Application Report
SPRAB52–July 2009

# TMS320DM36x SoC Architecture and Throughput Overview

*Manoj Bohra*

**ABSTRACT**

This application report provides information on the DM36x throughput performance and describes the DM36x System-on-Chip (SoC) architecture, data path infrastructure, and constraints that affect the throughput and different optimization techniques for optimum system performance. This document also provides information on the maximum possible throughput performance of different peripherals on the SoC.

**Contents**



**List of Figures**

**List of Tables**



# 1 SoC Architectural Overview

The ARM9 processor, enhanced direct memory access (EDMA3) transfer controllers, and the system peripherals are interconnected through switch fabrics. The switch fabrics allow for low-latency, concurrent data transfers between master peripherals and slave peripherals. Through a switch fabric, the master can send data to the slave without affecting a data transfer between the other master controller and slave peripherals. The switch fabrics also allow for seamless arbitration between the system masters when accessing system slaves. More information on SCR and bridges is provided in the following sections.

Figure 1 shows the connection between slaves and masters through the data switched central resource (SCR). Masters are shown on the right and slaves on the left. The data SCR connects masters to slaves via data buses having a width 32/64 running at a SYSCLK4 frequency, as shown in Figure 1. SYSCLK4 is supplied by the PLL1 controller and is fixed at a frequency equal to the CPU frequency divided by 3. Some peripherals, like ARM and the video image coprocessors (HDVICP), have both slave and master ports. Each EDMA3 transfer controller has an independent connection to the data SCR. Masters can access the configuration SCR through the data SCR.

Figure 2 shows the connection between the ARM and the configuration SCR, which is mainly used by the ARM to access peripheral registers. The data SCR also has a connection to the configuration SCR that allows masters to access most peripheral registers.

The following is a list of points that help to interpret Figure 1 and Figure 2.

- The arrow indicates the master/slave relationship.
- The arrow originates at a bus master and terminates at a bus slave
- The direction of the arrows does not indicate the direction of data flow. Data flow is typically bi-directional for each of the documented bus paths.

- Some peripherals may have multiple instances shown for a variety of reasons in the diagrams, some of which are described below:

– The peripheral/module has master port(s) for data transfers, as well as slave port(s) for register access, data access, and/or memory access. Examples of these peripherals are ARM, high-definition video image coprocessors (HDVICP), Ethernet media access controller (EMAC) subsystem, and host port interface (HPI).

– The peripheral/module has a master port as well as slave memories. Examples of these are the ARM and HDVICP.

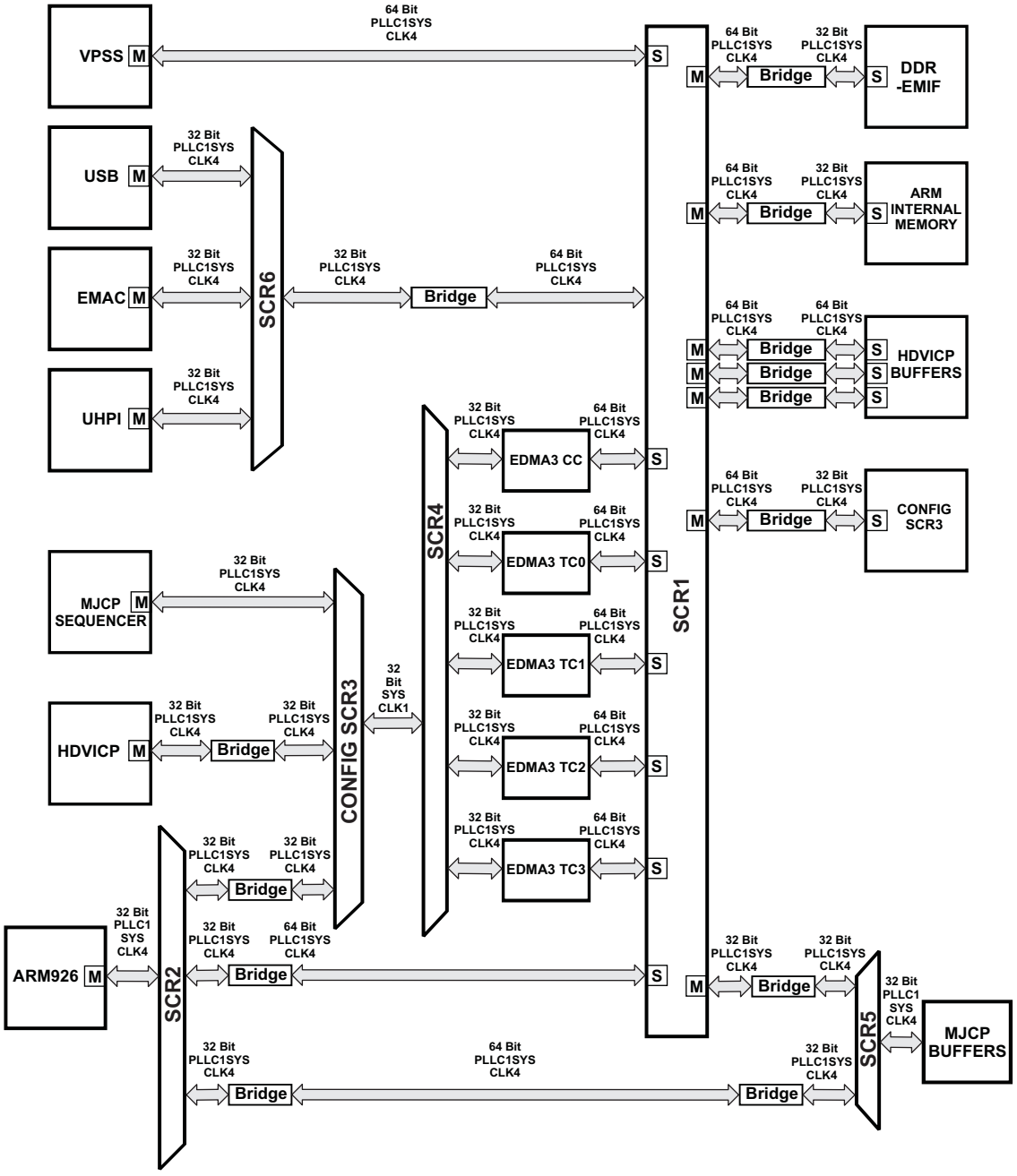



**Figure 1. TMS320DM36x System Interconnect Block Diagram**

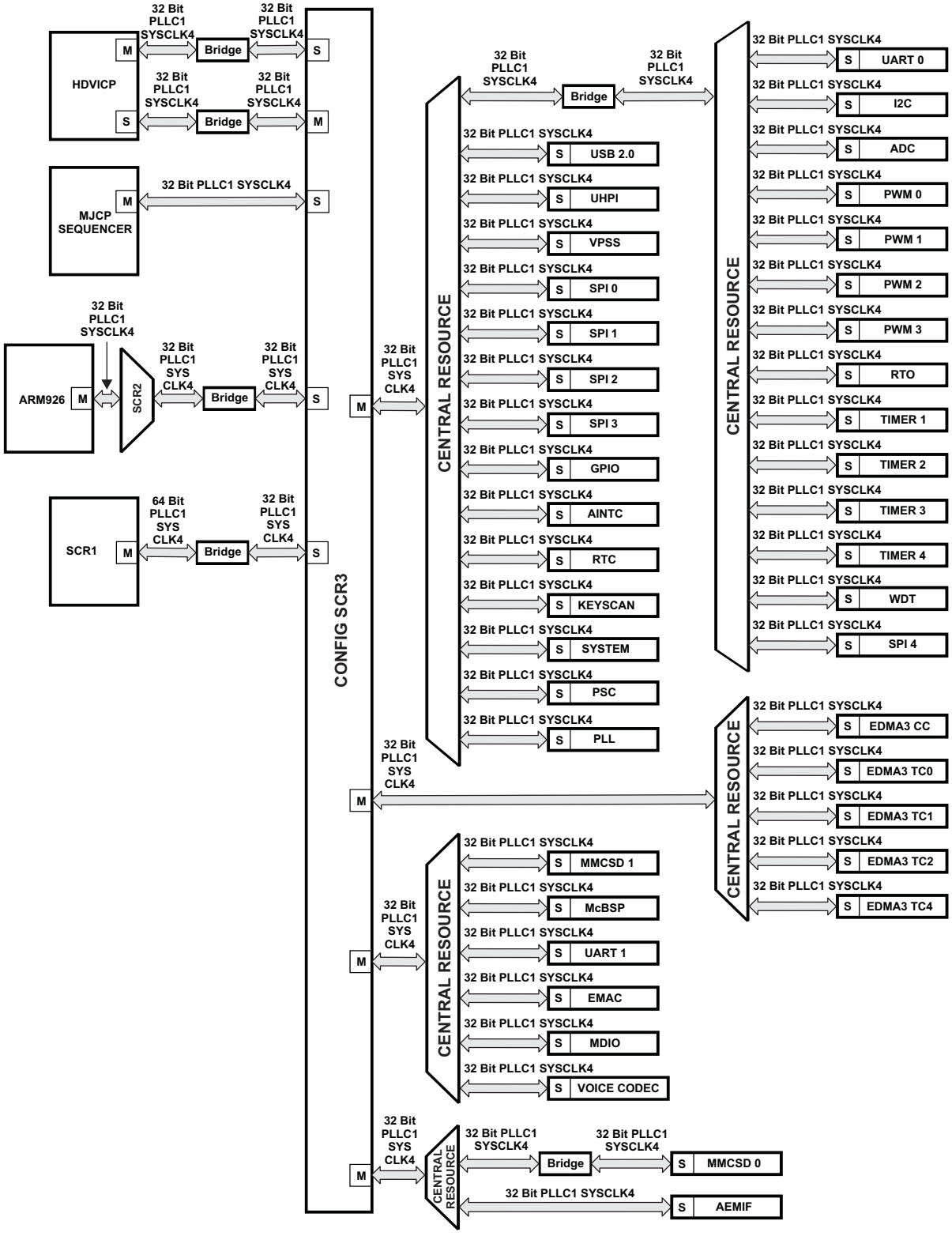



**Figure 2. TMS320DM36X Peripheral Configuration Bus**

## 1.1 Master Peripherals

The DM36x SoC peripherals can be classified into two categories: master peripherals and slave peripherals. Master peripherals are typically capable of initiating read and write transfers in the system and do not rely on the EDMA3 (system DMA) or CPU to perform transfers to and from them. Table 1 lists all master peripherals of the DM36x SoC. To determine the allowed connections between masters and slaves, each master request source must have a unique master ID (mstid) associated with it. The master ID for each DM36x SoC master is also shown in Table 1.

**Table 1. TMS320DM36x DMSoC Master Peripherals**

| Mstid | DM36x Master |
|---|---|
| 0 | ARM Instruction |
| 1 | ARM Data |
| 7-2 | Reserved |
| 8 | VPSS |
| 9 | MJCP (sequencer) |
| 10 | EDMA CC |
| 11 | HDVICP |
| 15-12 | Reserved |
| 16 | EDMA TC0 Read Port |
| 17 | EDMA TC0 Write Port |
| 18 | EDMA TC1 Read Port |
| 19 | EDMA TC1 Write Port |
| 20 | EDMA TC2 Read Port |
| 21 | EDMA TC2Write Port |
| 22 | EDMA TC3 Read Port |
| 23 | EDMA TC3 Write Port |
| 24-31 | Reserved |
| 32 | EMAC |
| 33 | UHPI |
| 34 | USB |
| 35-63 | Reserved |

## 1.2 Slave Peripherals

Slave peripherals service the read/write transactions that are issued by master peripherals. All DM36x SoC slaves are listed in Table 2. Note that memories are also classified as peripherals.

**Table 2. TMS320DM36x DMSoC Slave Peripherals**

| DM36x Slaves |
|---|
| Arm Internal Memory |
| MPEG/JPEG Coprocessor (MJCP) Memory |
| Config Bus Registers and Memory |
| DDR EMIF Memory |
| HDVICP Memory Port-1 |
| HDVICP Memory Port-2 |
| HDVICP Memory Port-3 |

## 1.3 Switched Central Resources (SCR)

The SCR is an interconnect system that provides low-latency connectivity between master peripherals and slave peripherals. More information on master and slave peripherals is provided in the following sections. It is the decoding, routing, and arbitration logic that enable the connection between multiple masters and slaves that are connected to it. Multiple SCRs are used in the DM36x SoC, as shown in Figure 1 and Figure 2, to provide connections among different peripherals. Look at Table 3 for supported master and slave peripheral connections. Additionally, the SCRs provide priority-based arbitration and facilitate concurrent data movement between master and slave peripherals. For example, as shown in Figure 3 (black lines), through SCR1, the VPSS (master) can send data to the DDR2 memory controller (slave) concurrently without affecting a data transfer between the EMAC (master) and ARM (TCM) internal memory (slave).

## 1.4 Bridge

Bridge in the DM36x SoC, different clock rates and bus widths are used in various parts of the system. To communicate between two peripherals that are operating at different clock rates and bus widths, there should be logic to resolve these differences. Bridges provide a means of resolving these differences by performing bus-width conversion as well as bus operating clock frequency conversion. Bridges are also responsible for buffering read and write commands and data. Figure 3 shows the typical connection of a bridge.


**Figure 3. Bridge**

Multiple bridges are used in the DM36x SoC. For example, as shown in Figure 4, bridge performs a bus-width conversion between a 32-bit bus and a 64-bit bus.

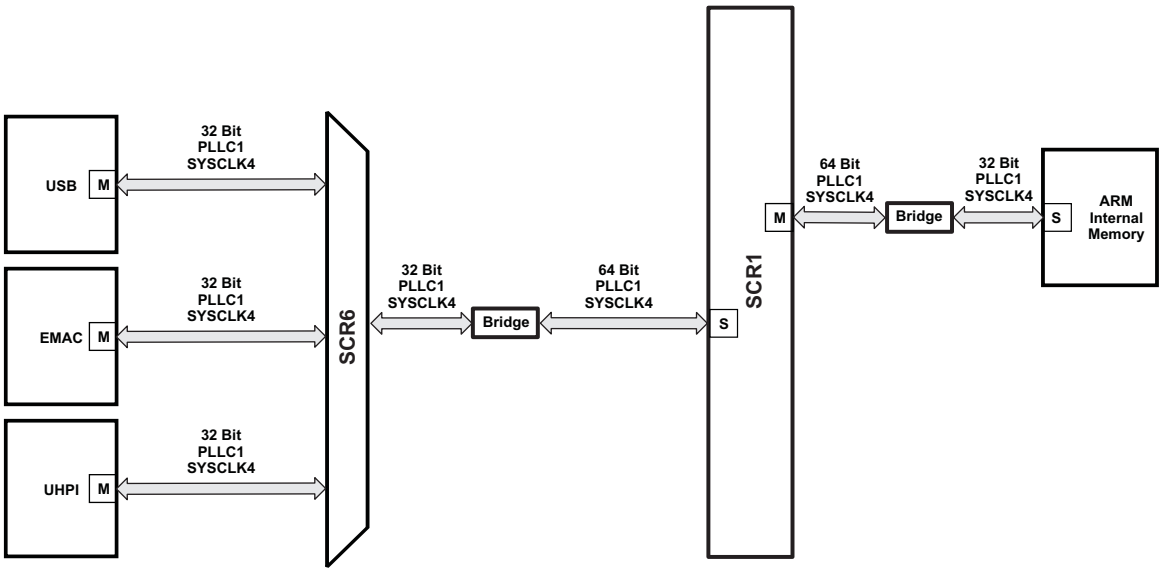



**Figure 4. Bus-Width and Clock Rate Conversion**

### 1.4.1 Head of Line Blocking

A command first-in-first-out (FIFO) is implemented inside the bridge to queue transaction commands. All requests are queued on FIFO basis; bridges do not reorder the commands. It is possible that a high priority request at the tail of a queue can be blocked by lower priority commands that could be at the head of the queue. This scenario is called bridge head of line blocking. In Figure 5, the command FIFO size is 4. FIFO is completely filled with low priority (7) requests before a higher priority request (0) comes in. In this case, the high priority request has to wait until all four lower priority (7) requests get serviced. When there are multiple masters vying for the same end point (or end points shared by the same bridge) , the bridge head of line blocking is one of the factors that can affect system throughput and a master's ability to service read/write requests targeted to a slave peripheral/memory.

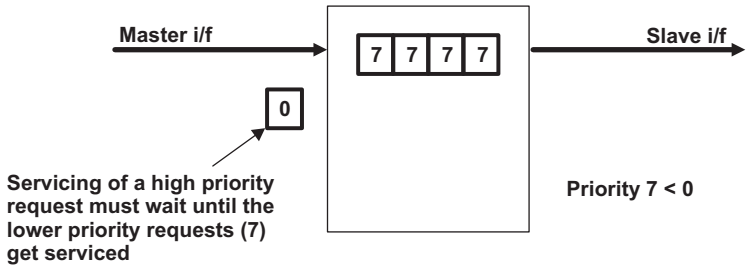



**Figure 5. Bridge Head of Line Blocking**

## 1.5 Master/Slave Connectivity

Not all masters on the device may connect to all slaves. Allowed connections are summarized in Table 3.

**Table 3. System Connection Matrix**

| | Slave Module | | | | |
|---|---|---|---|---|---|
| DMA Master | ARM Internal Memory | MPEG/JPEG Coprocessor Memory | HD Video Image Coprocessor Memory | Configuration Bus Registers and Memory | DDR EMIF Memory |
| ARM | √ | √ | √ | √ | √ |
| VPSS | | | √ | | √ |
| DMA Master Peripheral (USB, EMAC, HPI) | √ | | | | √ |
| EDMA3TC0 | √ | √ | √ | √ | √ |
| EDMA3TC1 | √ | √ | √ | √ | √ |
| EDMA3TC2 | √ | √ | √ | √ | √ |
| EDMA3TC3 | √ | √ | √ | √ | √ |

## 1.6 Data Bus (Widths/Speeds)

There are two main types of busses on the DM36x SoC: 64-bit bus and 32-bit bus.

- A 64-bit bus with separate read and write interface, allowing multiple outstanding read and write transactions, simultaneously. This bus is best suited for high-speed/high-bandwidth exchanges, Especially data transfers between on-chip and off-chip memories. On the DM36x device, the SCR, SCR1, interfaces with all the modules using this 64-bit bus. Most of the high bandwidth master peripherals, e.g., EDMA3 transfer controller (EDMA3TC), and slave memories, DDR, are directly connected to the main SCR through this bus. Peripherals that do not support the 64-bit bus interface are connected to the main SCR via bridges (responsible for protocol conversion from 64-bit to 32-bit bus interface).

- A 32-bit bus, with a single interface for both reads and writes. The read and write transactions serviced strictly in order. This bus is best suited for communication with the memory-mapped registers of all on-chip peripherals. Accesses to memory-mapped registers could be for configuration purposes, e.g., accesses to configure a peripheral) or for data accesses (e.g., read writes from/to multichannel audio serial port (McASP) receive/transmit buffer registers or writes to transmitter holding registers reads from receiver buffer registers on universal asynchronous receiver/transmitter (UART).

### 1.7 Default Burst Size

Burst size is another factor that affects peripheral throughput. A master's read/write transaction is broken down into smaller bursts at infrastructure level. The default burst size for a given peripheral (master) is the maximum number of bytes per read/write command. The burst size determines the intra-packet efficiency of a master's transfer. At system interconnect level, it also facilitates pre-emption as the SCR arbitrates at burst size boundaries.

Table 4 shows default burst sizes of some of the DM36x SoC masters.

**Table 4. Default Burst Sizes**

| Masters | Possible Burst Sizes |
|---|---|
| EDMA TC0 read/write | 32 byte |
| EDMA TC1 read/write | 32/64 byte |
| EDMA TC2 read/write | 16 byte |
| EDMA TC3 read/write | 16 byte |
| HPI | Eight -32-bit words |
| USB | 64 byte |
| EMAC | 64 byte |

## 2 SoC Constraints

This section describes the factors that constrain the system throughput.

### 2.1 Hardware Latency

Each master-slave transaction has to go through multiple elements in the system. Each element contributes to a hardware latency of the transaction. In Figure 1 and Figure 2, all masters, slave, SCRs, and bridges contribute latency. For example, consider a transfer from EMAC-to-DDR memory, as shown in Figure 6 (black line).

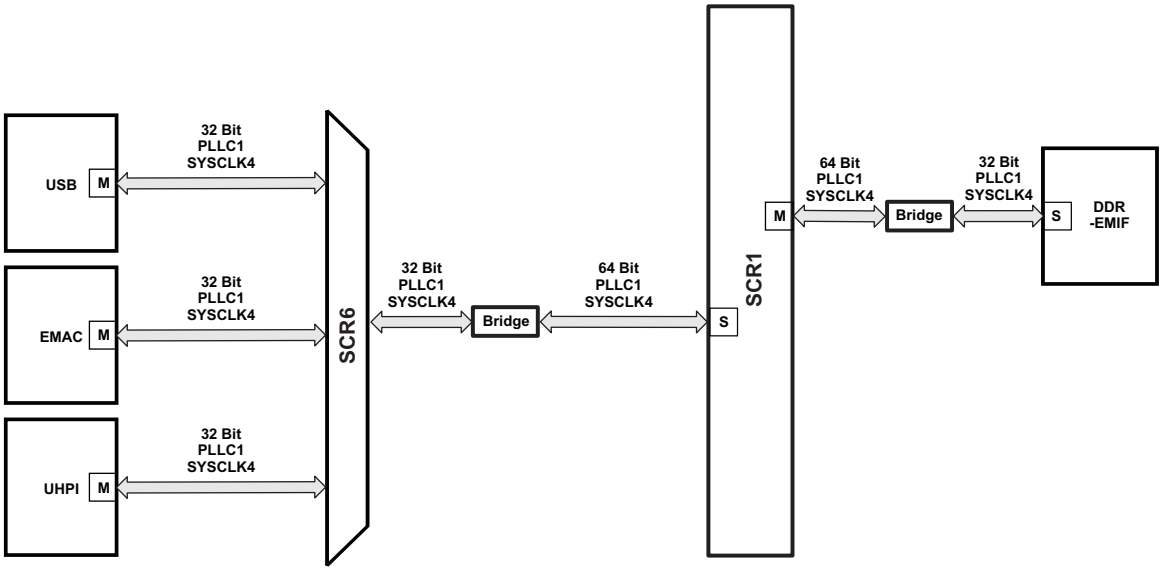



**Figure 6. EMAC-to-DDR Transfer**

This transaction experiences latencies in the master EMAC, SCR6, M-M Bridge - 2, SCR1, and the slave memory (DDR). Also, it is important to note that accessing registers is not a single cycle access. It has to go through multiple SCRs/bridges and experiences hardware latency. Polling on registers is more expensive.

The latency faced in the bridges is directly related to the default burst size and the command FIFO depth.

The worst latencies are due to SCR arbitration and bridge head of line blocking. The topology is optimized to minimize latency between critical masters and slaves. For example, notice that in Figure 1 there are no bridges or extra hops from critical masters such as EDMA master to critical slaves such as DDR memory.

## 2.2 Reads Vs Writes

Note that read transactions are more costly than writes. In case of a read transaction, the master has to wait until it gets the data back from the slave. However, in a write case, the master can issue a write transaction and go ahead with the next transaction without waiting for a response from the slave.

## 2.3 On-Chip Memory Vs Off-Chip Memory

On-chip memory access does not experience any hardware latency, whereas, off-chip memory access experiences hardware latency with SCR and bridges. It is recommended to keep frequently used code in on-chip memory for better system throughput performance.

## 2.4 Memory Maximum Bandwidths

Memory bandwidth has an effect on system throughput. More bandwidth gives better throughput performance. Table 5 shows all memories of the DM36x SoC and their maximum bandwidths.

**Table 5. Memory Maximum Bandwidths**

| Memory | Theoretical Maximum Bandwidth |
|---|---|
| ARM TCM | 1188 MB/s (ARM clock frequency * ARM bus width = 297 MHz* 32-bit bus) |
| DDR | 864 MB/s (DDR clock frequency*2 * DDR bus width = 216 MHz*2 * 16-bit bus) |
| DDR (in case of different operational frequency) | 972 MB/s (DDR clock frequency*2 * DDR bus width = 243 MHz*2 * 16-bit bus) |

# 3 SoC Level Optimizations

This section describes system level optimization techniques.

## 3.1 SCR Arbitration

SCR provides priority-based arbitration to select the connection between master and slave peripherals; this arbitration is based on the priority value of each master.

Each master can have a priority value between 0 and 7 with 0 being the highest priority and 7 being the lowest priority. The prioritization scheme works such that, at any given time, if there are read/write requests from multiple masters vying for the same end point (same slave peripheral/memory or infrastructure component like bridge/SCR connecting to multiple slave peripherals), then the accesses from the master at the highest priority are selected first. Additionally, if there are read/write requests from masters programmed at the same/equal priority, then one request from each master is selected in a round-robin manner.

The prioritization within the SCR is programmable for each master by configuring the Bus Master Priority Control 0 Register (MSTPRI0) and Bus Master Priority Control 1 Register (MSTPRI1). For more details on these registers, see the *TMS320DM36x Digital Media System-on-Chip (DMSoC) ARM Subsystem User's Guide* (SPRUFG5). The default priority levels for the DM36x SoC bus masters are shown in Table 6; lower values indicate higher priority.

**Table 6. Default Master Priorities**

| Master | Default Priority |
|---|---|
| VPSS | 0 |
| EDMA TC0 | 0 |
| EDMA TC1 | 0 |
| EDMA TC2 | 0 |
| EDMA TC3 | 0 |
| ARM (DMA) | 1 |
| ARM (CFG) | 1 |
| USB | 4 |
| EMAC | 4 |
| HPI | 4 |
| High-Definition Video Image Coprocessor | 5 |
| MPEG/JPEG Coprocessor | 5 |

Note that there is no priority set for the EDMA3 channel controller (EDMA3CC). This is because the EDMA3CC accesses only the TPTCs and is always given higher priority than the other masters on those Fast CFG SCR slave ports. Although the default priority values (for different masters) have been chosen based on the prioritization requirements for the most common application scenarios, it is prudent to adjust/change the master priority values based on application-specific needs to obtain optimum system performance and to ensure real-time deadlines are met.

## 3.2 DDR2 Prioritization Scheme

The DDR2 memory controller services all master requests on priority basis and reorders requests to service highest priority requests first, improving system performance. For more details on the DDR2 memory controller prioritization scheme, see the *TMS320DM36x Digital Media System-on-Chip (DMSoC) DDR2/mDDR Memory Controller User's Guide* (SPRUFI2).

# 4 Throughputs

This section describes the maximum throughput performance of different peripherals of the DM36x SoC. It also provides the factors that affect peripheral throughput and recommendations for optimum peripheral performance.

## *4.1 Enhanced Direct Memory Access (EDMA)*

This section provides a throughput analysis of the EDMA module integrated in the TMS320DM36x DMSoC.

### 4.1.1 Overview

The EDMA controller's primary purpose is to service user programmed data transfers between internal or external memory-mapped slave endpoints. It can also be configured for servicing event driven peripherals (such as serial ports), perform sorting or sub frame extraction of various data structures, etc. There are 64 direct memory access (DMA) event channels and 8 QDMA channels serviced by four concurrent physical channels. The block diagram of EDMA is shown in Figure 7.

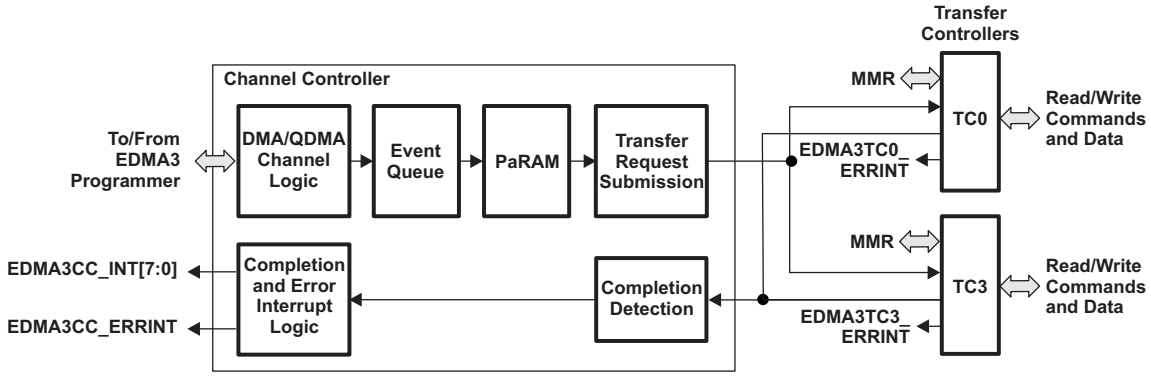



**Figure 7. EDMA3 Controller Block Diagram**

DMA channels are triggered by external event, manual write to event set register (ESR), or chained event. QDMA are auto triggered when write is performed to the user-programmable trigger word.

Once a trigger event is recognized, the event is queued in the programmed event queue. If two events are detected simultaneously, then the lowest-numbered channel has highest priority.

Each event in the event queue is processed in the order it was queued. On reaching the head of the queue, the PaRAM associated with that event is read to determine the transfer details. The transfer request (TR) submission logic evaluates the validity of the TR and is submits a valid transfer request to the appropriate transfer controller.

Figure 8 shows a block diagram of the channel controller.

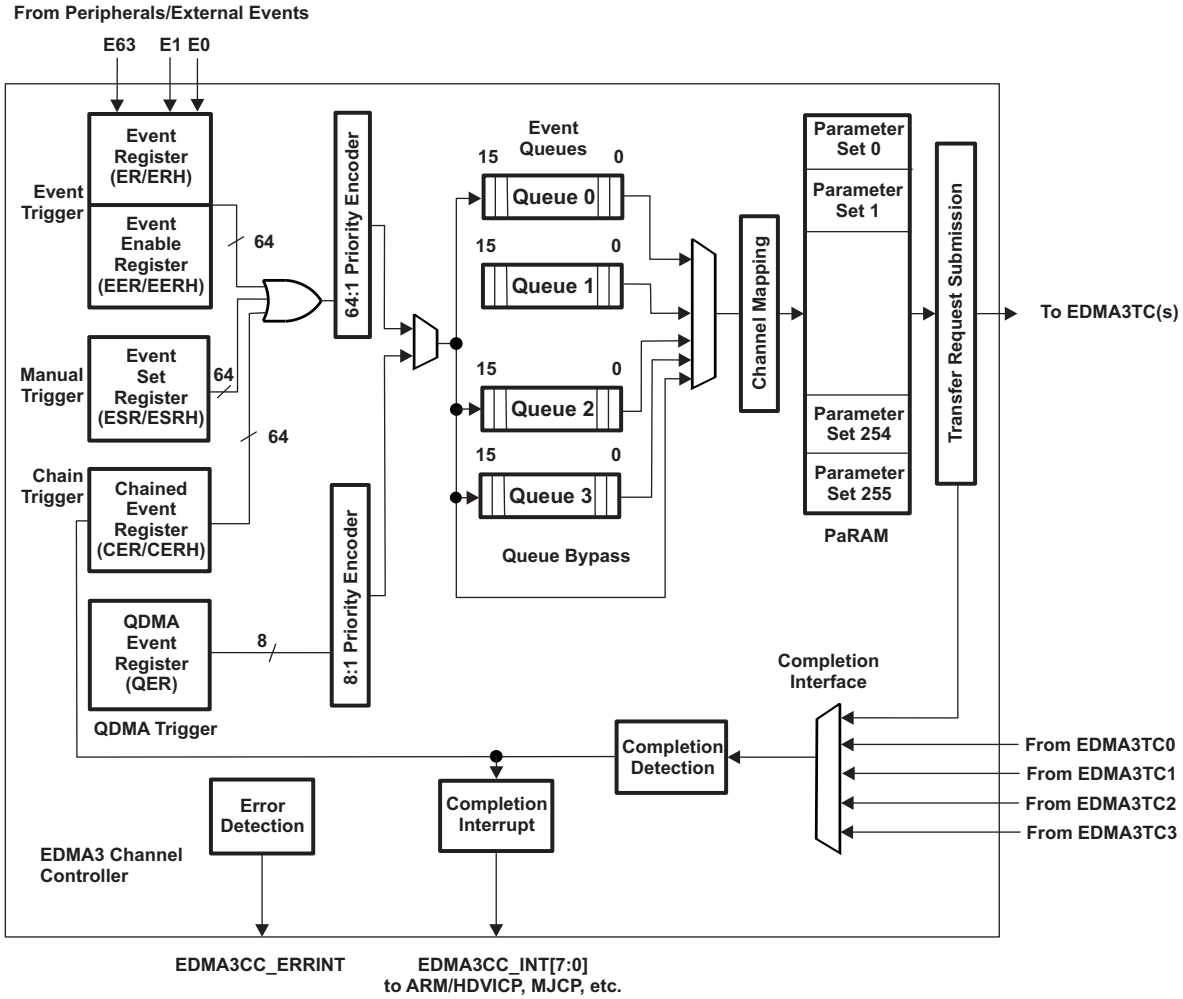



**Figure 8. EDMA3 Channel Controller (EDMA3CC) Block Diagram**

The transfer controller receives the request and is responsible for data movement as specified in the transfer request. Figure 9 shows a block diagram of the transfer controller.

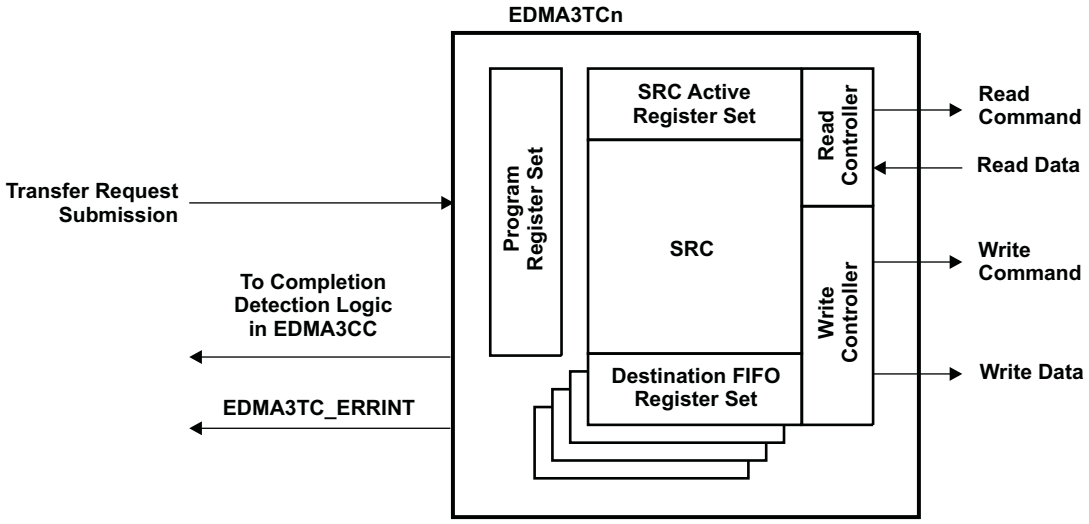



**Figure 9. EDMA3 Transfer Controller (EDMA3TC) Block Diagram**

The transfer controller receives the TR in the DMA program register set, where it transitions to the DMA source active set and the destination FIFO register set immediately. The read controller issues the read command when the data FIFO has space available for data read. When sufficient data is in the data FIFO, the write controller starts issuing the write command.

The maximum theoretical bandwidth for a given transfer can be found by multiplying the width of the interface and the frequency at which it transfers data. The maximum speed the transfer can achieve is equal to the bandwidth of the limiting port. The transfer never achieves the maximum theoretical bandwidth, due to the latency in the transmission. It is important to remember that latency will have a legacy impact for shorter transfers. Approximate latency for different memory accesses is equal to the time taken for a one byte transfer; this latency is not considered for throughput measurement in this document. Table 7 lists the internal bus frequencies at which different memories and slave end point operates and their bus widths.

**Table 7. Frequency and Bus Widths for Different Memory and Slave Endpoints**

| Module Name | Freq (MHz) | | Bus Width (bits) |
|---|---|---|---|
| | 270 MHz CPU | 300 MHz CPU | |
| DDR | 216 | 243 | 32 |
| MJCP | 216 | 243 | 32 |
| HDVICP | 270 | 297 | 64 |
| ARM | 270 | 297 | 32 |

The formulas used for the throughput calculations are shown below:

- Actual Throughput = (Transfer Size/Time Taken)
- Ideal Throughput = Frequency of Limiting Port * Data Bus Width in Bytes
- TC Utilization = (Actual Throughput/ Ideal Throughput) * 100

### 4.1.2 Test Environment

The common system setup for the EDMA throughput measurement is given below:

- ARM clock: 297 MHz
- DDR clock: 243 MHz
- Used timer operating at 24 MHz
- Throughput data collected is standalone. No other ongoing traffic.
- All profiling done with on-chip timer

### 4.1.3 Factors Affecting EDMA Throughput Value

EDMA channel parameters allow many different transfer configurations. Typical transfer configurations result in transfer controllers bursting the read write data in default burst size chunks, thereby, keeping the busses fully utilized. However, in some configurations, the TC issues less than optimally sized read/write commands (less than default burst size), reducing performance. To properly design a system, it is important to know which configurations offer the best performance for high-speed operations. These considerations are especially important for memory to memory/paging transfers. Single-element transfer performance is latency-dominated and is unaffected by these conditions.

The different factors considered for throughput calculation with its impact is given in Table 8.

**Table 8. Factors Considered for Throughput**

| Factors | Impact | Recommendations |
|---|---|---|
| Source/Destination Memory | The transfer speed depends on SRC/DST memory bandwidth | Use DDR for better results. Avoid AEMIF. |
| Transmit Size | Throughput depends on small transfers due to transfer overhead/latency. | Configure EDMA for larger transfer size as throughput. Small transfer size is dominated by transfer overhead. |
| A-Sync/AB-Sync | Performance depends on the number of TRs. More TRs would mean more overhead. | |
| Source/Destination Bidx | Optimization will not be done if BIDX is not equal to ACNT value | Configure BIDX equal to ACNT value |
| Queue TC Usage | Performance is the same for all four TCs | All four TCs have the same configuration and show the same performance |
| Burst Size | Decides the largest possible read/write command submission by TC | The default size for all transfer controllers is 32 bytes. This also results in most efficient transfers/throughput in most memory to memory transfer scenarios. |
| Source/Destination Alignment | Performance degrades if there is a mismatch in alignment | Set source destination alignment value to zero for better performance. |

### 4.1.4 Transfer Size

Throughput is low for smaller transfer size (less than 1KB) due to transfer overhead. Large transfer sizes give higher throughput. Figure 10 and Figure 11 describes the throughput for transfers between different memory like HDVICP , MJCP and DDR with different transfer sizes. Figure 12 and Figure 13 describes percentage of utilization different transfer sizes.

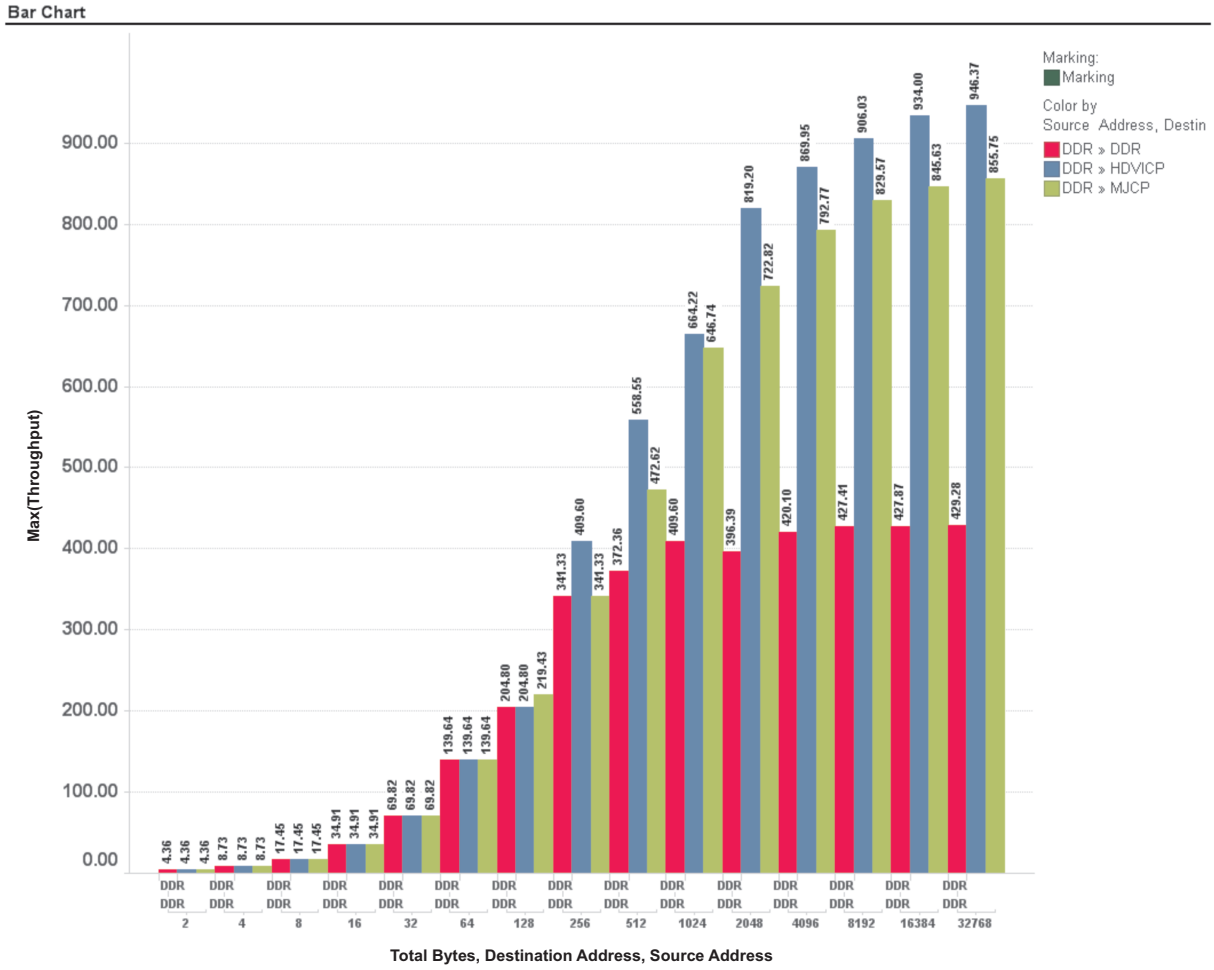



**Figure 10. Throughput of EDMA for DDR to HDVICP, MJCP and DDR Access**

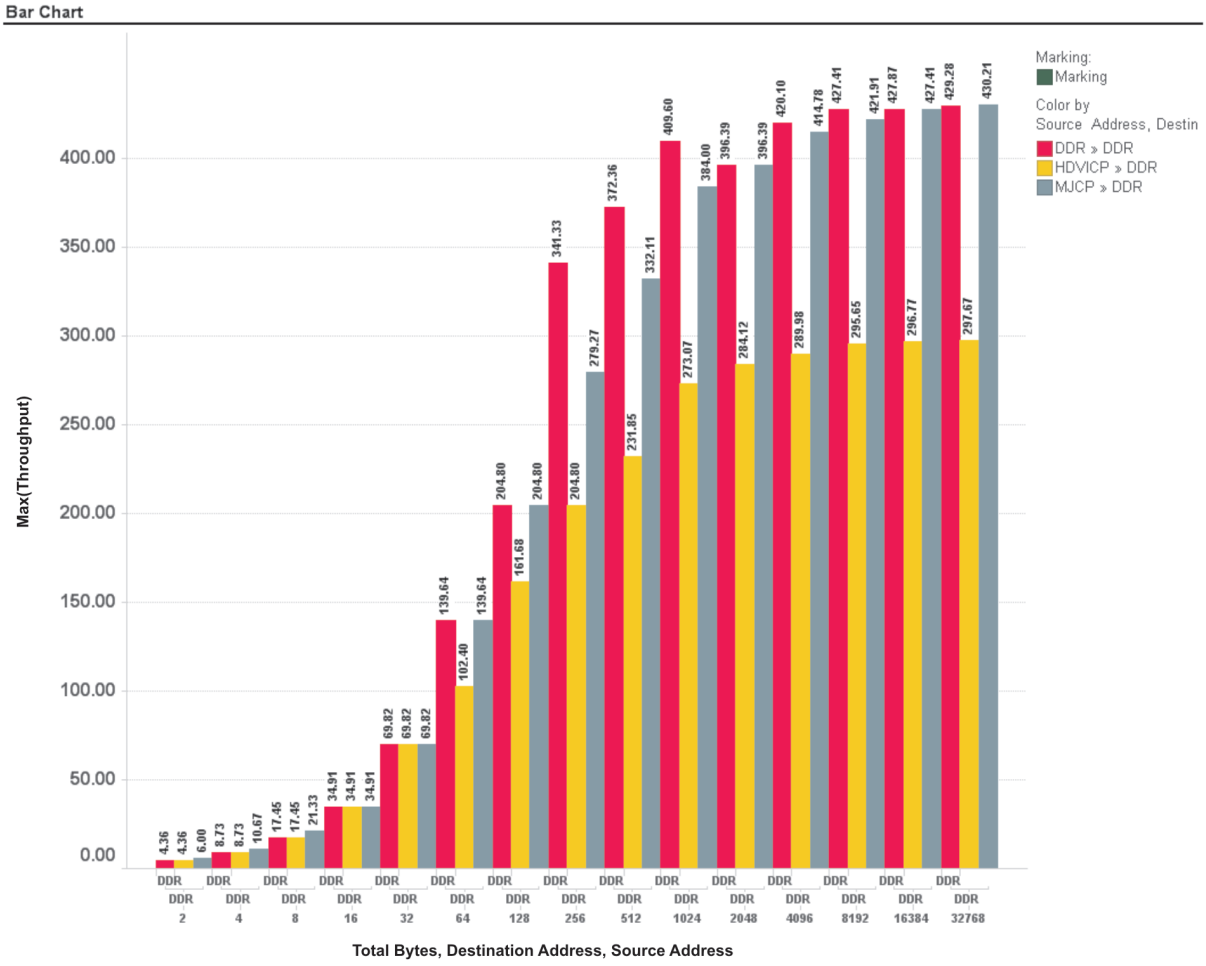



**Figure 11. Throughput of EDMA for HDVICP, MJCP to DDR Access**

As transfer size increases, the impact of the transfer overhead/latency reduces, therefore, the throughput also increases. In the analysis shown here, overhead includes the number of cycles from the logging of the start time stamp to event set, as well as the number of cycles from the EDMA completion interrupt to the logging of the end time stamp. The throughput is above 800 Mbytes per sec for a transfer with a data size greater than 2KB in DDR to HDVICP data transfer.

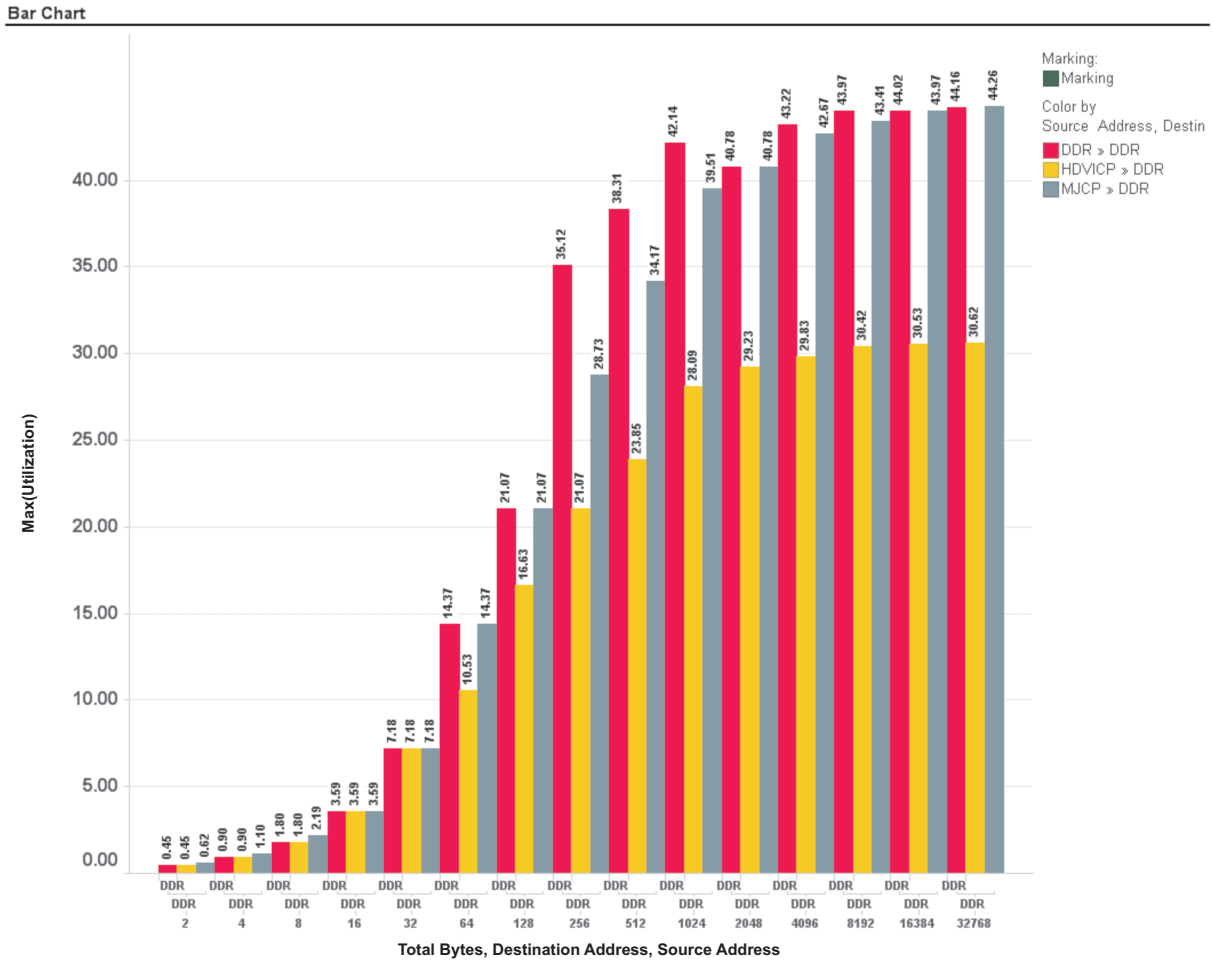



**Figure 12. Utilization of EDMA for HDVICP, MJCP to DDR Access**

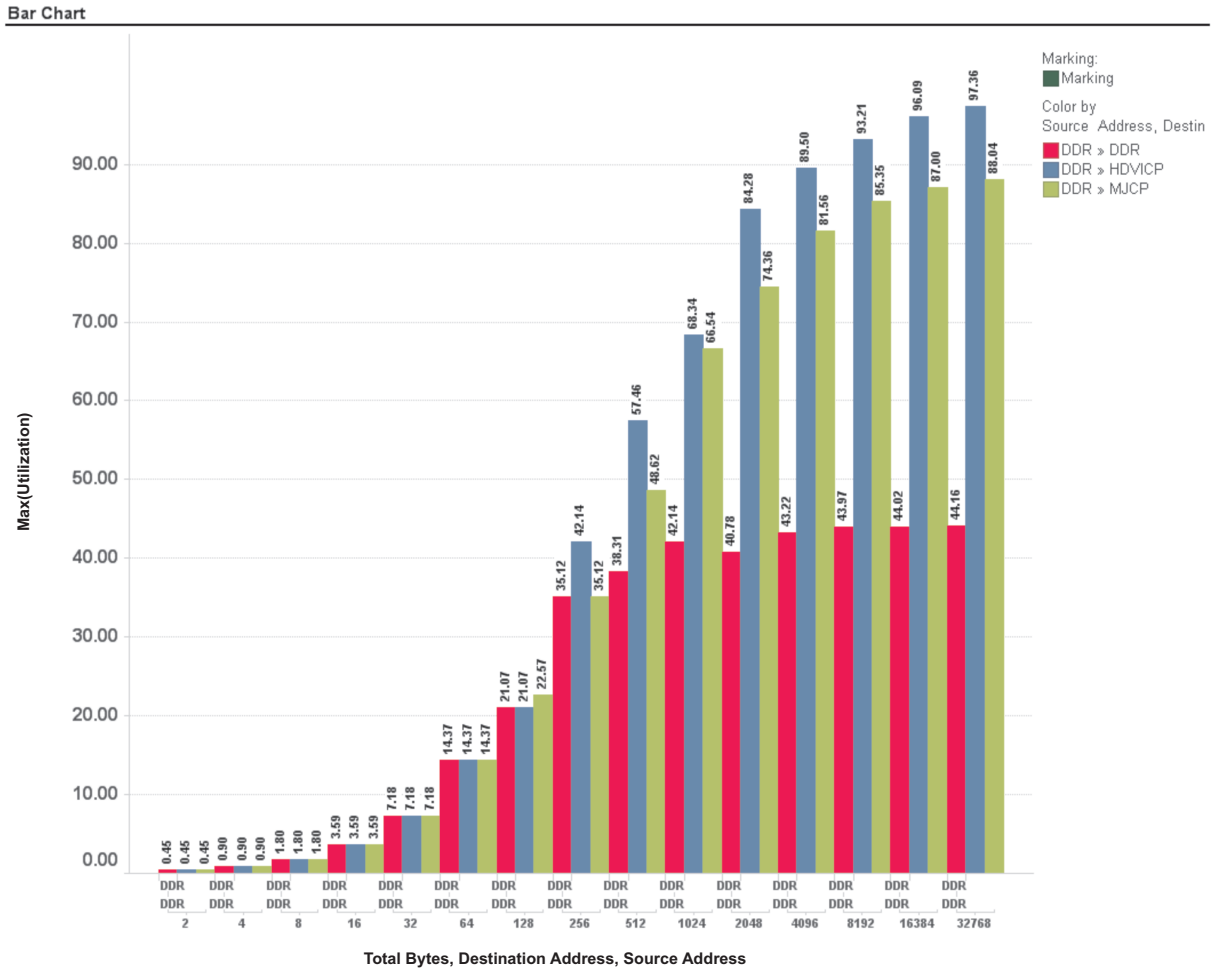



**Figure 13. Utilization of EDMA for DDR to HDVICP, MJCP and DDR Access**

As transfer size increases, the impact of the transfer overhead/latency reduces; therefore, the percentage of utilization also increases. In the analysis shown here, overhead includes the number of cycles from the logging of the start time stamp to event set, as well as the number of cycles from the EDMA completion interrupt to the logging of the end time stamp. The percentage of utilization is above 60% for a transfer with a data size greater than 2KB in DDR to HDVICP data transfer.

### 4.1.5 A-Sync/AB-Sync

An A-sync transfer is configured as follows:

- The number of TRs submitted equals BCNT * CCNT
- Each sync event generates a TR with a transfer size equal to ACNT bytes

Therefore, this configuration results in the following trends:

- Larger ACNT values results in higher bus utilization by submitting larger transfer sizes per sync event that reduces transfer overhead.

Figure 14 shows the percentage of utilization of EDMA for different ACNT value.

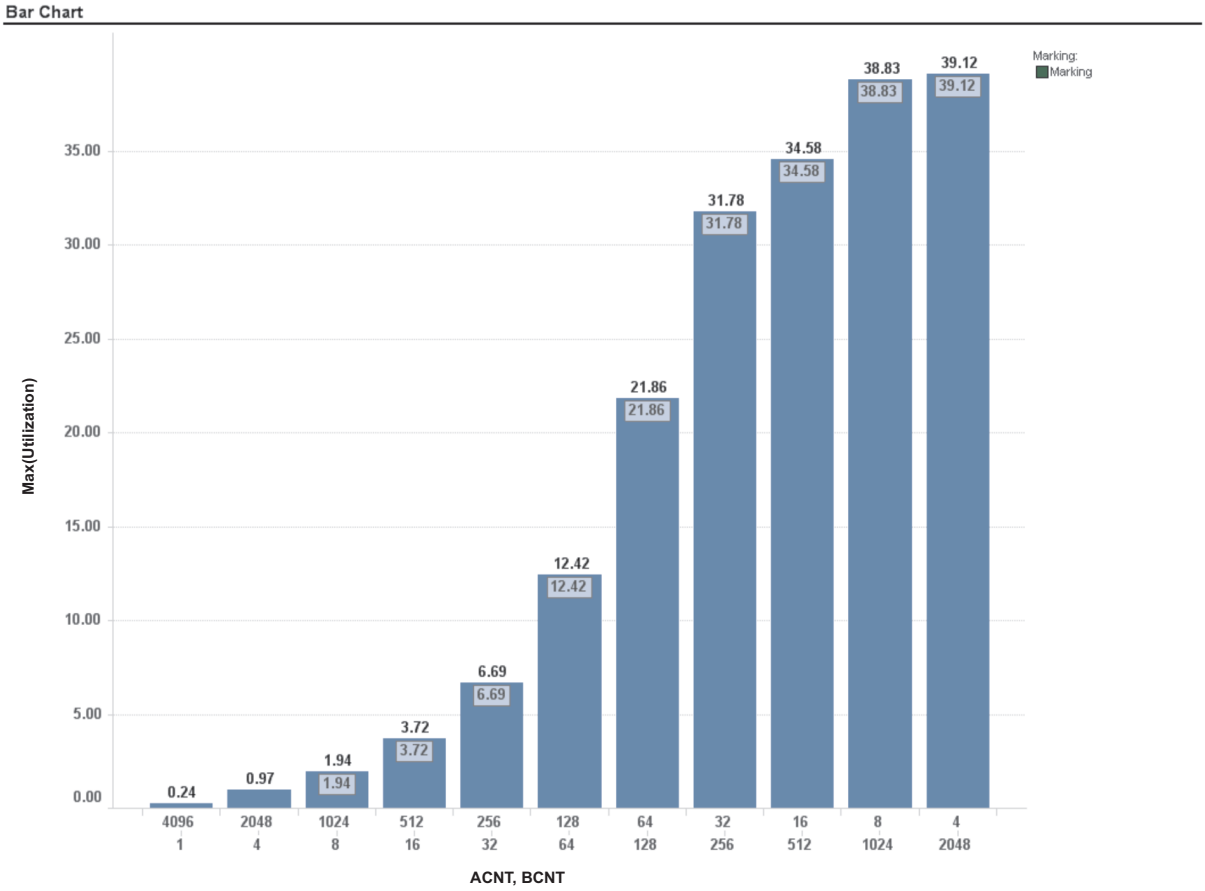



**Figure 14. Utilization for Different Element Size (ACNT)**

The Y-axis represents the percentage of utilization and X-axis represents the different configuration of ACNT and BCNT value to do 8KB transfer between L2 and DDR memory locations. In the case of an AB-sync transfer, the number of TRs submitted is equal to CCNT; each TR causes the transfer of ACNT*BCNT bytes. If the number of TRs submitted for both A-sync and AB-sync is the same, then the throughput value will be almost the same.

Figure 15 shows the effect of A-sync and AB-sync on the performance for different transfer size.

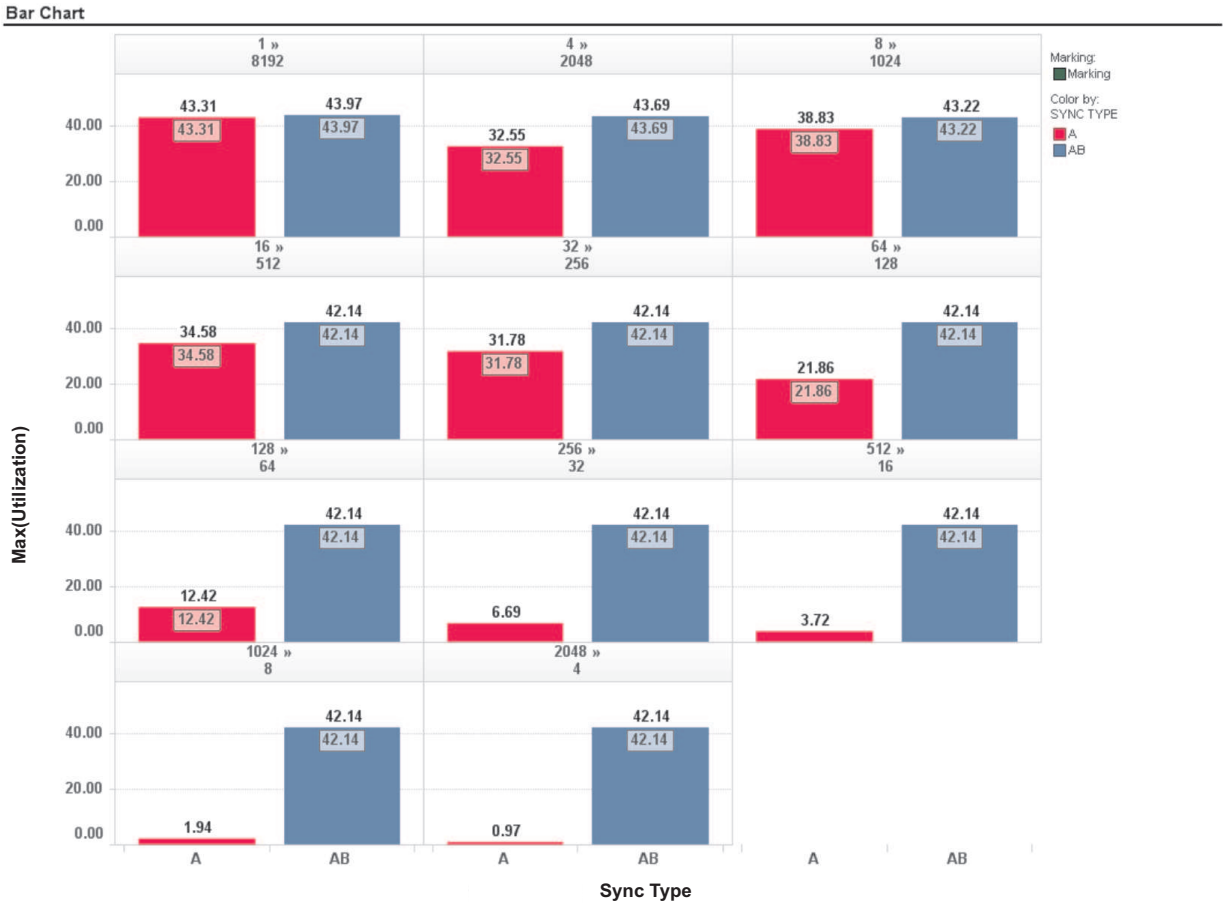



**Figure 15. Effect of A-Sync and AB-Sync (Transfer Size 8192)**

Figure 15 shows the comparison between A-Sync and AB-Sync transfers for same transfer size of 8192 bytes and varying BCNT (CCNT is always 1). The A-Sync transfers are done using chaining to self. The y-axis represents the percentage of utilization for different BCNT and ACNT; in the upper raw of the X Axis first line represents BCNT and second line represents ACNT.

For BCNT equal to 8, the number of TRs submitted for A-sync is eight times more than the number of TRs submitted for AB-sync, which shows slight degradation in performance; whereas, for BCNT equal to 512, the number of TRs submitted for A-sync transfer is 512 times more than AB-sync, which shows huge degradation in performance.

### 4.1.6 Queue TC Usage

On DM36x, there are four transfer controllers to move data between slave end points. The default configuration for the transfer controllers is shown in Table 9.

**Table 9. EDMA3 Transfer Controller Configurations**

| Name | TCO | TC1 | TC2 | TC3 |
|---|---|---|---|---|
| FIFOSIZE | 256 bytes | 256 bytes | 128 bytes | 128 bytes |
| BUSWIDTH | 8 bytes | 8 bytes | 8 bytes | 8 bytes |
| DSTREGDEPTH | 4 entries | 4 entries | 4 entries | 4 entries |
| Default DBS | 32 bytes | 32 bytes | 16 bytes | 16 bytes |

The individual TC performance for paging/memory to memory transfers is essentially dictated by the TC configuration. In most scenarios, the FIFOSIZE configurations for the TC have the most significant impact on the TC performance; the BUSWIDTH configuration is dependent on the device architecture and the DSTREGDEPTH values impact the number of in flight transfers. On the DM36x device, TC0, TC1 transfer controllers yield identical performance for all transfer scenarios and TC2, TC3 transfer controllers yield identical performance for all transfer scenarios Figure 16 and Figure 17 shows the throughput of TC0, TC1 and TC2, TC3, respectively.

The Y-axis represents the throughput in MBps and the X-axis represents the different transfer size for TC0, TC1, TC2 and TC3. On the DM36x device, TC0, TC1 transfer controllers yield identical performance for all transfer scenarios and TC2, TC3 transfer controllers yield identical performance for all transfer scenarios, but it is less than TC0 and TC1 because of the FIFO SIZE and DBS.

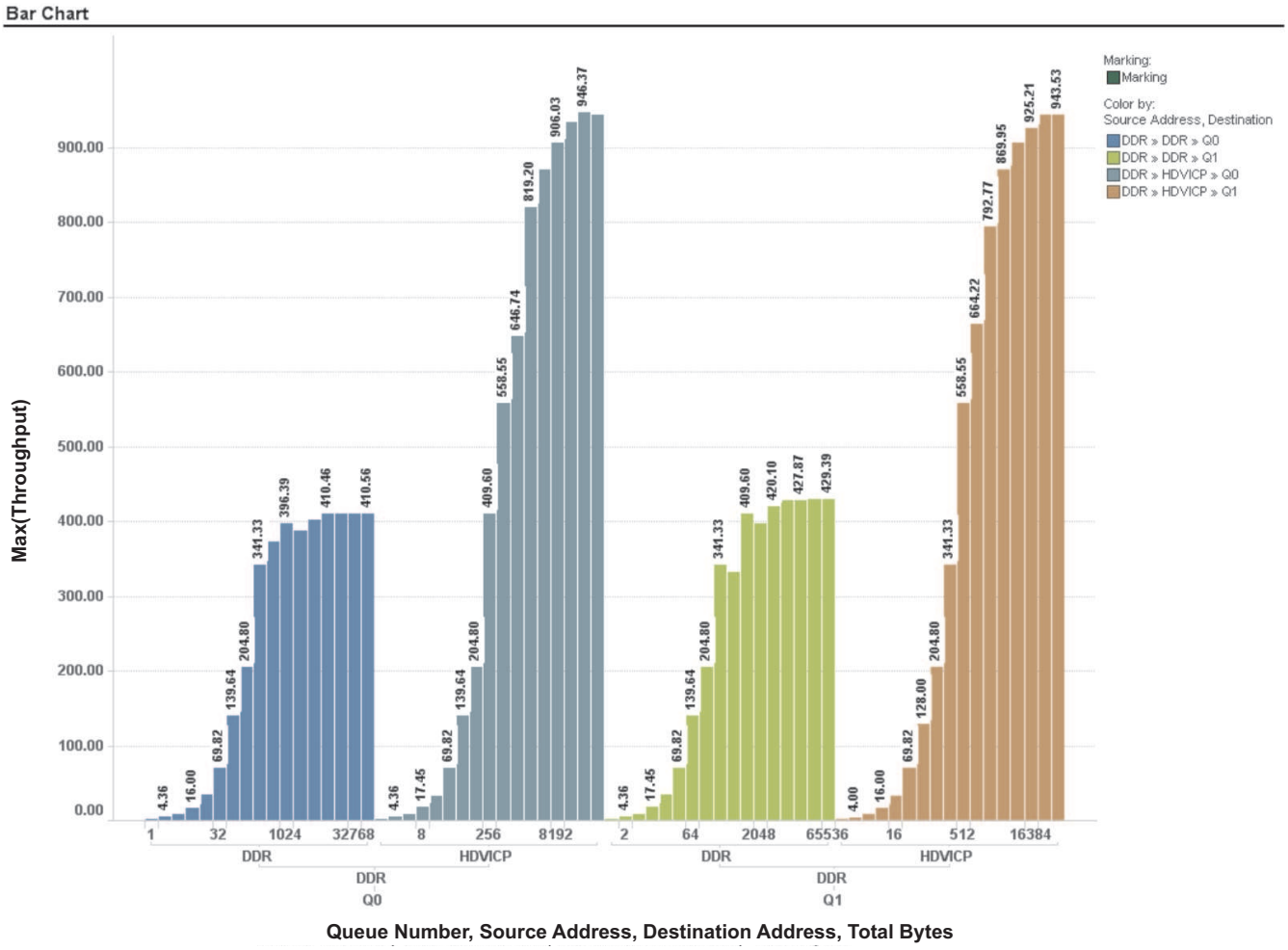



**Figure 16. Performance of TC0 and TC1**

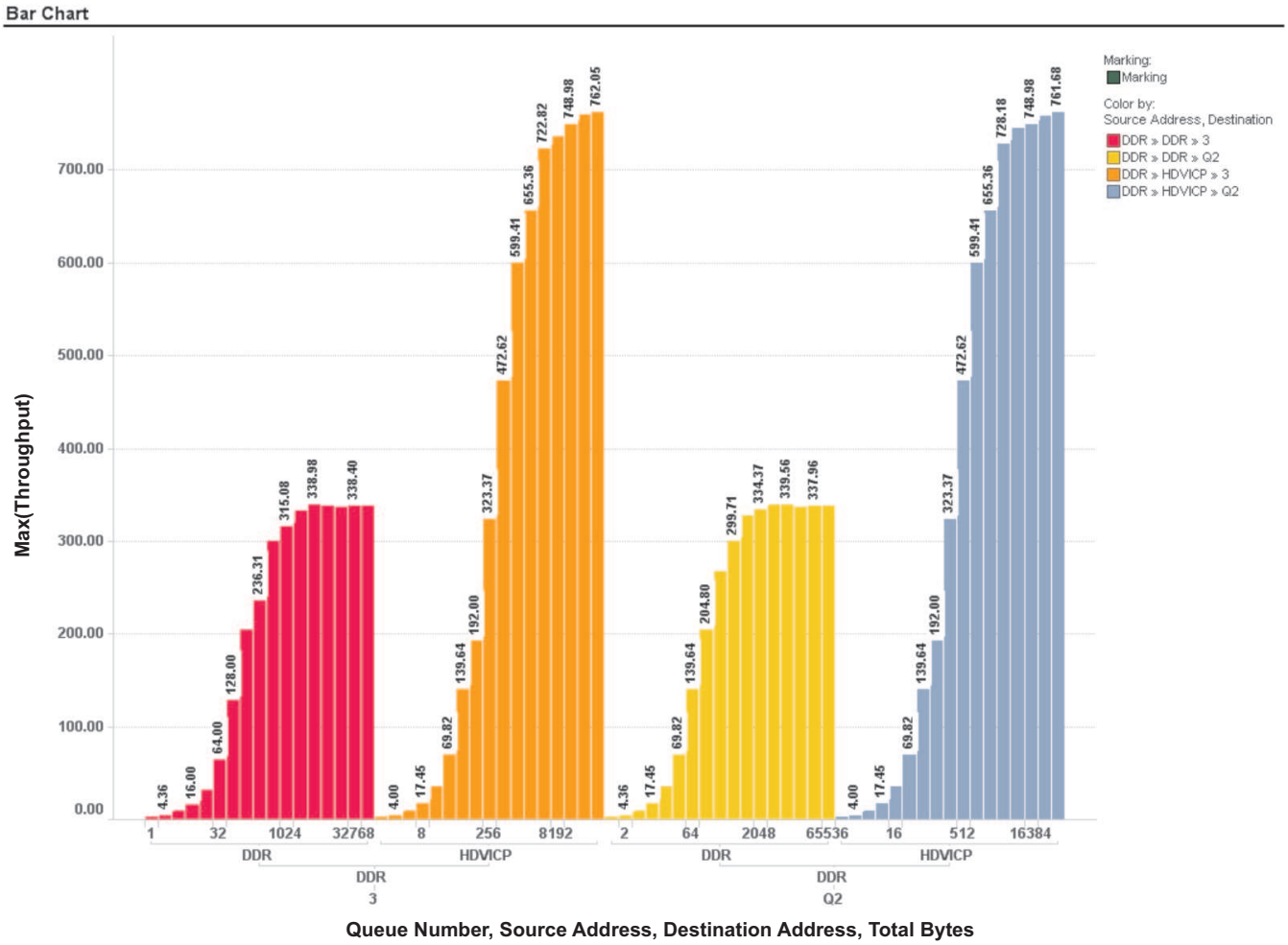



**Figure 17. Performance of TC2 and TC3**

### 4.1.7 Performance of EDMA

Figure 18, Figure 19, and Table 10 capture the best case throughput and bus utilization for various source and destination memory combinations. Table 10 summarizes actual throughput and maximum typical throughput obtained in MBytes/sec along with % of utilization for different source and destination memory combinations. All data shown with ACNT equal to 8KB, BCNT and CCNT equal to 1, A-Sync transfers with increment addressing mode and CPU/DDR/memory setup as specified in Section 4.1.2.

**Table 10. EDMA Performance of EDMA for 8KB Transfer**

| Source Memory | Destination Memory | Actual Throughput MBytes/sec) | Theoretical Maximum Throughput (MBytes/sec) | Utilization (%) |
|---|---|---|---|---|
| DDR | DDR | 429 | 972 | 44.18 |
| | HDVICP | 946 | 972 | 97.36 |
| | MJCP | 855 | 972 | 88.04 |
| HDVICP | DDR | 298 | 972 | 30.71 |
| | MJCP | 298 | 972 | 30.66 |
| MJCP | DDR | 430 | 972 | 44.26 |
| | HDVICP | 430 | 972 | 44.26 |

Figure 18 shows % of utilization on Y-axis and various source and destination memory combinations on X-axis, color coded for extra clarity.

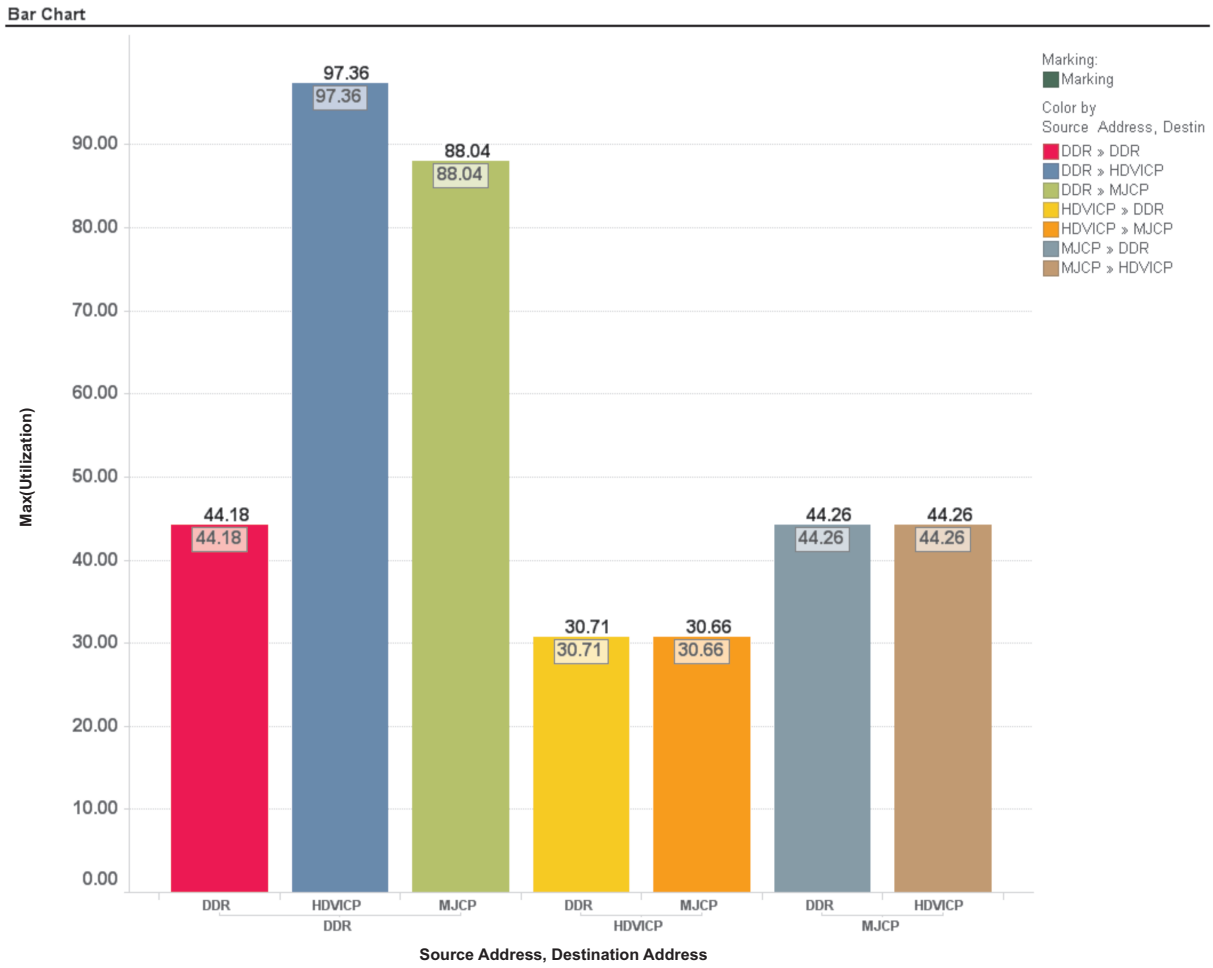



**Figure 18. EDMA Performance**

Figure 19 shows throughput on Y-axis and various source and destination memory combinations on X-axis, color coded for extra clarity

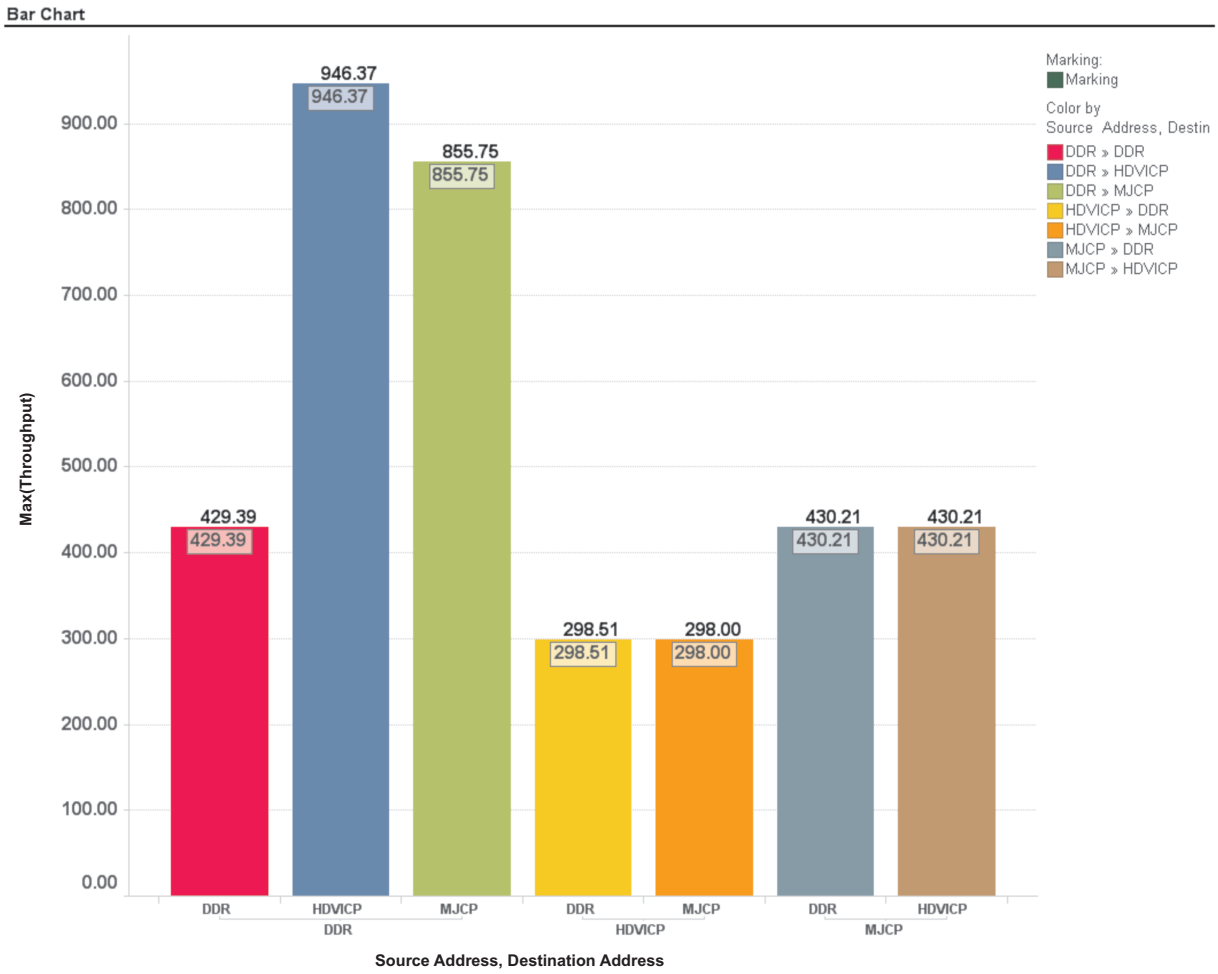



**Figure 19. EDMA Performance**

### 4.1.7.1 ARM and DDR Frequency Variation

Figure 20 and Figure 21 show the performance of EDMA for transfer between HDVICP and DDR for 16KB and 32KB transfer size with different ARM and DDR frequency configuration. Color code for different transfer size is shown in the left of the figure.

Figure 20 show the % utilization for TC2 and TC3.

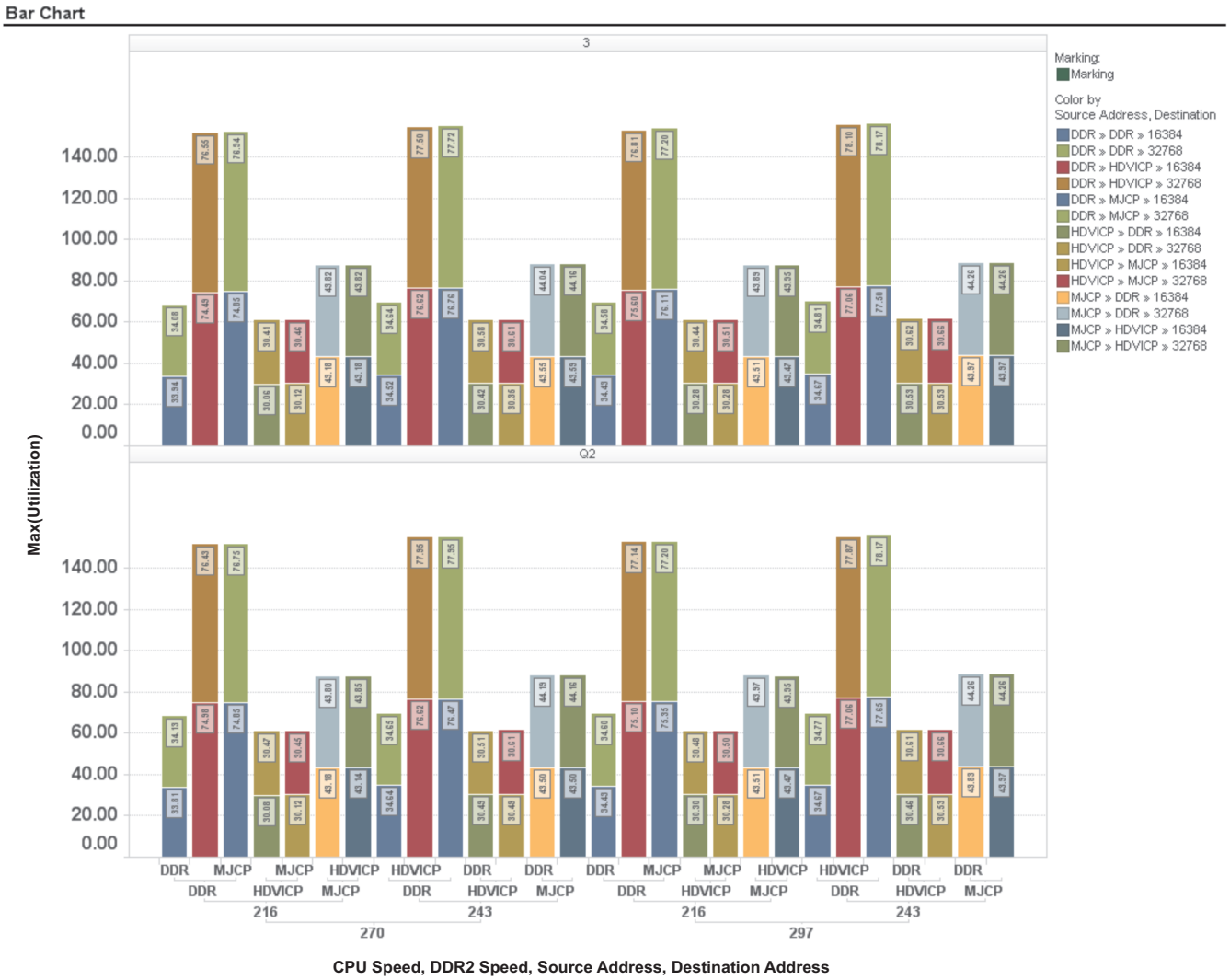



**Figure 20. Max Throughput for Different CPU and DDR Frequency**

The Y-axis represents the maximum throughput for transfer between DDR to HDVICP and DDR to DDR and the X-axis represents different CPU frequency configuration for given DDR frequency.

Figure 21 show the % utilization for TC0 and TC1.

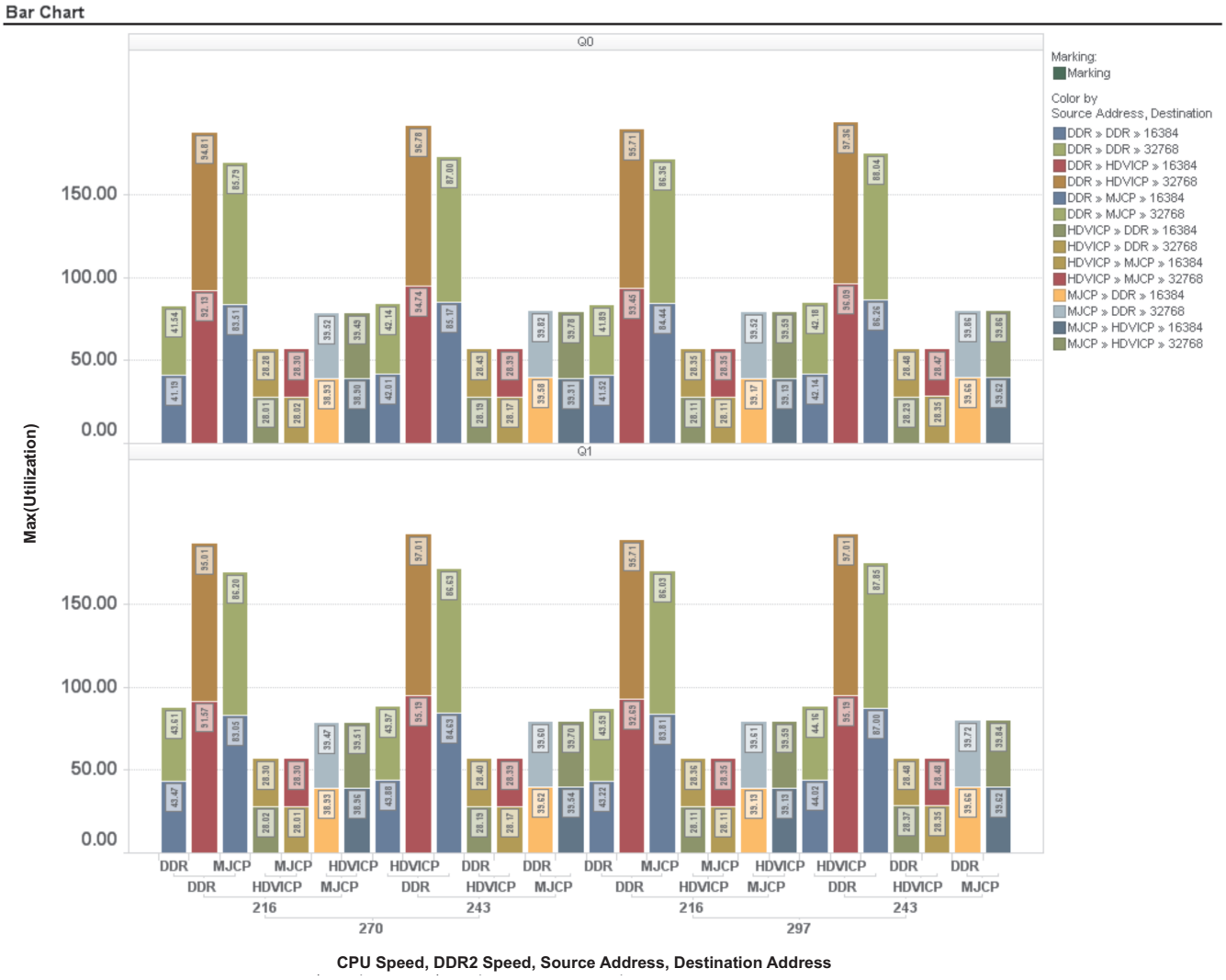



**Figure 21. Utilization for Different CPU and DDR Frequency**

The Y-axis represents the percentage of utilization for transfer between DDR to HDVICP and DDR to DDR and the X-axis represents different CPU frequency configuration for given DDR frequency.

The maximum throughput in MBps and utilization in % is shown in Table 11, when transfer size is 8KB.

**Table 11. EDMA Maximum Throughput for TC0 and TC2**

| Total Bytes | Queue Number | Source Address | Destination Address | CPU Speed | DDR2 Speed | Max of Throughput | Utilization |
|---|---|---|---|---|---|---|---|
| 4096 | Q0 | DDR | DDR | 270 | 216 | 337.81 | 39.1 |
| | | | | | 243 | 399.61 | 41.11 |
| | | | | 297 | 216 | 342.52 | 39.64 |
| | | | | | 243 | 401.24 | 41.28 |
| | | | HDVICP | 270 | 216 | 664.22 | 76.88 |
| | | | | | 243 | 833.08 | 85.71 |
| | | | | 297 | 216 | 687.44 | 79.56 |
| | | | | | 243 | 869.95 | 89.5 |
| | Q2 | DDR | DDR | 270 | 216 | 280.07 | 32.42 |
| | | | | | 243 | 330.99 | 34.05 |
| | | | | 297 | 216 | 289.13 | 33.46 |
| | | | | | 243 | 338.98 | 34.87 |
| | | | HDVICP | 270 | 216 | 574.88 | 66.54 |
| | | | | | 243 | 687.44 | 70.72 |
| | | | | 297 | 216 | 588.65 | 68.13 |
| | | | | | 243 | 728.18 | 74.92 |
| 8192 | Q0 | DDR | DDR | 270 | 216 | 350.46 | 40.56 |
| | | | | | 243 | 403.71 | 41.53 |
| | | | | 297 | 216 | 355.53 | 41.15 |
| | | | | | 243 | 410.46 | 42.23 |
| | | | HDVICP | 270 | 216 | 747.56 | 86.52 |
| | | | | | 243 | 889.63 | 91.53 |
| | | | | 297 | 216 | 768 | 88.89 |
| | | | | | 243 | 906.03 | 93.21 |
| | Q2 | DDR | DDR | 270 | 216 | 288.7 | 33.41 |
| | | | | | 243 | 336.08 | 34.58 |
| | | | | 297 | 216 | 294.32 | 34.07 |
| | | | | | 243 | 339.56 | 34.93 |
| | | | HDVICP | 270 | 216 | 620.21 | 71.78 |
| | | | | | 243 | 730.88 | 75.19 |
| | | | | 297 | 216 | 614.4 | 71.11 |
| | | | | | 243 | 744.73 | 76.62 |

## 4.2 Ethernet Subsystem

This section provides the throughput analysis of the ESS module integrated in the TMS320DM36x SoC.

### 4.2.1 Overview

The Ethernet media access controller (EMAC) provides an efficient interface between DM36x and the network. The DM36x EMAC supports both 10Base-T (10 Mbits/second [Mbps]) and 100Base-TX (100 Mbps) in either half- or full-duplex mode. The EMAC module also supports hardware flow control and quality of service (QOS) support.

The frequencies supported for transmit and receive clocks are fixed by the IEEE 802.3 standard as:

- 2.5 MHz for 10 Mbps
- 25 MHz for 100 Mbps

The EMAC controls the flow of packet data from the DM36x device to the PHY. The management data input/output (MDIO) module controls PHY configuration and status monitoring.

The EMAC module conforms to the IEEE 802.3-2002 standard, describing the *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer* specifications. The IEEE 802.3 standard has also been adopted by ISO/IEC and re-designated as ISO/IEC 8802-3:2000(E).

Deviating from this standard, the EMAC module does not use the Transmit Coding Error signal MTXER. Instead of driving the error pin when an underflow condition occurs on a transmitted frame, the EMAC intentionally generates an incorrect checksum by inverting the frame CRC, so that the transmitted frame is detected as an error by the network

Both the EMAC and the MDIO modules interface to the DM36x device through a custom interface that allows efficient data transmission and reception. This custom interface is referred to as the EMAC control module, and is considered integral to the EMAC/MDIO peripheral. The control module is also used to multiplex and control interrupts.

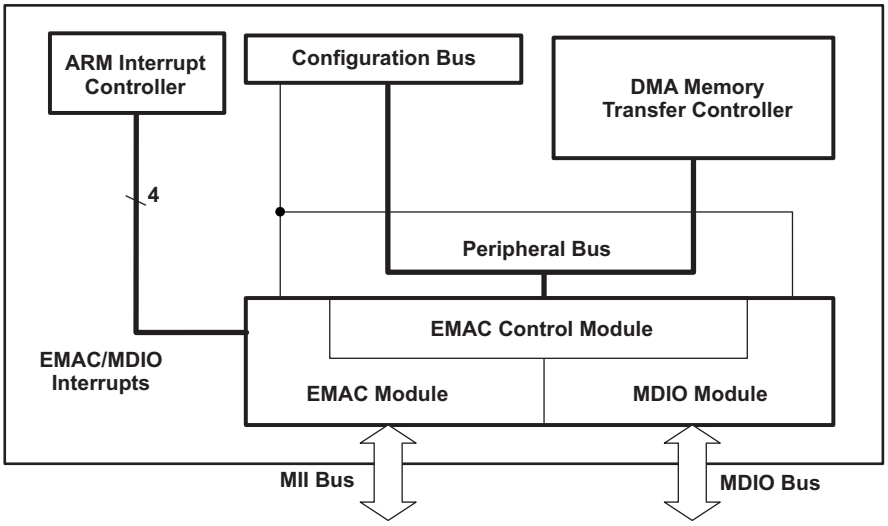



**Figure 22. EMAC and MDIO Block Diagram**

#### 4.2.1.1 Ethernet Media Access Controller (EMAC) Control Module

The basic functions of the EMAC control module (see Figure 23) are to interface the EMAC and MDIO modules to the rest of the system, and to provide for a local memory space to hold EMAC packet buffer descriptors. Local memory is used to help avoid contention to device memory spaces. Other functions include the bus arbiter, and interrupt control and pacing logic control.

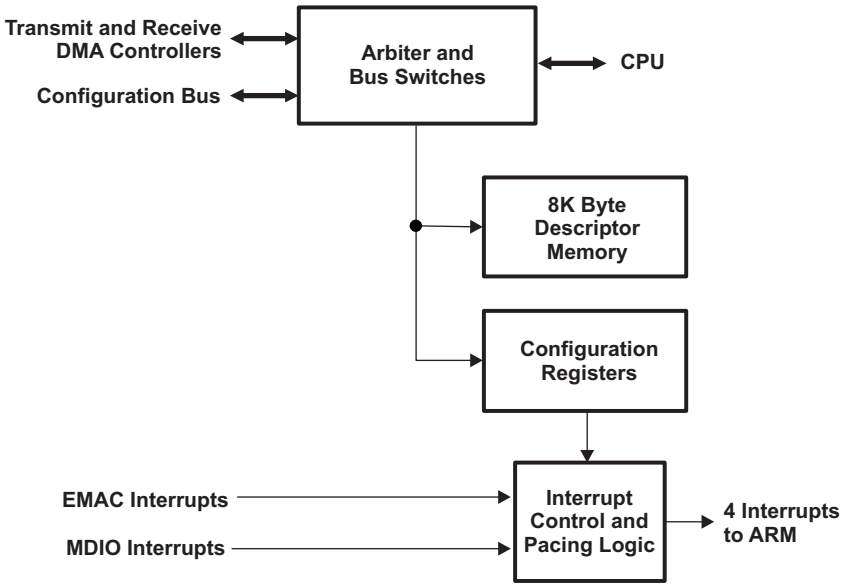



**Figure 23. EMAC Control Module Block Diagram**

#### 4.2.1.2 Management Data Input/Ouput (MDIO) Module

The DM36x device supports a single PHY being connected to the EMAC at any given time. The MDIO module is designed to allow almost transparent operation of the MDIO interface with little maintenance from the CPU. The MDIO module continuously polls 32 MDIO addresses in order to enumerate all PHY devices in the system. Once a PHY device has been detected, the MDIO module reads the MDIO PHY link status register (LINK) to monitor the PHY link state. Link change events are stored in the MDIO module, which can interrupt the CPU. This storing of the events allows the CPU to poll the link status of the PHY device without continuously performing MDIO module accesses. However, when the CPU must access the MDIO module for configuration and negotiation, the MDIO module performs the MDIO read or write operation independent of the CPU. This independent operation allows the processor to poll for completion or interrupt the CPU once the operation has completed.

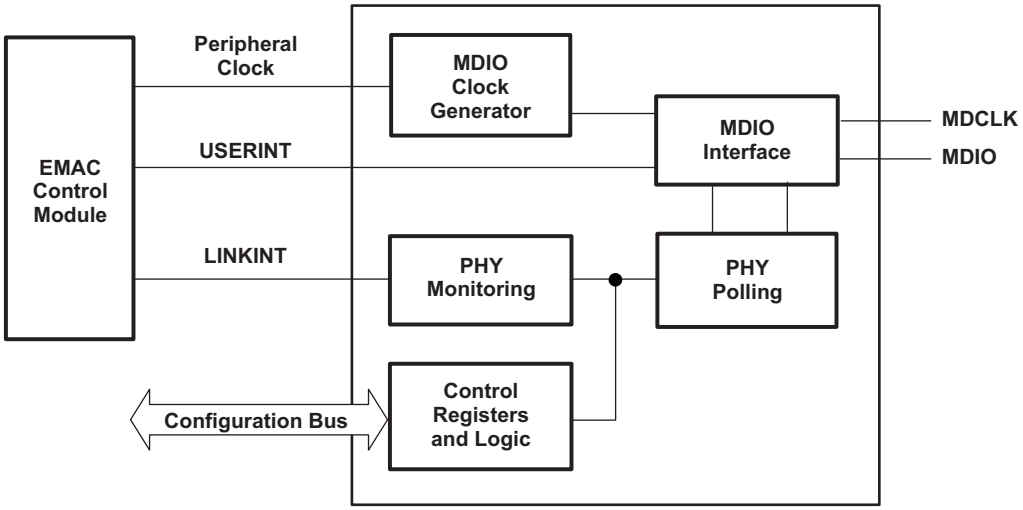



**Figure 24. MDIO Module Block Diagram**

#### 4.2.1.3 *Ethernet Media Access Controller (EMAC) Module*

The EMAC module provides an efficient interface between the processor and the networked community. The EMAC on the device supports 10BaseT (10Mbits/second) and 100BaseT (100Mbits/second) in either half-duplex with hardware flow control and quality-of service (QOS) support.

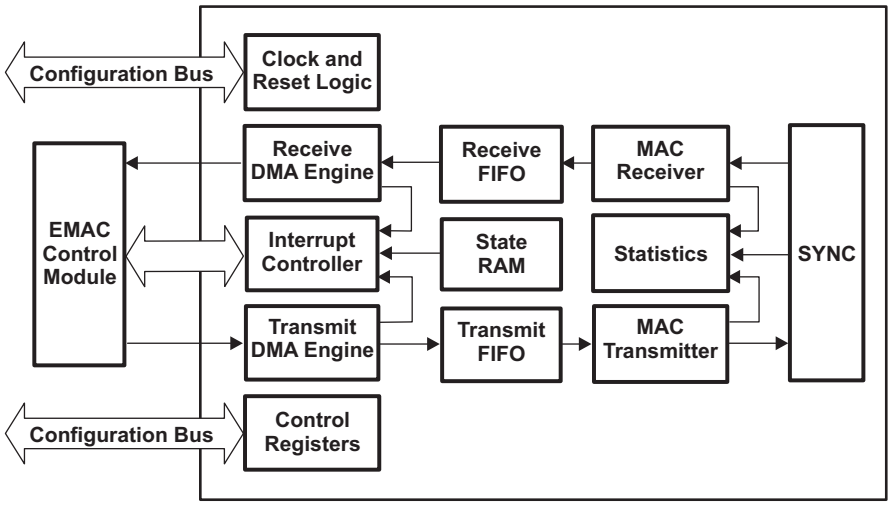
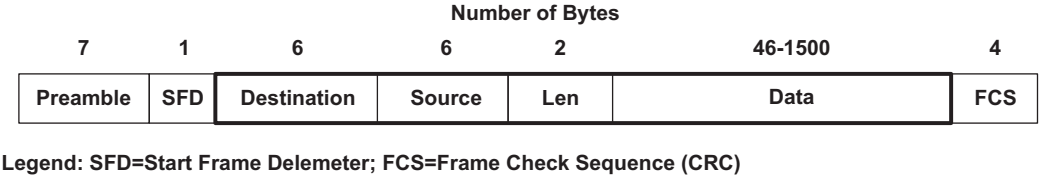



**Figure 25. EMAC Module Block Diagram**

The format of an Ethernet frame is shown in Figure 26 and described in Table 12. The data portion of a single Ethernet frame on the wire is shown outlined in bold. The Ethernet frames are of variable lengths, with no frame smaller than 64 bytes or larger than RXMAXLEN bytes.




**Legend: SFD=Start Frame Delemeter; FCS=Frame Check Sequence (CRC)**

**Figure 26. Ethernet Frame Format**

**Table 12. Ethernet Frame Format Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Preamble | | Preamble. These 7 bytes have a fixed value of 55h and serve to wake up the receiving EMAC ports and to synchronize their clocks to that of the sender's clock. |
| 1 | SFD | | Start of Frame Delimiter. This field with a value of 5Dh immediately follows the preamble pattern and indicates the start of important data. |
| 6 | Destination | | Destination address. This field contains the Ethernet MAC address of the EMAC port that the frame is intended. It may be an individual or multicast (including broadcast) address. When the destination EMAC port receives an Ethernet frame with a destination address that does not match any of its MAC physical addresses, an no promiscuous, multicast or broadcast channel is enabled, it discards the frame. |
| 6 | Source | | Source address. This field contains the MAC address of the Ethernet port that transmits the frame to the local area network. |
| 2 | Len | | Length/Type Field. The length field indicates the number of EMAC client data bytes contained in the subsequent data field of the frame. This field can also be used to identify the type of data the frame is carrying. |
| 46 to (RXMAXLEN 18) | Data | | Data Field. This field carries the datagram containing the upper layer protocol frame, that is, the IP layer datagram. The maximum transfer unit (MTU) of Ethernet (RXMAXLEN - 18) bytes. This means that if the upper layer protocol datagram exceeds (RXMAXLEN - 18) bytes, then the host has to fragment the datagram and send it in multiple Ethernet packets. The minimum size of the data field is 46 bytes. This means that if the upper layer datagram is less than 46 bytes, the data field has to be extended to 46 bytes by appending extra bits after the data field, but prior to calculating and appending the FCS. |
| 4 | FCS | | Frame Check Sequence. A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence covers the 60 to (RXMAXLEN - 4) bytes of the packet data. Note that this 4-byte may or may not be included as part of the packet data, depending on how the EMAC is configured. |

### 4.2.2 Test Environment

The common system setup for the EMAC throughput measurement is given below:

- All tests are performed on DM36x
- ARM clock frequency is 270 MHz
- DDR clock frequency is 216 MHz
- AEMIF clock configuration is:
  - Read time cycles (setup/strobe/hold): 35 (6/26/3)
  - Write time cycles (setup/strobe/hold): 20 (6/11/3)
  - Data bus width
- TCM memory: WAIT cycles enabled, takes 5 ARM cycles to access memory
- Throughput data collected is standalone, no other ongoing traffic
- Readings are collected for 10 Mbps and 100 Mbps

### 4.2.3 Factors Affecting the EMAC Throughput

The EMAC throughput depends on the packet size, the type of memory it accesses through DMA, and the descriptors memory location. To properly configure a system, it is important to know which configuration offers the best performance.

The different factors and their impact on the EMAC throughput are given in Table 13.

**Table 13. Factors Considered for Throughput**

| Factor | Impact | General Recommendation |
|---|---|---|
| Packet Size | Performance is less for small packet size due to transfer overhead/latency | Configure EMAC for Image packet size |
| Descriptor Memory Location | Performance is less for slow memory like AEMIF | Configure descriptor memory location to the EMAC internal RAM for best performance |
| Source Memory Location | Performance is less for slow memory like AEMIF | Configure source memory location to the DDR |
| Destination Memory Location | Performance is less for slow memory like AEMIF | Configure source memory location to the DDR |

#### 4.2.3.1 Packet Size

Figure 27 shows the effect of the different packet sizes on the EMAC throughput for 100 Mbps mode. The number of packets transferred is 10. As the packet size increases, the throughput value also increases irrespective of source address, destination address, and descriptor address. Once the packet is transferred, the EMAC or CPU accesses the next descriptor address until the end of all the packets. The descriptor memory is accessed more frequently if the packet size is less than the EMAC or CPU, which adds more delay in total transfer time and causes degradation in the throughput for smaller packet size. Figure 27 shows that the EMAC performance is better if the packet size is above 400 bytes.

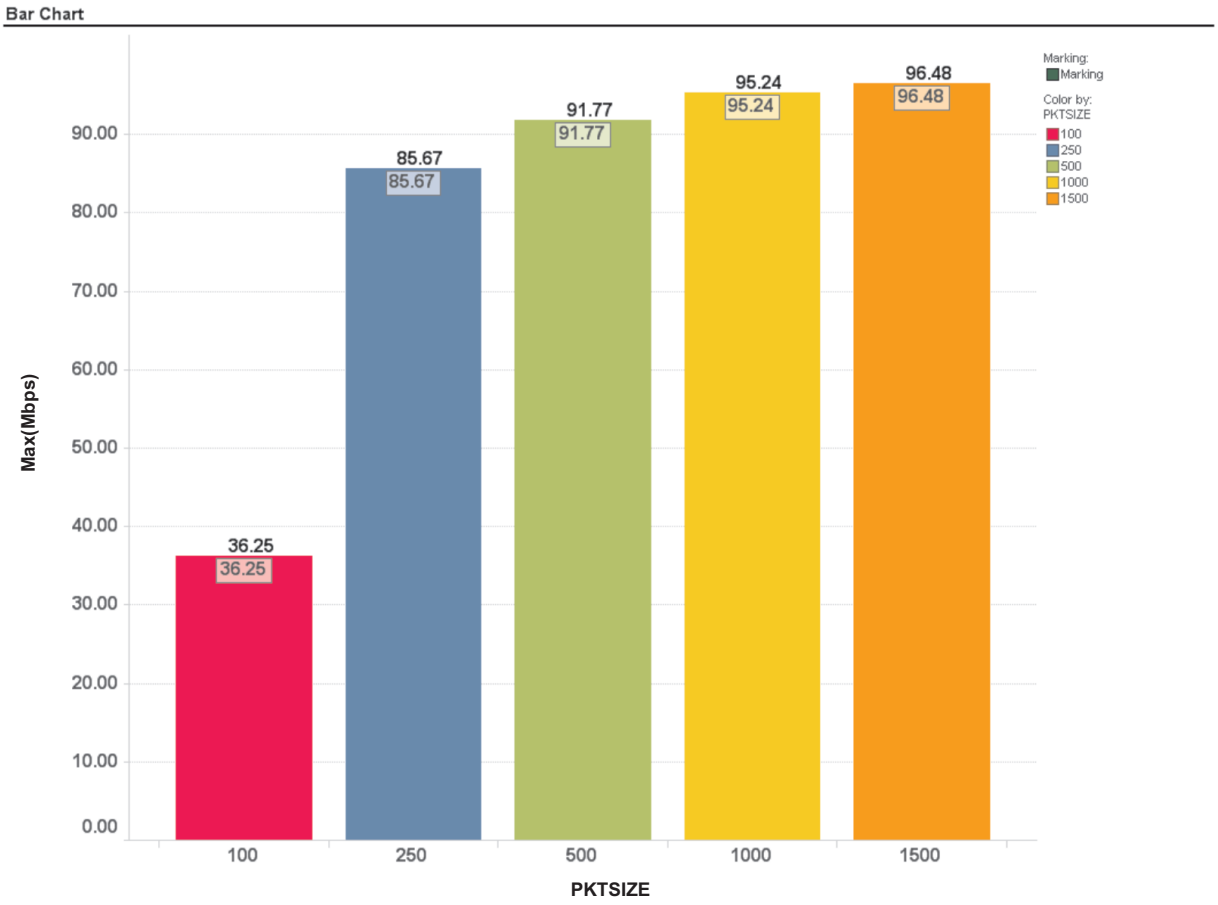



**Figure 27. Effect of Packet Size on the EMAC Throughput for 100 Mbps Mode**

### 4.2.3.2 Descriptor Memory Location

Figure 28 shows the effect of the descriptor memory location on the EMAC throughput. The EMAC or CPU accesses the descriptor memory for each transfer. It takes more time if it is placed in slow memory, which causes degradation in the performance. For better performance, the descriptor memory should be placed in fast memory, i.e., DDR. EMAC has internal memory of 8KB to hold the descriptors and can be configured to transfer up to 1500 packets without any CPU intervention. Figure 28 shows that the performance is better if the descriptors are kept in the EMAC internal RAM.

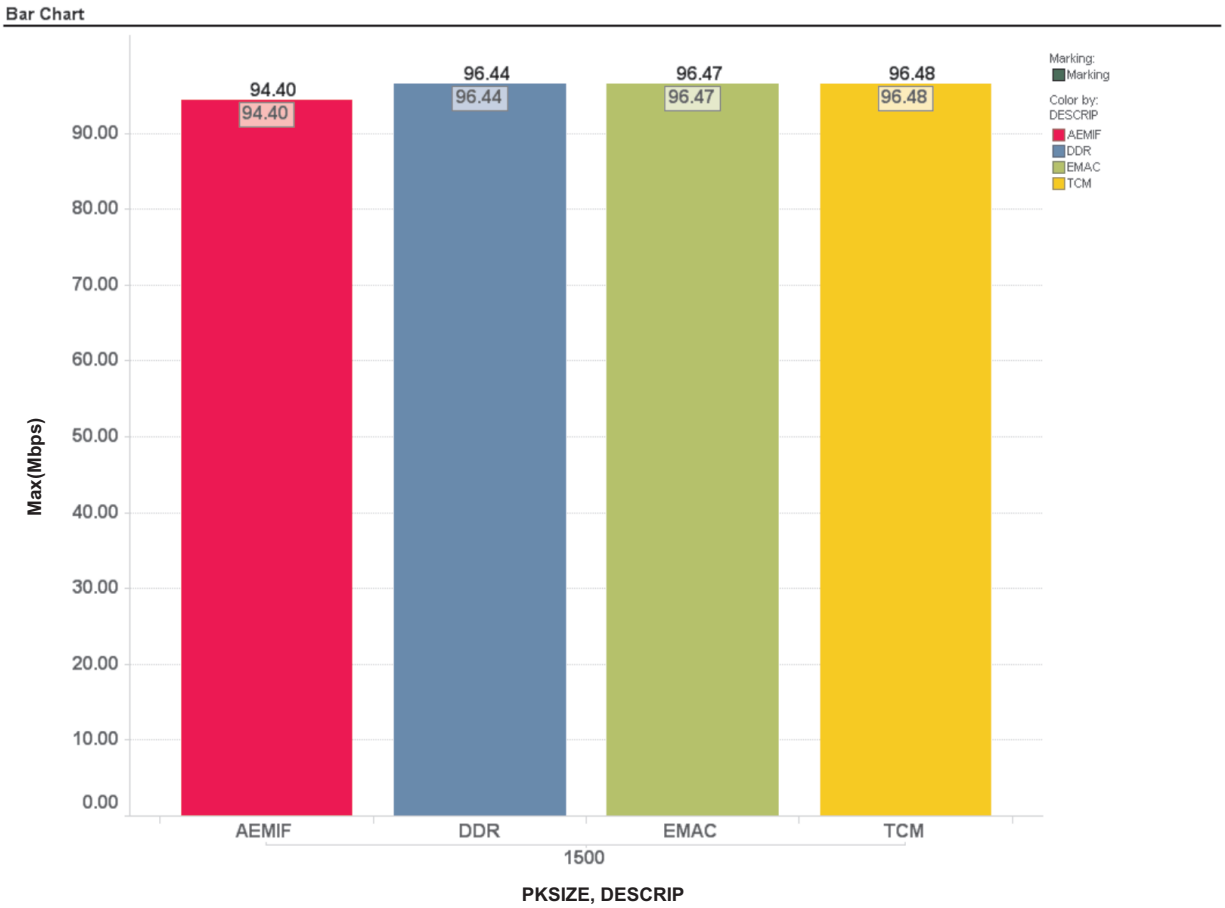



**Figure 28. Effect of Descriptor Memory Location on the EMAC Throughput for 100 Mbps Mode**

### 4.2.3.3 Source Memory Location

Figure 29 shows the effect of the source memory location on the EMAC throughput for 100 Mbps. Once the EMAC transfer is triggered, the DMA accesses the data from the source memory address. DMA takes more time to access If the source data is kept in the slow memory, i.e., AEMIF, which causes degradation in the performance. For better performance, source memory must be kept in fast memory, i.e., DDR.

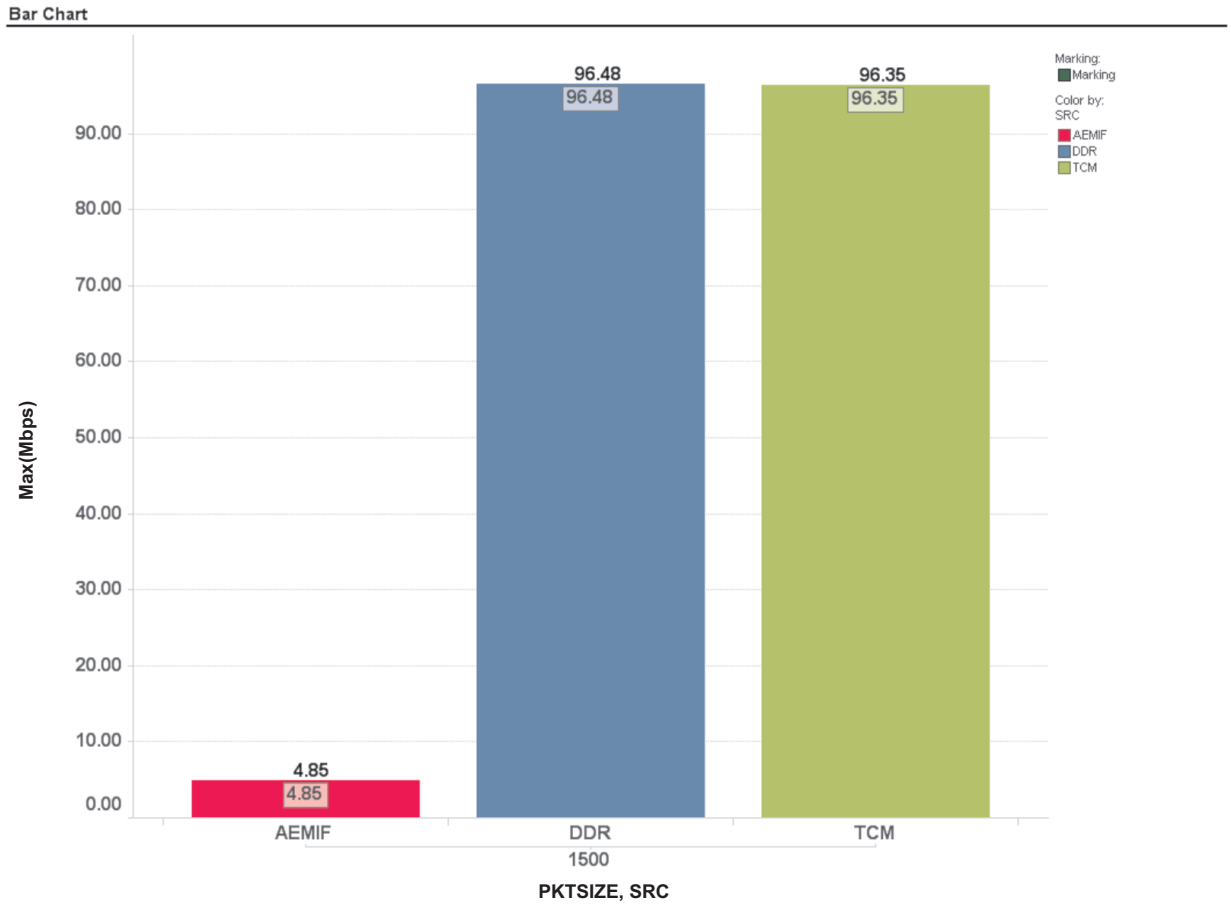



**Figure 29. Effect of Source Memory Location on the EMAC Throughput for 100 Mbps Mode**

#### 4.2.3.4 Destination Memory Location

Figure 30 shows the effect of the destination memory location on the EMAC throughput for 100 Mbps. Once the EMAC transfer is triggered, DMA puts the data to the destination memory address. It takes more time to access DMA if the destination memory is configured to slow memory, i.e., AEMIF, which causes degradation in the performance. For better performance, destination memory must be kept in fast memory like DDR.

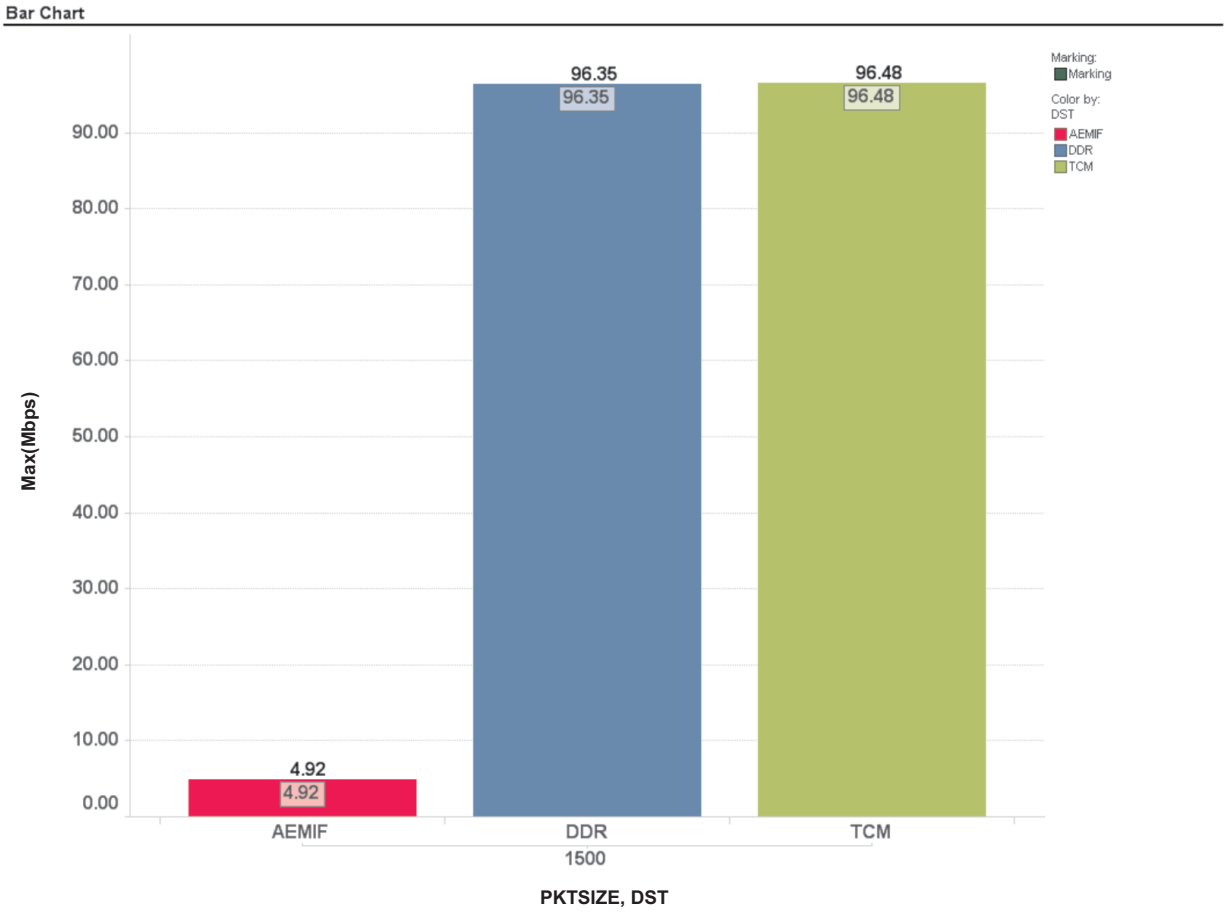



**Figure 30. Effect of Destination Memory Location on the EMAC Throughput for 100 Mbps Mode**

### 4.2.4 The Best EMAC Configuration

Figure 31 shows the EMAC throughput value for all combinations of descriptor memory locations, source memory locations, and destination memory locations. The packet size is configured for 1500 bytes and the packet number is configured for 10 packets.

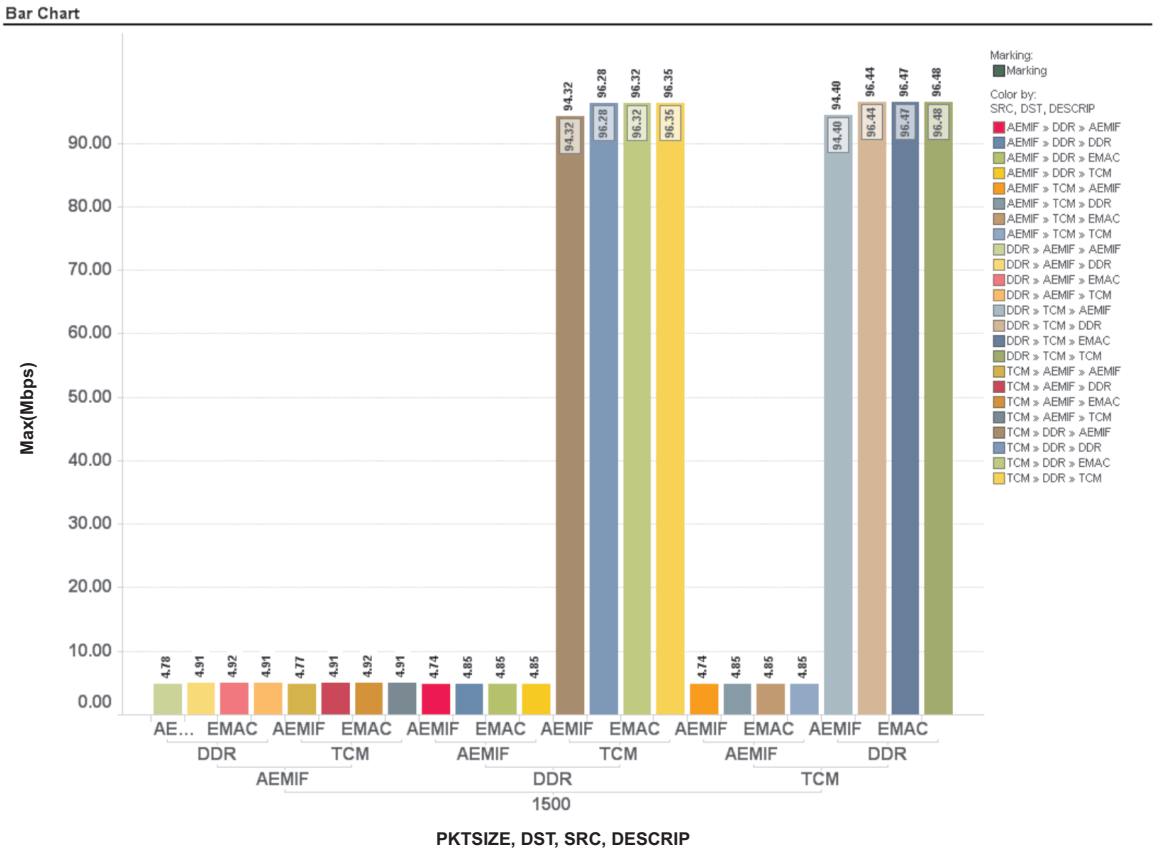



**Figure 31. Effect of Different Memory Locations on the EMAC Throughput**

Table 14 shows the throughput values for all combinations of descriptor memory locations, source memory locations, and destination memory. The best EMAC configuration is shown in bold in Table 14.

**Table 14. Effect of Different Memory on the EMAC Throughput**

| Source | Destination | Descriptor | Mbps |
|---|---|---|---|
| AEMIF | DDR | AEMIF | 4.74 |
| | | DDR | 4.85 |
| | | EMAC | 4.85 |
| | | TCM | 4.85 |
| | TCM | AEMIF | 4.74 |
| | | DDR | 4.85 |
| | | EMAC | 4.85 |
| | | TCM | 4.85 |

**Table 14. Effect of Different Memory on the EMAC Throughput (continued)**

| Source | Destination | Descriptor | Mbps |
|---|---|---|---|
| DDR | AEMIF | AEMIF | 4.78 |
| | | DDR | 4.91 |
| | | EMAC | 4.92 |
| | | TCM | 4.91 |
| | TCM | AEMIF | **94.4** |
| | | DDR | **96.44** |
| | | EMAC | **96.47** |
| | | TCM | **96.48** |
| TCM | AEMIF | AEMIF | 4.77 |
| | | DDR | 4.91 |
| | | EMAC | 4.92 |
| | | TCM | 4.91 |
| | DDR | AEMIF | **94.32** |
| | | DDR | **96.28** |
| | | EMAC | **96.32** |
| | | TCM | **96.35** |

## *4.3 4.3 Multimedia Card/Secure Digital Card (MMC/SD)*

The MMC/SD card controller supports the industry standards with the exceptions noted below:

- Multimedia Card (MMC) Specification V3.31
- Secure Digital (SD) Physical Layer Specification V1.1
- Secure Digital Input/Output(SDIO) Specification V2.0

### 4.3.1 MMC/SD Overview

The MMC/SD controller uses the MMC/SD protocol to communicate with the MMC/SD cards. The MMC/SD controller can be configured to work as an MMC or SD controller, based on the type of card with which the controller is communicating. Figure 32 summarizes the MMC/SD mode interface and illustrates how the controller is interfaced to the cards in MMC/SD mode. The MMC/SD controller contains a single 512-bit FIFO that is used for both reading data from the memory card and writing data to the memory card.

In the MMC/SD mode, the controller supports one or more MMC/SD cards. When multiple cards are connected, the MMC/SD controller selects one by using identification broadcast on the data line. The following MMC/SD controller pins are used:

- CMD: This pin is used for two-way communication between the connected card and the MMC/SD controller. The MMC/SD controller transmits commands to the card and the memory card drives responses to the commands on this pin.
- DAT0 or DAT0-3: MMC cards use only one data line (DAT0) and SD cards use one or four data lines. The number of DAT pins (the data bus width) is set by the WIDTH bit in the MMC control register.
- CLK: This pin provides the clock to the memory card from the MMC/SD controller.

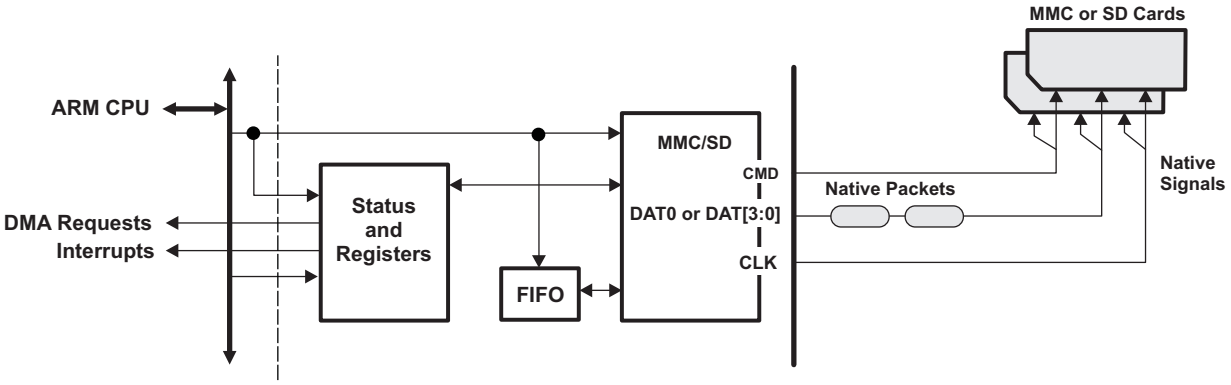



**Figure 32. MMC/SD Card Controller Block Diagram**

### 4.3.2 MMC/SD Clock Overview

The MMC/SD controller has two clocks, the function clock and the memory clock. The function clock determines the operational frequency of the MMC/SD controller and is the input clock to the MMC/SD controller on the device. The MMC/SD controller is capable of operating with a function clock up to 121.5 MHz.

The memory clock appears on the SD_CLK pin of the MMC/SD controller interface. The memory clock controls the timing of communication between the MMC/SD controller and the connected memory card. The memory clock is generated by dividing the function clock in the MMC/SD controller. The divide-down value is set by CLKRT bits in the MMC Memory Clock Control Register (MMCCLK) and is determined by the equation:

```
Memory clock frequency = function clock frequency/(2 x (CLKRT + 1)), (When div4 bit of MMCCLK is
0)
```

or

```
Memory clock frequency = function clock frequency/(4 x (CLKRT + 1)), (When div4 bit of MMCCLK is
1)
```

The MMC/SD card throughput is tightly related to the serial clock. In the current test environment, the MMC/SD serial clock can only be sourced internally. Therefore, the maximum serial clock rate is 30 MHz, obtained by setting the two clock dividers to 1. The theoretical maximum throughput is 120 Mbps (30 MHz x 4 line communication), regardless of receiving or transmitting.

### 4.3.3 Test Environment

The common system setup in this throughput analysis is as follows:

- ARM clock rate: 270 MHz
- DDR clock rate: 243 MHz
- MMC/SD master clock rate: 30 MHz

### 4.3.4 Factors Affecting MMC/SD Throughput

Table 15 lists the factors that might affect MMC/SD throughput.

**Table 15. Factors Affecting MMC/SD Throughput**

| Factor | Impact | General Recommendation |
|---|---|---|
| SRC/DST Buffer Location | Different memories have different EDMA access latencies. Too long an EDMA access might result In untimely service. | Avoid locating SRC/DST buffers in AEMIF memory due to its long access delay. |
| FIFO Trigger Level Used to Generate EDMA Event | MMCSD has 512 bit FIFO. The EDMA transmit or receive event can be generated when FIFO is full up to 256 bit or 512 bit. If we select 256 bit as a trigger level, then we can utilize the FIFO effectively. | FIFO trigger level is 256 so the FIFO is well utilised between the EDMA and the MMC/SD card. |

### 4.3.5 SRC/DST Buffer Location

Figure 33 and Figure 34 shows the % utilization for MMC/SD write and read operation when write or read happens from different buffer locations.

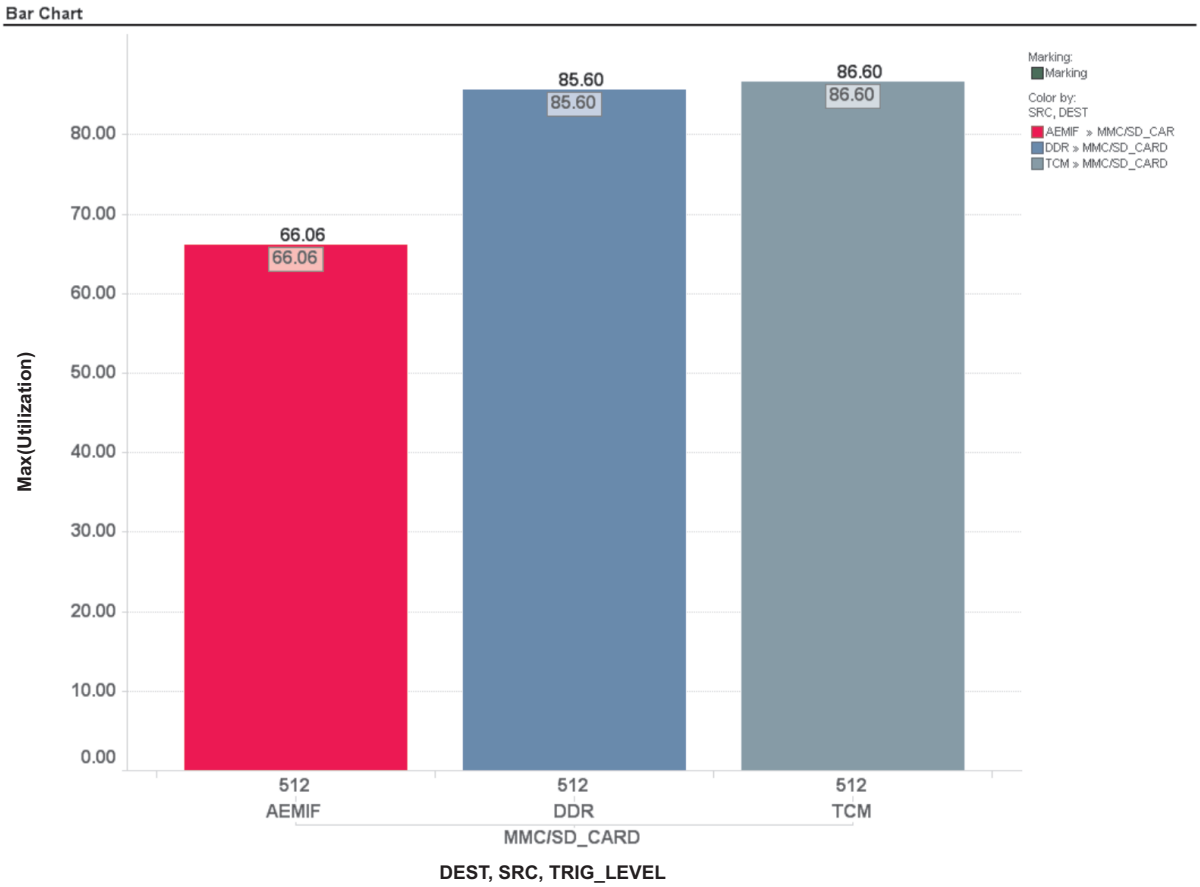



**Figure 33. Percentage Utilization for MMC/SD Write**

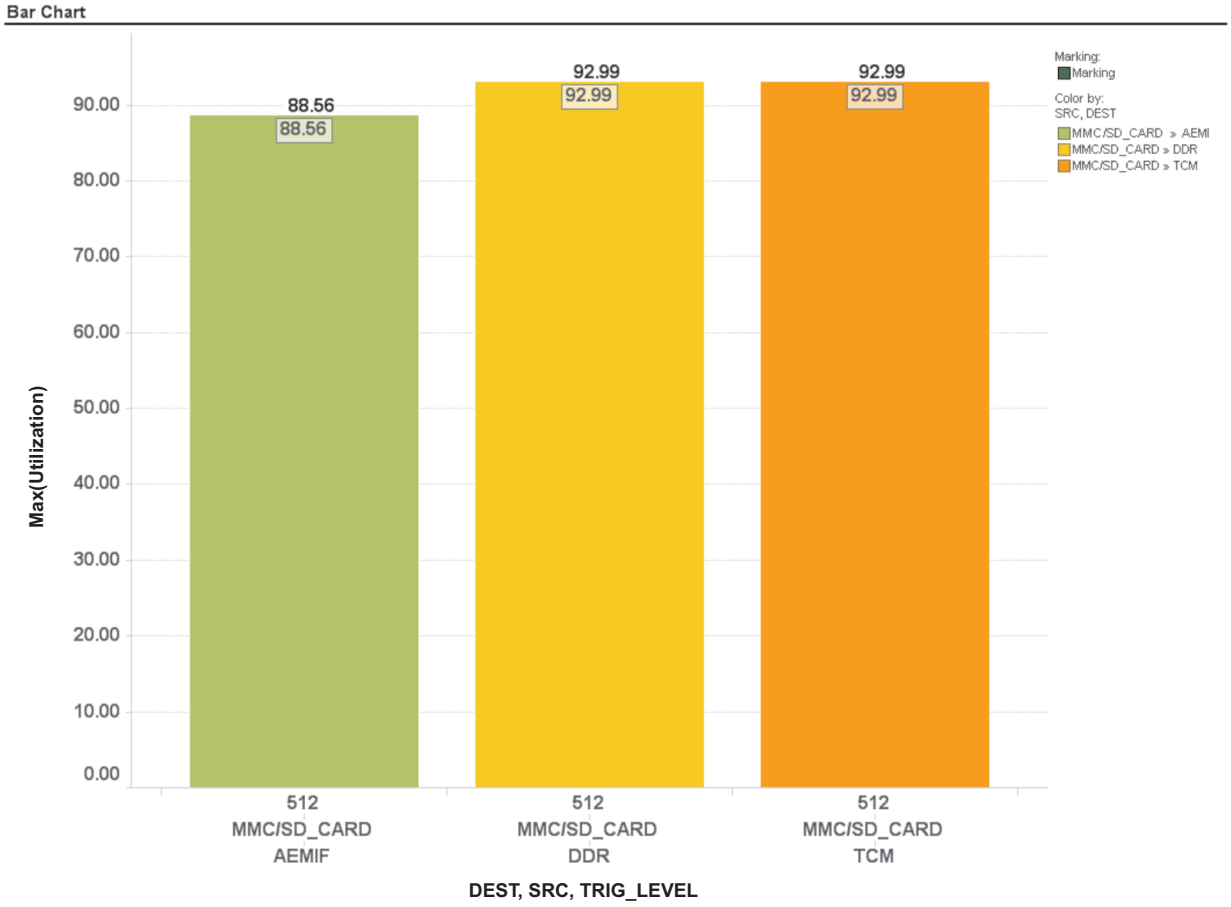



**Figure 34. Percentage Utilization for MMC/SD Read**

Figure 35 and Figure 36 shows the throughput in Mbps for MMC/SD write and read operation when write or read happens from different buffer locations.

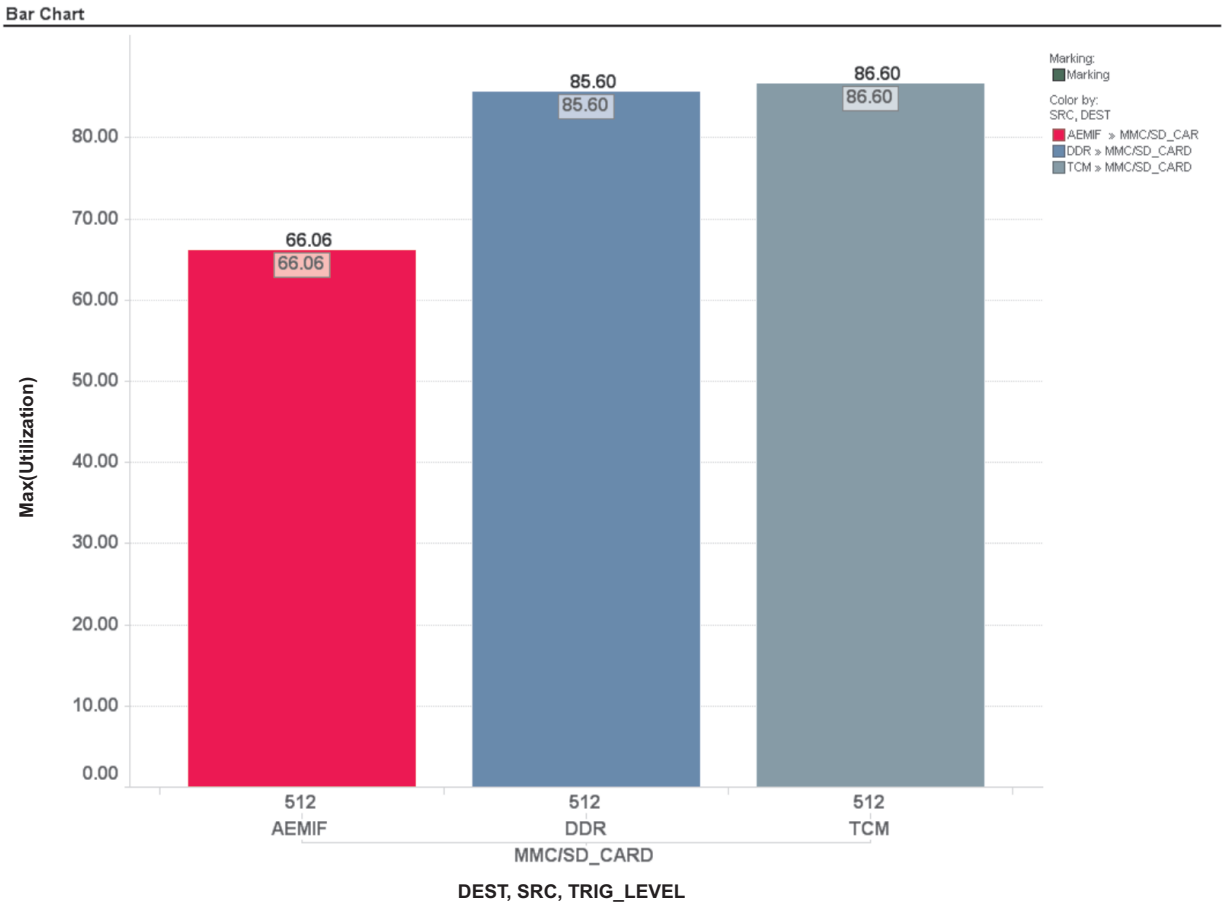



**Figure 35. Throughput for MMC/SD Write**

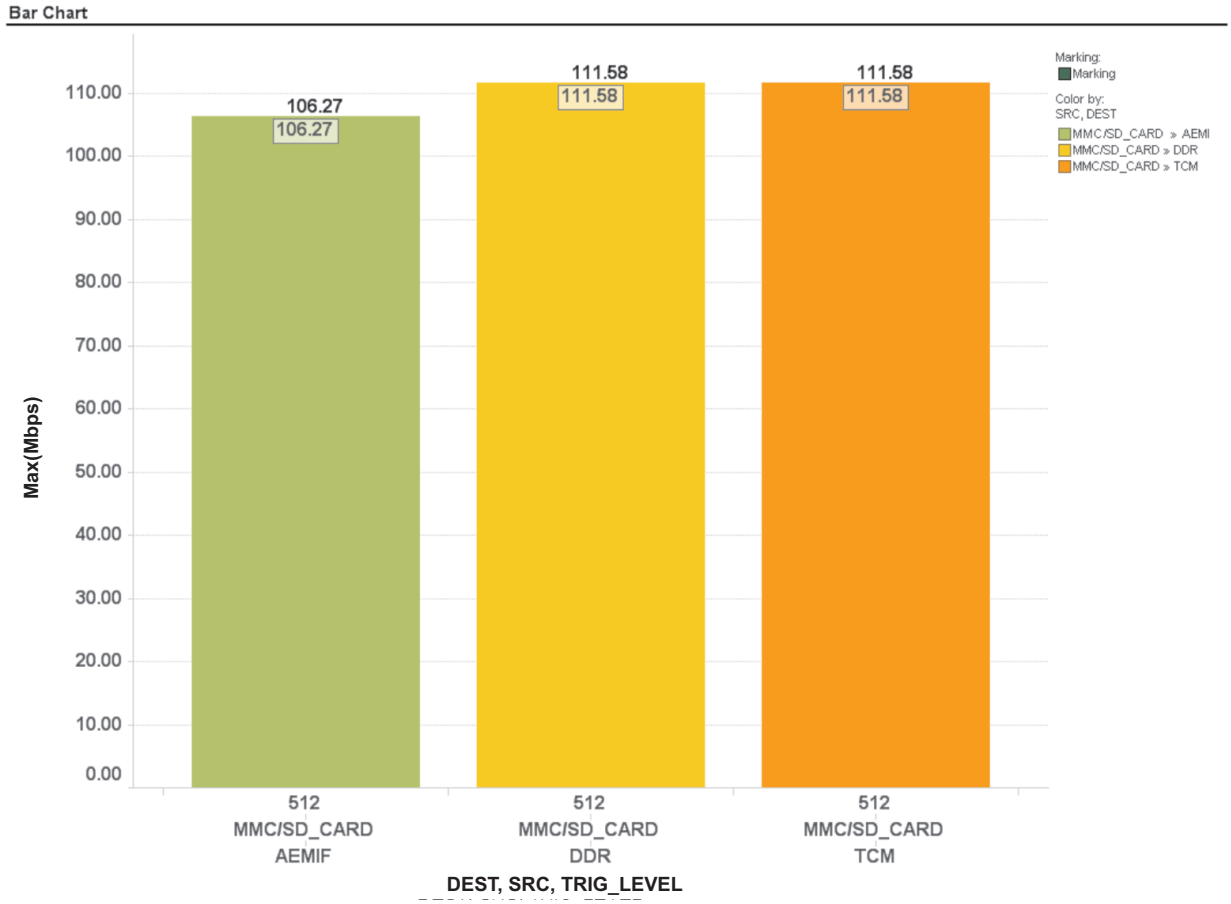



**Figure 36. Throughput for MMC/SD Read**

Internal memory has the shortest EDMA access latency for both read and write. Compared to that, DDR memory has a longer latency; AEMIF memory has the longest latency of all. For example, when SRC/DST buffers are both in internal memory or DDR memory, MMC/SD can service data at the maximum bit clock, achieving 115 Mbps throughput. When the SRC buffer is in AEMIF memory, MMC/SD can only maintain accurate transfer at 115 Mbps.

### 4.3.6 FIFO Trigger Levels

MMCSD has 512 bit FIFO. EDMA transmit or receive event can be generated when FIFO is full up to 256 bit or 512 bit. We can utilize the FIFO effectively If we select 256 bit as a trigger level. When the MMC/SD card controller uses half of the FIFO for card read and write operation, the other half of the FIFO is utilized by EDMA to continue the transfer of data to or from the DDR\TCM\AEMIF buffer location. Both the EDMA and the MMC/SD card controller can work concurrently.

Figure 37 and Figure 38 shows the throughput in Mbps for MMC/SD write and read operation when write or read happens with different FIFO trigger levels.

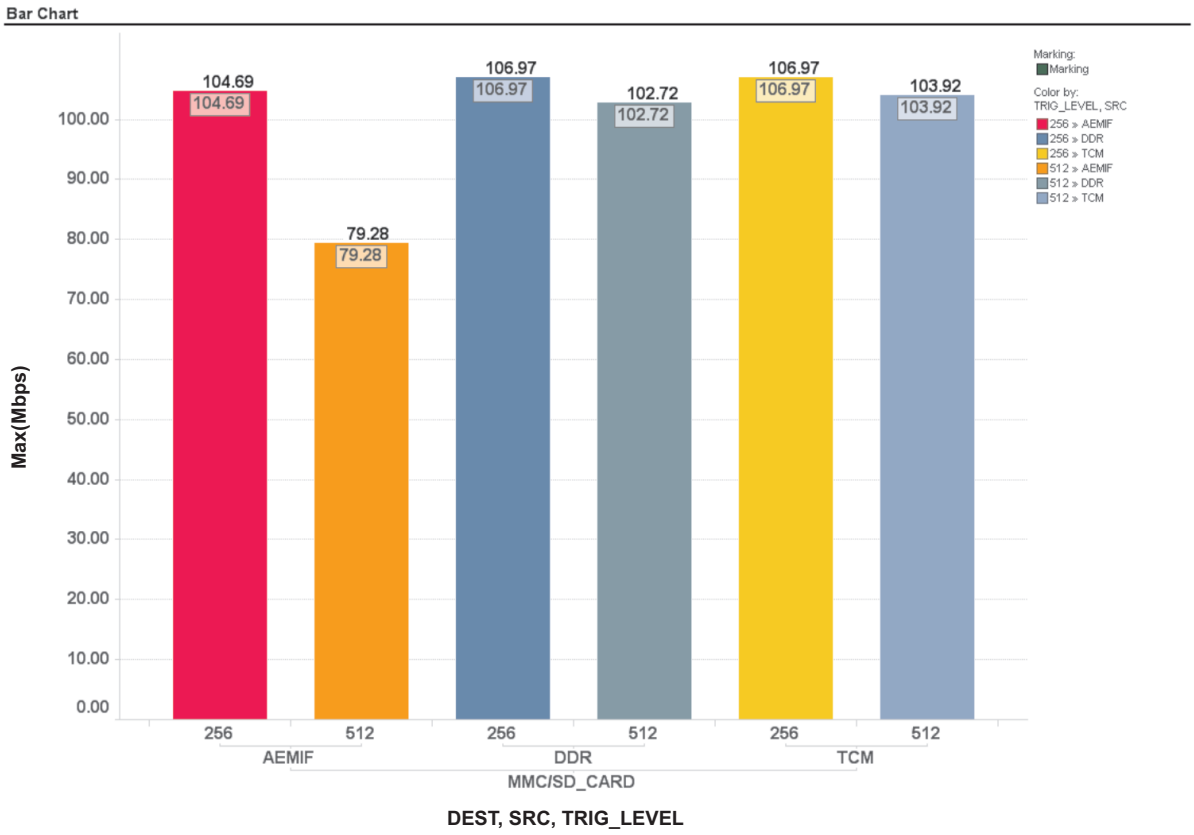



**Figure 37. Throughput for MMC/SD Write With Different FIFO Trigger Levels**

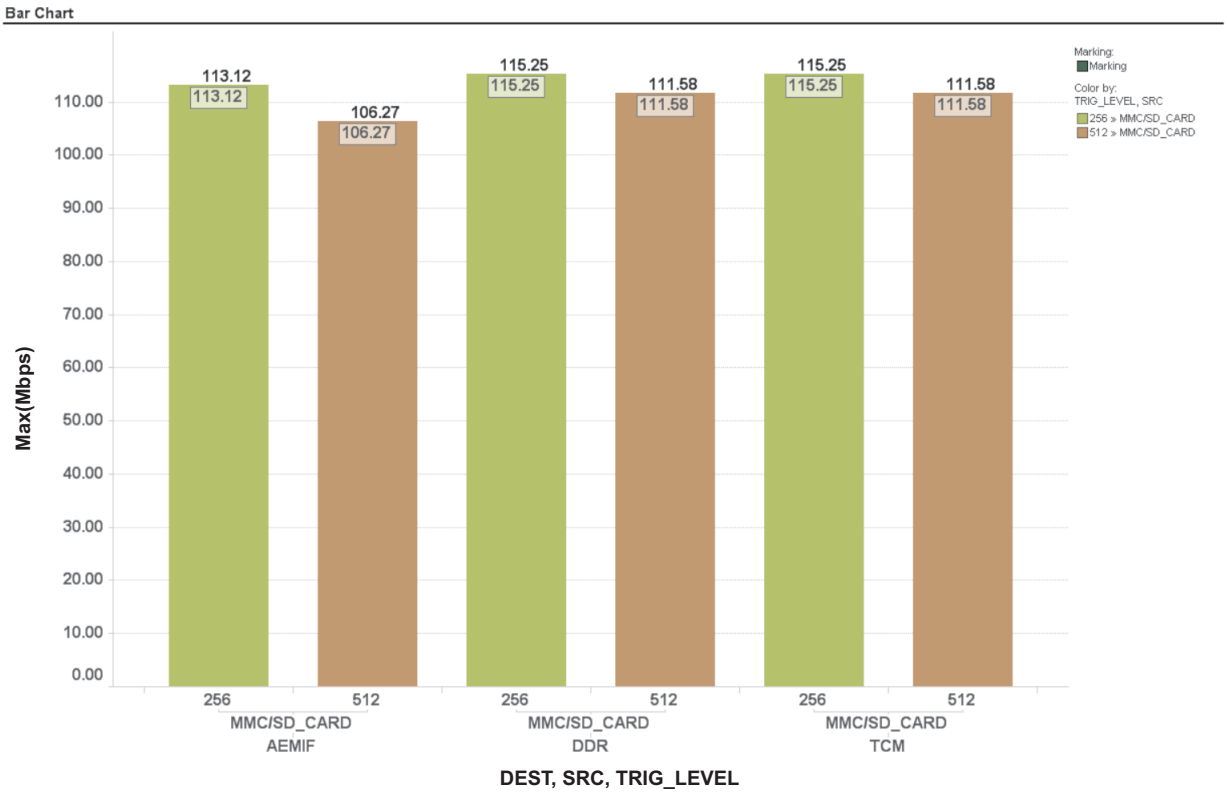



**Figure 38. Throughput for MMC/SD Read With Different FIFO Trigger Levels**

### 4.3.7 Optimization Recommendations

It is not recommended to set SRC/DST buffer in AEMIF memory due to its limited access speed. If space is sufficient, SRC/DST should be set in internal memory. If not, SRC/DST can be set in DDR memory provided that there is a small amount of traffic at the DDR bus. It is also recommended to use 256-bit FIFO trigger level to maximize FIFO utilization. Following the previous recommendations produces a throughput equal to the theoretical number.

## 5 References

- *TMS320DM36x Digital Media System-on-Chip (DMSoC) DDR2/mDDR Memory Controller User's Guide* (SPRUFI2)
- *TMS320DM36x Digital Media System-on-Chip (DMSoC) ARM Subsystem User's Guide* (SPRUFG5)
- *TMS320DM36x Digital Media System-on-Chip (DMSoC) Enhanced Direct Memory Access (EDMA) Controller Reference Guide* (SPRUFI0)
- *TMS320DM36x Digital Media System-on-Chip (DMSoC) Ethernet Media Access Controller (EMAC) User's Guide* (SPRUFI5)
- *TMS320DM36x Digital Media System-on-Chip (DMSoC) Multimedia Card (MMC)/Secure Digital (SD) Card Controller User's* Guide (SPRUFH5)

## IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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

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

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

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