3.33\ nsec$, other time calculations are done in a similar manor.

Destination duration is calculated by $\frac{64\ Bytes}{800\ MBytes/sec} = 80\ nsec$, other duration calculations are done in a similar manor.

2.1.4.2 Transfer Duration: Greater Destination Bandwidth

When transferring to a destination with greater bandwidth, transfer duration is equal to the source duration plus the offset. In this case, the source duration is equal to the transfer size divided by the source bandwidth (the limiting bandwidth factor). The offset is equal to the size of the final burst of data to the destination port divided by the destination bandwidth. The final burst size has an upper limit of the destination port write command buffer size.

For example, suppose a 600-MHz C64x DSP is transferring 64 bytes from a 32-bit, 133-MHz SBSRAM on EMIF-A to the L2 memory. The SBSRAM has a bandwidth of 533 MB/sec while the L2 has a faster bandwidth of 2.4 GB/sec. The final burst size is 2 words (L2 write common buffer size), so the offset is 3.33 nS. The source duration is 120 nS. Thus, the transfer duration is 123.33 nS in the ideal case.

2.1.4.3 Bandwidth Consideration

The transfer bandwidth is equal to the bandwidth of the slower of the source or destination resource.

From an overall bandwidth perspective, these transfer duration estimates assume that the EDMA is able to maintain the transfer bandwidth, and that the only limiting factor is the speed of the resources involved. The EDMA performance measurements below compare ideal to actual bandwidth under various conditions.

2.1.4.4 Transfer Latency

Transfer latency is the delay from an event to the start of the transfer. In the ideal case (no TRs currently queued on the same priority level) transfer latency is based on the propagation delay from the event to the start of data movement at the source peripheral. This can be affected by the readiness of the source, destination, and the transfer requestor, as well as the status of the priority queue.

This propagation delay varies by device. See *TMS320C64x EDMA Performance Data* (SPRAA02) or *TMS320C621x/TMS320C671x Performance Data* (SPRAA03) for information on propagation delay.

2.1.5 Deadlines and Periodicity

The deadline is the time by which the TR must be complete. TR deadlines can be defined by a number of system-specific performance requirements and deadlines defined by external hardware or transfer standards. It is important to meet standards or hardware requirements for the DSP to interface with other devices in a system. Transfers with specific hardware requirements are those that service data flow in and out of the chip, such as serial port servicing. If a transfer doesn't have an associated hardware restriction, other factors such as desired system performance may define a deadline.

Deadlines are often dictated by periodicity. On average, if an event must be serviced every N CPU cycles, then the average deadline is also N CPU cycles. Buffers allow short-term deadlines to exceed average deadlines, as the example below demonstrates. When considering the worst-case scenario, it is often permissible to consider the short-term deadline if it can be shown that there is always sufficient recovery time before the following worst-case scenario. (Keep in mind that, during this recovery period, the transfer will have to slightly outperform the average deadline.)

For example, assume the video port requires 720 bytes of data every 35 uS. The average deadline for this transfer is 35 uS. However, the video port has an internal, self-maintained buffer. Assume a transfer is triggered whenever the buffer level drops below 4096 bytes. While an average deadline of 35 uS must be maintained, any single TR has a short-term deadline of about 199 uS (35 uS x 4096/720), because 4096 bytes are in the buffer.

The deadline will be the measure of proper IO scheduling. Once traffic is scheduled, if deadlines are not met, traffic is re-evaluated.

If a TR has no defined periodicity, its periodicity is said to be random. However, when estimating the worst cases, it can be useful to consider that a TR of random periodicity may have a minimum recurrence interval.

For example, the L2 controller will submit cache TRs for only one cache action (allocate, evict) at a time. The minimum recurrence interval is the time it takes the cache TR to complete (transfer latency plus transfer duration). For a single L1 cache line (64 bytes) allocation on a 600-MHz C64x CPU from a 133-MHz, 64-bit SBSRAM, this is about 150.8 nS.

2.2 Summarizing System Traffic

Summarizing system traffic into tables helps analysis of the transfers and scheduling traffic. Example tables are shown below in the example section.

To summarize system traffic, list each TR and its properties in a table. Include columns that define the transfer's description, stage of operation, TR requestor, source port, destination port, transfer size, and periodicity. This is the system traffic summary (see Table 1 in the example section).

Also, create a table of system transfers that shows which transfers utilize which source and destination ports. Create one row for each port in use, and one column for each TR. Under each TR, place 'read' in the row of the port from which the TR reads, and 'write' in the row of the port to which the TR writes. This is the port usage summary (see Table 2 in the example section).

Finally, create a table that compares transfer size, transfer bandwidth, ideal transfer latency and duration, deadlines (both short-term and average), and delay tolerance. Delay tolerance is equal to the short-term deadline minus the sum of ideal transfer latency and duration. This is a table of temporal TR characteristics (see Table 3 in the example section), and it is helpful in determining which transfers are most sensitive to interference.

When examining Table 3 in the example section, note that transfer bandwidth for non-EMIF peripheral ports is higher than expected for the peripherals. This is because transfers from most peripheral ports consist of short bursts from an internal buffer with long delays while the buffer is refilling. So from an internal perspective, the transfer bandwidth is the high internal rate (at EDMA speed).

2.3 Analyzing System Traffic

System traffic should be analyzed from a number of perspectives. Keep in mind that the overall goal is to meet deadlines by minimizing TR interference, which will increase system performance.

Note the three places transfers are arbitrated in the EDMA: at the transfer request nodes, in the priority queues, and in the queue register sets. The EDMA architecture documents explain the arbitration process at these places in detail.

2.3.1 Transfer Duration and Interference

Transfers with relatively long durations have a greater possibility of interfering with other transfers. Care should be taken to insure delay sensitive TRs are not submitted behind long transfers on a given priority queue. To avoid this, delay sensitive TRs should not be assigned to the same priority level as long transfers. If a large transfer is considered high priority, it may be best to split it into multiple, shorter transfers. For an example of this, see the TMS320C6000 DSP EDMA Controller Reference Guide (SPRU234).

An Active TRs using the same ports as a long transfer may be delayed by the long transfer. This concept of port blocking is explained in the EDMA architecture documents. If an active TR is of lower priority, it may be delayed for the entire duration of the long transfer. If an active TR is of higher priority, it may be delayed for a short time waiting for command buffers to flush. Regardless of priority, transfers may share port bandwidth as described below, or as dictated by read/write parallelism (see the EDMA architecture documents for details).

2.3.2 Sharing Peripheral Port Bandwidth

2.3.2.1 TRs Issuing the Same Command Type

Sometimes active TRs on different priority levels can share the bandwidth of a peripheral port. This occurs when a transfer is not taking up the full bandwidth of a port. If this is the case, the excess bandwidth of the port may be used by a lower priority transfer in between bursts for the higher priority transfer. The port buffers are designed to insure that delays are not experienced by the higher priority transfer and increase EDMA bandwidth available to other transfers.

For example consider a traffic situation on a TMS320C6416 DSP. If an urgent priority (Q0) transfer is reading from L2 and writing to EMIF-B, the L2 bandwidth is not fully utilized. The L2 port is busy for only a fraction of the duration of the transfer. A lower priority transfer from L2 to EMIF-A could be interleaved with the above transfer, taking up any remaining L2 bandwidth.

If the sum of the ideal individual bandwidths of the concurrent transfers approaches or exceeds the total bandwidth of a shared port, the transfer of lowest priority is throttled back to maintain the higher priority transfer(s). See the EDMA performance documents for more information.

2.3.2.2 TRs Issuing Different Command Types

Sharing port bandwidth can also occur in the case of read/write parallelism (see the EDMA architecture documents for details). If a port is receiving read commands and write commands concurrently, that port will service the commands in the order of their arrival. The bandwidth of the port will be split between the reading TR and writing TR. This is primarily a concern for EMIF memories because of their relatively slow bandwidths and high usage in a typical system.

For example, consider a situation on a C64x CPU at 600 MHz. An active TR on priority Q2 is reading from L2 and writing to a 64-bit 100-MHz SBSRAM on EMIF-A. Another active TR on priority Q0 is reading from the SBSRAM and writing to the video port. The ideal transfer bandwidth of each transfer is 800 MB/sec (limited by the SBSRAM bandwidth), but they will actually share the bandwidth of the SBSRAM. See the EDMA performance documents for details on the bandwidth split.

This also applies to a single TR reading from and writing to a single peripheral port, such as an L2 to L2 transfer.

2.3.3 Schedulable With Respect to Time

Transfers which can be triggered at times specified by the system designer are schedulable with respect to time. These types of transfers increase the flexibility of the system by allowing the designer to manually avoid possible TR interference.

Most transfers can be forced onto a time schedule. For example, transfers can be scheduled by:

- Configuring the timer peripheral to trigger a transfer or series of transfers.
- Using transfer A to trigger transfer B via chaining or linking.
- Placing two transfers on the same priority level will, in the case that both TRs are submitted in close proximity, cause them to occur serially, thus “scheduling” the second after the first.

If the above methods of scheduling with respect to time are insufficient for a particular transfer, the following methods offer more control. The DSP has added flexibility because its resources are memory-mapped, allowing EDMA transfers to define and modify operation on the fly. In an extreme case, a designer may rely on these mechanisms to schedule transfers:

- The CPU/Timer/event triggers an ISR that polls serial ports, buffers, etc. and triggers transfers intelligently according to some set of rules. For example, if two transfers may interfere with each other, submit only one at a time.
- The chain and link mechanisms can be used to schedule transfers. The chaining feature can start a preprogrammed EDMA transfer upon the conclusion of a transfer. The linking feature can be used to load a new set of parameters for the EDMA transfer upon its completion, and subsequent events for the channel will trigger the new transfer.
- Transfers can enable other transfers by directly modifying the PaRAM, or a memory-mapped control register. For example, transfer A does one of the following: 1) updates another transfer EDMA parameter set or 2) updates the EDMA channel enable register.

There are always tradeoffs when scheduling transfers in these ways. Sometimes the overhead involved in scheduling transfers outweighs the benefit of traffic flexibility. In other instances, a transfer is initiated by a host or the cache controller or has random periodicity. In those cases, a transfer is not considered to be schedulable with respect to time (though they may still have schedulable priority).

2.3.4 Schedulable With Respect to Priority

The priority scheme in the EDMA is meant to allow time-critical transfers to take precedence over others. Here are a few key things to remember when considering assigning priority.

- When two active TRs target the same source or destination port, priority determines which transfer takes precedence.
- TRs are sorted into queues based on priority, so transfers of the same priority level occur serially.
- A priority setting does not affect the speed at which a transfer occurs. A high or low priority transfer, uninterrupted by others, will have the same transfer duration.

2.3.5 Transfer Requestor Priority Allocation

Each transfer requestor has a limited number of TRs that it can submit to each priority queue. This is dictated by the priority queue allocation registers (PQARs). Programmability of this limit varies by requestor and device family. For details on programming the PQARs, see the appropriate EDMA architecture document.

2.4 Scheduling System Traffic

The above analysis gives insight into transfer interaction and helps define a starting place for scheduling. With that insight, TRs can be intelligently scheduled. Once scheduled with respect to priority and time (where applicable), use transfer timetables (described below) to determine the worst-case interference. If deadlines are met under the worst-case conditions, then scheduling is successful.

2.4.1 TR Timetables

A transfer timetable is a graphical tool that will help in determining the interference between TRs and actual transfer latency and duration (see example timetable in Figure 2). After determining the interference between TRs in the worst case and calculating the resulting transfer latency and duration, transfer deadlines can then be assessed.

The x-axis represents time, and the y-axis has 4 divisions that represent priority level (C64x only). TRs will be depicted on their respective priority levels.

Transfer latency represents the first stage of a TR which should be shown on the timetable. The TR can be delayed during this stage if its requestor is stalled (a situation which should be avoided), and if it is waiting behind other TRs in the same priority queue.

Transfer duration represents the second stage of a TR which should be shown on the timetable. The TR can be delayed during this stage by other active TRs according to the priority scheme, read/write parallelism, or port blocking.

When a TR is actively using a port, keep track of how much bandwidth it utilizes. When multiple TRs access the same port, the bandwidth sharing section above describes how bandwidth is split between them. The bandwidth available to a transfer will determine its progress and affect its duration. When the behavior of the EDMA is in question, refer to the EDMA architecture document. Understanding the priority queues, queue registers, commands, ports, and command buffers is the key to understanding how transfers interact.

The timetable does not need to cover every possible TR interaction. It only needs to show the worst cases for each TR (or each delay-sensitive TR).

2.4.2 Finding the Worst-Case Scenario for a TR

For the TR in question (TRIQ − the most delay sensitive TR), the worst case is the scenario in which it is delayed the most by other TRs. It must be assumed that other TRs are interfering with the TRIQ in the worst possible way (except for those TRs guaranteed not to interfere by scheduling with respect to time or by other constraints).

Create a timetable depicting the worst-case scenario. Start with the TRIQ. Show the ideal transfer latency and duration on the timetable. Review the three places TRs interfere, and determine the worst combination of other TRs that will delay the TRIQ in each place. The following guidelines will help in determining the worst-case scenario:

- The transfer requestor nodes:

  TR nodes can stall if they submit more than their allotted limit to any of the priority queues. When stalled, no TRs are submitted to any priority level. This scenario should be avoided by properly programming the priority queue allocation registers.

  To check for stalls, assume that the requestor of the TRIQ has submitted all possible TRs on any priority level, just before the TRIQ is submitted.

  Has the requestor exceeded its limit on outstanding TRs? If so, reevaluate the priority queue allocation register programming and transfer priority levels to avoid requestor stalls.

- The priority queues:

  TRs are serviced serially in the FIFO priority queue.

  Assume that all requestors have submitted all possible TRs on the same priority level as the TRIQ, just before the TRIQ is submitted.

  Is the TRIQ stalled behind any other TRs in the queue ahead of it? Keep in mind that those TRs may be delayed by others.

- The queue registers (active TRs):

  Active TRs on different priority levels are those being processed in the queue registers. These active TRs can interfere according to the priority scheme, read/write parallelism, or port blocking.

  Assume that all other TRs (on any priority level) using the same resources as the TRIQ (source/dest ports) were submitted just before the TRIQ. Delay the TRIQ with port blocking and read/write parallelism as much as possible.

  Is the TRIQ delayed beyond its deadline by other active TRs which share/block its resources? If it is then traffic needs to be re-evaluated.

Add TRs to the timetable according to the above worst-case interference guidelines, accounting for interference in the latencies and durations of all TRs on the timetable. All other TRs need not be added; only those which interfere with the TRIQ.

Also account for any transfers that can recur multiple times within the timeframe of the single TRIQ. For example, if the TRIQ is a relatively long (20 uS) data transfer from EMIF, it may be interrupted multiple times by a QDMA transfer recurring every 3 uS.

Once the timetable is complete, determine the actual transfer latency and duration of the TRIQ in the worst case, and compare this to its deadline. If any deadlines are violated, traffic scheduling must be re-evaluated.

3 Proper IO Scheduling – Example System

3.1 System Definition

In this example system a 600-MHz DM642 DSP is used to process video. It receives compressed audio/video data from a PCI host, decompresses and processes this data, delivers the uncompressed video to the video port for display, and sends the audio output to the audio codec via the McBSP. The program code is on-chip, and data is stored in a 133-MHz 32-bit SDRAM connected via the EMIF. This SDRAM has an ideal bandwidth of 533 MB/sec.

The compressed audio/video input data is a fixed-rate 2 Mbps stream. The PCI host sends data in 128-byte (32-word) blocks every 488 uS. The PCI transfers this data to a buffer, located in SDRAM. The PCI peripheral submits TRs after every 8-words received, for a total of four TRs for the 32 word block. Transfers from a PCI host operating at 33 MHz to 133 MHz SDRAM should take 9-PCI cycles (272.7 nS) for each 8-word transfer (source: *TMS320C64x DSP Peripheral Component Interconnect (PCI) Performance*, SPRA965). The TMS320C64x PCI contains a 16 word write FIFO to buffer the PCI data stream that should allow this transfer speed to be sustained. However, if internal traffic prevents these transfers from being serviced at this speed, then the PCI transfer will stall and wait states will be inserted. So the only deadline for the PCI is the 488 uS repeat rate of the input data.

The output video format is 720x480 video at 30 fps. In this video format each pixel is 16 bits deep, making each video frame 675 kb. To sustain this output, the video port is configured to request 720 byte blocks of data (half a video line) every 34.72 uS. The video port has an output buffer that it will maintain at a level of 4400 bytes, so the maximum short-term deadline (assuming a full buffer) is about 212 uS.

The video decompression algorithm uses the QDMA for all data transfers, and accesses data in 16x16 pixel blocks (single 2-D TR for all 256 pixels). Processing a single video frame of output data requires the algorithm to transfer, in the worst case, 6 video frames worth of data (4050 kb) between buffers and internal memory for motion estimation purposes. This worst-case would transfer one block (512-byte TR) every 4.12 uS; which is the short-term deadline for the algorithm transfers. These accesses may be reads (EMIF to L2) or writes (L2 to EMIF). When determining the worst case, the type of access that causes the longest delay will be assumed.

The audio output format is 44 kHz, 16-bit stereo audio. Left and right channel samples are sent to the audio codec via the McBSP. One word (both left and right samples) is transferred every 22.72 uS to support this format.

The audio decompression algorithm uses the QDMA for all data transfers, and sends data in 8-sample blocks (16-byte transfers). Assume that in the worst case it must transfer 880 kb to decompress a second worth of audio output data. In this worst case, the algorithm would submit TRs for 16-byte transfers every 17.76 uS. These transfers may be reads (EMIF to L2) or writes (L2 to EMIF). When determining the worst case, the type of access that causes the longest delay will be assumed.

The data in SDRAM will be defined as non-cacheable, as the algorithms access the external buffers in a somewhat random fashion.

Assume the decoding stage of operation is the only one with multiple sensitive transfers occurring, therefore it will be the only stage considered during IO scheduling.

3.2 Video System Traffic

Summarizing system traffic yields Table 1 and Table 2.

Table 1. Video System Traffic Summary

	Incoming Stream	Outgoing Video	Outgoing Audio	Video Algorithm QDMAs	Audio Algorithm QDMAs
TR requestor	PCI node	EDMA channel node	EDMA channel node	L2 controller node	L2 controller node
Source port	PCI port	EMIF (SDRAM)	EMIF (SDRAM)	L2/EMIF	L2/EMIF
Destination port	EMIF (SDRAM)	Video peripheral	McBSP	EMIF/L2	EMIF/L2
Transfer size (per TR)	32 bytes	720 bytes	4 bytes	512 bytes	16 bytes
Periodicity	4 TRs recurring every 488 uS	1 TR every 34.72 uS	1 TR every 22.72 uS	Irregular: 1 TR (min. recur: 4.12 uS)	Irregular: 1 TR (min. recur: 17.76 uS)

Table 2. Video System TR Port Usage

Port	Incoming Stream	Outgoing Video	Outgoing Audio	Video Algorithm	Audio Algorithm
L2 Controller				R/W	R/W
EMIF-A	WRITE	READ	READ	W/R	W/R
PCI	READ				
Video		WRITE			
McBSP			WRITE		

Table 3 compares the TRs temporal characteristics. Transfer duration is found by finding the duration of the transfer for the slower port, then adding in the offset.

Table 3. TR Temporal Characteristics

	Incoming Stream	Outgoing Video	Outgoing Audio	Video Algorithm	Audio Algorithm
Transfer Size	32 bytes	720 bytes	4 bytes	512 bytes	16 bytes
Source Bandwidth	PCI: 1.2 GB/sec	EMIF: 533 MB/sec	EMIF: 533 MB/sec	533 MB/sec or 2.4 GB/sec	533 MB/sec or 2.4 GB/sec
Destination Bandwidth	EMIF: 533 MB/sec	VP: 2.4 GB/sec	McBSP: 1.2 GB/sec	2.4 GB/sec or 533 MB/sec	2.4 GB/sec or 533 MB/sec
Transfer Duration	66.7 nS	1353.3 nS	10.8 nS	963.9 nS	33.3 nS
Transfer Latency	83.3 nS	117.5 nS	117.5 nS	117.5 nS or70 nS	117.5 nS or70 nS
Deadline	Avg.: 122 uS Short-term: 122 uS	Avg.: 34.72 uS Short-term: ~212 uS	Avg.: 22.72 uS Short-term: 22.72 uS	Avg.: 4.12 uS (min. recurrence interval) Short-term: flexible	Avg.: 17.76 uS (min. recurrence interval) Short-term: flexible
Delay Tolerance	Avg.: 121.8 uS Short-term: 121.8 uS	Avg.: 33.25 uS Short-term: 210.5 uS	Avg.: 22.59 uS Short-term: 22.59 uS	Avg.: 3.03 uS Short-term: flexible	Avg.: 17.61 uS Short-term: flexible

3.3 Examining System Traffic

The EMIF is used by every transfer and should be considered carefully. It is necessary to decide which transfers will have priority, which should share the bandwidth (if any), and how to schedule traffic to best use EMIF bandwidth.

The incoming and outgoing audio/video data streams have strict hardware deadlines, are delay sensitive, and should be placed in the higher priority range.

The fact that the outgoing video TR is relatively large (720 bytes) is significant. Other sensitive TRs on the same priority could be delayed for the duration of the transfer. Since the outgoing video TR has more short-term delay tolerance than the other sensitive TRs (due to the large video buffer) it will be prioritized just below them.

As the algorithm QDMAs are the least delay sensitive, they will be given the lowest priority. However, even on low priority, keep in mind that they may interfere via read/write parallelism (see the EDMA architecture document for details). Read/write parallelism, however, will not be a major concern for small transfers, as they may complete in a single read/write command.

3.4 Scheduling System Traffic

In light of the above considerations the following priority schedule will be used:

- The incoming stream will be placed on the high priority queue (Q1).
- The outgoing video TRs will be placed on the medium priority (Q2).
- The outgoing audio TRs will be placed on the urgent priority (Q0).
- The audio and video algorithm QDMAs will be on low priority (Q3).

This layout puts the short TRs and sensitive TRs on the higher priorities, and longer TRs are placed on their own priority levels.

Table 4. TR Priorities and Requestors

	Incoming Stream	Outgoing Video	Outgoing Audio	Video Algorithm	Audio Algorithm
Priority	Q1	Q2	Q0	Q3	Q3
Requestor	PCI node	EDMA channel node	EDMA channel node	L2 node	L2 node

The incoming stream TR are generated by the PCI. The priority queue allocation register in the PCI peripheral (TRCTL) must be programmed to submit TRs to the high priority level. Also, the allocation limit should be left at the default value of 4.

The QDMA submits TRs to Q3. The number of QDMA TR requests submitted to this queue is not expected to exceed 6 TR in this application. Therefore, L2ALLOC3 should be set to 6. The other L2ALLOCn registers will be left at default (to accommodated cache and for future expansion).

The EDMA submits TRs to Q0 and Q2. The number of requests to these queues is expected to be six or less in this application. Therefore, PQAR0 and PQAR2 should be set to 6.

Remember that each priority queue has a maximum depth of 16. Table 5 summarizes priority queue allocation registers and shows that this maximum is not exceeded.

Table 5. Priority Queue Allocation Registers

	L2 Node	EDMA Channel Node	Master Peripherals†	Total
Priority Q0	L2ALLOC0 = 6	PQAR0 = 6		12
Priority Q1	L2ALLOC1 = 2	PQAR1 = 6	PCI TRCTL: priority = 1, allocation = 4	12
Priority Q2	L2ALLOC2 = 2	PQAR2 = 6		8
Priority Q3	L2ALLOC3 = 6	PQAR3 = 6		12

† Allocation registers for unused master peripherals can be ignored.

3.5 Determining Worst Case for TRs

Since the outgoing audio TR is the most delay sensitive, it will be examined in depth here with a TR timetable.

3.5.1 TRIQ: Outgoing Audio

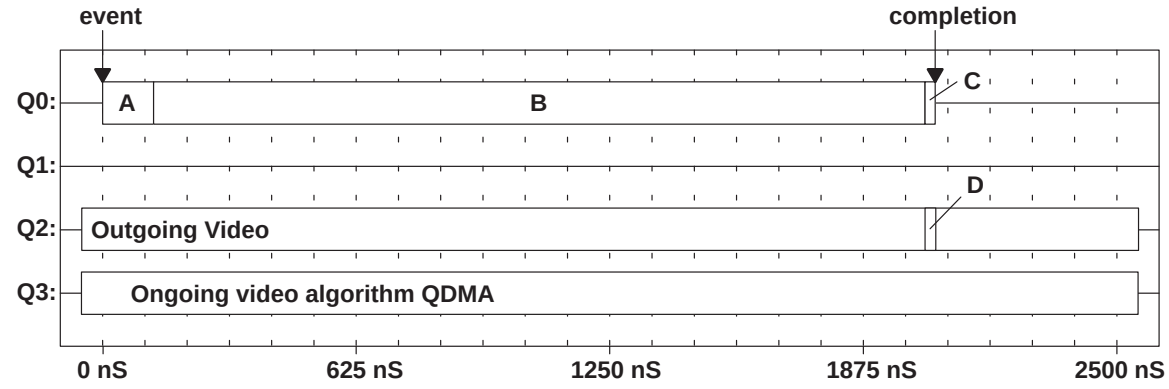
Outgoing audio data has a short term delay tolerance of 22.59 uS for the transfer to complete. Once the transfer begins, it will only take 10.8 nS to complete. With a transfer this small the only real concern is for the transfer to gain access to the resources. This is why the audio transfer was placed on the urgent priority queue. In the worst case situation, the audio transfer could be in a port blocking scenario (see the EDMA architecture document for details) where the 4 EMIF read command buffers are full from the outgoing video transfer. If an audio TR reached the queue registers at that point it would be given priority, however, its read command would be delayed behind the 4 read commands already in the EMIF command buffers. The length of the delay is equal to the amount of data in those 4 buffers (16 words each in the worst case) divided by the SDRAM bandwidth of 533 MB/sec. The incoming stream TR could be delayed 609 nS (480 nS transfer duration + 129 nS transfer latency).

The delay could be increased further due to read/write parallelism (see *TMS320C64x DSP EDMA Performance Data*, SPRAA02) if an ongoing video algorithm was writing to EMIF-A at the same time as the outgoing video was reading from EMIF-A. If the outgoing audio TR arrived in this situation (both read and write EMIF-A command buffers full), then the port blocking delay from the outgoing video transfer would still exist, and additionally this delay would increase due to the ongoing video algorithm write to EMIF-A. So rather then a 609 nS delay, there would be a delay of 2049 nS (the read transfer only receives 25% of EMIF-A bandwidth, 480 nS/0.25 +129 nS). Even with this addition to the worst case situation the transfer would still complete with 20 uS to spare.

If there had been a conflict, then the outgoing audio could be stored in L2 rather then external memory; then the outgoing audio would experience negligible delay.

The following TR timetable depicts the worst-case scenario described above.

Worst-Case Scenario, TRIQ: Outgoing Audio



A = The TRIQ latency period = 129 nS
B = The TRIQ read command is stalled behind the outgoing video read commands = 1920 nS
C = The TRIQ outgoing audio data is read from the EMIFand written to the McBSP = 10.8 nS
D = The outgoing video read commands are stalled behind the outgoing audio read commands = 10.8 nS

Figure 2. Example TR Timetable

4 EDMA Performance

This section discusses EDMA performance in general. The bandwidth of the various peripherals are discussed. For device specific information regarding performance measurements and test data, see *TMS320C64x EDMA Performance Data* (SPRAA02) or *TMS320C671x/C621x Performance Data* (SPRAA03).

4.1 EDMA Bandwidth

The EDMA bandwidth varies by device, and is calculated by the fact that it can write an element of data to the peripheral ports every EDMA clock cycle.

The size of the element depends on the peripheral port involved. The L2 port, video port, EMIF port, and TCP/VCP port are all 64-bit ports. The UTOPIA, McBSP, and master peripheral ports are 32-bit. The port size affects the EDMA bandwidth available to the peripherals.

The EDMA frequency is CPU frequency divided by 2 (C64x) or equal to the CPU frequency (C621x/C671x).

For example, a C64x with CPU frequency of 600 MHz has an EDMA frequency of 300 MHz. This results in a bandwidth of 1.2 GB/sec for 32-bit ports, and 2.4 GB/sec for 64-bit ports. In contrast, a 133-MHz 32-bit SDRAM has an ideal throughput of 533 MB/sec.

The EDMA bandwidth is generally not the limiting factor in a transfer, though it must be considered. The most important factors are the source and destination peripheral bandwidths. The slower of the two limits the overall rate at which the transfer can progress. The slower port is considered busy for the entire duration. The faster port is only considered busy for a fraction of the time, roughly equal to the ratio of bandwidths. It is able to service other transfers in the interim.

Another important factor is transfer arbitration and interaction, which was introduced in the EDMA architecture document and explored in depth in the IO Scheduling section of this document. The effects of transfer arbitration and interaction can be approximated by the techniques described in the IO Scheduling section.

4.2 Peripheral Bandwidth

To find the duration of a transfer, it is necessary to know the bandwidth of the peripherals involved. Ideal peripheral bandwidths are given here, though it may be necessary to account for any transfer interaction and external peripheral activity (such as page misses in an SDRAM).

4.2.1 L2 and Video Port

The L2 memory controller and the video port have the fastest memory accessible by the EDMA. Both contain 64-bit memories accessible at a frequency of CPU divided by 2. In the case of a 600-MHz CPU, the resulting bandwidth is 2.4 GB/sec. For the L2 memory, remember that this bandwidth is shared between CPU accesses, EDMA accesses, and cache coherency operations.

4.2.2 EMIF

In the case of the EMIF, data bursts into the command buffers at the EDMA bandwidth. From the command buffers, data is conveyed to/from the external memory interface at the rate of the attached device. This bandwidth is defined by the EMIF control registers, which specify external memory bus width, frequency, and programmable wait states (asynchronous setup, strobe, and hold times). For example, a 64-bit, 100-MHz external SBSRAM has an ideal bandwidth of 800 MB/sec.

4.2.3 Utopia, McBSP/McASP, and Master Peripherals

The UTOPIA, McBSP/McASP, and master peripheral ports behave differently than memory ports. This is because transfers to and from these ports typically have a short, defined length that fit into the respective command buffers in the port. As a result, the bandwidth of transfers to or from these ports is not dependent on upon external data flow. The bandwidth depends only on the EDMA, and it is equal to the EDMA frequency times 32-bits (the width of these ports).

For example, when the McBSP peripheral receives 1 word, it generates a receive event and moves that data to its read command buffer. The EDMA reads that command buffer (at the EDMA clock rate through the McBSP port which is 32-bit wide). At an EDMA clock rate of 300 MHz, giving this port has an effective bandwidth of 1.2 GB/sec.

The same is true for a McBSP write. The EDMA writes data to the McBSP write command buffer at the EDMA clock rate. The same effective bandwidth is achieved. Once the data enters the command buffer, the EDMA has finished the transfer. The McBSP peripheral is left to send the data out the serial port at the serial rate.

The UTOPIA and master peripherals operate the same way, achieving the same effective bandwidth from an EDMA transfer perspective.

4.2.4 Other Peripherals

EDMA transfers accessing peripherals that do not have ports (such as the I2C) are routed through the L2 controller, utilizing the L2 peripheral port. The data is passed to the peripheral configuration bus where it is routed to its destination. Generally these are single-word accesses. For writes, the TR is finished as soon as the data is transferred to the L2 controller. Thus, the transfer bandwidth is the same as a normal L2 access. For reads, the data is fetched from the peripheral configuration bus

5 Summary

This application note discusses efficient EDMA IO scheduling for the TMS320C6000 line of DSPs. Several techniques to organize system transfers to aid in scheduling are presented. These techniques are demonstrated in an example to properly schedule system traffic. Understanding the EDMA architecture, and the expected EDMA performance will greatly aid a design in properly scheduling EDMA traffic. For more information on these topics please consult the application notes listed in the reference section.

6 References

1. *TMS320C64x EDMA Architecture* (SPRA994)
2. *TMS320C621x/TMS320C671x EDMA Architecture* (SPRA996)
3. *TMS320C6000 DSP Peripherals Overview Reference Guide* (SPRU190)
4. *TMS320C6000 EDMA Controller Reference Guide* (SPRU234)
5. *TMS320C64x DSP Two-Level Internal Memory Reference Guide* (SPRU610)
6. *TMS320C621x/TMS320C671x DPS Two-Level Internal Memory Reference Guide (*SPRU609)

Appendix A Terminology

- **Active TR:** TR that is currently being processed in a queue register set in the EDMA transfer controller. There can be up to four simultaneous, active TRs. Technically, it is no longer in the priority queues, and no longer outstanding.
- **Commands:** Active TRs submit commands to the peripheral ports. These commands are for a burst of data to be read or written.
- **Data transfer, access, or transfer:** High-level, abstract terms that refer to data movement in a system. Can refer to an entire data flow, or part of a data flow, depending on context. This is in contrast to a transfer request (see definition below).
- **EDMA transfer controller:** The transfer engine of the C64x, C671x, and C621x DSPs, that handles all data movement in the system.
- **EDMA channel controller:** A user-programmable mechanism for submitting TRs to the EDMA transfer controller based on internal and external events.
- **Master peripheral:** The EMAC, HPI, and PCI peripherals are considered master peripherals.
- **Outstanding TR:** A TR that is awaiting processing by in the priority queues. A single TR requestor is limited in the number of outstanding TRs it can have per priority level.
- **Read/Write parallelism:** Active TRs in the EDMA queue registers submit read and write commands to source and destination ports. Because the read and write command pipelines are independent of one another, it is possible for a single port to receive read commands from one TR and write commands from another TR, regardless of the TRs priorities. In this way, a peripheral may interleave servicing of two TRs of different priority. See the EDMA architecture document for further details and an example.
- **Transfer duration:** Transfer duration is measured from the first read from the source peripheral to the last write on the destination peripheral.
- **Transfer latency:** Transfer latency is the time it takes for the following: an event to generate a TR, for that TR to reach the EDMA queue registers for processing, and for the first command to start transferring data at the source peripheral port. Thus, it can be measured from the time the event goes active to the time the source peripheral begins reading data.
- **Transfer request (TR):** Refers to a specific data transfer request submitted to the EDMA Transfer Controller by a transfer requestor.
- **Transfer requestor:** Module that submits TRs. There are three transfer requestors: the EDMA channel controller, the L2 controller, and the master peripheral.
- **Transfer/IO scheduling:** The process of defining transfer priority and timing.

Appendix B Transfer Interaction Summary: Priorities and Ports

The two basic resources inside the EDMA are priority levels and peripheral ports. The rules that define interaction of transfer requests are based on these two resources:

- TRs within a priority level occur serially in the order of their arrival, because there can be only one active TR from each priority level at any given time.
- Ports receive commands from the active TR with the highest priority which utilizes that port. Note, however, that reads and writes are considered separately by the source (read commands) and destination (write commands) pipelines. Also note that if a higher priority transfer isn't fully utilizing the peripheral bandwidth, lower priority transfers are allowed to submit commands.
- The L2 port can often service multiple active TRs because it is often ready faster than other ports.

Note that the priority level of cache servicing and master peripherals is programmable in the C64x family, but not in the C67x/C62x family.

Appendix C Units

Bandwidth units are as follows:

- MB/sec equals 1,000,000 bytes per second
- GB/sec equals 1,000,000,000 bytes per second

Data size units are as follows:

- kB equals kilobyte equals 1,024 bytes
- MB equals megabyte equals 1,048,576 bytes

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