Design and Verification of a Multichip Coherence Protocol

Shahid Ikram, Isam Akkawi, Richard Kessler, Jim Ellis, David Asher
Cavium, Inc., 600 Nickerson Road, Marlborough, MA 01752.

Abstract

This paper describes an industrial experience of the development of a multichip coherence interconnect protocol (OCI). The protocol specification was formulated as extended state transition tables. We used formal as well as simulation tools in different protocol development's phases like discovery, design, modeling, validation and maintenance. A number of design bugs as well as holes (missing features) were found in these phases. Furthermore, the protocol models were used to validate micro-architectural protocols and constraints. The tools and processes developed were employed in the design of two major multi-billion transistors chips and helped us in achieving a quicker convergence of verification work by saving many months of effort.

Overview

Objectives:
1. A robust set of of high-level specifications
2. Tape-out in time

Challenges:
1. Limited time to tape-out
2. Limited resources in terms of man-power, software licenses, compute power
3. Complete proof of the protocol correctness not possible in given time

Solutions:
1. A compromise between formal verification and simulation
2. Common specification for formal verification and simulations
3. Fast design revisions loop

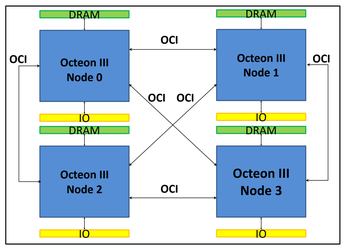
Results:
1. Number of protocol bugs found and fixed
2. No protocol errors found during RTL regressions
3. Protocol was operational on first silicon samples

Methodology

1. Identify the components of the protocol
2. Create specifications of each of the protocol components
3. Identify the services provided by each of the components
4. Model the protocol
5. Create a comprehensive set of correctness and completeness properties
6. Create an automated flow of protocol validation that supports revisions

The OCI Protocol

1. Multi-chip L2
2. Cache Coherence
3. Directory-based



Protocol Specifications

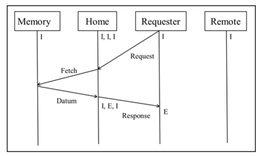
Home Table

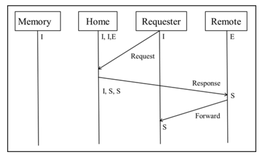
Current State					Next State					Outputs			Inputs
Cmd	**H**	**N1**	**N2**	**N3**	**Cmd**	**H**	**N1**	**N2**	**N3**	**N1**	**N2**	**N3**	**Req/Resp/Frwd**
none	E	I	I	I	none	S	S	I	I	RD_RSP	-	-	OCI_RD (Read from remote node)
none	S	S	I	I	LCL_WR	S	S->I	I	I	INVAL	-	-	LCL_WR_LCL_A (local write to home address)
LCL_WR	S	S->I	I	I	none	M	I	I	I	-	-	-	INV_RSP (Invalidate response)

Remote Table for Node 1

Current State		Next State		Outputs			Inputs
Cmd	**St**	**Cmd**	**St**	**H**	**N2**	**N3**	**Req/Resp/Frwd**
none	I	OCI_RD	I	OCI_RD (Read from remote node – i.e. home)	-	-	LCL_RD_RMT_A (local read – remote address)
OCI_RD	I	none	S	-	-	-	RD_RSP (read response)
none	S	none	I	INV_RSP (Invalidate Response)	-	-	INV (Invalidate)

Protocol Transactions



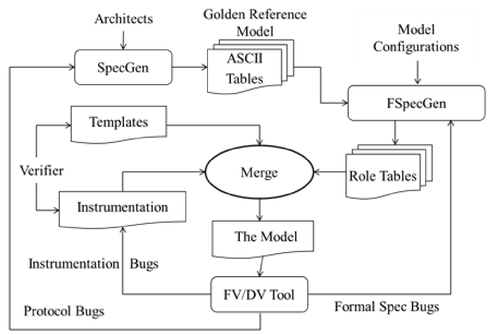


Protocol Properties

Completeness
- Under-Specification
- Over-Specification
- Legal States Reachability
- Transactions' Reachability
- Missing/hidden assumptions

Correctness
- Deadlock freedom
- Cache coherence
- Data consistency
- Designer properties

Validation Flow



Modeling OCI

For a given address,
- Two types of transition table, home and remote.
- Six unique roles/services.

```
//The Polling Loop, for a given address
start:                          //Starting point of the polling loop
                                //Choose randomly among
available choices.
case(true)
 (Home node has no pending request):   goto LocalAddressRequest;
(Remote node has no pending request):  goto RemoteAddressRequest;
(There is a request from a remote):    goto HomeReceiveRequest;
(There is a response message for remote):  goto RemoteReceiveResponse;

(There is a forwarding message for remote): goto RemoteReceiveForwardRequest;
(There is response message for home): goto HomeReceiveResponse;
Default:   goto deadlockchecker;    //No options for this address; It is a deadlock.
endcase
goto start;
```

Handling Complexity

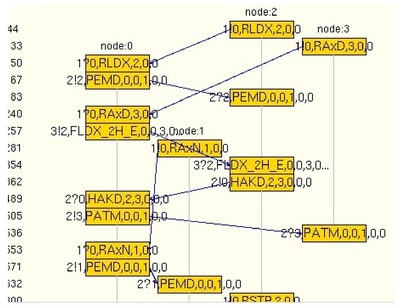
- Parameterization
- Address abstraction
- Initial state abstraction
- Configurations

Properties\Models	RLDD		RLDX		RLDD-RLDX		RLDD-RLDX-RLDT		
Nodes	3	4	3	4	3	4	2	3	4
Parallel (total/proved/fail)	6/6	6/6	6/6	6/6	6/6	6/5	6/6	6/6	6/6
Full(total/proved/fail)	6/6	6/3/1	6/6	6/5	6/6	6/1	6/6	6/3/3	6/2/1
Coverage (covered/ unreachable)	34/-	206/0	49/-	67/0	39/-	568/-	59/2333	871/30	268/26
RTG Good State	√	√	√	√	√	-	√	√	√
Cache Coherence	√	-	√	√	√	-	√	-	-
Deadlock	√	-	√	-	√	-	√	-	×
Total Time(before killed.)	1m	30m	5s	20m	15m	2h	15m	24h	14h

Validating OCI using Spin

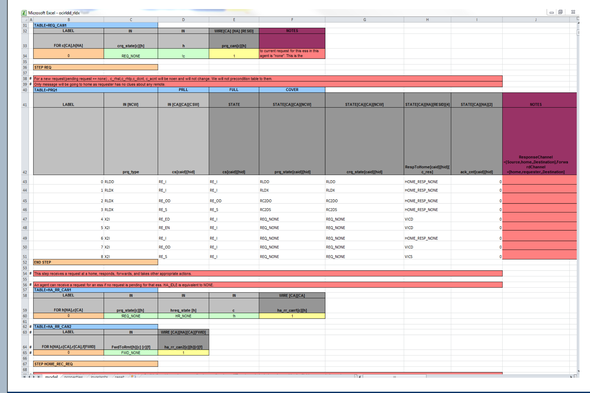
Spin model checker:
1. Continues development
2. Documentation
3. Excellent support from Gerard Holzmann

Spin modes of operation:
1. Interactive simulation
2. Formal verification
3. Random simulation



Validating OCI using JasperGold

Jasper Arch
1. A state of the art tool tailored for architectural validation
2. Host of engines and great debugging capabilities
3. Built in transition coverage, parallel and full assertions
4. Cut the validation time in half



Results

- Found missing actions in the protocol
- Found missing/hidden assumptions of the protocol
- Found missing transitions causing deadends
- Found unreachable transitions
- Proved that the protocol is cache coherent and data consistent
- Proved that the protocol is deadlock free

Bugs Categorization

Category	Percentage
Missing Cases/Dead-Ends	30%
Unreachable cases	30%
Deadlocks	10%
Missing Assumptions	10%
Data/Masking	10%
Miscellaneous	10%

Contact information

Dr. Shahid Ikram
Cavium, Inc.

600 Nickerson Road,
Marlborough, MA 01752

T: 5084796513
E: shahid.ikram@cavium.com