Application Report
SPRA893 – January 2003

OMAP ARM Data Throughput Analysis

Carrie Rhodes — *Technical Staff, DSP Applications*
Chuck Farrow

ABSTRACT

This application report describes techniques for measuring throughput and latency of OMAP™ TIARM925T data accesses through the on-chip traffic controller. Complete examples are given using external memories such as SDRAM and SRAM for reads, writes and back-to-back, read-write sequences.

Contents

List of Figures

List of Tables

1 Introduction

The OMAP architecture devices contain two external memory interfaces, external memory interface fast (EMIFF) and external memory interface slow (EMIFS), for glueless connections to a variety of memories. These external memories and internal SRAM are accessible by the TIARM925T MPU, the C55x DSP, and the system DMA. Access to each memory location is controlled by the traffic controller (TC). Access times can be minimized by using the ARM data cache and carefully designing your system to effectively utilize the tri-bus architecture. This application note provides throughput and latency results for a variety of ARM data accesses which should be used as guidelines when placing data in the memory map. Since the results presented are for ARM accesses only without bus contention, care should be taken when also utilizing the DSP and System DMA.

See Appendix A for a block diagram of the OMAP architecture.

2 System Setup

The following components were used in the course of creating this application report:

- Texas Instruments test board with external Samsung K4S561632C-TL75 16Mx16 SDRAM and 512KB Samsung K6T4016U3C-TB70 SRAM
- Code Composer Studio™ v2.1 for OMAP
- Tektronix TLA700 Logic Analyzer
- POMAP5910CGZG

OMAP5910 ARM and DSP clocks were set up for 150 MHz operation, while the TC was clocked at 75 MHz. The SRAM connected to the internal memory interface (IMIF) runs at the same speed as the TC. The EMIFS, connected to asynchronous SRAM, was clocked at 37.5 MHz, with three wait-states (WS) for reads and two WS for writes, and a write enable length (WELEN) of 2. EMIFF, connected to SDRAM, was clocked at 75 MHz with a CAS latency of 2. All clock domains, peripherals, and System DMA channels unnecessary for performing the throughput analysis were disabled. The clock related settings are summarized in Table 1.

Table 1. Clock Setup

DSP	ARM	TC/IMIF	EMIFF	EMIFS				
150MHz	150MHz	75MHz	75MHz	CAS 2	37.5MHz	WELEN 2	3WS Rd	2WS Wr

A combination of ARM assembly and unoptimized C code was used for performing the throughput analysis. ARM program code was placed in the external SRAM and test setup and results tables were stored in internal SRAM.
Note: The DSP and system DMA were not used for the purposes of this application note.

Iterations of ARM data access tests were all run with the ARM instruction cache enabled. Throughput is defined as the average number of TC cycles it takes for a single access out of 10K performed. For the read-write sequences, a single access is defined as a back-to-back read and a write. The read-write sequence can start with either a read or write. Both cases are studied in this application report. The average cycle measurement was verified using the logic analyzer by observing the specific signals related to the type of memory being tested.

Latency measurements were taken using the logic analyzer and are defined as the number of cycles to perform the first read or write access in a test sequence. To mark the beginning of a test, an ARM GPIO pin was toggled for every 100 read, write, or read-write operations. Other signals used in the measurement varied depending on the memory type and are described below.

Average throughput was measured using an ARM timer running at 6 MHz. The timer was enabled in code before the call to the test sequence. Timer values were recorded at the completion of a test and then scaled to derive the TC cycle count using the following equation:

$$\text{TC Cycles per access} = \left(\frac{\text{Timer value}}{6\ \text{MHz}}\right) * \left(\frac{75\ \text{MHz}}{10000\ \text{accesses}}\right)$$

An example code segment is shown in Figure 1.

```
// Enable ARM timers
      RunTimersOneTwoThree();
// Run the test
      Data100Write_32(TempDataP,TempIter);
// Disable ARM timers
     StopTimersOneTwoThree();
_Data100Write_32:
      MOV   R2, R0            ; Save address

                              ; Do 100 Writes
Loopwrite32:
                              ; Set armio3 high, use this for logic analyzer
trigger
      LDR  R12, armio_reg     ;  and latency measurement
      LDRH R12, [R12]
      ORR  R10, R12, #0x8
      LDR  R12, armio_reg
      STRH R10, [R12]
      STRT   R4, [R0], #4     ; Write one word from R4; R0++
                            .
                            .
                            . Repeat instruction to complete 100 writes
                            .
                            .
      STRT   R4, [R0], #4     ; Write one word from R4; R0++
     LDR  R12, armio_reg      ; Set armio3 low
     LDRH R12, [R12]
     BIC  R10, R12, #0x8
     LDR  R12, armio_reg
     STRH R10, [R12]
     MOV   R0, R2             ; Restore address
     SUB   R1, R1, #0x1
     CMP   R1, #0x0
     BNE   Loopwrite32
     BX    R14                ; Return with Branch Link
```

Figure 1. Code Segment for Logic Analyzer Trigger and Write Test

2.1 Details on Latency Measurement using the Logic Analyzer

Latency measurements were recorded for EMIFF and EMIFS accesses only. For EMIFF measurements, signals SDRAM.RAS and SDRAM.CAS were used along with a GPIO to determine the cycle count for the first read or write sequence in a test. The latency for EMIFF was defined as the number of TC cycles from when the GPIO was driven high until two cycles after the SDRAM.CAS signal was driven low (CAS latency = 2). Latency for EMIFS accesses to external SRAM was measured as the number of cycles from when the GPIO was driven high until the FLASH.CS signal transitioned. See Figures 8 and 9 for more details. This technique was used for all three types of accesses: read, write, and read-write sequences.

2.1.1 Details on Throughput Measurement using the Logic Analyzer

Average throughput measurements taken using an ARM timer were verified using the logic analyzer. Throughput of SDRAM memory on the EMIFF was measured using the SDRAM.CAS and SDRAM.WE signals. TC cycles were then computed by multiplying the access time by the TC clock (75 MHz). Throughput for EMIFS accesses were measured as the period of the FLASH.CS signal and converted to TC cycles.

2.1.1.1 Peripheral Registers Setup

TC was configured as follows:

- EMIFS_PRIO, EMIFF_PRIO, and IMIF_PRIO settings
  - 31:16 reserved
  - 15:12 LB host set to 000b for single transfers, not used in this analysis
  - 11:8 System DMA set to 0000b
  - 7 reserved
  - 6:4 DSP (CPU or DMA) set to 000b for single transfers, not used in this analysis
  - 3 reserved
  - 2:0 ARM set to 000b for single transfers, ARM EMIFS code fetch (EMIFS_PRIO) affected the transfers in this analysis
- EMIFS_CONFIG_REG = 0x00000012
  - Boot mode was set for CS3 space at 0x00000000
- EMIFS_CS3_CONFIG = 0x00002239
  - WELEN=2, WRWST=2, RDWST=3, FLCLKDIV=TC/2
- EMIFF_SDRAM_CONFIG = 0x101D4F4
  - SDF1 frequency range, auto-refresh enabled and calculated from SDRAM datasheet for minimum, 16-bit bus, 256M bits, 4 banks
- EMIFF_MRS = 0x00000027
  - CAS latency 2, full-page burst length
- TIMEOUT1 = TIMEOUT2 = TIMEOUT3 = 0x00000000
- ENDIANISM = 0x00000000
  - Little-endian

3 Throughput Analysis Examples

Throughput examples were generated for ARM data accesses to memories on the EMIFF, EMIFS, and IMIF busses. Each test case was a block transfer of 10K accesses and the average was computed to determine the TC cycle counts reported in the following tables. The logic analyzer was used to verify these numbers as well as determine the latency of the first access. Table 2 summarizes the types of accesses and the ARM memory and cache (I$ – instruction cache, D$ – data cache) settings. The two different data cache modes were used and results in the following tables highlight their benefits. For the write-through (WT) mode, a write-cache hit always generates a memory access where the copy back (CB) mode does not. Several cycles can be saved, by using the CB mode. However, it requires that the cache be cleaned in order to update memory.

Table 2. Access Type, Cache Selections, ARM Addresses

ARM Data Access		I$ ON, D$ OFF	I$ ON, D$ ON as WT	I$ ON, D$ ON as CB
Memory				
	EMIFS	Addr:0x00060000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite	Addr:0x00060000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite	Addr:0x00060000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite
	EMIFF	Addr:0x10600000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite	Addr:0x10600000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite	Addr:0x10600000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite
	IMIF	Addr:0x20010000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite	Addr:0x20010000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite	Addr:0x20010000 Read, Write, ReadWrite Read, Write, ReadWrite Read, Write, ReadWrite

Throughput data was derived for 8,16, and 32-bit data read and write accesses. The read-write test is further subdivided to show data for the case where the access begins with a read or a write. In the following sections, examples are given for the various accesses identified in Table 2.

3.1 ARM Data Reads for EMIFS, EMIFF and IMIF Memories

ARM data reads for EMIFS, EMIFF and IMIF memories with various data cache (Dcache) modes are shown in Table 3, Table 4, and Table 5. Instruction cache is always enabled.

Table 3. EMIFS Reads: Latency, Throughput

EMIFS Addr:0x00060000		**Latency (TC cycles)**	**Average Throughput (TC cycles)**		
Access Size	**Access Type**	**First Read**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	Read	17.0	12.1	0.5	0.5
16-bit	Read	17.0	12.0	0.5	0.5
32-bit	Read	23.1	18.0	0.5	0.5

The numbers shown indicate that 8 and 16-bit reads take the same amount of time but 32-bit reads take longer when Dcache is disabled. The extra cycle time is encountered because the EMIFS data bus is only 16-bits wide. When Dcache is enabled, the 32-bit access penalty is eliminated and the ARM can access one data value per cycle (the TC is running at 75 MHz, half the ARM speed).

Table 4. EMIFF Reads: Latency, Throughput

EMIFF Addr:0x10600000		Latency (TC cycles)	Average Throughput (TC cycles)		
Access Size	**Access Type**	**First Read**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	Read	11.0	10.2	0.5	0.5
16-bit	Read	11.0	10.2	0.5	0.5
32-bit	Read	12.0	9.9	0.5	0.5

The same benefit as observed for the EMIFS accesses is shown here when Dcache is enabled (single cycle ARM data accesses). It can also be seen that accesses with Dcache disabled are quicker here as compared to EMIFS accesses.

Table 5. IMIF Reads: Latency, Throughput

IMIF Addr:0x20010000		Latency (TC cycles)	Average Throughput (TC cycles)		
Access Size	**Access Type**	**First Read**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	Read	Internal	4.0	0.5	0.5
16-bit	Read	Internal	4.0	0.5	0.5
32-bit	Read	Internal	4.0	0.5	0.5

When reading from internal memory, there is a 6–14 cycle savings. Some of the benefit comes from the 32-bit data bus to the internal SRAM. Additional cycles are saved by avoiding off-chip accesses. Again, in order to get single cycle ARM accesses, the Dcache must be enabled.

3.2 ARM Data Writes for EMIFS, EMIFF and IMIF Memories

ARM data writes for EMIFS, EMIFF and IMIF memories with various Dcache modes are shown in Table 6, Table 7, and Table 8. Instruction cache is always enabled.

Table 6. EMIFS Writes: Latency, Throughput

EMIFS Addr:0x00060000		Latency (TC cycles)	Average Throughput (TC cycles)		
Access Size	**Access Type**	**First Write**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	Write	14.0	14.0	14.0	0.5
16-bit	Write	14.0	14.0	14.0	0.5
32-bit	Write	23.1	23.0	23.0	0.5

EMIFS writes with Dcache enabled in WT mode have no advantage over Dcache disabled, because the data is written out to external SRAM memory, even on cache hits. When using the CB mode, the same single cycle ARM accesses are attainable.

Table 7. EMIFF Writes: Latency, Throughput

EMIFF Addr:0x10600000		Latency (TC cycles)	Average Throughput (TC cycles)		
Access Size	**Access Type**	**First Write**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	Write	8.9	6.2	6.2	0.5
16-bit	Write	9.0	6.2	6.2	0.5
32-bit	Write	10.0	4.2	4.2	0.5

One interesting observation is that 32-bit ARM Data writes are quicker than 8 and 16-bit writes for Dcache off and WT modes.

Table 8. IMIF Writes: Latency, Throughput

IMIF Addr:0x20010000		Latency (TC cycles)	Average Throughput (TC cycles)		
Access Size	**Access Type**	**First Write**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	Write	Internal	3.0	3.0	0.5
16-bit	Write	Internal	3.0	3.0	0.5
32-bit	Write	Internal	3.0	3.0	0.5

Writes to internal memory are two ARM cycles faster than reads. Again, Dcache in WT mode has no benefit as the data is written out to memory during each access.

3.3 ARM Data Write Read Sequence for EMIFS, EMIFF and IMIF Memories

ARM data read write sequences for EMIFS, EMIFF and IMIF memories with various Dcache modes are shown in Table 9, Table 10, and Table 11. The Icache is always enabled. As defined above, the read write test sequence is10k alternating reads and writes from/to the specified memory. The test sequence determines the latency of the first access and the average throughput of a read and write access. Each case, write first and read first is tested and is reflected in the latency column. Table 9, Table 10, and Table 11 are a summary of the results.

Table 9. EMIFS WriteRead Sequence: Latency, Throughput

EMIFS Addr:0x00060000		Latency (TC cycles)		Average Throughput for One Read Write Sequence (TC cycles)		
Access Size	**Access Type**	**First Write**	**First Read**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	read write seq. (write first)	14.0		27.0	14.0	2.1
8-bit	read write seq. (read first)		18.0	25.8	14.0	2.1
16-bit	read write seq. (write first)	14.0		27.0	14.0	2.1
16-bit	read write seq. (read first)		18.0	25.8	14.0	2.1
32-bit	read write seq. (write first)	23.1		42.0	23.0	2.1
32-bit	read write seq. (read first)		24.0	41.1	23.0	2.1

When the sequence started with a write:

- 8- and 16-bit analysis: The first write took 14.0 TC cycles, the first read took 18.0 TC cycles, and the second write took 9.0 TC cycles, yielding an average sequence time of 27.0 TC cycles.
- 32-bit analysis: The first write took 23.1 TC cycles, the first read took 23.9 TC cycles, and the second write took 18 TC cycles, yielding a sequence time of 41.9 TC cycles.

When the sequence started with a read:

- 8 and 16-bit analysis: The first read took 18.0 TC cycles, the first write took 9.0 TC cycles, and the second read took 18.0 TC cycles, yielding sequence time of 27.0 TC cycles.
- 32-bit analysis: The first read took 24.0 TC cycles, the first write took 18.0 TC cycles, and the second read took 24.0 TC cycles, yielding a sequence time of 42.0 TC cycles.

Table 10. EMIFF WriteRead Sequence (Write First): Latency, Throughput

EMIFF Addr:0x10600000		Latency (TC cycles)		Average Throughput for One Read Write Sequence (TC cycles)		
Access Size	**Access Type**	**First Write**	**First Read**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	read write seq. (write first)	9.0		17.1	6.2	2.1
8-bit	read write seq. (read first)		11.0	17.1	6.2	2.1
16-bit	read write seq. (write first)	9.0		17.1	6.2	2.1
16-bit	read write seq. (read first)		11.0	17.1	6.2	2.1
32-bit	read write seq. (write first)	10.1		14.7	4.1	2.1
32-bit	read write seq. (read first)		12	14.7	4.1	2.1

When the sequence started with a write:

- 8 and 16-bit analysis: The first write took 9.0 TC cycles, the first read took 9.0 TC cycles, and the second write took 9.0 TC cycles yielding a sequence time of 18.0 TC cycles, which is comparable to the average.
- 32-bit analysis: The first write took 10.1 TC cycles, the first read took 6 TC cycles, and the second write took 9.0 TC cycles.

When the sequence started with a read:

- 8 and 16-bit analysis: The first read took 11.0 TC cycles, the first write took 9.0 TC cycles, and the second read took 8.0 TC cycles, yielding a sequence time of 17.0 TC cycles.
- 32-bit analysis: The first read took 12.0 TC cycles, the first write took 9.0 TC cycles, and the second read took 6.0 TC cycles, yielding a sequence time of 15.0 TC cycles.

Table 11. IMIF WriteRead Sequence (Write First): Latency, Throughput

IMIF Addr:0x10600000		Latency (TC cycles)		Average Throughput for One Read Write Sequence (TC cycles)		
Access Size	**Access Type**	**First Write**	**First Read**	**Dcache OFF**	**Dcache ON (WT)**	**Dcache ON (CB)**
8-bit	WriteRead	Internal	Internal	8.1	3.0	2.1
16-bit	WriteRead	Internal	Internal	8.1	3.0	2.1
32-bit	WriteRead	Internal	Internal	8.1	3.0	2.1

When Dcache was enabled the average ARM access time dropped from 8.1 to 3.0 and 2.1 TC cycles for WT and CB modes, respectively. TC latency could not be measured since the bus is internal to the OMAP core.

4 Logic Analyzer Screen Shots

This section contains a variety of screen shots taken off the Tektronix 704 Logic Analyzer. The following results and additional traces not shown were used to determine latencies and verify the average throughput for each test. The beginning of the latency period is denoted as the rising edge of armio3 which was asserted immediately before the first read or write access. For 8- and 16-bit read accesses to SDRAM through the EMIFF, the end of the latency measurement was taken as two TC cycles (CAS latency = 2) after the SDRAM.CAS signal toggled from low to high. The 32-bit reads require an additional TC cycle for the second 16-bit word. EMIFS latencies were complete when the FLASH.CS signal was driven low (see Figure 2). Refer to the markers labeled 1 and 2 as well as the reported “Delta Time” for actual latencies.

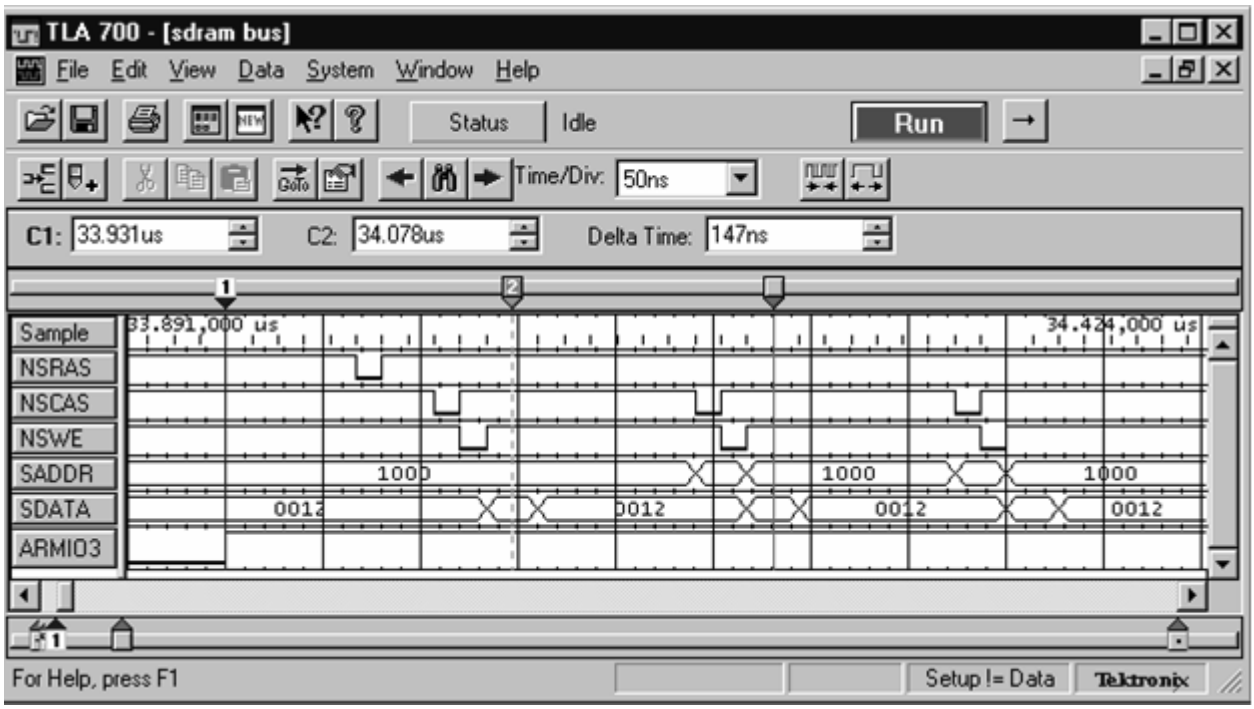

Figure 2. 16-Bit ARM Data Read on the EMIFF Bus

The latency for the first 16-bit data read in a series of 10K reads is 147 ns or 11.0 TC cycles. A second unlabeled marker in the screen shot above indicates the completion of the second 16-bit read. The time elapsed from marker 2 to this unlabeled marker was measured as 134 ns, which translates to 10.1 TC cycles per 16-bit read. The average throughput measured using the 6 MHz ARM timer was 10.2. Similar results were observed for 8-bit reads.

Figure 3 indicates that the latency for the first 32-bit read cycle is 160 ns or 12 TC cycles. The second read is 133 ns or 10.0 TC cycles, which is similar to the 16-bit case.

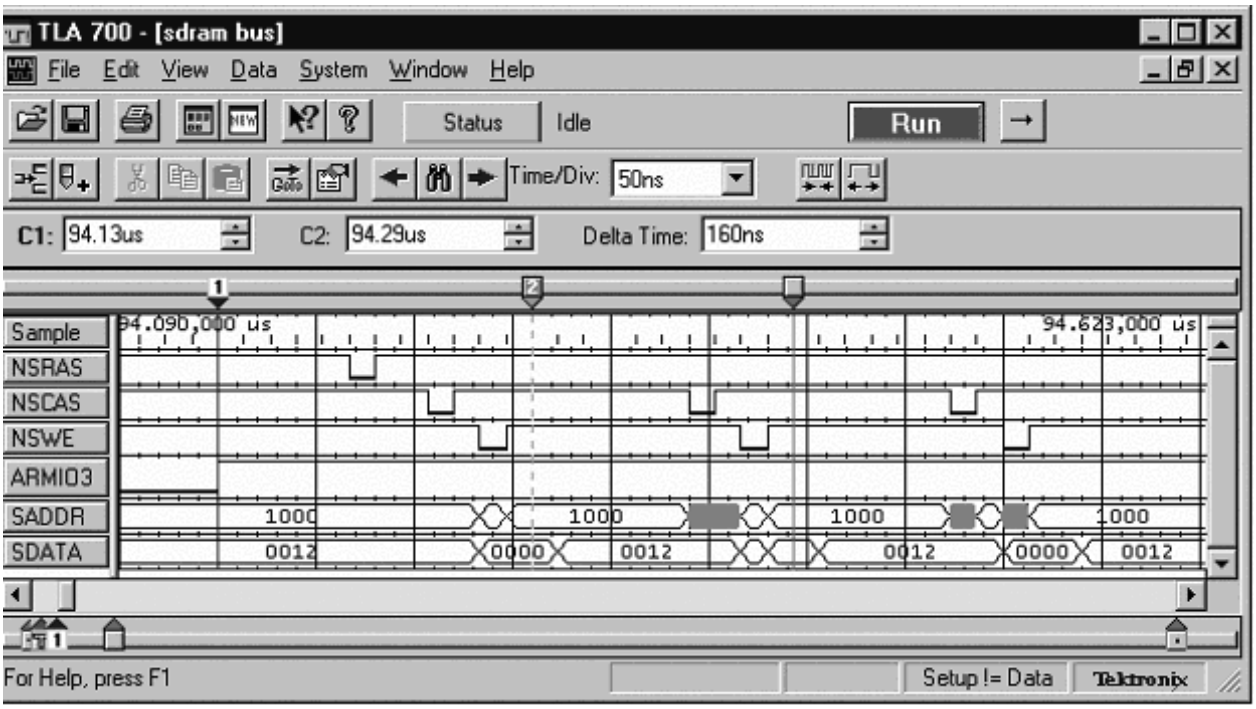

Figure 3. 32-Bit ARM Data Read on the EMIFF Bus

The difference between marker 1 and 2 is the measured latency for an 8-bit data write on the EMIFF bus (see Figure 4). The observed value of 119 ns is equivalent to 8.9 TC cycles. Marker 2 and the unlabeled marker denote the time required for the second 8-bit write in the 10K write sequence. According to the logic analyzer, a single 8-bit write took 7.1 TC cycles which is a approximately one more cycle than the average throughput reported in Table 7.

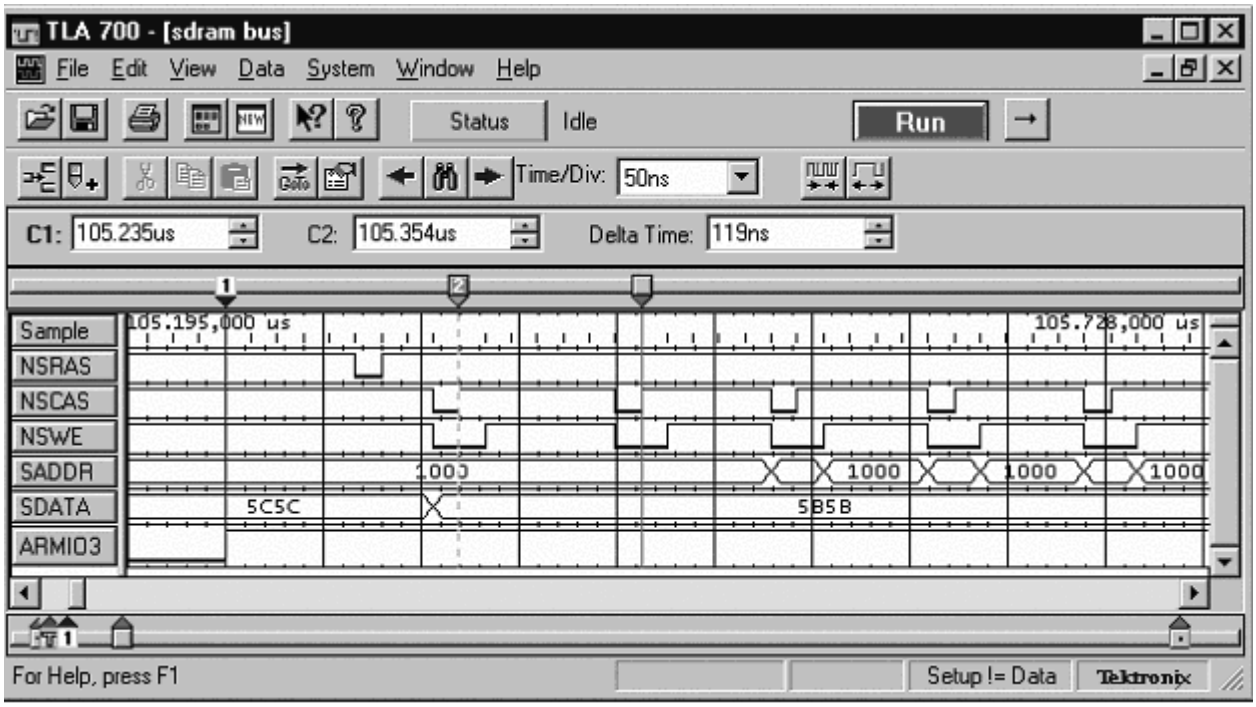

Figure 4. 8-Bit ARM Data Write on the EMIFF Bus

The latency for the first write in a sequence of 10K 32-bit data writes to SDRAM is 133 ns or 0.0 TC cycles (see Figure 5). Adjacent 32-bit writes denoted by the second falling edge of the NSWE signal require 4.1 TC cycles, which is approximately the same cycle time as reported in Table 7. This result indicates that 32-bit writes through the EMIFF are actually faster than 8- or 16-bit data writes through the same bus.

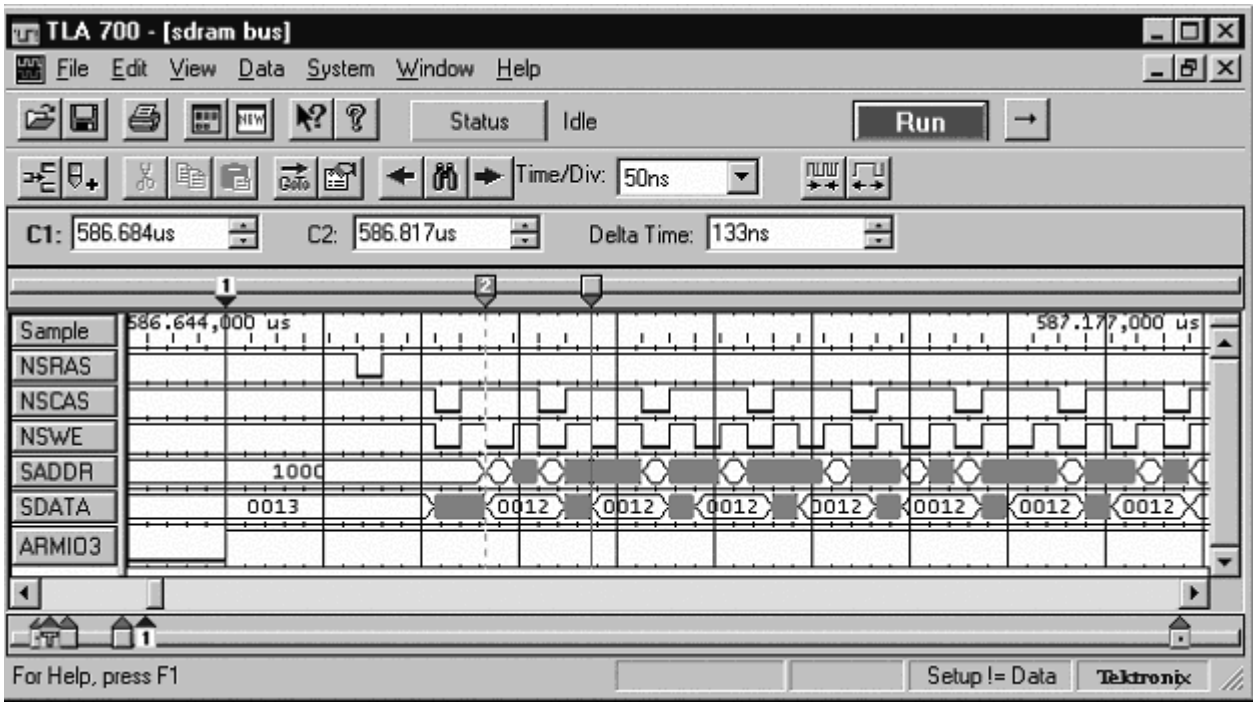

Figure 5. 32-Bit ARM Data Write on the EMIFF Bus

Figure 6 shows a 32-bit read write sequence which starts with a write. Markers 1 and 2 indicate that the first write takes 134 ns or 10.1 TC cycles. The second access is a read which takes 6 TC cycles to complete and the following write access takes 120 ns or 9 TC cycles.

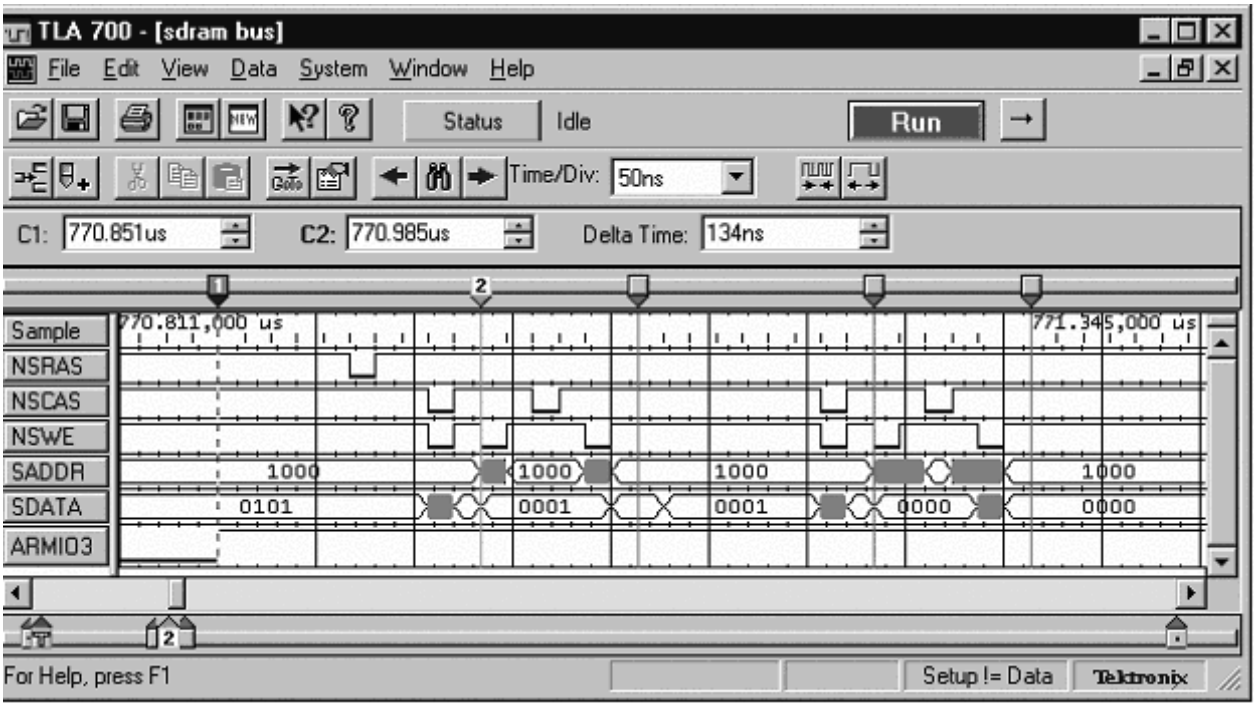

Figure 6. 32-Bit ARM Data Read Write Sequence (Write First) on the EMIFF Bus

Figure 7 shows the EMIFF bus activity for a 32-bit read-write sequence when a read is the first access. Markers 1 and 2 frame the first read in the test sequence. The delta between these markers indicates that the first read took 160 ns or 12 TC cycles. The adjacent write took 120 ns or 9 TC cycles. The next read access can be observed to have taken 80 ns or 6 TC cycles.

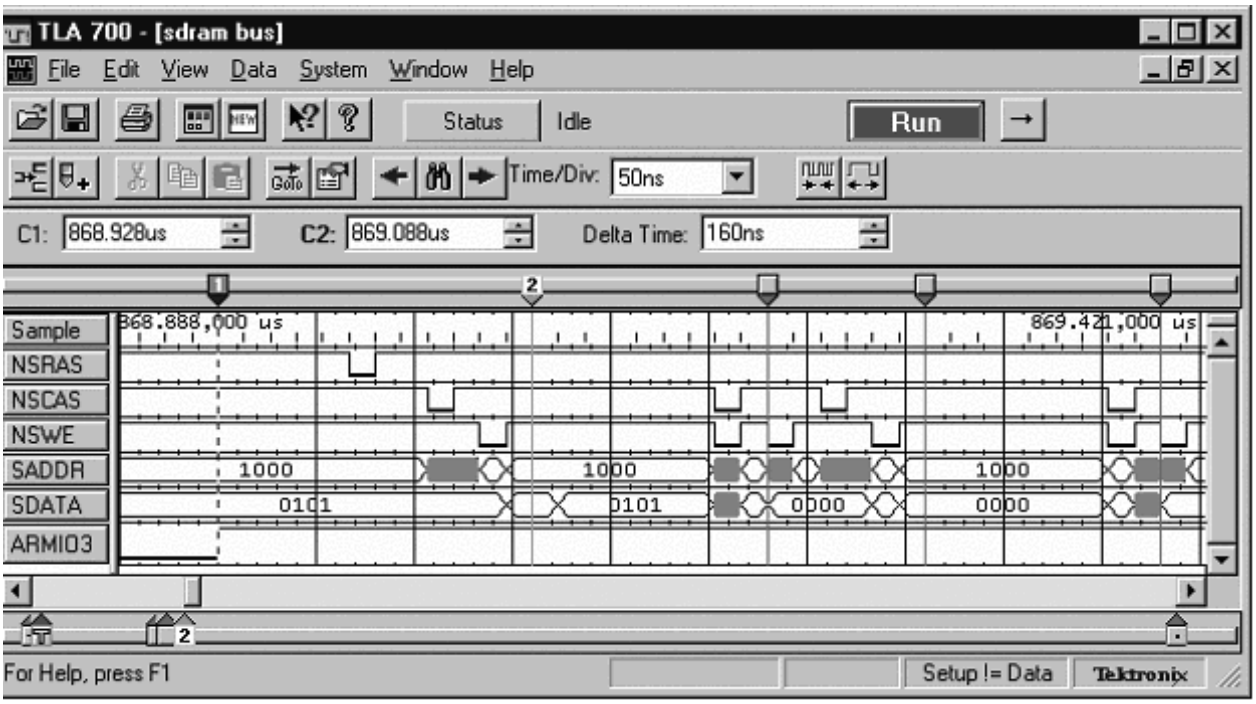

Figure 7. 32-Bit ARM Data Read Write Sequence (Read First) on the EMIFF Bus

Figure 8 provides an example of the timing sequence on the EMIFS bus when accessing data from external SRAM. The latency of the first read in this sequence, shown by Markers 1 and 2, is 226 ns or 17 TC cycles. The second read in the sequence completes at the falling edge of NFCS_3 which is 12 TC cycles after the previous falling edge.

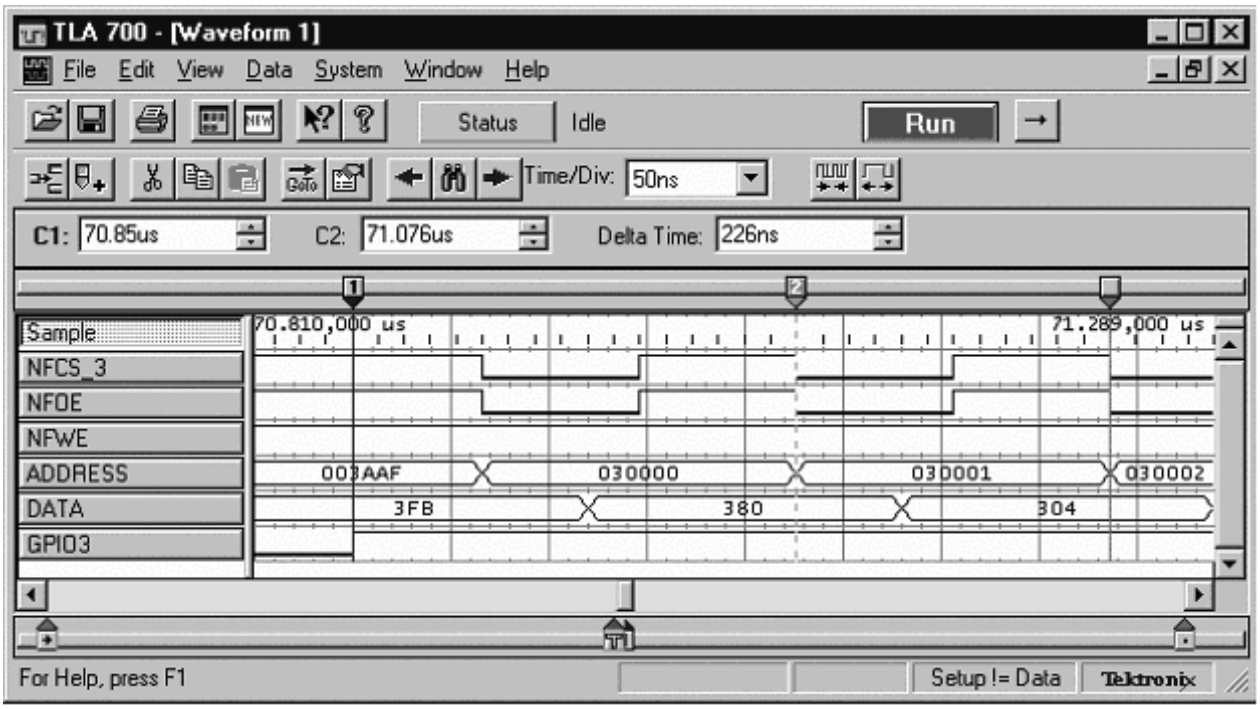

Figure 8. 16-Bit ARM Data Read Sequence on the EMIFS Bus

A 32-bit write sequence to SRAM is shown in Figure 9. The write sequence begins when the chip select becomes active which is at the falling edge of NFCS_3. The completion of the write occurs 1 TC cycle after the rising edge of the write enable, NFWE. The latency for this access is 308 ns or 23.1 TC cycles.

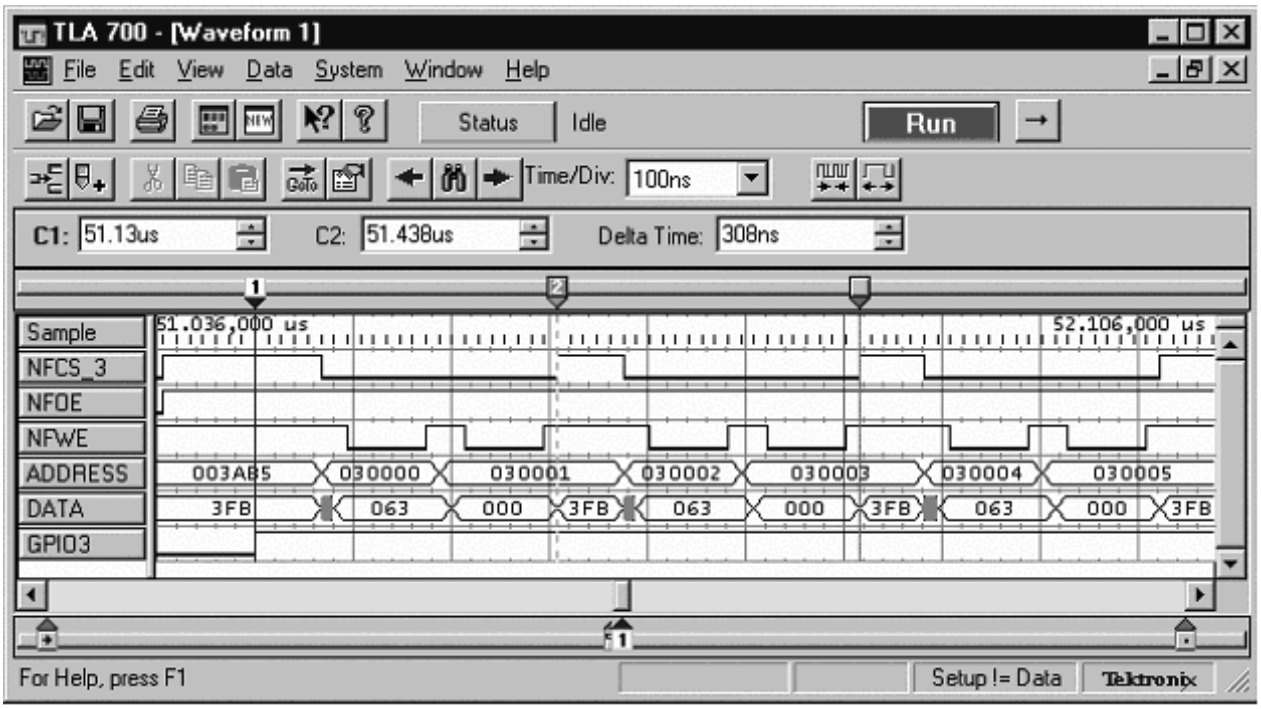

Figure 9. 32-Bit ARM Data Write Sequence on the EMIFS Bus

5 Conclusion

The OMAP architecture devices offer flexibility in your design by providing several locations for placing program and data memory. The two external busses, EMIFF and EMIFS, allow you to take advantage of both SDRAM and FLASH memories. Additional internal SRAM provides a third location for placing program and data memory and requires approximately 2–7 times fewer TC cycles compared to external memory. Further cycle savings can be achieved by using Dcache in both CB and WT modes. It was also observed that when accessing SDRAM on the EMIFF bus, an additional 40 ns was needed to access a new row in the memory . All test cases for EMIFF show the condition for when a new row selection was required, thereby, giving a worst case scenario.

Examples presented in this application note demonstrate a method for benchmarking ARM data accesses on the OMAP architecture devices. These examples should be used as guidelines for benchmarking your own system to determine where to place data in the memory map. In addition, the examples do not include simultaneous ARM, system DMA, and DSP accesses which could introduce contention on each bus. Refer to the priority schemes for the traffic controller to determine extra delays that may be encountered when contention occurs.

6 References

1. *OMAP5910 Dual-Core Processor Data Manual* (SPRS197)
2. *OMAP5910 Dual-Core Processor Technical Reference Manual* (SPRU602)
3. Samsung Electronics *http://www.samsungelectronics.com*

Appendix A OMAP Block Diagram

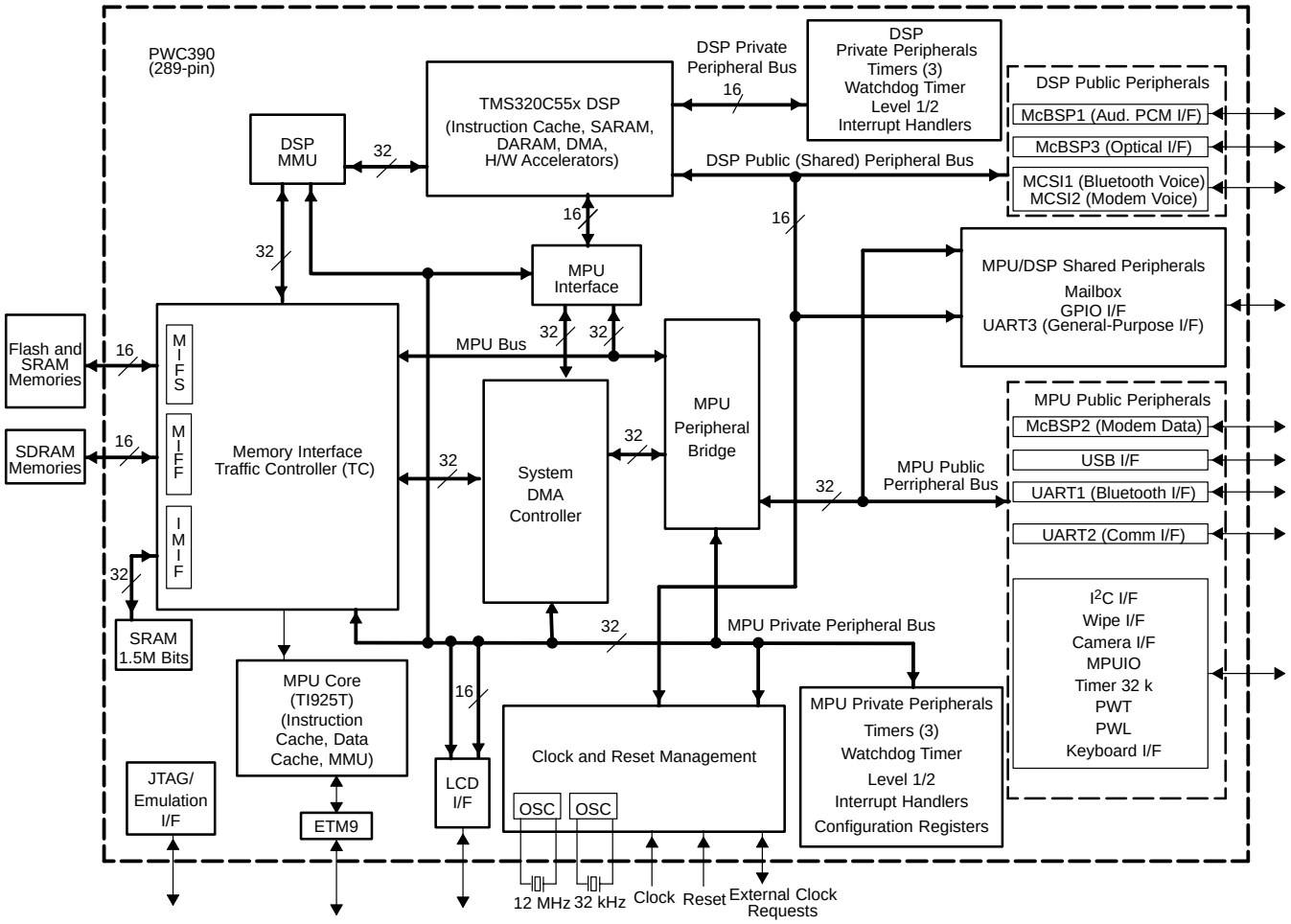

Figure A–1. OMAP Block Diagram

Appendix B MPU Memory Map

Table B–1. MPU Memory Map

Address Range	On-Chip	External Interface
0x0000 0000 0x01FF FFFF		FLASH CS0 32 Mbytes
0x0200 0000 0x03FF FFFF	reserved	
0x0400 0000 0x05FF FFFF		FLASH CS1 32 Mbytes
0x0600 0000 0x07FF FFFF	reserved	
0x0800 0000 0x09FF FFFF		FLASH CS2 32 Mbytes
0x0A00 0000 0x0BFF FFFF	reserved	
0x0C00 0000 0x0DFF FFFF		FLASH CS3 32 Mbytes
0x0E00 0000 0x0FFF FFFF	reserved	
0x1000 0000 0x13FF FFFF		SDRAM 64Mbytes
0x1400 0000 0x1FFF FFFF	reserved	
0x2000 0000 0x2002 FFFF	Internal SRAM 192Kbytes	
0x2003 0000 0x2FFF FFFF	reserved	
0x3000 0000 0x7FFF FFFF		Local bus space for USB host
0x8000 0000 0xDFFF FFFF	reserved	
0xE000 0000 0xE0FF FFFF	DSP public memory space (accessible by MPUI) 16Mbytes	
0xE100 0000 0xEFFF FFFF	DSP public peripherals (accessible by MPUI)	
0xFFFB 0000 0xFFFC FFFF	MPU public, MPU/DSP shared peripherals	
0xFFFD 0000 0xFFFE FFFF	MPU private peripherals	
0xFFFF 0000 0xFFFF FFFF	reserved	

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.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265