5 INSTRUCTION CACHE CONTROLLER

5.1 INTRODUCTION

The instruction cache may be viewed as a buffer memory between the main (external and probably slow) memory, and the fast CPU. The cache is used to store the program instructions that are frequently used. An increase in throughput may result when instruction words required by a program are available in the on-chip cache, and the time required to access them on the external bus is eliminated. Cache specific instructions are provided in the instruction set, permitting the user to lock sectors of the cache, and to flush the cache contents under software control. The instruction cache controller in the DSP56300 core is capable of controlling 1k or 2k of instruction cache memory array, with the following features:

Following is a summary of the instruction cache features:

- Software controlled enable bit in the chip extended mode register
- Mask programming selection between 1024 or 2048 locations
- 8 Way, Fully Associative, Sectored Placement Policy
- One to Four Word Transfer Granularity
- LRU Sector Replacement Algorithm
- User Transparent - No user management required
- Individual Sector Locking/Unlocking
- Software controlled Global Cache Flush
- Cache controller status observability via the On-Chip Emulator

5.2 INSTRUCTION CACHE ARCHITECTURE

5.2.1 Instruction Cache Structure

The Instruction Cache is composed of the Memory Array and the Cache Controller. Figure 5-1 shows a block diagram of the instruction cache controller.

The Instruction Cache memory array contains 1024 (or 2048, mask programmed) 24-bit words, logically divided into eight 128 (256)-word cache sectors. Since there are 8 sectors of 128 (256) words each, in the internal program RAM, the 24 bit address is divided into the following two fields:

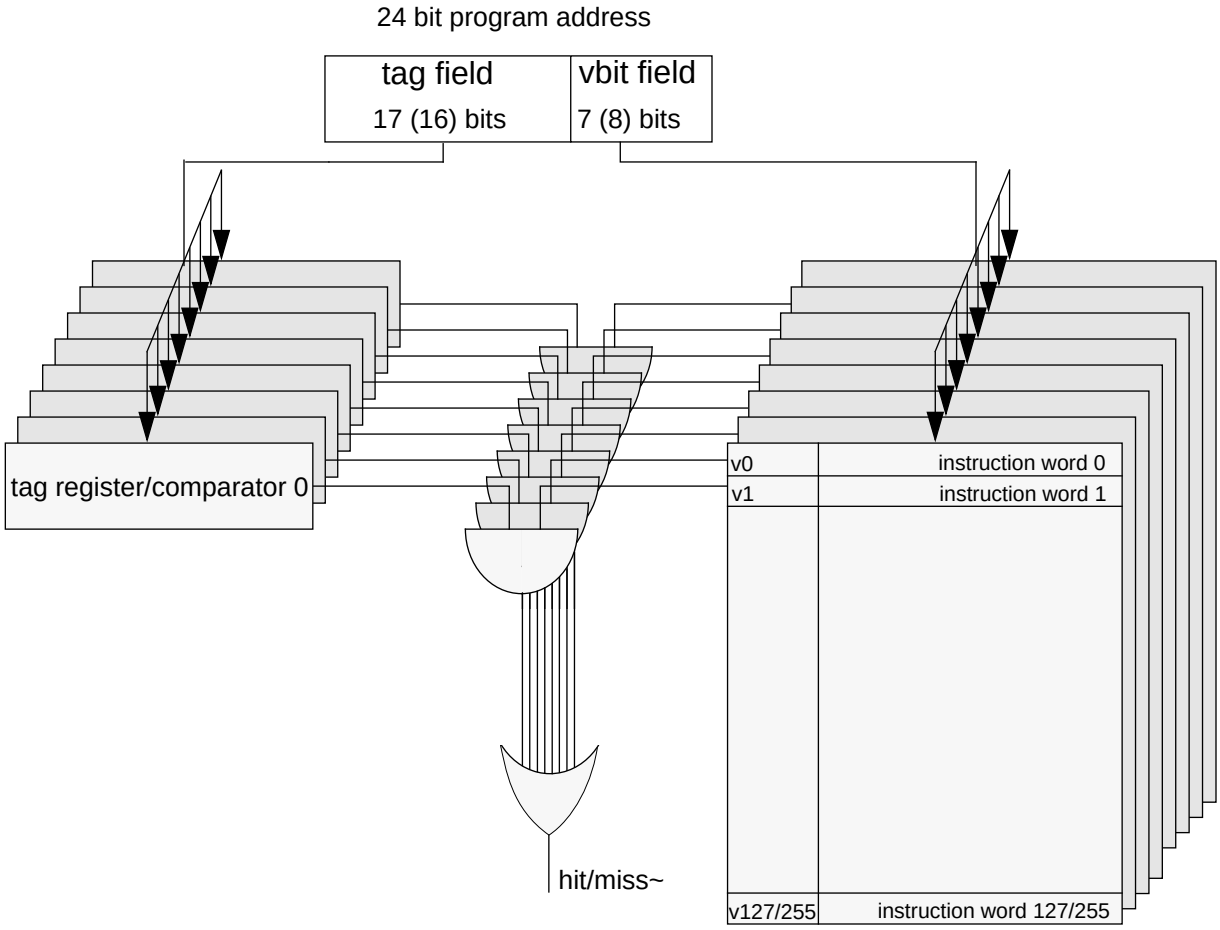
- vbit field; 7 (8) LSBits for the word displacement in the sector
- tag field; 17 (16) MSBits for the sector base address

The sectors placement algorithm is fully associative.

A 17 (16) bit tag is associated with every one of the 8 internal program memory RAM sectors. When the Cache Controller searches for a tag equal to the tag field of the current address, it compares it to the 8 tags in parallel using the 8 comparators.

Each word in each cache sector is associated with a cache word valid bit, that specifies whether or not the data in that word has already been fetched from external memory and is therefore valid. There is a total of 2048 (1024 of them are not used if the Icache size is 1024 bytes) valid bits, arranged as 8 banks of 128 or 256 valid bits each, one bank for every sector. Note that these valid bits are not available to the user, for direct use. The valid bits are cleared by the processor hardware RESET to indicate that the PRAM context has not been initialized.

Figure 5-1. Instruction Cache Block Diagram



5.2.2 Cache Programmer's Model

The Instruction Cache is controlled by two control bits:

- Cache Enable (CE) bit in the EMR part of the Chip Status Register (SR, bit 19). When CE is cleared the Instruction Cache is disabled. When CE is set the Instruction Cache is enabled.
- Burst Enable (BE) bit in the EOM part of the Chip Operating Mode Register (OMR, bit 10). When BE is cleared the Instruction Cache Transfer Granularity on miss is a single-word. When BE is set, the Instruction Cache Transfer Granularity on miss is increased to a one to four burst block.

NOTE To assure proper operation, cache enable mode (CE bit in SR) should not be cleared while burst mode is enabled (BE bit in OMR is set).

- The Instruction Cache is supported by the instruction set via the following instructions: PLOCK, PLOCKR, PUNLOCK, PUNLOCKR, PFREE, PFLUSH, PFLUSHUN.

5.2.3 Cache Operation

5.2.3.1 program fetch

When the core generates an address for an instruction fetch, the cache controller compares its tag field to the tag values currently stored in the tag register file. This tag values are the tag fields of the base addresses of the memory sectors currently mapped into the cache.

5.2.3.2 hit

If there is a tag match, i.e. sector hit, then the valid bit of the corresponding word in that cache sector is checked by using the vbit field as an address to the valid bit array. If the valid bit is set, meaning the word in the cache has already been brought to the cache and is valid, then that word is fetched from the cache location corresponding to the desired address. This situation is called a cache hit meaning that both corresponding sector and corresponding instruction word are present and valid in the instruction cache. The Sector Replacement Unit (SRU) flags the sector as the Most Recently Used (MRU).

5.2.3.3 word miss when burst mode is disabled

If there is a tag match, i.e. sector hit, but the desired word is not flagged as valid (corresponding valid bit is cleared), than the cache initiates a read access to the external program memory, introducing wait states into the pipeline. The amount of wait states will be 1 plus the number of wait states that are programmed into the bus interface unit's control registers, reflecting the type of memory used. The Sector Replacement Unit (SRU) flags the sector as the Most Recently Used (MRU) and the fetched instruction is sent both to the core and copied to the relevant sector location. Then the valid bit of that word is set.

5.2.3.4 word miss when burst mode is enabled

If there is a tag match, i.e. sector hit, but the desired word is not flagged as valid (corresponding valid bit is cleared), than the cache initiates a burst of up to 4 read accesses to the external program memory. The exact number of fetch requests depends

on the two least significant bits of the address of the initiating fetch that was detected as miss -

- ‘11’ - only one request will be initiated as if the burst mode was disabled
- ‘10’ - two requests will be initiated and the number of wait states required by the memory type and speed will be added by two.
- ‘01’ - three requests will be initiated and the number of wait states required by the memory type and speed will be added by three.
- ‘00’ - four requests will be initiated and the number of wait states required by the memory type and speed will be added by four.

These external read accesses will introduce wait states into the pipeline. The amount of wait states for each fetch will be 1 plus the number of wait states that are programmed into the bus interface unit’s control registers, reflecting the type of memory used. The Sector Replacement Unit (SRU) flags the sector as the Most Recently Used (MRU) and each of the fetched instruction is copied to the relevant sector location. Then the valid bit of that word is set.

5.2.3.5 sector miss

If there is no match between the tag field and all sector tag registers, meaning that the memory sector containing the requested word is not present in the cache, the situation is called a sector miss, which is another form of a cache miss. If a sector miss occurred, the cache’s SRU selects the sector to be replaced. The cache controller then flushes the selected cache sector by clearing all corresponding valid bits, loads the corresponding tag register with the new tag field, and at the same time initiates an access to the external program memory, as described in Section 5.2.3.3 and Section 5.2.3.4. The sector is flagged as MRU, the fetched instruction is sent both to the core and copied to the relevant sector location and the valid bit of that word is set.

5.2.4 Default Mode On Hardware Reset

After hardware RESET the Instruction Cache is disabled and the DSP56300 Core operates in the PRAM Mode. The cache is initialized to the following initial condition:

- All valid bits will be cleared.
- All tag registers are initialized to hold the value $1FFFF ($FFFF).
- The LRU stack will hold a default descending order of sectors.
- All cache sectors are in the unlocked state.

5.2.5 Cache Locking

Cache locking is useful for locking some time-critical code parts in the cache memory. When a cache sector is locked, the Sector Replacement Unit (SRU) can not replace this sector even if it becomes the least recently used sector (bottom of LRU stack).

A sector can be locked by the instructions PLOCK or PLOCKR. Their operand is an effective memory address (absolute or PC-relative). The cache sector to which this address belongs (if there is such one), is locked. If the specified effective address does not belong to one of the current cache sectors, a memory sector containing this address will be allocated into the cache, thereby replacing the least recently used cache sector. This cache sector will be locked but empty. If all the cache sectors are already locked, this memory sector will not be allocated into the cache and the lock operation will not be executed. The locked cache sector becomes MRU.

Locking a cache sector, if it is already in the cache, does not affect the contents of it, the value of its valid bits or the corresponding tag register contents.

Note: PLOCK and PLOCKR are detected as illegal opcodes in PRAM Mode. Issuing these instructions when the cache is disabled will initiate the Illegal Interrupt. A distance of at least three instruction cycles (equivalent to three NOP instructions) should be maintained between an instruction that changes the value of the cache enable bit (CE) and one of the instructions PLOCK and PLOCKR.

5.2.6 Cache Unlocking

A locked sector can be unlocked to allow sector replacement from that cache sector. Unlocking can be performed in three different ways.

A locked sector can be unlocked by the PFREE, PUNLOCK or PUNLOCKR instructions. PUNLOCK/R's operand is an effective memory address (absolute or PC-relative). The memory sector containing this address is allocated into a cache sector (if its not already in a cache sector) and this cache sector is unlocked. If all the cache sectors are already locked, this memory sector will not be allocated into the cache and the unlock operation will not be executed. The unlocked cache sector becomes MRU, and is enabled for replacement by the LRU algorithm. Unlocking a locked cache sector using these instructions does not affect its contents, its tag, or its valid bits.

All the locked sectors can be unlocked simultaneously using the instruction PFREE, which provides the user the ability to reset the locking mechanism. Unlocking the sectors using PFREE, does not affect their contents (instructions already fetched into the sector storage area), their valid bits, their tag register contents or the LRU stack status. If a PFREE instruction is executed when none of the sectors are locked, than none of the tag registers, valid bits and LRU status will be changed.

The locked sectors could also be unlocked by the PFLUSH instruction. Unlocking the sectors, via PFLUSH, clears all the sector's valid bits and sets the LRU stack and tag

registers to their default values.

Note: PFREE, PUNLOCK and PUNLOCKR are detected as illegal opcodes in PRAM Mode. Issuing these instructions when the cache is disabled will initiate the Illegal Interrupt. A distance of at least three instruction cycles (equivalent to three NOP instructions) should be maintained between an instruction that changes the value of the cache enable bit (CE) and one of the instructions PFREE, PUNLOCK and PUNLOCKR.

5.2.7 Cache Flush

This operation is performed by executing the instructions PFLUSH or PFLUSHUN. The execution of PFLUSH causes a global cache flush that brings the cache to the hardware reset initial condition:

- All valid bits will be cleared.
- All tag registers are initialized to hold the value $1FFFF ($FFFF).
- The LRU stack will hold a default descending order of sectors.
- All cache sectors are in the unlocked state.

The execution of PFLUSHUN causes a flush only to the unlocked sectors and brings the cache to the following initial condition:

- All valid bits of the unlocked sectors will be cleared.
- All tag registers of the unlocked sectors are initialized to hold the value $1FFFF ($FFFF).
- The LRU stack will hold a default descending order of sectors.

Note: Coherency between PRAM mode and CACHE mode is not supported by the Instruction Cache Controller. It is not possible to fill the cache while in PRAM mode, and use the contents after switching to CACHE mode. The cache is automatically flushed when switching from CACHE to PRAM mode.

Note: PFLUSH and PFLUSHUN are detected as illegal opcodes in PRAM Mode. Issuing these instructions when the cache is disabled will initiate the Illegal Interrupt. A distance of at least three instruction cycles (equivalent to three NOP instructions) should be maintained between an instruction that changes the value of the cache enable bit (CE) and one of the instructions PFLUSH and PFLUSHUN.

5.2.8 Sector Replacement Unit

When a sector miss occurs, a cache sector must be selected to contain the new memory sector. The Sector Replacement Unit (SRU) determines which sector would be flushed from the cache, by constantly monitoring the requested addresses and the sectors usage.

The sector replacement policy is to replace the Least Recently Used (LRU) sector.

The LRU stack status is effected only in Cache Enable Mode by instruction fetch operations and by execution of the PFLUSH, PLOCK and PUNLOCK instructions. Locked cache sectors continue to "move" up and down the LRU stack. This implies that when picking the Least Recently Used sector (the one at the bottom of the LRU stack), locked sectors are skipped.

When the cache is initialized to the reset condition, the LRU stack holds a default descending order of sectors, i.e. sector number 0 is the Most Recently Used and sector number 7 is the Least Recently Used.

5.2.9 Data Transfers to/from ICACHE Space

Data transfers to/from the program memory can be accomplished by the DMA or by software, using MOVE instructions.

5.2.9.1 DMA transfers

DMA transfers have no effect on the Tag Register File, Valid Bit Array and LRU Stack even when the cache is enabled.

When the cache is disabled, the instruction cache memory space is considered a part of the internal program memory space. DMA transfers to/from this space will be executed without any limitation.

When the cache is enabled, the instruction cache memory space is considered a part of the external program memory space. DMA transfers to/from this space will be executed through the external memory expansion port. Coherency between the external program memory and the contents of the instruction cache is not maintained.

5.2.9.2 Software-Controlled transfers

The term "PMOVE" is used to indicate a MOVE instruction used to transfer data between the program memory space and any other source/destination. PMOVE transfers have no effect on the Tag Register File and LRU Stack even if the cache is enabled. The term "PMOVEW" is used to indicate a PMOVE transfer with the program memory space being the destination. The term "PMOVER" is used to indicate a PMOVE transfer with the program memory space being the source.

When the cache is disabled, the instruction cache memory space is considered a part of the internal program memory space. PMOVER from this space or PMOVEW to this space

will be executed without any limitation.

When the cache is enabled, PMOVER transfers are checked for a HIT or MISS:

- If the cache controller generates a HIT on the program memory space address, the data will be read from the cache memory array. Since PMOVE is not considered an instruction fetch operation, the LRU state will not be changed due to this transfer.

- If the cache controller generates a MISS on the program memory space address, the data will be read from the external program memory causing the amount of wait states as specified in the bus control register of the memory expansion port. The Cache state will not be changed due to this transfer. When in the Burst Mode, no burst will be initiated.

When the cache is enabled, PMOVEW transfers are also checked for a HIT or MISS:

- If the cache controller generates a SECTOR-HIT on the program memory space address, the data will be written both to the cache memory array and to the external program memory causing the amount of wait states as specified in the bus control register of the memory expansion port. The valid bit of the word will be set.The LRU stack will not be changed due to this transfer.

- If the cache controller generates a SECTOR-MISS on the program memory space address, the data will be written only to the external program memory causing the amount of wait states as specified in the bus control register of the memory expansion port. The Cache state will not be changed due to this transfer. When in the Burst Mode, no burst will be initiated.

WARNING

For proper operation, none of the three instructions before a PMOVE transfer should clear or set the Cache-Enable bit in the Status Register.

5.2.10 Cache Observability Via OnCE

The user is supplied with full non-intrusive system debug capability when in cache mode, having the ability to observe the cache status:

- what are the memory sectors that are currently mapped into cache

- which cache sectors are locked

- which cache sector is the Least Recently Used

- indication on the occurrence of HIT

This is accomplished in the debug mode, by reading the tag registers contents, lock bits, LRU bits and hit-status serially via the OnCE.

It is also possible to read the valid bits of specific cache locations. To check, whether an address, which MSBs are in a tag register, is in the cache, one should send the opcode of a MOVEM from this address. The bit 5 of OSCR will indicate the value of the valid bit.

- Each read of the cache status via the OnCE should access all the 9 registers, so that such a read starts every time from the tag #0.
- Each read of the cache status via the OnCE should be made only when the chip is in the debug mode of operation.