Linearization Protocol

Detailed Protocol Specification

N. A. Oswald

1 Introduction

This document provides a detailed description of the Linearization Protocol.

2 Controller

The following summarizes the notation and assumptions used in the transition table:

- Format of receive message action: ?Message
- Format of send message action: destination!Message
  If no destination is given the action corresponds to a broadcast.
- If a message that corresponds to a blank field arrives at the controller it is discarded.
- Format of information taken from received message: m.Information
  e.g. Message source: m.src, Message timestamp: m.TS
- Format of information taken from cache: b.Information
  e.g. Local timestamp: b.TS, Cache identifier: b.ID
- Maximum possible timestamp TS value: $\mathsf{TS}_{\mathsf{max}}$
  If the local TS of a cache equals $\mathsf{TS}_{\mathsf{max}}$ it is reset.
- Transition into next State: $\rightarrow$ State

3 Interconnect

3.1 Property

The Linearization Protocol can be used together with an ordered or unordered interconnect. In an unordered interconnect all caches can theoretically become invalid if a timestamp overflow occurs and an invalidation message still exists in the network. To avoid timestamp counter overflows, sufficiently large timestamps must be chosen.

3.2 Simulation Model

In the simulation a single unordered network is used. The Linearization Protocol does not require the implementation of virtual channels to be free of deadlocks. There exists no need to separately simulate possible FIFO buffers, because the network is chosen to be unordered. In case of a timestamp counter overflow, all counters are reset to zero after the last message within the network has been served. This corresponds to a global system reset.

4 Model Checking

4.1 System Requirements

The provided model can be simulated using the Murphi model checker. After installing Murphi, the path in the provided Makefile must be updated. Afterwards the provided Murphi code can be build using make. The output of the created executable is the model checking report of the Linearization Protocol.

4.2 Simulation Model Modifications

In the simulation model, the number of processors, the number of addresses, the maximum size for the timestamp TS and the network buffer size can be set. The corresponding constants can be found at the start of the provided Murphi file.

4.3 SWMR Invariant

The Single-Writer-Multiple-Reader (SWMR) invariant must be maintained. If a write becomes globally visible through a cache transitioning into the V (Valid) state and broadcasting the Update, then all other caches must be in the I (Invalid) state waiting for updates.

4.4 Cache Consistency Violation Invariant

If a cache is in the V state it must also have the most recent TS value. The most recent TS is stored in a global variable. If a cache transitions from the W (Write) state to the V state it updates the global TS variable.

4.5 All Caches Invalid Invariant

At any given time there must exist a cache, which state is different from the I (Invalid) state.

4.6 Deadlocks

The Murphi model checking tool checks the protocol by default for deadlocks. It is important to remove messages that cannot be served in a specific state immediately from the input buffer, as otherwise a deadlock would exist.

Table 1: Controller

State	Read	Write	?Inv(TS,ID)	?InvAck(TS)	?Update(TS,ID,Data)

V	hit	b.TS++;
		b.LID=b.ID;
		!Inv(b.TS,b.ID);
		→ W^A
			b.TS=m.TS;
			b.LID=m.ID;
			m.src!InvAck(b.TS);
			→ I^U		

I^U	stall	stall	if (m.TS>b.TS |
			(m.TS=b.TS & m.ID>b.LID)) {
			b.TS=m.TS;
			b.LID=m.ID;
			m.src!InvAck(b.TS)
			}
					if (m.TS=b.TS & m.ID=p.LID) {
					if (m.TS=TS_{max}) {
					b.TS=0
					}
					→ V
					}

W^A	stall	stall	if (m.TS>b.TS |
			(m.TS=b.TS & m.ID>b.LID)) {
			b.TS=m.TS;
			b.LID=m.ID;
			reset b.acnt;
			m.src!InvAck(b.TS);
			→ I^U
			}
				b.acnt++;
				if (b.acnt=N_{Caches}){
				!Update(b.TS,b.ID,Data);
				reset b.acnt;
				→ V
				}	