5. *Secondary Cache Interface*

The processor supports a mandatory secondary cache by providing an internal secondary cache controller with a dedicated secondary cache port.

The cache's tag and data arrays each consist of an external bank of industry-standard synchronous SRAM (SSRAM). This SSRAM must have registered inputs and outputs, asynchronous output enables, and use the late write protocol (data is expected one cycle after the address).

5.1 Tag and Data Arrays

The secondary cache consists of a 138-bit wide data array (128 data bits + 9 ECC bits + 1 parity bit) and a 33-bit wide tag array (26 tag bits + 7 ECC bits), as shown in Figure 5-1. ECC is supported for both the data and tag arrays to improve data integrity.

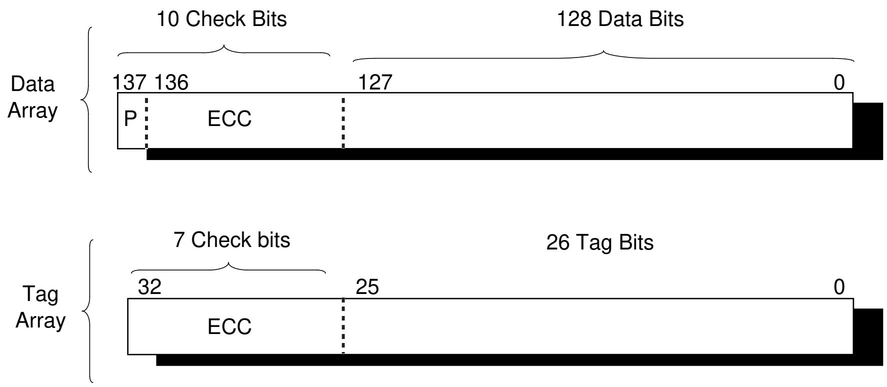

Figure 5-1 Secondary Cache Data and Tag Array

The secondary cache is implemented as a two-way set associative, combined instruction/data cache, which is physically addressed and physically tagged, as described in Chapter 4, the section titled "Cache Organization and Coherency."

The **SCSize** mode bits specify the secondary cache size; minimum secondary cache size is 512 Kbytes and the maximum secondary cache size is 16 Mbytes, in power of 2 (512 Kbytes, 1 Mbyte, 2 Mbytes, etc.). (See page 54 in Errata.)

The **SCBlkSize** mode bit specifies the secondary cache block size. When negated, the block size is 16 words, and when asserted, the block size is 32 words.

5.2 Secondary Cache Interface Frequencies

The secondary cache interface operates at the frequency of **SCClk**, which is derived from **PClk**. The **SCClkDiv** mode bits select a **PClk** to **SCClk** divisor of 1, 1.5, 2, 2.5, or 3, using the formula described in Chapter 7, the section titled "Secondary Cache Clock."

Synchronization between the **PClk** and **SCClk** is performed internally and is invisible to the system. The processor supplies six complementary copies of the secondary cache clock on **SCClk(5:0)** and **SCClk(5:0)***.

5.3 Secondary Cache Indexing

The secondary cache data array width is one quadword, and therefore **PA(3:0)**, which specify a byte within a quadword, are unused by the Secondary Cache interface.

Indexing the Data Array

Since the maximum secondary cache size is 16 Mbytes (8 Mbytes per way), each way requires a maximum of 23 bits to index a byte within a selected way, or 19 bits to index a quadword within a way. Consequently, the processor supplies **PA(22:4)** on **SC(A,B)Addr(18:0)** to index a quadword within a way. The processor selects a secondary cache data way with the **SC(A,B)DWay** signal.

Table 5-1 presents the secondary cache data array index for each secondary cache size; for instance, a 4 Mbyte cache uses the 17 address bits, **PA(20:4)** on **SC(A,B)Addr(16:0)**, concatenated with the way bit, **SC(A,B)DWay**, to index a quadword within a 2 Mbyte way.

Table 5-1 Secondary Cache Data Array Index

SCSize Mode Bits	Secondary Cache Size	Secondary Cache Data Array Index	Physical Address Bits Used
0	512 Kbyte	**SC(A,B)DWay || SC(A,B)Addr(13:0)**	**PA(17:4)**
1	1 Mbyte	**SC(A,B)DWay || SC(A,B)Addr(14:0)**	**PA(18:4)**
2	2 Mbyte	**SC(A,B)DWay || SC(A,B)Addr(15:0)**	**PA(19:4)**
3	4 Mbyte	**SC(A,B)DWay || SC(A,B)Addr(16:0)**	**PA(20:4)**
4	8 Mbyte	**SC(A,B)DWay || SC(A,B)Addr(17:0)**	**PA(21:4)**
5	16 Mbyte	**SC(A,B)DWay || SC(A,B)Addr(18:0)**	**PA(22:4)**

Indexing the Tag Array

The processor supplies the secondary cache tag array's least significant index bit on **SCTagLSBAddr** to support two block sizes without system hardware changes. This signal functions normally as a least significant index bit when the secondary cache block size is 16 words. However, when the secondary cache block size is 32 words, this signal is always negated, since only half as many tags are required. The processor supplies the secondary cache tag way on **SCTWay**.

Table 5-2 presents the secondary cache tag array index for each secondary cache size; it shows each index is composed of a physical address loaded onto **SC(A,B)Addr()**, concatenated with **SCTWay** and **SCTagLSBAddr**.

Table 5-2 Secondary Cache Tag Array Index

SCSize Mode Bits	Secondary Cache Size	Secondary Cache Tag Array Index
0	512 Kbyte	**SCTWay || SC(A,B)Addr(13:3) || SCTagLSBAddr**
1	1 Mbyte	**SCTWay || SC(A,B)Addr(14:3) || SCTagLSBAddr**
2	2 Mbyte	**SCTWay || SC(A,B)Addr(15:3) || SCTagLSBAddr**
3	4 Mbyte	**SCTWay || SC(A,B)Addr(16:3) || SCTagLSBAddr**
4	8 Mbyte	**SCTWay || SC(A,B)Addr(17:3) || SCTagLSBAddr**
5	16 Mbyte	**SCTWay || SC(A,B)Addr(18:3) || SCTagLSBAddr**

For a system design that only supports a secondary cache block size of 32 words, the secondary cache tag array need not use **SCTagLSBAddr** as an index bit.

5.4 Secondary Cache Way Prediction Table

The primary and secondary caches are two-way set associative. However, the implementation of the secondary cache is different than the primary caches.

The primary caches read simultaneously from two separate tag arrays, corresponding to each way in the cache, and then select the data based on the result of two parallel tag compares.

The secondary cache does not use this implementation because it would either require too many pins to read in two full copies of the data and tags, or add latency to externally multiplex two banks of memory. Instead, a way prediction table is used to determine which way to read from first.

The way prediction table is internal to the processor and has 8K one-bit entries, each entry corresponding to a pair of secondary cache blocks. The bit entry indicates which way of the addressed set has been most-recently used (MRU). When the secondary cache is accessed, this prediction bit is used as an address bit; thus the two ways in the secondary cache are shared in the same SSRAM bank.

The secondary cache way prediction table is indexed with a subset of 11 to 13 bits of the physical address, based on both the secondary cache block size, and the secondary cache size, as shown in Table 5-3. “0 || ” indicates a zero bit concatenated to the address to pad the index out to a full 13-bits.

Table 5-3 Secondary Cache Way Prediction Table Index

SCSize Mode Bits	Secondary Cache Size	SCBlkSize Mode Bit	Secondary Cache Block Size	Secondary Cache Way Prediction Table Index
0	512 Kbyte	0	16-word	0 || PA(17:6)
		1	32-word	0 || 0 || PA(17:7)
1	1 Mbyte	0	16-word	PA(18:6)
		1	32-word	0 || PA(18:7)
2 to 5	2M to 16 Mbyte	0	16-word	PA(18:6)
		1	32-word	PA(19:7)

Three states are possible in the way prediction table:

- the desired data is in the predicted way
- the desired data is in the non-predicted way
- the desired data is not in the secondary cache

The tags for both ways are read "underneath" the data access cycles in order to discern as rapidly as possible which of these states are valid. This reading is possible because it takes two accesses to read a primary data block (8 words) and 4 cycles to read a primary instruction block (16 words); thus the bandwidth needed to read the tag array twice exists in all cases. Only an extra address pin to the tag array is needed to make this operation parallel and this is implemented by the **SCTWay** pin.

The three possible states are handled in the following manner:

- If, after reading the tags for both ways, it is discovered that the data exists in the predicted way, the processor continues normally.
- If the data exists in the non-predicted way, the processor accesses this non-predicted way in the secondary cache and updates the way prediction table to point to this way.
- If the access misses in both ways of the secondary cache, the data is fetched from the system interface. If the state of the predicted way is found to be *invalid*, the fetched data is placed in it and the MRU is unchanged. However, if the state of the predicted way is found to be *valid* then the fetched data is placed into the non-predicted way, and the way prediction table is updated to point to this way since it is now the most-recently-used. (See page 58 in Errata.)

The way prediction table can cover up to a 2 Mbyte secondary cache when the secondary cache block size is 32 words. If the secondary cache exceeds this size, the accuracy of the way prediction table diminishes slightly. However, the extremely large performance gain made by making the secondary cache larger far outstrips any performance loss in the way prediction table.

5.5 Secondary Cache Tag

The secondary cache tag, transferred on the **SCTag(25:0)** bus, is divided into three fields, as shown in Figure 5-2 below.

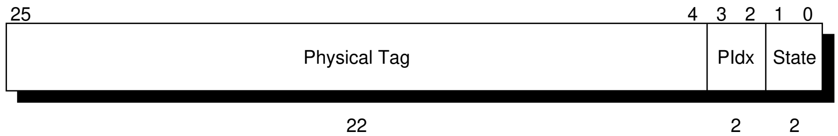

Figure 5-2 Secondary Cache Tag Fields

SCTag(25:4), Physical Tag

The minimum secondary cache size is 512 Kbytes (256 Kbytes per way), so a minimum of 18 bits are required to index a data byte within a selected way. Since the processor supports 40 physical bits, a maximum of 22 bits are required for the physical tag:

```
40 physical address bits - 18 minimum required = 22
```

Consequently, the processor supplies the 22 physical address bits, **PA(39:18)**, on **SCTag(25:4)** for the physical tag.

When the secondary cache is larger than the minimum size, the secondary cache tag array must still maintain the full physical tag supplied by the processor, even though some bits are redundant.

SCTag(3:2), PIdx

Bits **SCTag(3:2)** of the secondary cache tag contain the primary cache index, *PIdx*.

The *PIdx* field contains **VA(13:12)**, which are the two lowest virtual address bits above the minimum 4 Kbyte page size. This field is written into the secondary cache tag during a secondary cache refill. For each processor-initiated secondary cache access, the virtual address bits are compared with the *PIdx* field of the secondary cache tag. If a mismatch occurs, a virtual coherency condition exists and the value of the *PIdx* field is used by internal control logic to purge primary cache locations, so that all primary cache blocks holding valid data have indices known to the secondary cache. This mechanism, unlike that of the R4400 processor, is implemented in hardware. It helps preserve the integrity of cached accesses to a physical address using different virtual addresses, an occurrence called **virtual aliasing**. For each external coherency request, the *PIdx* field of the secondary cache tag provides a mechanism to locate subset lines in the primary caches.

SCTag(1:0), Cache Block State

The lower two bits of the secondary cache tag, **SCTag(1:0)**, contain the cache block state, which can be *Invalid*, *Shared*, *CleanExclusive*, or *DirtyExclusive* as shown in Table 5-4.

Table 5-4 Secondary Cache Tag State Field Encoding

SCTag(1:0)	State
0	*Invalid*
1	*Shared*
2	*CleanExclusive*
3	*DirtyExclusive*

Since the secondary cache tags are updated immediately for stores to the primary data cache, and all caches use a write back protocol, the data in the secondary cache may not always be consistent with data in the primary cache even though the tags always reflect the correct state of a secondary cache block.

5.6 Read Sequences

There are five basic read sequences:

- a 4-word read
- an 8-word read
- a 16-word read
- a 32-word read
- a tag read

4-Word Read Sequence

A 4-word read sequence is performed by a CACHE Index Load Data (S) instruction to read a doubleword of data and 10 check bits from the secondary cache data array.

Figure 5-3 depicts a secondary cache 4-word read sequence. A quadword is read from the index specified by **PA(23:6)**, and the way specified by **VA(0)** of the CACHE instruction.

The doubleword specified by **VA(3)** is then stored into the CP0 *TagHi* and *TagLo* registers, and the corresponding check bits are stored into the CP0 *ECC(9:0)* register. The data may be examined by copying the CP0 *TagHi*, *TagLo*, and *ECC* registers to the general registers with the MTC0 instruction.

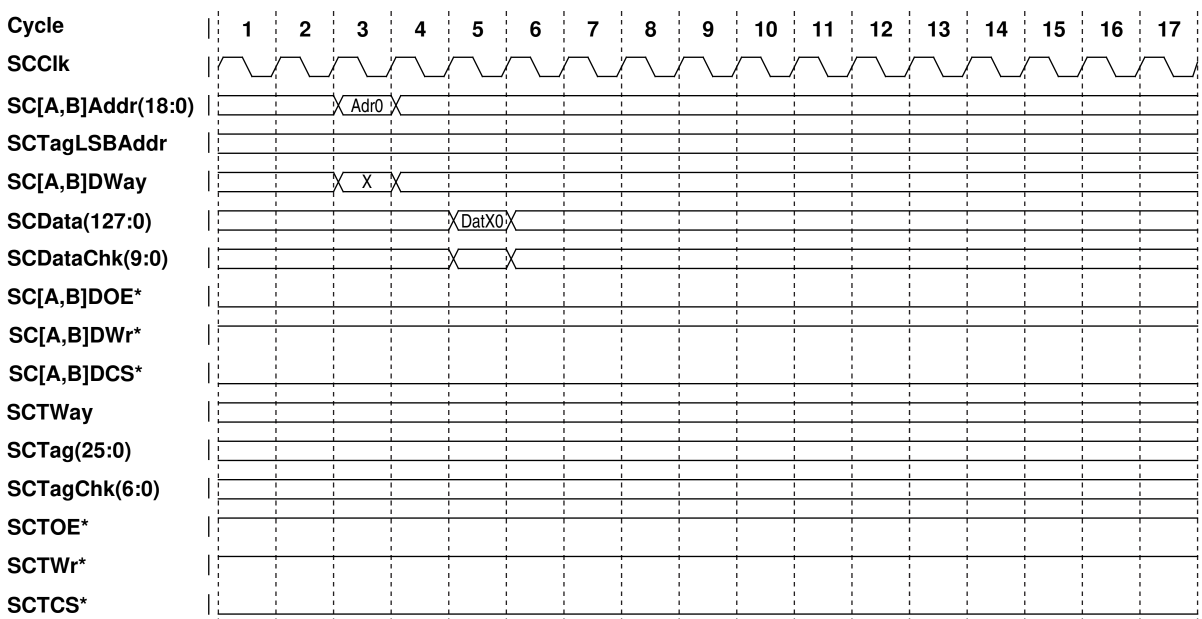

Figure 5-3 4-Word Read Sequence

8-Word Read Sequence

An 8-word read sequence refills the primary data cache from the secondary cache after a primary data cache miss.

Figure 5-4 depicts a secondary cache 8-word read sequence. In it, **SC(A,B)DWay** and **SCTWay** are driven with value X on the first address cycle, which is obtained from the way prediction table.

On the next address cycle, **SCTWay** is complemented in order to read the tag from the non-predicted way of the addressed set. **SC(A,B)DWay** is not changed since it is assumed that the way prediction table is correct and the read is likely to hit in the predicted way.

The tag for the non-predicted way is returned to the processor in the same cycle as the second quadword of data. Reads that miss in the predicted way, but hit in the non-predicted way, are noted by the internal control logic and reissued to the secondary cache as soon as possible.

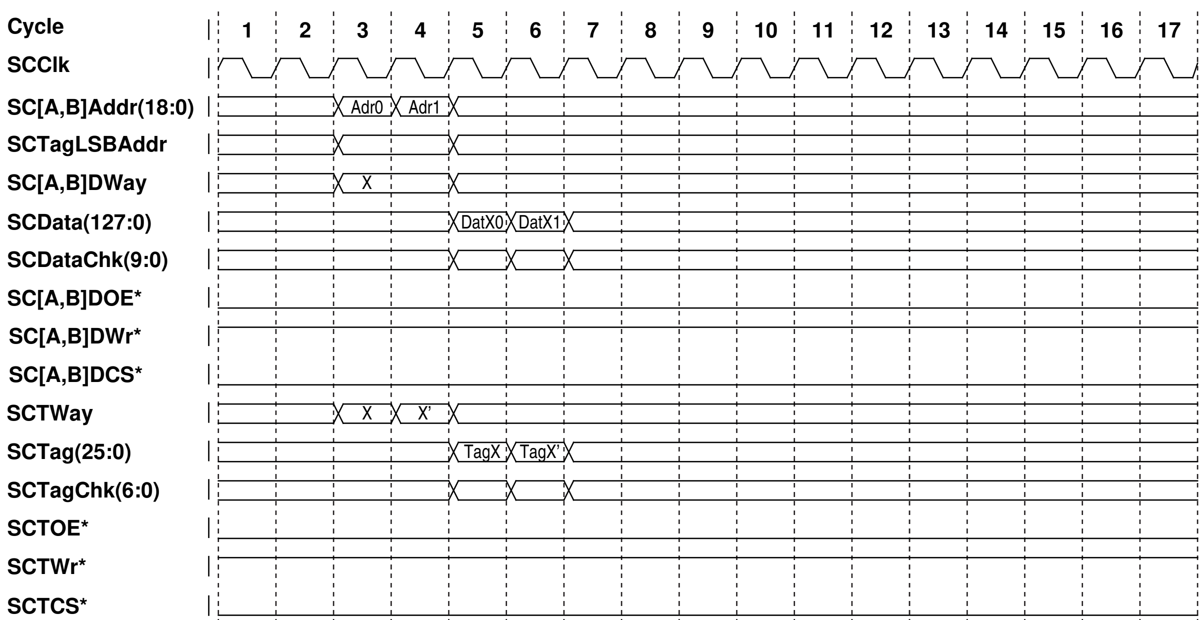

Figure 5-4 8-Word Read Sequence

16 or 32-Word Read Sequence

A 16-word read sequence refills the primary instruction cache from the secondary cache after a primary instruction cache miss. A 16-word read sequence is also performed when the secondary cache block size is 16 words, and a *DirtyExclusive* secondary cache block must be written back to the System interface.

A 32-word read sequence is performed when the secondary cache block size is 32 words, and a *DirtyExclusive* secondary cache block must be written back to the System interface.

Figure 5-5 depicts a secondary cache 16 or 32-word read sequence. This is similar to an 8-word read sequence except that more addresses must be issued, in order to read the appropriate number of quadwords.

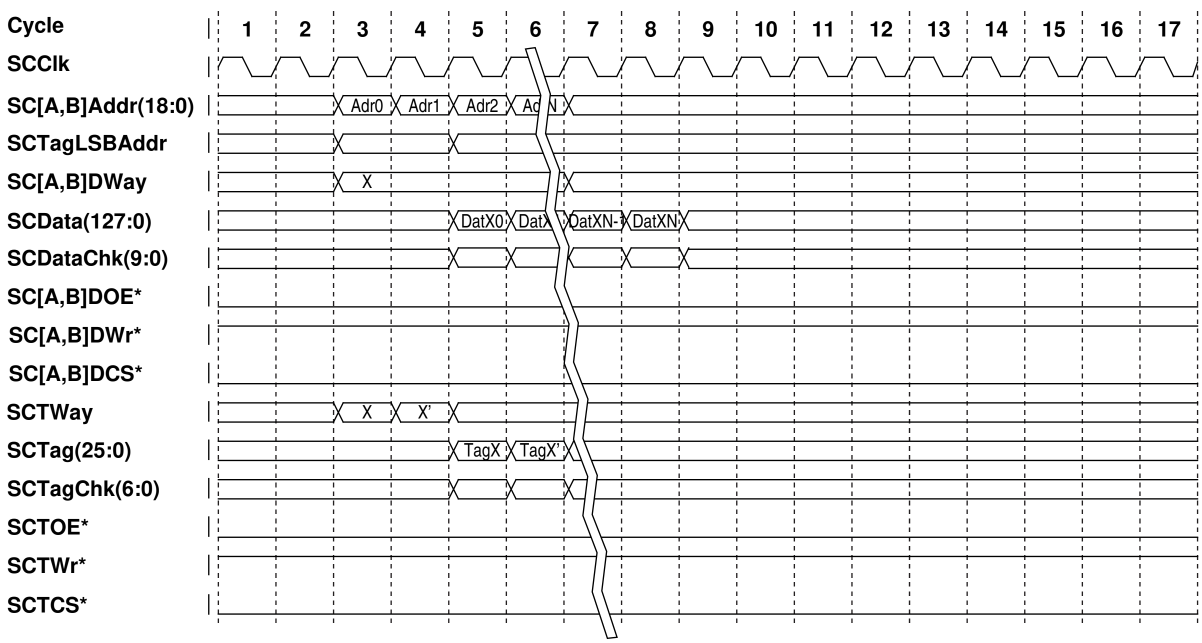

Figure 5-5 16 or 32-Word Read Sequence

Tag Read Sequence

A tag read sequence is performed when the state of a secondary cache block is required, but it is not necessary to access the data array. This sequence is used for the CACHE Index Load Tag (S) instruction.

Figure 5-6 depicts a secondary cache tag read sequence.

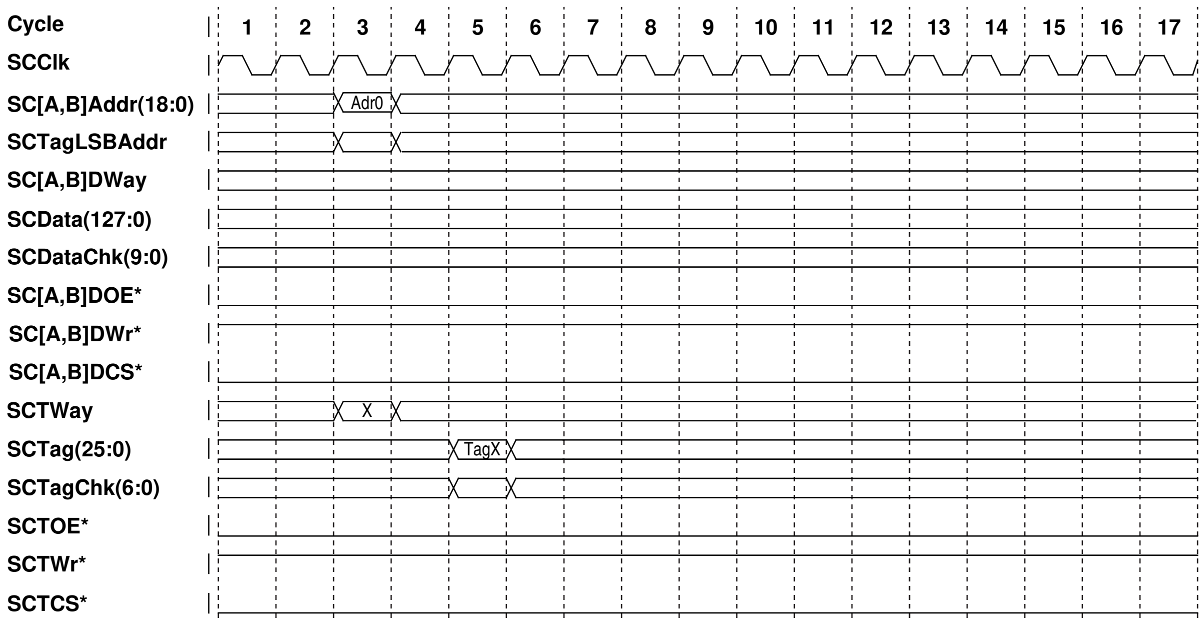

Figure 5-6 Tag Read Sequence

5.7 Write Sequences

There are five basic write sequences:

- a 4-word write.
- an 8-word write
- a 16-word write
- a 32-word write
- a tag write

4-Word Write Sequence

A 4-word write sequence is performed by a CACHE Index Store Data (S) instruction to store a quadword of data and 10 check bits into the secondary cache data array.

Figure 5-7 depicts a secondary cache 4-word write sequence. A quadword is written to the index specified by **PA(23:6)**, and the way specified by **VA(0)** of the CACHE instruction.

A doubleword specified by **VA(3)** is obtained from the CP0 *TagHi* and *TagLo* registers, and the other half of the doubleword is padded to zeros. Normal ECC and parity generation is bypassed and the check field of the data array is written with the contents of the CP0 *ECC(9:0)* register.

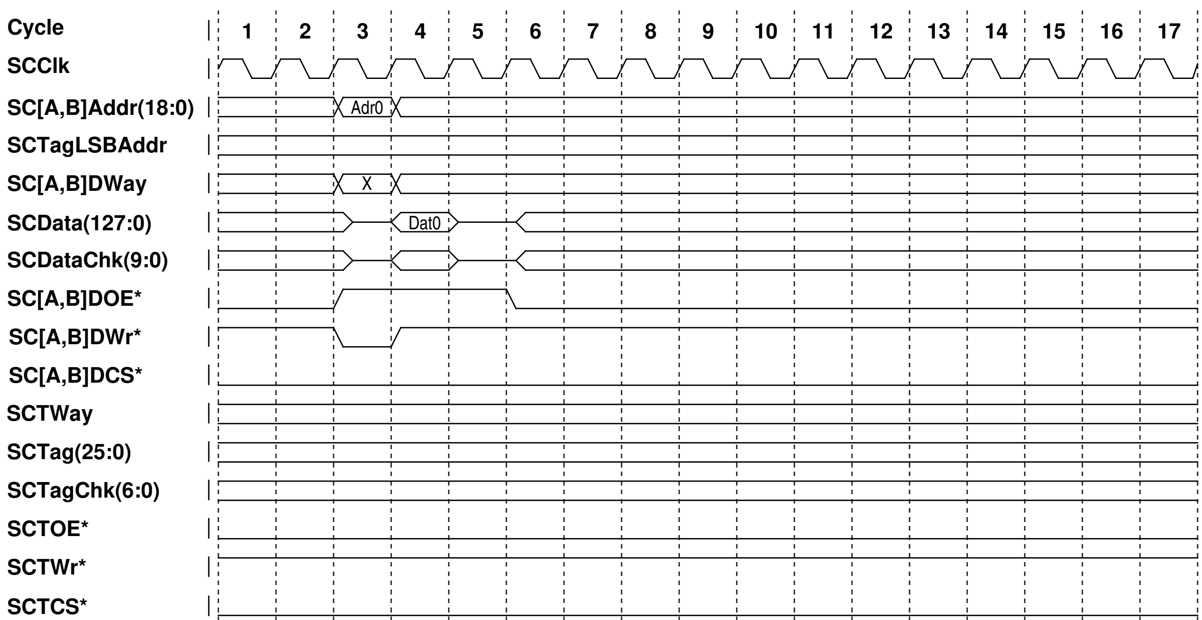

Figure 5-7 4-Word Write Sequence

8-Word Write Sequence

An 8-word write sequence writes back a dirty block from the primary data cache to the secondary cache.

Figure 5-8 depicts a secondary cache 8-word write sequence. **SC(A,B)DWay** are driven with the way bit obtained from the primary data cache tag. The secondary cache tag is not written since it was previously updated when the primary data cache block was modified.

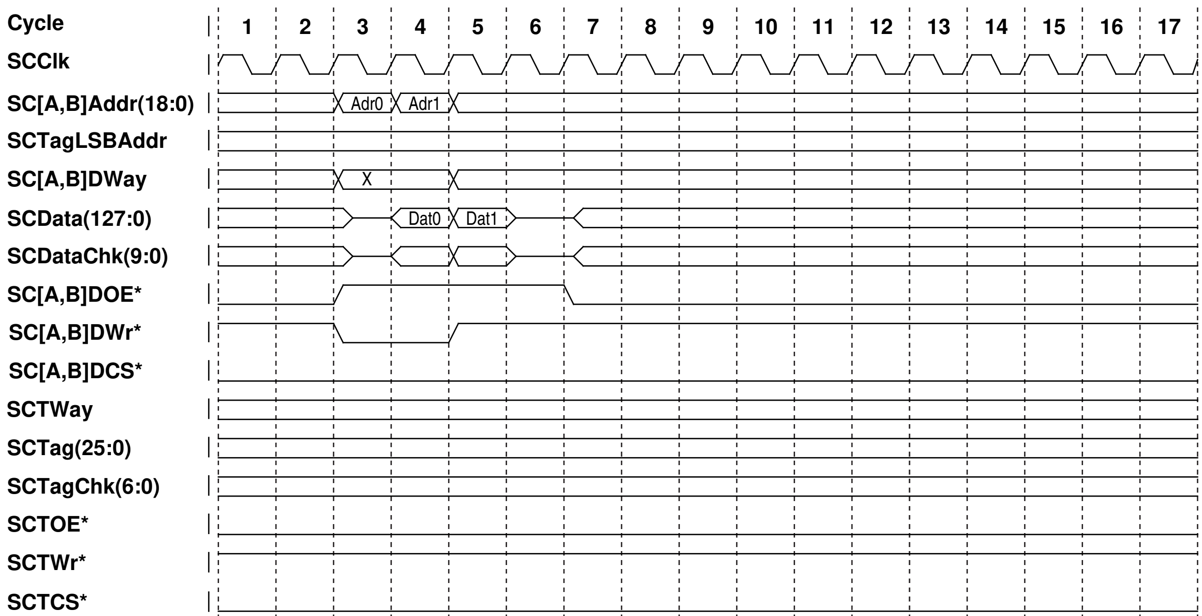

Figure 5-8 8-Word Write Sequence

16 or 32-Word Write Sequence

A 16- or 32-word write sequence refills a secondary cache block from the System interface after a secondary cache miss. A 16-word write sequence is performed when the secondary cache block size is 16 words, and a 32-word write sequence is performed when the secondary cache block size is 32 words.

Figure 5-9 depicts a secondary cache 16 or 32-word write sequence.

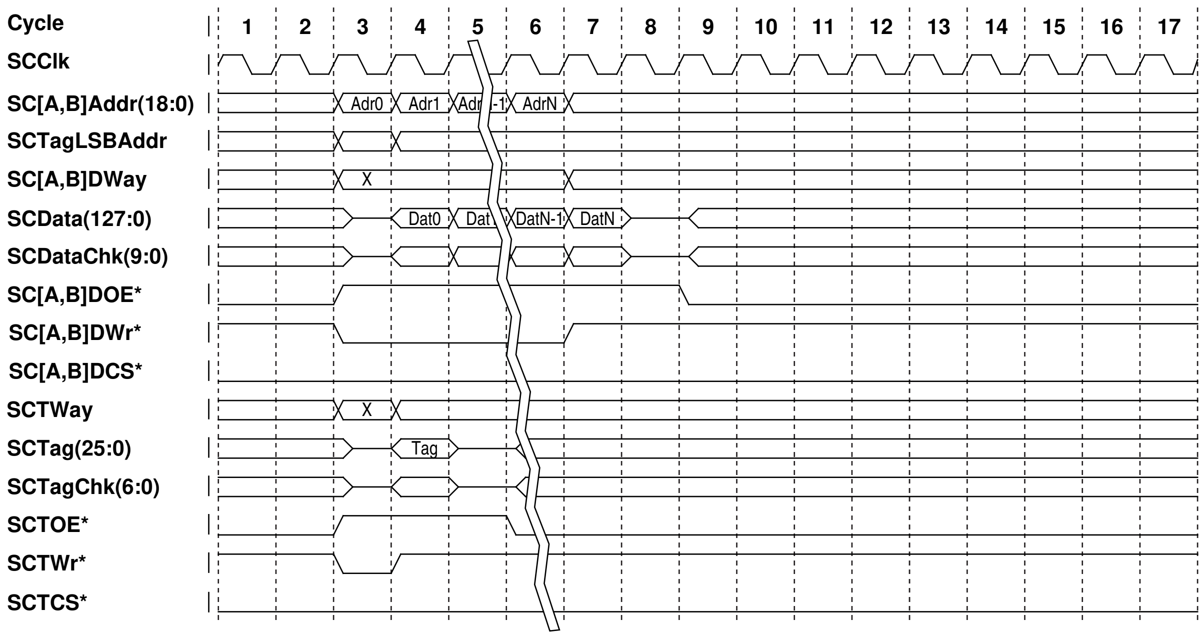

Figure 5-9 16/32-Word Write Sequence

Tag Write Sequence

A tag write sequence updates the secondary cache tag array without affecting the data array. This sequence is used for the following:

- to reflect primary cache state changes in the secondary cache
- for external coherency requests
- for the CACHE Index Store Tag (S) instruction

Figure 5-10 depicts the secondary cache tag write protocol.

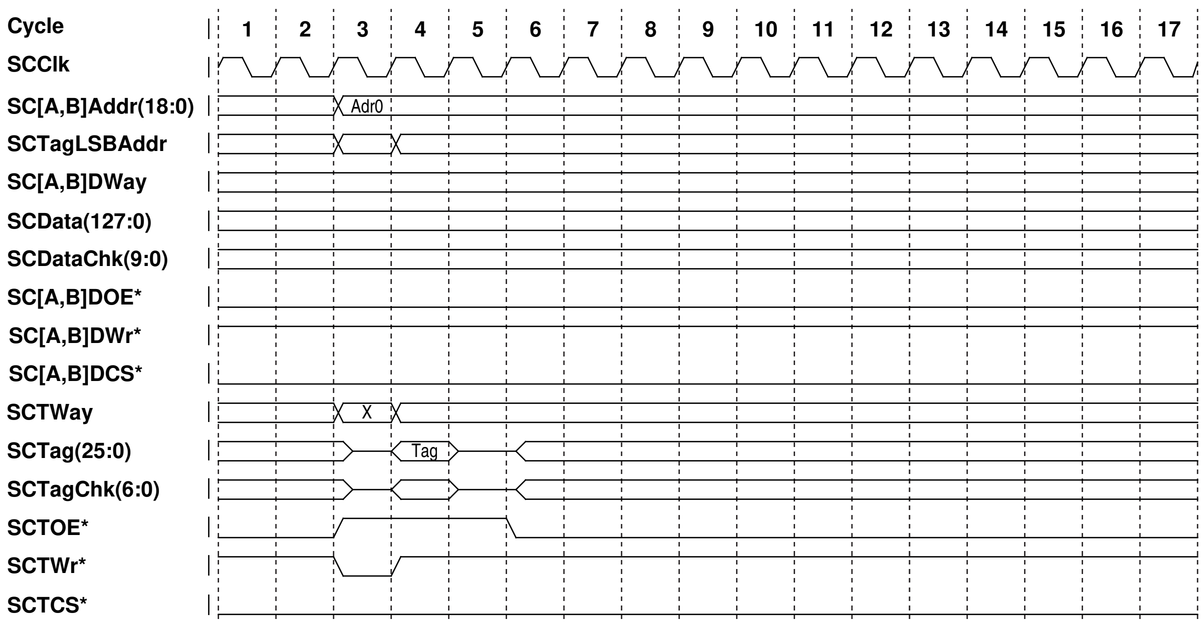

Figure 5-10 Tag Write Sequence