Freescale Semiconductor
Application Note

Document Number: AN3060
Rev. 0, 01/2006

MSC711x Optimization Techniques

by *Barbara Johnson*
Digital Systems Division
Freescale Semiconductor, Inc.
Austin, TX

This application note discusses methods to optimize the performance of an MSC711x application. It provides guidelines for grouping memory, allocating memory for program and data, and configuring the DDR controller, DMA controller, and crossbar switch for highest system performance. System designers can implement these techniques to tune application performance. This document assumes a working knowledge of the MSC711x DSP. For details on MSC711x, consult the device reference manual.

Contents

1 Reducing M1 Memory Contentions

The 256 KB M1 memory is organized into four 64 KB groups, each accessible to the core through the program (P) bus, data (X_a and X_b) buses. Each group is also accessible for DMA and Ethernet MAC transfers through the ASM1 bus.

Figure 1 shows M1 memory organization. Each group contains eight 8 Kbyte modules, each organized as an array of 2048 32-bit rows. The eight modules are interleaved so that the next row of an address is always at the next module. The interleaving increases the probability of performing two simultaneous data accesses to a memory group. The interleaving is shown in the following examples:

- Addresses 0x0000–0x0003 are in row 0 of module 0.
- Addresses 0x0004–0x0007 are in row 0 of module 1.
- Addresses 0x0008–0x000B are in row 0 of module 2.
- Addresses 0x000C–0x000F are in row 0 of module 3.

Contention occurs when a memory group cannot service all requested simultaneous accesses to M1 memory. To resolve the contention, the core inserts one or more stall cycles. For accesses to different modules, there is no contention and the core does not stall.

	Group 0 64 KB	Group 1 64 KB	Group 2 64 KB	Group 3 64 KB	
Module 7 8 KB	0x0FFFC–0x0FFFF ... 0x0001C-0x0001F	0x1FFFC–0x1FFFF ... 0x1001C–0x1001F	0x2FFFC–0x2FFFF ... 0x2001C–0x2001F	0x3FFFC–0x3FFFF ... 0x3001C–0x3001F	Row 2047 .... Row 0
Module 6 8 KB	0x0FFF8–0x0FFFB ... 0x00018-0x0001B	0x1FFF8–0x1FFFB ... 0x10018–0x1001B	0x2FFF8–0x2FFFB ... 0x20018–0x2001B	0x3FFF8–0x3FFFB ... 0x30018–0x3001B	Row 2047 .... Row 0
Module 5 8 KB	0x0FFF4–0x0FFF7 ... 0x00014-0x00017	0x1FFF4–0x1FFF7 ... 0x10014–0x10017	0x2FFF4–0x2FFF7 ... 0x20014–0x20017	0x3FFF4–0x3FFF7 ... 0x30014–0x30017	Row 2047 .... Row 0
Module 4 8 KB	0x0FFF0–0x0FFF3 ... 0x00010-0x00013	0x1FFF0–0x1FFF3 ... 0x10010–0x10013	0x2FFF0–0x2FFF3 ... 0x20010–0x20013	0x3FFF0–0x3FFF3 ... 0x30010–0x30013	Row 2047 .... Row 0
Module 3 8 KB	0x0FFEC–0x0FFEF ... 0x0000C-0x0000F	0x1FFEC–0x1FFEF ... 0x1000C–0x1000F	0x2FFEC–0x2FFEF ... 0x2000C–0x2000F	0x3FFEC–0x3FFEF ... 0x3000C–0x3000F	Row 2047 .... Row 0
Module 2 8 KB	0x0FFE8–0x0FFEB ... 0x00008-0x0000B	0x1FFE8–0x1FFEB ... 0x10008–0x1000B	0x2FFE8–0x2FFEB ... 0x20008–0x2000B	0x3FFE8–0x3FFEB ... 0x00008–0x3000B	Row 2047 .... Row 0
Module 1 8 KB	0x0FFE4–0x0FFE7 ... 0x00004-0x00007	0x1FFE4–0x1FFE7 ... 0x10004–0x10007	0x2FFE4–0x2FFE7 ... 0x20004–0x20007	0x3FFE4–0x3FFE7 ... 0x30004–0x30007	Row 2047 .... Row 0
Module 0 8 KB	0x0FFE0–0x0FFE3 ... 0x00000–0x00003	0x1FFE0–0x1FFE3 ... 0x10000–0x10003	0x2FFE0–0x2FFE3 ... 0x20000–0x20003	0x3FFE0–0x3FFE3 ... 0x30000–0x30003	Row 2047 .... Row 0

Figure 1. M1 Memory Organization

Figure 2 shows how the M1 memory addressing relates to the memory group. Bits 17–16 of an M1 memory address specify the group selected. Bits 4–2 of an M1 memory address specify the module selected, as follows:

- Addresses 0x00000–0x0FFFF are in group 0.
- Addresses 0x10000–0x1FFFF are in group 1.
- Addresses 0x20000–0x2FFFF are in group 2.
- Addresses 0x30000 - 0x3FFFF are in group 3.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17	16	15 14 13 12 11 10 9 8 7 6 5	4 3 2	1 0
Upper Bits	Group	Row	Module	Offset

Figure 2. M1 Addressing

Following are recommended practices to reduce M1 memory contentions:

- Place the data accessed by the core on the X_a and X_b data buses into a different memory group from the data buffers accessed by the DMA controller or Ethernet MAC.
- Keep any program code in M1 memory together in its own memory group.
- Place data to be accessed in parallel into different memory groups where possible. If parallel data cannot be placed in different groups, place data buffers at an offset from each other so different modules are accessed. A suitable offset, for example, would be N × 32 + 16 bytes, where N is an integer less than 1024.

2 Allocating Memory for Program and Data

M1 memory is optimal for both high-speed program and data storage. It is multi-ported and it allows parallel accesses from the core and the crossbar switch. The core can access M1 memory through the P bus for program fetches and through the X_a and X_b buses for reading/writing data. Also, the DMA controller and Ethernet MAC can access M1 memory through the crossbar switch ASM1 bus. The core can access M1 memory at zero wait states and operates at the core frequency. The core can read or write a single 32-bit data in one core clock cycle. **Figure 3** shows the M1 code and data flow diagram.

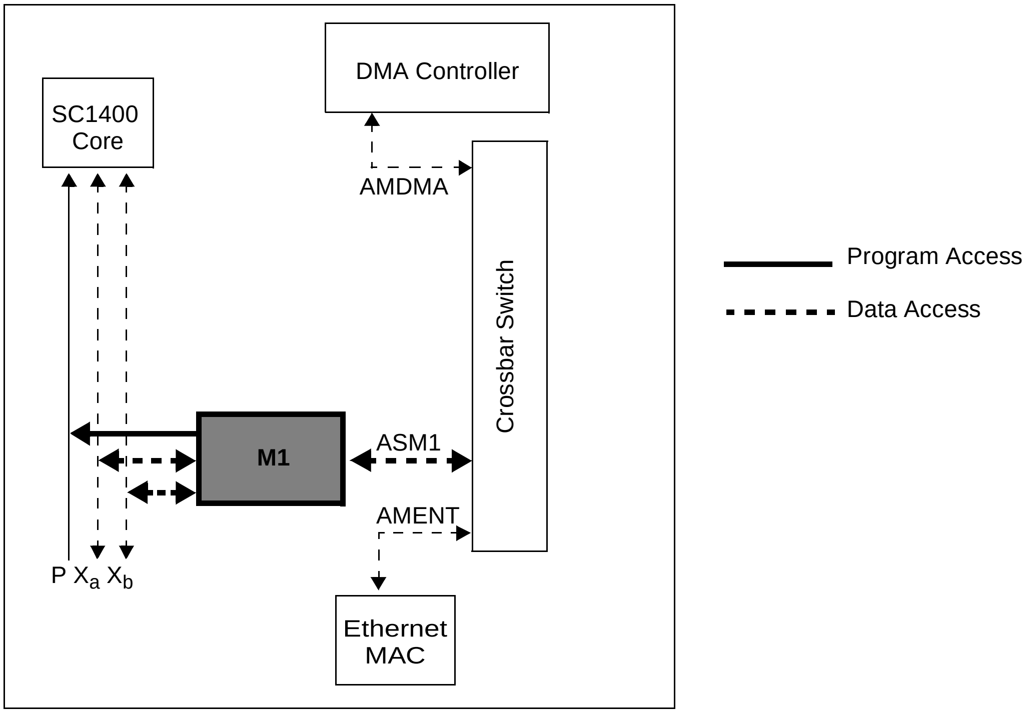

Figure 3. M1 Code and Data Flow

M2 memory is a secondary memory that is also optimal for program storage because the instruction cache can be used to boost overall performance. The instruction fetch unit (IFU) handles all program accesses to external address space, including memory locations beyond M1 memory. The IFU also conducts all instruction cache update activity. The IFU connects to the crossbar switch via the AMIC bus. The instruction cache allows code to be placed into M2 memory and external DDR memory, both of which have higher access latencies than M1 memory yet achieve high performance. The prefetch mechanism greatly improves cache performance in a system running a program with sequential code because in the next access to the code area, the code is probably already in the cache.

M2 memory can also be used for data storage, but it is not as optimal for data storage as M1 memory. The extended core interface (ECI) handles the switching between the core buses and the AMEC bus connected to the crossbar switch. Data can be written to and read from M2 memory via the AMEC bus at half the core frequency. Because stalls can occur when the SC1400 core writes to locations outside M1 memory, the write buffer in the ECI improves performance by releasing the core and allowing it to continue processing while it completes the write operation when the destination becomes available. An immediate write access to M2 memory that bypasses the write buffer requires eight cycles because the core stalls until the write transfer completes. However, if the write buffer is enabled, the access time is reduced to one cycle. The core can read a single 32-bit unit of data from M2 memory in nine core clock cycles. The M2 memory code and data flow diagram is shown in **Figure 4**.

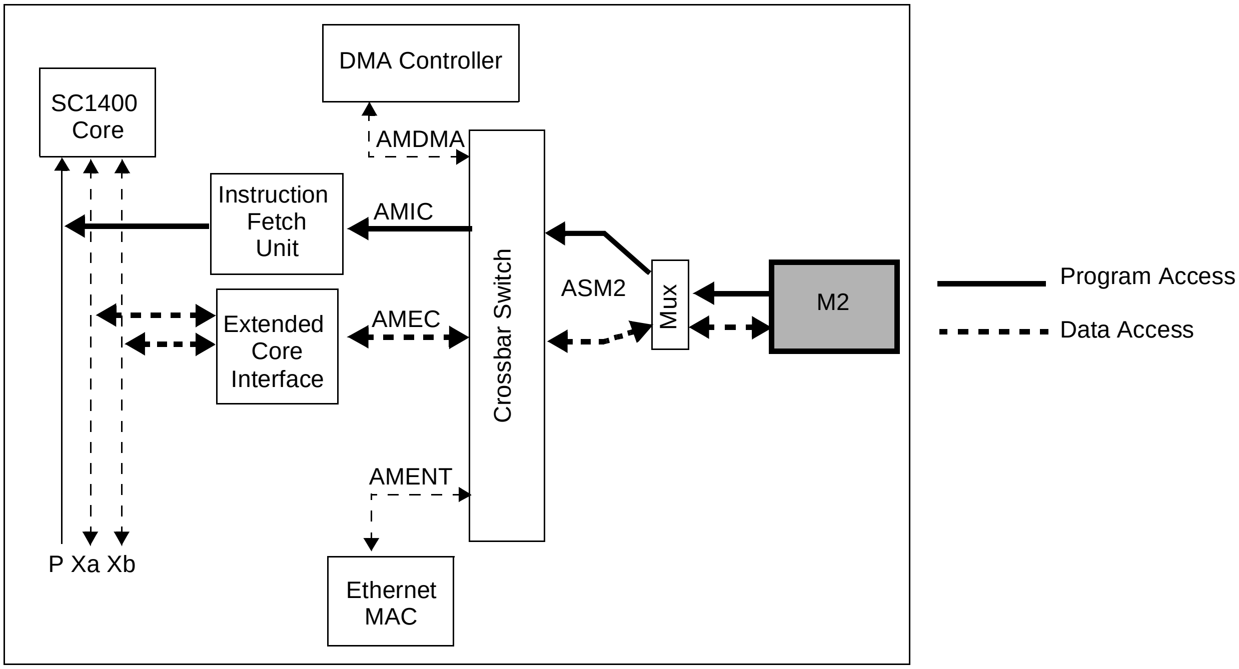

Figure 4. M2 Code and Data Flow

The external DDR memory can also be used for program storage because program code can be brought into the instruction cache. However, it is even less optimal than M2 memory for data storage. External memory is accessed through the ASEMI bus, which runs at half the core frequency. An immediate write access to external memory that bypasses the write buffer requires eight cycles. However, if the write buffer is enabled, access time is reduced to one cycle. The SC1400 core can read a single 32-bit unit of data from external memory in 23 core clock cycles. The external memory code and data flow diagram is shown in **Figure 5**.

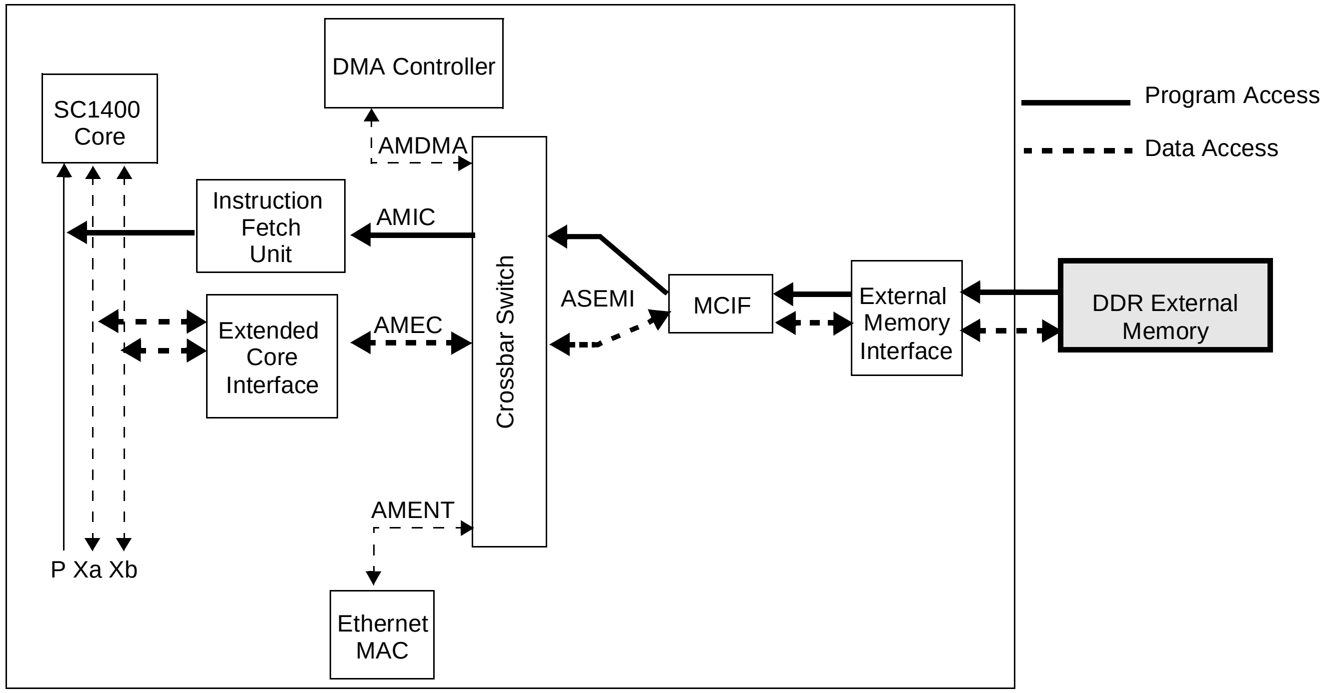

Figure 5. External Memory Code and Data Flow

Table 1 summarizes the SC1400 core access times for read and write transfers to memories. The M1 memory access times assume that no M1 contention occurs. The M2 memory access times assume there are no other accesses on the ASM2 and AMEC buses and that the write buffer is empty. The external memory access times assume there are no other accesses on the ASEMI and AMEC buses, the write buffer is empty, and the MCIF ECI predictive read is enabled.

Table 1. SC1400 Core Access Times

SC1400 Core Access	Access Type	Number of Core Clocks		
		M1 Memory	M2 Memory	External Memory
Single 32-bit Access	Read	1	9	23
	Write	1	N/A	N/A
	Write - Immediate	N/A	8	8
	Write - WB	N/A	1	1
8 Consecutive 32-bit Accesses	Read	8	40	84
	Write	8	N/A	N/A
	Write - Immediate	N/A	40	40
	Write - WB	N/A	12	13

The recommended practices for memory allocation are as follows:

- M1 memory. Place all critical data into M1 memory. Follow the M1 memory grouping recommendations discussed in **Section 1**, *Reducing M1 Memory Contentions*, on page 1. Any space left over in M1 memory can be used to store critical program code.
- M2 memory:
  - Place program code that needs to run efficiently into M2 memory.
  - Program one of the write buffer data areas (WBDAR*n*) assigned to the M2 memory address space memory for normal write buffer operation.
  - Program the instruction cacheable area configuration register (IRCR*n*) as follows:
    - Set primary set size to 4 fetch sets. When a cache miss occurs, a portion of the cache line is loaded from M2 as a primary set of bursts. The size of the primary set is always greater than or equal to the burst size.
    - Set burst size to 4 fetch sets. When a cache miss occurs, the new data is fetched in bursts of 4 fetch sets or 64 bytes.
    - Enable prefetch to the end of the cache line. This mode takes advantage of the spatial locality of the code.
  - Any space left over in M2 memory can be used for additional data.
  - Use the last 64 bytes of M2 memory for data only. Code fetches from this area by the SC1400 core can result in an attempt to access the reserved areas beyond the end of the M2 memory. Such fetches may cause the system to stop operation. For example, for MSC711x devices with 192 KB M2 memory, the memory range 0x0102FFC0–0x0102FFFF should be reserved for data only.
- External memory:
  - Place additional noncritical program code into external memory.
  - Program one of the write buffer data areas (WBDAR*n*) assigned to the external DDR memory address space memory for normal write buffer operation.
  - Program the instruction cacheable area configuration register (IRCR*n*) as follows:
    - Set primary set size to 4 fetch sets.
    - Set burst size to 4 fetch sets.
    - Enable prefetch to the end of the cache line.

Figure 6 shows an example memory allocation for program and data. Hot code/data indicate the most critical and cold code/data are the least critical.

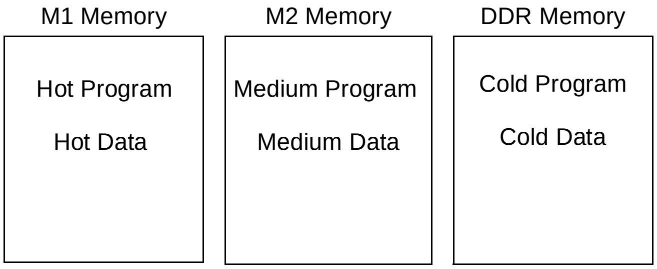

Figure 6. Example Memory Allocation for Program and Data

3 Configuring Write Buffer Areas

As noted earlier, the ECI write buffer improves performance of write accesses by the core areas outside the extended core. For best system performance, assign a write buffer area to the M2 and external DDR memory spaces as well as to the peripheral address spaces. Because the SC1400 core directly accesses M1 memory, the write buffer is bypassed in M1 write accesses. Programming recommendations for assigning write buffer areas, WBDAR*n*, are as follows:

- One WBDAR*n* to the M2 memory address space memory for normal write buffer operation.
- One WBDAR*n* to the external DDR memory address space memory for normal write buffer operation.
- One WBDAR*n* to the TDM/HDI ports through the ASTH bus memory space for normal write buffer operation.
- One WBDAR*n* to the IPBus and APB peripheral memory space for write immediate operation.

4 Improving DDR Performance

The memory controller interface (MCIF) increases the efficiency of accesses through the DDR interface. A write buffer improves write access performance on the ASEMI. It supports zero wait-state write accesses when it is not full. It accepts 8 × 64-bit write accesses before stalling the ASEMI bus. The write buffer is always enabled and it is used for all write accesses. No special configuration is needed.

The MCIF also supports predictive reading for critical program and data accesses through the four dedicated read buffers shown in **Table 2**. These programmable buffers improve efficiency through the DDR memory controller. The MCIF can perform program predictive reads and store the program in the IFU read buffer, which supports 4 × 128-bits of program accesses from external memory. The MCIF also performs data predictive reads and stores the data in the ECI, DMA, and ALT read buffers. The ECI read buffer supports 2 × 64-bits of read data access by the core from external memory. **Table 3** shows that the read access time from external memory by the core is 84 cycles in 16-pin mode or 72 cycles in 32-pin mode when the ECI predictive read is enabled. When ECI predictive read is disabled, the access time increases to 104 core cycles. Both the DMA and ALT read buffers support 4 × 64-bit of read data access by the DMA controller and the Ethernet MAC.

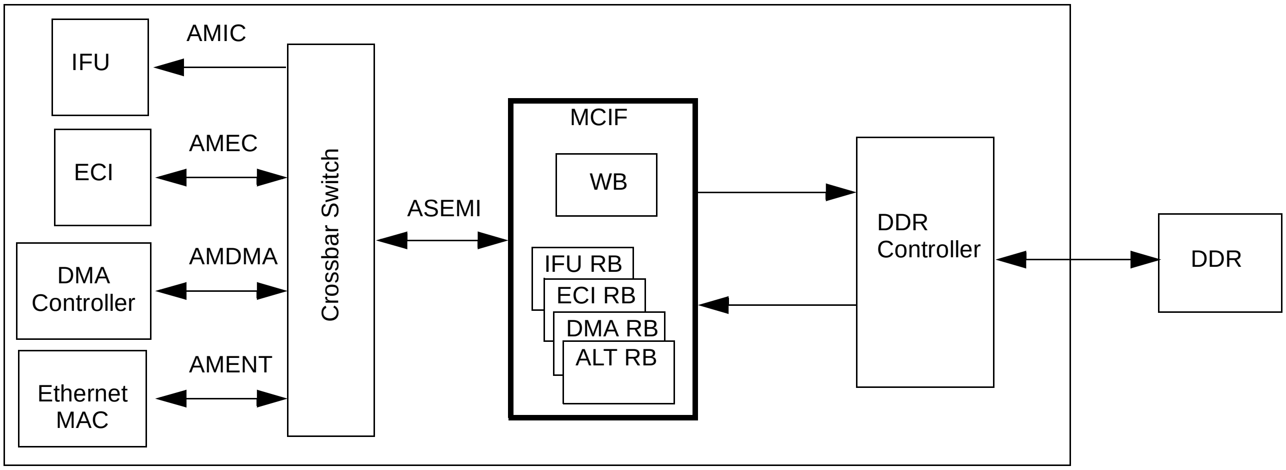

Figure 7. MCIF Block Diagram

Table 2. MCIF Read Buffers

Read Buffer	Predictive Reads	Size
IFU	Program accesses by core	4 × 128 bits
ECI	Data read accesses by core	2 × 64 bits
DMA	Data read accesses by DMA	4 × 64 bits
ALT	Data read accesses by DMA or Enet MAC	4 × 64 bits

Table 3. SC1400 Data Read from External Memory Access Times

Access Type	DDR Mode	ECI Predictive Read	
		On	Off
8 Consecutive 32-bit Read Accesses by Core	16-pin	84 cycles	104 cycles
	32-pin	72 cycles	104 cycles

Recommended practices for improving DDR performance are as follows:

- To optimize program accesses from external memory by the core, enable predictive reads for the IFU read buffer: set MCIFCTRL[IPRE] = 01.
- To optimize data accesses from external memory by the core, enable predictive reads for the ECI read buffer: set MCIFCTRL[EPRE] = 1
- To optimize DMA read accesses, use both the DMA and ALT read buffers as needed and configure them to serve DMA channels that stream data from external memory. Because each read buffer supports five DMA channels, using only the DMA read buffer is probably adequate:
  - DMA accesses using the DMA read buffer:
    - Set MCIFCTRL[DPRE] = 1 to enable DMA predictive reads.
    - Set MCIFCTRL[DCOE] = 1 to enable DMA channel select operation.
    - Set DMA channel select fields in the DCHSEL register to correspond with DMA channels that stream data from external memory.
    - Use DMA transfer size of 32 bytes.
  - DMA accesses using the ALT read buffer (as needed):
    - Set MCIFCTRL[AMSEL] = 0001 to select the ALT read buffer to serve the DMA controller.
    - Set MCIFCTRL[APRE] = 1 to enable ALT predictive reads.
    - Set MCIFCTRL[ACOE] = 1 to enable DMA channel select operation.
    - Set DMA channel select fields in the ACHSEL register to correspond with DMA channels that stream data from external memory.
    - Use a DMA transfer size of 32 bytes.

- To optimize read accesses by the Ethernet MAC, use the ALT read buffer.
  - — Give ALT read buffer service preference to the DMA controller if more than five DMA channels are used for DDR streaming. Otherwise, enable the ALT buffer to serve the Ethernet MAC.
  - — MCIFCTRL[AMSEL] = 0011 to select the ALT read buffer to serve the Ethernet MAC.
  - — MCIFCTRL[APRE] = 1 to enable ALT predictive reads.
- Configure the DDR for open page when most DDR accesses are DMA bursts:
  - — Accesses are efficient because subsequent accesses do not need to issue an ACTIVATE command when the DDR controller opens a page.
  - — Set the precharge interval in DDR_SDRAM_INTERVAL[BSTOPRE].
- Configure the DDR for auto-precharge mode when most DDR accesses are random accesses:
  - — The DDR controller must issue a PRECHARGE command to a logical bank after every read or write transaction.
  - — Set DDR_SDRAM_INTERVAL[BSTOPRE] = 0.
- For DMA transfers from any internal memory to external DDR memory, do not use a DMA transfer size of 32 bytes because incorrect data may be written to external DDR memory under certain conditions. Instead, use a DMA transfer size of 8 bytes. This workaround has no performance impact. For all other transfers, use a DMA transfer size of 32 bytes.
- The user should experiment with enabling the 2T timing in the DDR_SDRAMCFG[2TEN] if incorrect data is being written to external memory. This mode holds the address and command signals valid for one additional cycle. It is expected to have a minor impact on performance.

5 Avoiding Nested Preemption of DMA Channels

A DMA channel can be preempted if it is configured for fixed priority arbitration. Preemption can be enabled for long, lower-priority DMA transfers. However, nested preemption is not supported, so long latencies can occur before a high-priority DMA channel is serviced. **Figure 8** shows the nested preemption scenario, which proceeds as follows:

1. A DMA channel with a large byte count and low priority is in progress.
2. A second DMA request comes from a DMA channel with a slightly higher priority and also a large byte count.
3. The second DMA channel preempts the first.
4. A third DMA request comes from a peripheral such as the TDM with a priority higher than the first and the second DMA channels.
5. The third DMA channel must wait until the preempting DMA channel (the second channel) completes before it can preempt because nested preemption is not allowed. A high-priority DMA channel may miss its real-time deadline because of the long latency. This is an issue only when priority is fixed and preemption is enabled on a subset of the channels.
6. When the third DMA channel completes, the first DMA channel becomes active again to complete its transfer.

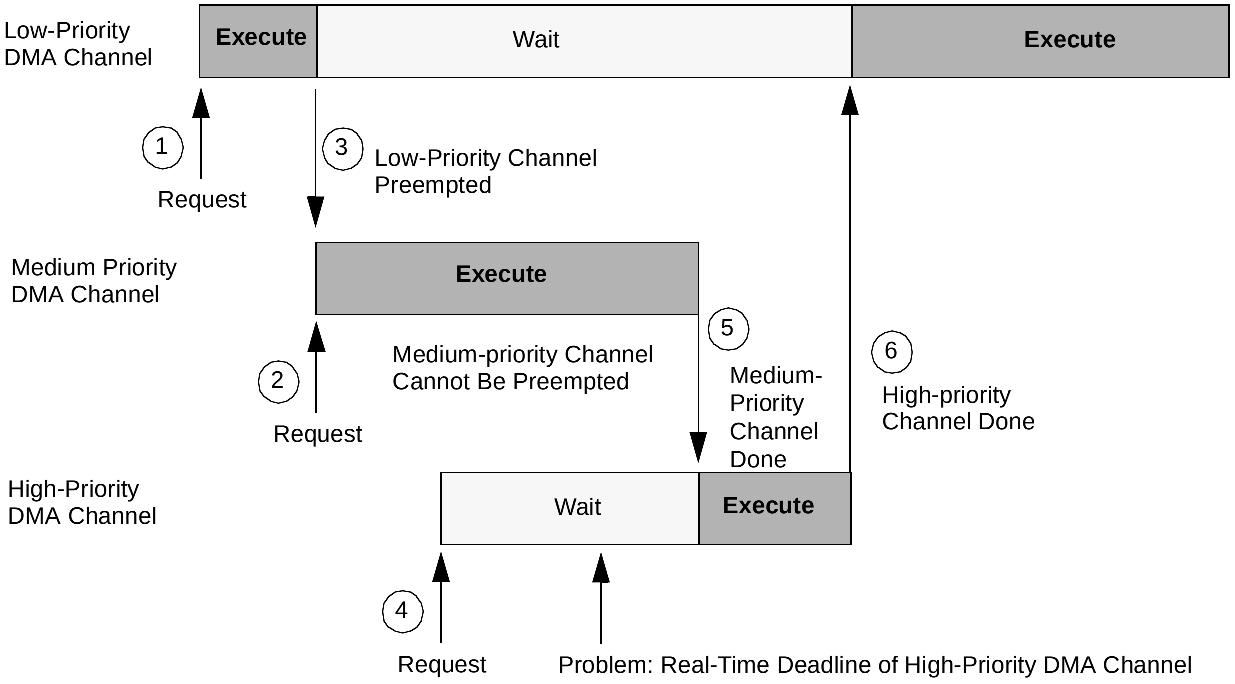

Figure 8. DMA Nested Preemption Scenario

Use one of the following techniques to ensure that only one medium- or low-priority DMA channel with large byte counts or long transfer times is active at a time:

- Schedule all lower-priority preemptable DMA transfers so that only one low-priority channel is active at a time. For example, filter all lower-priority DMA transfer requests through one software driver to ensure that no other channel with this characteristic is active when a channel starts.
- Link all the low- or medium-priority channels so that when one preemptable channel completes, it continues with the next in the chain.
- Instead of configuring a low-priority DMA channel for a single minor loop and a large byte count, break up the transfer into several minor loops using the DMA major-minor looping structure. This allows the DMA controller to rearbitrate at the end of each minor loop so that the highest-priority channels have less time to wait if the DMA preemption issue is encountered. For example, a 2048 byte DMA transfer can be broken into sixteen 128-byte transfers.

Figure 9 shows the same scenario as **Figure 8**, but the medium-priority DMA channel is split into multiple transactions. This causes stall and re-arbitration but the high-priority channel now waits less time and therefore can meet its real-time deadline.

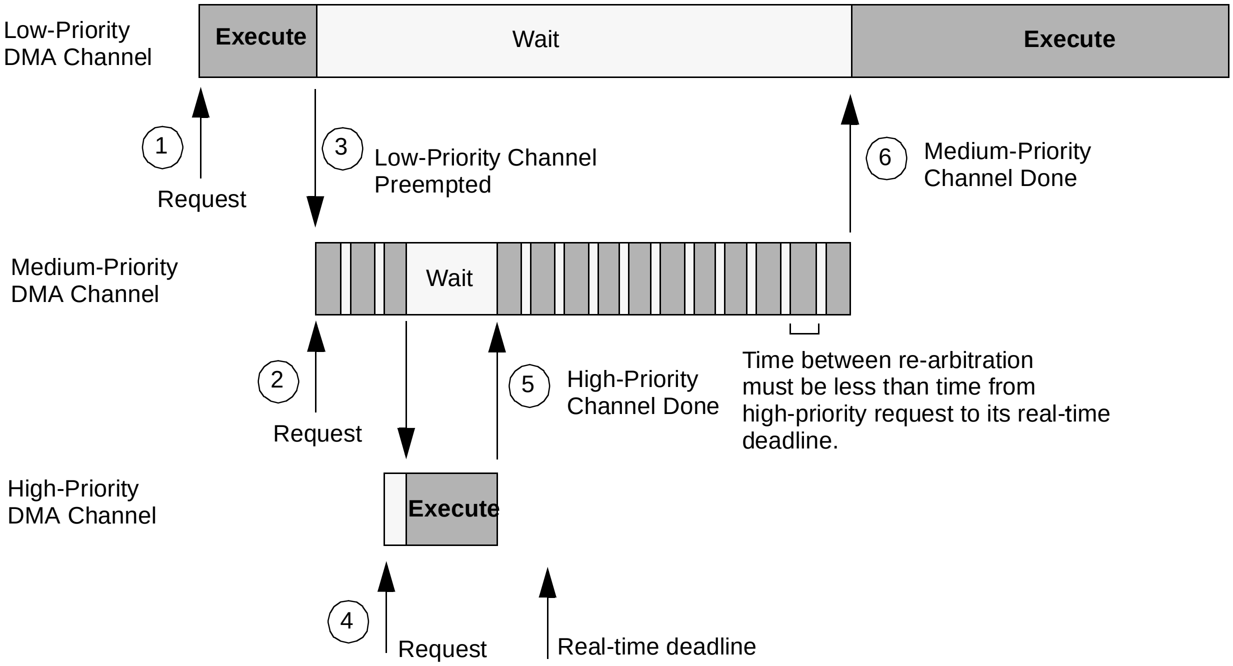
Low-Priority DMA Channel
Execute
Wait
Execute
1
Request
3
Low-Priority Channel Preempted
6
Medium-Priority Channel Done
Medium-Priority DMA Channel
Wait
2
Request
5
High-Priority Channel Done
Time between re-arbitration must be less than time from high-priority request to its real-time deadline.
High-Priority DMA Channel
Execute
4
Request
Real-time deadline

Figure 9. Meeting Real-Time Deadlines

6 Configuring the Crossbar Switch

The multi-port crossbar switch allows up to four parallel data transfers between four masters and six slaves as shown in **Figure 10**. For example, the crossbar switch supports the following operations in parallel:

- Cache bursts from the M2 memory via the instruction fetch unit (IFU)
- Core writes to a peripheral register via the extended core interface (ECI)
- DMA bursts data to external memory
- Ethernet MAC DMA bursts data to M1 memory

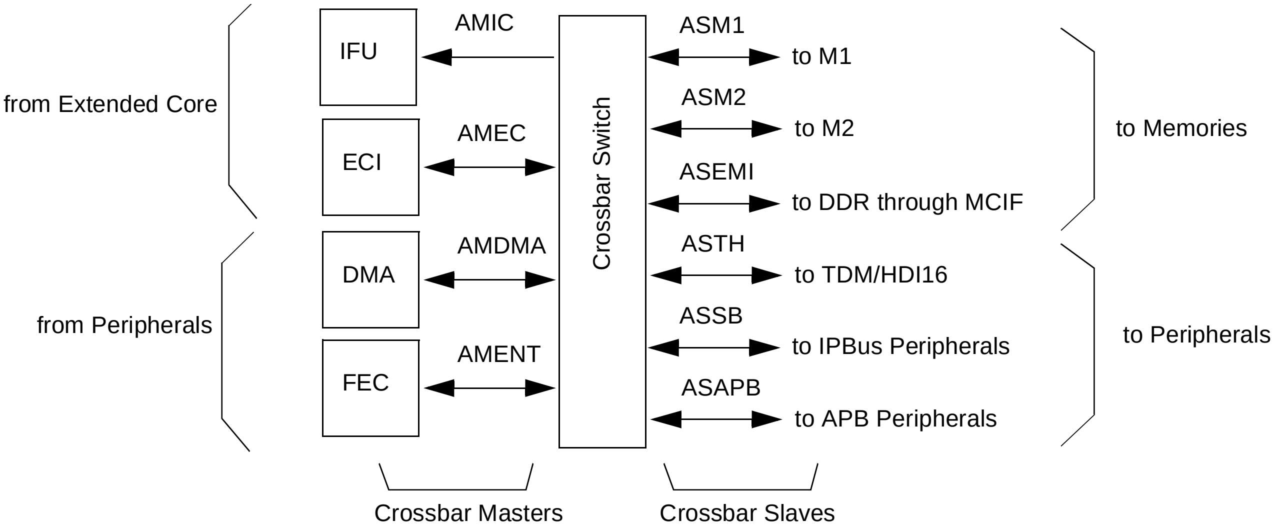

Figure 10. Crossbar Masters and Slaves

The master and slave ports and their corresponding port numbers are shown in **Table 4** and **Table 5**. At reset, the ECI has the highest priority, followed by the DMA, the IFU and the Ethernet MAC. When multiple masters access the same slave port, arbitration determines which master is granted access to the slave port. Depending on the user application, the default priorities should be changed according to the system requirements. Proper crossbar priority settings is critical for optimal system performance.

Table 4. Crossbar Switch Masters

Master	Master Port	Crossbar Port Number	Priority at Reset
Extended Core Interface	AMEC	0	Highest
DMA	AMDMA	1	--
Instruction Fetch Unit	AMIC	2	--
Ethernet MAC	AMENT	3	Lowest

Table 5. Crossbar Switch Slaves

Slave	Slave Port	Crossbar Port Number
M1 Memory	ASM1	0
M2 Memory	ASM2	1
External Memory Interface	ASEMI	2
TDM/HDI Ports	ASTH	3
IPBus Peripherals	ASSB	4
APB Peripherals	ASAPB	5

6.1 Correct Arbitration Scheme

Correct system operation requires the correct arbitration scheme at each slave port in the crossbar switch. The arbitration scheme is configured in each slave port through the following two registers:

- Slave General-Purpose Control Register (SGPCR*n*) for selecting fixed-priority or round-robin arbitration.
- Master Priority Register (MPR*n)* for setting the priority for each master port.

The following practices are recommended to ensure usage of the correct arbitration scheme. It is important to experiment with these settings to determine the best system configuration:

- The Ethernet controller does not elevate any of its accesses. As a result, all Ethernet accesses are normally at a lower priority than priority-elevated accesses from the IFU, ECI, and DMA controller. To address this situation, the control bit DEVCFG[ENTP] allows you to define all Ethernet accesses as always elevated or never elevated. This bit is cleared by default.
- The ASM1 slave port supports accesses from the AMDMA and AMENT masters. The AMIC and AMEC should never access this slave port. **Figure 11** shows the ASM1 supported accesses.
  - — *Priority scheme 1*. The AMDMA has a higher priority than the AMENT, so all DMA transfers have a higher priority than Ethernet MAC DMA transfers:
    - – SGPCR0[ARB] = 00 for fixed-priority arbitration

    - MPR0[MSTR_1] = 000 to give AMDMA highest priority when ASM1 is accessed
    - MPR0[MSTR_3] = 001 to give AMENT second highest priority when ASM1 is accessed
  - *Priority scheme 2*. The AMENT has a higher priority than the AMDMA. Only DMA transfers with priority elevation have a higher priority than an Ethernet MAC DMA transfer.
    - SGPCR0[ARB] = 00 for fixed-priority arbitration
    - MPR0[MSTR_3] = 000 to give AMENT highest priority for ASM1 accesses
    - MPR0[MSTR_1] = 001 to give AMDMA second highest priority for ASM1 accesses
  - *Priority scheme 3*. The crossbar masters have round-robin priority.
    - SGPCR0[ARB] = 01 for round-robin arbitration

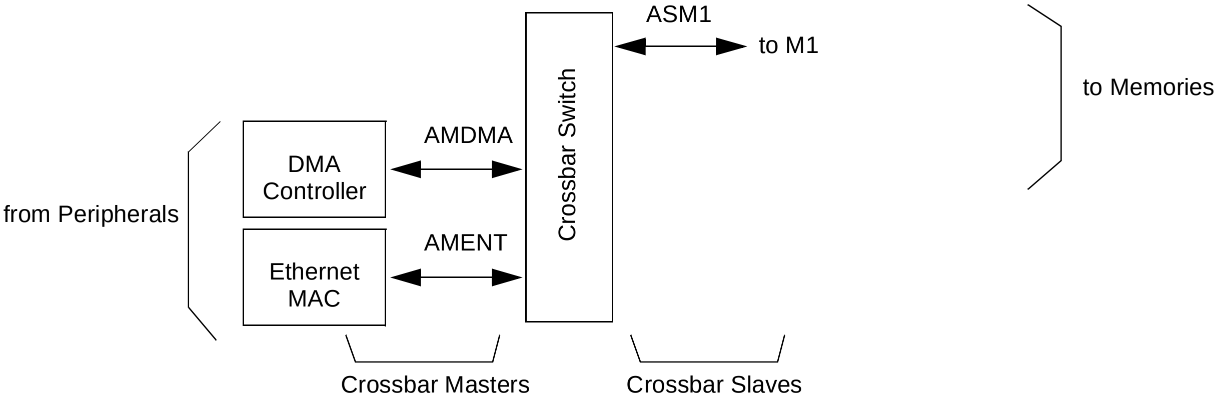

Figure 11. Supported ASM1 Accesses

- Both ASM2 and ASEMI slave ports support accesses from all masters. **Figure 12** and **Figure 13** show the supported accesses. The following settings apply to both ASM2 and ASEMI accesses:
  - *Priority scheme 1*. The AMIC has a higher priority than the other masters. This scheme favors servicing cache misses that stall the core.
    - SGPCR1/SGPCR2[ARB] = 00 for fixed-priority arbitration
    - MPR1/MPR2[MSTR_2] = 000 to give AMIC highest priority
    - MPR1/MPR2[MSTR_0] = 001 to give AMEC second highest priority
    - MPR1/MPR2[MSTR_3] = 010 to give AMENT third highest priority
    - MPR1/MPR2[MSTR_1] = 011 to give AMDMA fourth highest priority
  - *Priority scheme 2*. The AMEC has a higher priority than the other masters. This scheme favors core data accesses that can stall the SC1400 core.
    - SGPCR1/SGPCR2[ARB] = 00 for fixed-priority arbitration
    - MPR1/MPR2[MSTR_0] = 000 to give AMEC highest priority
    - MPR1/MPR2[MSTR_2] = 001 to give AMIC second highest priority
    - MPR1/MPR2[MSTR_3] = 010 to give AMENT third highest priority
    - MPR1/MPR2[MSTR_1] = 011 to give AMDMA fourth highest priority
  - *Priority scheme 3*. The AMENT has a higher priority than the other masters. This scheme favors the Ethernet MAC accesses when elevated.
    - SGPCR1/SGPCR2[ARB] = 00 for fixed-priority arbitration

- MPR1/MPR2[MSTR_3] = 000 to give AMENT highest priority
- MPR1/MPR2[MSTR_2] = 001 to give AMIC second highest priority
- MPR1/MPR2[MSTR_0] = 010 to give AMEC third highest priority for ASM2/ASEMI accesses
- MPR1/MPR2[MSTR_1] = 011 to give AMDMA fourth highest priority

— *Priority scheme 4*. The crossbar masters have round-robin priority.

- SGPCR1/SGPCR2[ARB] = 01 for round-robin arbitration

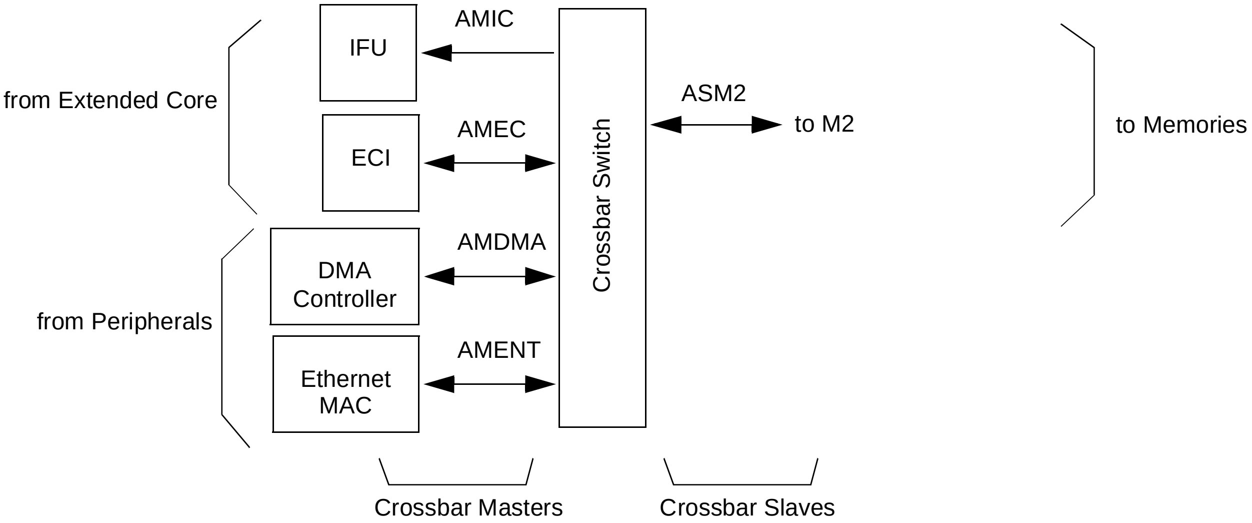

Figure 12. Supported ASM2 Accesses

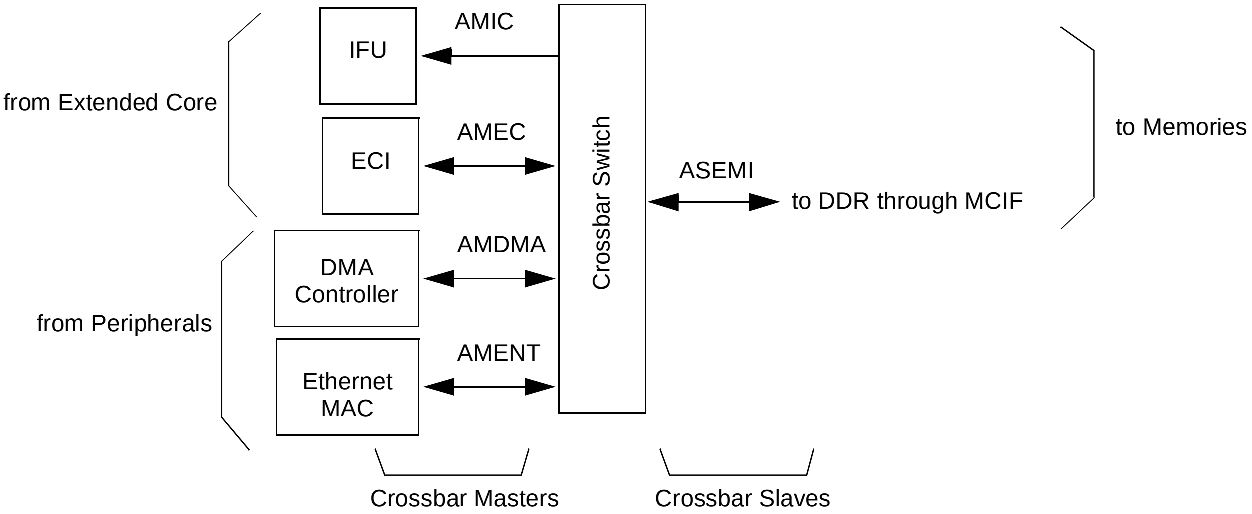

Figure 13. Supported ASEMI Accesses

- The ASTH slave port supports accesses from the AMEC and AMDMA masters. The AMIC and AMENT should never access this slave port. **Figure 14** shows the supported ASTH accesses.

— *Priority scheme 1*. The AMDMA has a higher priority than the AMEC. This scheme favors accessing the TDM and HDI16 registers via the DMA controller.

- SGPCR3[ARB] = 00 for fixed-priority arbitration

  - MPR3[MSTR_1] = 000 to give AMDMA the highest priority for ASTH accesses
  - MPR3[MSTR_0] = 001 to give AMEC second highest priority for ASTH accesses
- *Priority scheme 2*. The AMEC has a higher priority than the AMDMA. This scheme favors accessing the TDM and HDI16 registers via the DMA controller.
  - SGPCR3[ARB] = 00 for fixed-priority arbitration
  - MPR3[MSTR_0] = 000 to give AMEC the highest priority for ASTH accesses
  - MPR3[MSTR_1] = 001 to give AMDMA the second highest priority for ASTH accesses
- *Priority scheme 3*. The AMDMA and AMEC have a round-robin priority.
  - SGPCR3[ARB] = 01 for round-robin arbitration

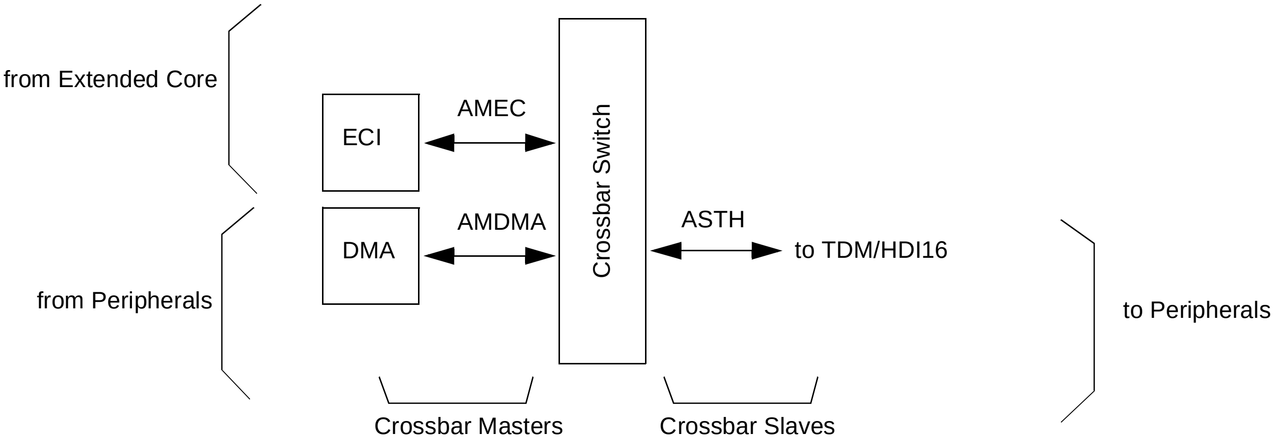

Figure 14. Supported ASTH Accesses

- Both the ASSB and ASAPB slave ports handle accesses from the AMEC and AMDMA masters. **Figure 15** and **Figure 16** show the supported ASSB and ASAPB accesses. The following settings apply to both ASSB and ASAPB:
  - *Priority scheme 1*. The AMEC has a higher priority than the AMDMA because DMA transfers do not typically occur on the AHB slave buses.
    - SGPCR4/SGPCR5[ARB] = 00 for fixed-priority arbitration
    - MPR4/MPR5[MSTR_0] = 000 to give AMEC the highest priority for ASM2/ASEMI accesses
    - MPR4/MPR5[MSTR_1] = 001 to give AMDMA second highest priority for ASM2/ASEMI accesses
  - *Priority scheme 2*. The crossbar masters have round robin priority.
    - SSGPCR4/SGPCR5[ARB] = 01 for round-robin arbitration

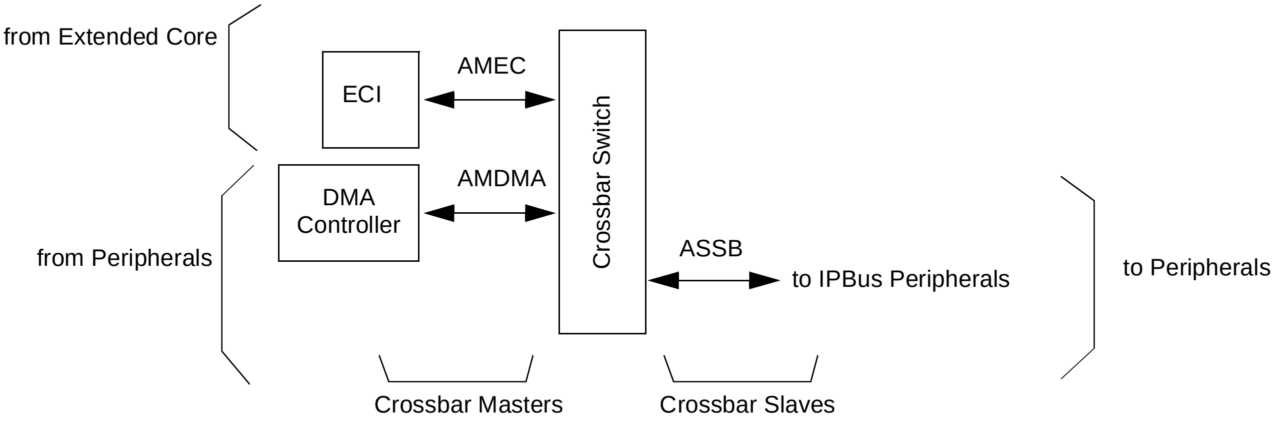

Figure 15. Supported ASSB Accesses

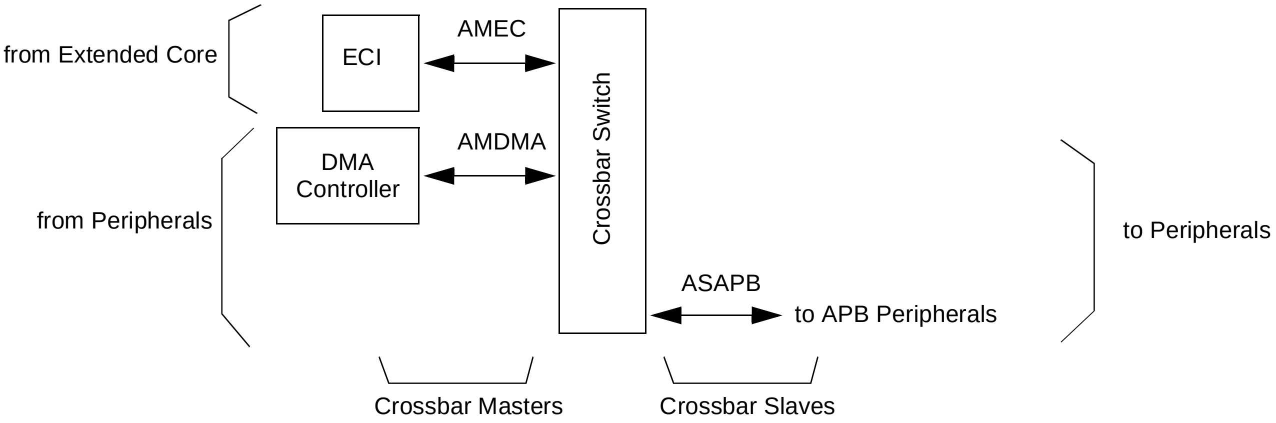

Figure 16. Supported ASAPB Accesses

Table 6 summarizes the recommended crossbar master priority settings.

Table 6. Summary of Crossbar Priority Schemes

Slave Port	Supported Master Accesses				Recommended Priorities			
	AMEC	AMDMA	AMIC	AMENT	Scheme 1	Scheme 2	Scheme 3	Scheme 4
ASM1	N	Y	N	Y	1. AMDMA 2. AMENT	1. AMENT 2. AMDMA	Round Robin	N/A
ASM2	Y	Y	Y	Y	1. AMIC 2. AMEC 3. AMENT 4. AMDMA	1. AMEC 2. AMIC 3. AMENT 4. AMDMA	1. AMENT 2. AMIC 3. AMEC 4. AMDMA	Round Robin
ASEMI	Y	Y	Y	Y	1. AMIC 2. AMEC 3. AMENT 4. AMDMA	1. AMEC 2. AMIC 3. AMENT 4. AMDMA	1. AMENT 2. AMIC 3. AMEC 4. AMDMA	Round Robin
ASTH	Y	Y	N	N	1. AMDMA 2. AMEC	1. AMEC 2. AMDMA	Round Robin	N/A

Table 6. Summary of Crossbar Priority Schemes (continued)

Slave Port	Supported Master Accesses				Recommended Priorities			
	AMEC	AMDMA	AMIC	AMENT	Scheme 1	Scheme 2	Scheme 3	Scheme 4
ASSB	Y	Y	N	N	1. AMEC 2. AMDMA	1. AMDMA 2. AMEC	Round Robin	N/A
ASAPB	Y	Y	N	N	1. AMEC 2. AMDMA	1. AMDMA 2. AMEC	Round Robin	N/A

6.2 Required High-Priority Enable Bit Usage

The high-priority bits in the SGPCR*n*[HPE7–0] enable the high-priority inputs for the respective master. These bits must be set according to the requirements to ensure that the crossbar switch works with the masters as desired. **Table 7** summarizes the required crossbar SGPCR*n* settings, which are as follows:

- SGPCR*n*[HPE7–4] bits that correspond to unimplemented master ports must be reset to 0 on all slave ports.
- The SGPCR*n*[HPE3] bit that corresponds to the Ethernet MAC must be cleared to 0 on all slave ports.
- SGPCR0[HPE2], SGPCR3[HPE2], SGPCR4[HPE2], and SGPCR5[HPE2] bits that correspond to the IFU must be reset to 0. The slave ports that do not support program access are ASM1, ASTH, ASSB, and ASAPB.
- SGPCR1[HPE2] and SGPCR2[HPE2] bits that correspond to the IFU must be set to 1. The slave ports that support program access are ASM2 and ASEMI.
- The SGPCR0[HPE0] bit that corresponds to the ECI must be set to 0. The slave port that does not support ECI access is ASM1.
- SGPCR1[HPE0] and SGPCR2[HPE0] bits that correspond to the ECI must be set to 1. The slave ports that support ECI access are ASM2 and ASEMI
- The remaining SGPCR*n*[HPE*x*] bits can be programmed as desired.

Table 7. Required SGPCRn[HPE7:0] Bit Settings

Slave General-Purpose Control Register	Slave Port	Master Port				
		Unused Masters (HPE7–4)	AMENT (HPE3)	AMIC (HPE2)	AMDMA (HPE1)	AMEC (HPE0)
SGPCR0	ASM1	0000	0	0	as desired	0
SGPCR1	ASM2	0000	0	1	as desired	1
SGPCR2	ASEMI	0000	0	1	as desired	1
SGPCR3	ASTH	0000	0	0	as desired	as desired
SGPCR4	ASSB	0000	0	0	as desired	as desired
SGPCR5	ASAPB	0000	0	0	as desired	as desired

7 Preventing Master Port Time-outs

In DMA transfers for which both the source and destination access the same memory space, a time-out can occur in the bus error detection units. The time-out can occur in M1-to-M1, M2-to-M2, or DDR-to-DDR transfers through the DMA controller. The DMA controller reads from one slave bus and writes to the same slave bus, which prevents other crossbar masters from arbitrating for that particular slave bus. When the other masters are locked out of the crossbar switch for a long time, a time-out occurs and the bus error interrupt is asserted. Following are recommendations for avoiding a bus time-out error:

- Program the memory slave port SGPCR to designate the DMA controller as the lowest-priority master:
  - For M1-to-M1 transfers via DMA, configure SGPCR0[MSTR_1] to give the AMDMA master a lower priority than AMENT, as shown in scheme 2 in Table 6 on page 16.
  - For M2-to-M2 transfers via DMA, configure SGPCR1[MSTR_1] to give the AMDMA master a lower priority than the other masters, as shown in any of the schemes in **Table 6**.
  - For DDR-to-DDR transfers via DMA, configure SGPCR2[MSTR_1] to give the AMDMA master a lower priority than the other masters, as shown in any of the schemes in **Table 6**.
- Ensure that the DMA channel performing the same memory-to-memory transfer is not elevated:
  - The DMA channel bandwidth control bit TCD7[BWC] must not have a value of 01 for dynamic priority elevation. If the TCD7[BWC] = 01, the DMA priority is elevated, which can potentially lock out the other masters.
  - Configure the TCD7[BWC] = 10 to stall the DMA for four cycles or set the TCD7[BWC] = 11 to stall the DMA for eight cycles after each read/write access completes.

8 Glossary

- *AHB Master DMA (AMDMA)*. A 64-bit wide, single-master bus connecting the DMA controller to the crossbar switch. The DMA controller is the master and the crossbar switch is a slave. This bus is used for transferring data when the DMA controller initiates transfers within the MSC711x.
- *AHB Master Extended Core (AMEC)*. A 64-bit wide, single-master bus connecting the extended core interface to the crossbar switch. The bus switch and write buffer in the extended core interface are multiplexed to form a single master, and the crossbar switch is a slave. When the SC1400 core accesses locations outside the extended core, this unit controls the access.
- *AHB Master Ethernet MAC (AMENT)*. A 32-bit wide, single-master bus connecting the Ethernet MAC's DMA to the crossbar switch. The Ethernet MAC is the master, and the crossbar switch is a slave. This bus is used for transferring data when the Ethernet MAC DMA initiates transfers within the MSC711x.
- *AHB Master Instruction Cache (AMIC)*. A 128-bit wide, read-only, single-master bus connecting the instruction fetch unit to the crossbar switch. The instruction fetch unit is the master, and the crossbar switch is a slave. When the instruction cache misses, the fetch unit issues an address to M2 memory or external memory to begin a cache burst.
- *AHB Slave to APB (ASAPB)*. A 32-bit wide, single-master bus connecting the crossbar switch to the interface bridge for the APB bus. The crossbar switch is the master, and the bridge is the slave. The switch accesses peripherals on the APB bus through this bridge. These peripherals include the UART, GPIO, TDM, HDI, interrupt controller, and software watchdog timer.

- *AHB Slave to External Memory Interface (ASEMI).* 64-bit wide, single-master bus connecting the crossbar switch to MSC711x external memory interface. The crossbar switch is the master and the external memory interface is the slave.
- *AHB Slave to M1 (ASM1)*. 64-bit wide, single-master bus connecting the crossbar switch to M1 memory. The crossbar switch is the master and the M1 memory is the slave. The M1 memory can be accessed by the DMA and Ethernet MAC.
- *AHB Slave to M2 (ASM2)*. 128-bit reads, 64-bit writes, single-master bus connecting the crossbar switch to M2 memory and Boot ROM. The crossbar switch is the master and the M2 and Boot ROM are the slaves. The M2 memory can be accessed by the core via the write buffer. The instruction fetch unit can burst from M2 or Boot ROM into the instruction cache. The M2 memory can be accessed by the DMA and Ethernet MAC.
- *AHB Slave to IPBus (ASSB)*. 32-bit wide, single-master bus connecting the crossbar switch to the interface bridge for the IPBus. The crossbar switch is the master and the bridge is the slave. Peripherals on the IPBus are accessed by the crossbar switch through this bridge. These peripherals include the DMA, DDR memory controller, MCIF, Ethernet MAC, I2C, and Timers.
- *AHB Slave to TDM/HDI16 Interfaces (ASTH)*. 64-bit wide, single-master bus connecting the crossbar switch to the HDI16 and TDM peripherals. The crossbar switch is the master and the HDI16 is the slave. ASTH provides a high speed port for accessing data through these peripherals.
- Crossbar Switch. The multi-port crossbar switch allows multiple data transfers to occur in parallel outside the extended core.
- *Double Data Rate (DDR)*. The external memory controller interface provides a 32-bit wide bus that connects the crossbar switch to the external system bus that connects to DDR memory devices.
- *Direct Memory Access (DMA)* . The 32-channel DMA controller that transfers data to and from M1, M2, external DDR memory, HDI and TDM interfaces.
- *Extended Core Interface (ECI)*. Handles the core requests to resources outside the extended core through the crossbar switch over the AMEC bus.
- *Ethernet Media Access Controller (MAC)*. Provides 10/100 Ethernet access. It has direct access to the MSC711x internal memories through its own DMA controller using the AMENT bus.
- *Instruction Fetch Unit (IFU)*. Program fetches to locations outside the M1 address space occur through the IFU via the AMIC bus. It also conducts all instruction cache update activity when a cache miss occurs.
- - *Memory Controller Interface (MCIF)*. Supports a write buffer and four dedicated read buffers to increase external DDR memory accesses. It also supports programmable predictive read capability.

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.