SSS Memory Coherence Proposal

Timothy Knight

July 1, 2002

1 Introduction

Several different memories in an SSS node can contain copies of global memory, specifically:

- SRF: indexed via stream instructions.
- Stream cache: physically addressed.
- Scalar cache: virtually addressed.

This document describes a proposal for memory coherence in the SSS.

2 Cacheable Segments

Each segment, as configured by a segment register, can be marked as one of the following:

- Non-cacheable.
- Cacheable by the stream cache.
- Cacheable by the scalar cache.
- Read-only; a read-only segment will be cached in the scalar cache, and may also be cached in the stream cache, depending upon the stream instructions issued.

3 Scalar Cache Coherence

The scalar cache is restricted to caching only addresses local to its node, and it is coherent with respect to those addresses. Specifically:

- If a either a remote node or the stream unit on the current node write to an address which is contained in the scalar cache, the scalar cache is updated with the newly written value.
- If either a remote node or the stream unit on the current node read from an address which is contained in the scalar cache, the data is read from the scalar cache instead of the memory.

Note that the scalar cache is virtually addressed, while the requests from remote nodes are for physical addresses; to handle this, there is a reverse address translation unit between the scalar cache and the memory system.
4 Stream Cache Coherence

The stream cache is non-coherent. It supports the following explicit stream instructions:

- Load/store with or without using the stream cache, in read-only or write-back mode. The stream cache supports write-allocate on miss.
- Gang-invalidation of read-only data in the stream cache.
- Flushing of write-back data in the stream cache. A flush-no-invalidate instruction is supported in addition to the usual flush operation.

5 Scatter-Add

The SSS node supports a scatter-add feature in which both ‘store’ and ‘add-and-store’ requests can be issued. The following are the specifications of the scatter-add feature:

- An add-and-store request is processed atomically by the hardware, using dedicated 64-bit floating point adders in the memory system.
- Only the stream unit in each node may issue add-and-store requests; the scalar processor is restricted to issuing only store requests.
- Neither the scalar cache nor the stream cache support add-and-store; thus, to ensure correct behaviour, the program should only issue add-and-store requests to addresses in segments marked as non-cacheable. However, add-and-store requests may actually be issued to any segment which is not read-only. In the case of cacheable segments the program must take appropriate precautions to guarantee that the address which is the subject of the add-and-store request is not contained in either cache at the time that the request is issued.

6 Memory Locking

The memory system supports a memory-locking feature to enable atomic scalar operations, in which a small table contains a list of addresses which are locked. Only addresses in segments marked as either non-cacheable or cacheable by the scalar cache may be locked.

A lock can be either a write-lock, in which only the scalar processor local to the locked address may write it, or a read-write-lock, in which only the scalar processor local to the locked address may read or write it. If a remote node issues a request which is prevented by the lock, the request will fail.

Note that the stream unit local to the locked address is also prevented from accessing the address while it is locked, just as the stream units on remote nodes are locked out; thus, the scalar processor should only lock addresses which are not the subject of stream operations during the period of the lock.