STE Based Verification of the Look-Aside Interface

Donglin Li
Department of Electrical and Computer Engineering, Concordia University
1455 de Maisonneuve, West Montreal, Québec, H3G 1M8, Canada
e-mail: li_don@ece.concordia.ca

Asif Iqbal Ahmed
Department of Electrical and Computer Engineering, Concordia University
1455 de Maisonneuve, West Montreal, Québec, H3G 1M8, Canada
e-mail: a_ahmed@ece.concordia.ca

Otmane Ait Mohamed
Department of Electrical and Computer Engineering, Concordia University
1455 de Maisonneuve, West Montreal, Québec, H3G 1M8, Canada
e-mail: ait@ece.concordia.ca

Abstract

In this paper we investigate the model checking of the Look-Aside Interface (LA-1 Standard) using STE, a model checking technique based on a form of quaternary symbolic simulation. The Look Aside interface is intended for devices located adjacent to a network processing device (NPE) that off load certain tasks from the network processor. For our verification purpose, we first extracted some safety properties of the LA-1 Interface based on its design specification. Then, we developed a synthesizable RTL model of the LA-1 Interface using Verilog, which was then translated to Exlif format in order to be accepted by the STE tool. Afterwards, we described those safety properties as STE assertions and performed the verification. Finally, to evaluate our approach, we verified the same properties using the RuleBase model checker from IBM.

Keywords *— Model checking, STE, Look-Aside Interface, Forte, RuleBase.*

1 Introduction

As the modern designs past one-million and move towards 10-million gates, the design-verification engineer require state-of-the art tools and methodology to verify these devices. Most of the case studies have shown that functional verification consumes more than 60 percent of the design-cycle time. In order to confront this issue, a number of innovative ideas have emerged. For instance, the use of model checking in order to verify designers assumptions i.e. safety properties and check for functional errors in RTL code is one such idea. Nowadays, the practice of model checking is gaining interest in the industry. In this paper, we address the application of a novel model checking technique, named Symbolic Trajectory Evaluation (STE) in order to verify the Look-Aside Interface (LA-1) [7].

The work on the LA-1 specification, the Look Aside Interface (Fig.1), started at the beginning of 2001 and became an approved Network Processing Forum (NPF) specification in April 2004 [2]. Basically, the LA-1 interface standardizes the movement of data between network processing elements and co-processors. Due to increasing demand of detailed lookups on packets and flows for the new IPv6 systems and carriers, there is a need of a faster memory mapped interface between network processors and other components, such as external co-processors and memory devices. Currently, the Look-Aside Interface [7] is the de-facto standard interface for linking these components [2]. It is being the key to several networking-specific applications, including packet forwarding, packet classification, admission control, and security. The LA-1 interface was first verified A.Habibi et al. [5] at a behavioral level. In our knowledge that work consisted the full RTL design verification of LA-1 Interface.

STE offers an alternative to classical symbolic model checking technology and it has shown its excellence in verifying medium to large scale industrial hardware designs, with a high degree of automation at both the gate level and the transistor level. It's widely used at Intel, Compaq, IBM, and Motorola. In Motorola, it has been used to verify several memory units, some with millions of transistors [11]. Also in [8] [9] STE has been used to verify CAMs (Content Addressable Memories) and PowerPC microprocessors.

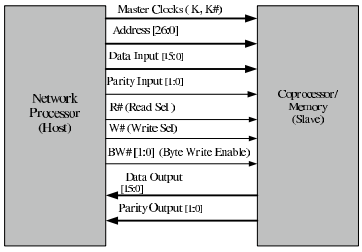

Figure 1: Look-Aside Interface

Our goal, was two fold: First we wanted to validate our verification flow and second, we wanted to compare the efficiency of the STE verification versus traditional model checking using a commercial model checking tool from IBM (RuleBase) [6] by verifying the same properties.

The rest of the paper is organized as follows: Section 2 explains Symbolic Trajectory Evaluation. Section 3 illustrates the design specification of the Look-Aside Interface (LA-1 Standard). Section 4 and 5 describe Model Checking using STE and RuleBase respectively. Section 6 contains the experimental results. Finally, we conclude in Section 7.

2 Symbolic Trajectory Evaluation

STE is a symbolic simulation combined with quaternary abstraction, a model checking technique first introduced by Seger and Bryant [10].

STE is used to verify behaviors of a system over bounded, finite time intervals. In STE, properties of the circuit are specified by trajectory assertions expressed in a restricted temporal logic. This temporal logic is so restricted that it

0-7803-8886-0/05/$20.00 ©2005 IEEE
CCECE/CCGEI, Saskatoon, May 2005

allows us to express properties over trajectories: bounded-length sequences of circuit states, saving the efforts of representing a systems transition relation and calculating its reachable state set, two very expensive operations, and thus makes STE suitable for the verification of large, data intensive systems, such as memory arrays, which is an important part of our LA-1 design.

STE offers an effective alternative to classical symbolic model checking [3] which often suffers from the state explosion problem. The strength of STE comes largely from the property that the complexity of an STE run depends only on the complexity of the STE assertion itself rather than of the circuit being checked.

A normalized form of basic STE assertions is

$$A_1 \& N(A_2) \& ... \& N^{k-1}(A_k) \rightarrow C_1 \& N(C_2) \& ... \& N^{k-1}(C_k)$$

where A_i and C_i are simple predicates or conjunctions of these, and N is the next time operator. For example, A_1 can be $D = d$, a simple predicate which states that node D of a circuit has the value of the variable d at the present time. The left side of the STE implication $\rightarrow$ is called the antecedent of the assertion, and the right side specifies the consequent. The antecedent instructs the initialization of signals for the symbolic simulation in STE and the consequent defines the expected response of the circuit which is then checked against the simulation result. Each signal may have one of the four values 0, 1, X and T, where X denotes an unknown and T denotes an over-constrained value, and a signal is initialized to X at all times if not specified.

A model-checking algorithm is applied in STE for proving such assertions valid, in that a system is checked whether it is a model of an assertion.

3 Look-Aside Interface

The LA-1 standard is the look-aside interface to network-processing elements (NPEs). It targets look-up-tables and memory-based coprocessors and emphasizes as much as possible on the use of the existing technology. It is based on Quad Data Rate (QDR) technologies. Although modeled on an SRAM interface, the LA-1 specification aims to accommodate other devices as well, such as CAMs, classifiers and encryption co-processors.

The LA-1 interface major features include:

- Concurrent read and write operation
- Unidirectional read and write interfaces
- Single address bus
- 18 pin DDR data output path transfers 32 + 4 bits of even byte parity per read.
- 18 pin DDR data input path transfers 32 + 4 bits of even byte parity per write
- Byte write control for writes

The LA-1 interface requires a master-clock pair. The master clocks (K and K#) are ideally 180 degrees out of phase with each other, and they are outputs for the host device and inputs for the slave device. A write cycle is initiated by asserting WRITE_SEL (W#) low at rising edge of K (K clock). The address of the Write cycle is provided at the following edge of K (K# clock which 180 degrees out phase from clock K). A read cycle is initiated by asserting READ_SEL (R#) low at rising edge of K (K clock) and the read address is presented on the same rising edge.

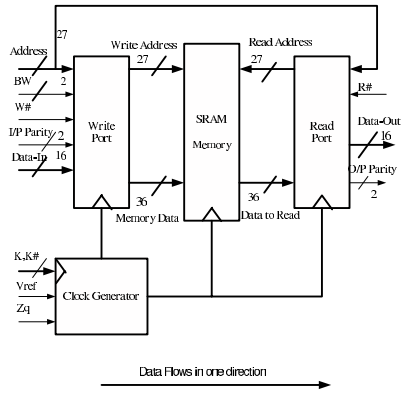

Figure 2: Look-Aside Interface single bank

A synthesizable RTL implemented in Verilog is derived from the design specification of LA-1 Interface. All verifications were performed on the original unpruned gate-level RTL models. Based on the design specification we chalked out the following properties:

Property 1

The first property is based on the operation of Write Port (Fig.3). The control pin BW1# and BW0# are used to enable or block write of a specific byte in a write cycle initiated with W# low at the rising edge of the master-clock.

Based on the above fact, Property 1 states that by asserting W# low at the rising edge of CLK_K and with control bus inputs BW0# and BW1# set to active low the full input data will be written at the rising edge of CLK_K and CLK_K1. Note that CLK_K and CLK_K1 are 180 degree out of phase. This above scenario is known to be the passthrough mode of the Write Port.

Property 2

This Property verifies the operation of the read port of the design. According to the specification a read cycle is initiated by asserting R# low at CLK_K rising edge. Data is delivered after the next rising edge of CLK_K.

4 Using STE for Model Checking

In this section, we present how we verified the LA-1 interface using the STE model checking technique mentioned above. The STE tool we used here is Forte. Forte is Intel's custom-built verification environment, evolved from Seger's Voss system. Forte integrates model-checking engines (STE), BDDs, circuit manipulation functions, theorem proving, and a functional programming language called

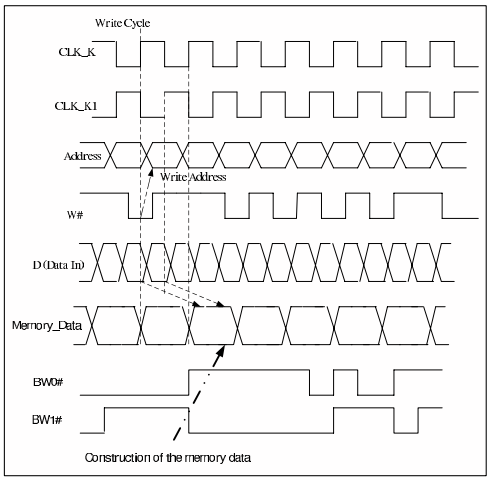

Figure 3. Timing diagram of the Write Port of LA-1 Interface

FL. Forte compiles standard HDL source code into formal circuit models, and includes tightly integrated graphical interfaces for the display of circuit structures and waveforms [4].

The model to be verified in Forte is written in Exlif format, where the RTL design is flattened to the gate level netlist. It is necessary to write a translator from the Verilog RTL to Exlif format. The high level description of the translation is given in Fig.4. The Verilog code is translated to Blif-mv using the VIS tool, then we developed a Perl script which converts the Blif-mv to Exlif. The converter is straight forward since the two formats Exlif and Blif-mv are similar modulo certain syntactic differences. The correctness of our translation is guaranteed, then by the VIS tool.

With the design translated to Exlif and its properties expressed by STE assertions of the form $Antecedent \rightarrow Consequent$, both the design and it's properties were loaded into Forte system and then the STE simulation was invocated which ended up with a value T/F to indicate the success/fail of the simulation.

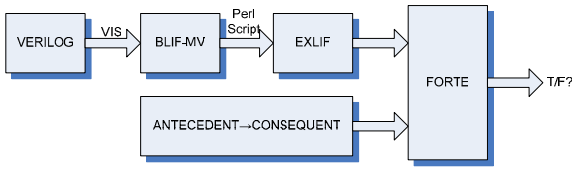

Figure 4: Verification in Forte

The STE formulations of the two properties which were described earlier in section 3 are given below:

Property 1

$A_1\&N(A_2)\&N^2(A_3)\&N^3(A_4)\&N^4(A_5)\&N^5(A_6) \rightarrow N^5(C_6)$, where
$A_1 = \neg CLK\_K \wedge CLK\_K1 \wedge \neg W\# \wedge \neg BW0\# \wedge \neg BW1\#$,
$A_2 = CLK\_K \wedge \neg CLK\_K1 \wedge D[15:0] = d1[15:0]$,
$A_3 = \neg CLK\_K \wedge CLK\_K1 \wedge D[15:0] = d2[15:0]$,
$A_4 = CLK\_K \wedge \neg CLK\_K1$,
$A_5 = \neg CLK\_K \wedge CLK\_K1$,
$A_6 = CLK\_K \wedge \neg CLK\_K1$,
$C_6 = Memory\_Data[31:16] = d1[15:0] \wedge$
$Memory\_Data[15:0] = d2[15:0]$.
Note that the $d1[15:0]$ and $d2[15:0]$ here are vectors of boolean variable which will cover all the 2^{16} cases of the input nodes.

Property 2

$A_1\&N(A_2)\&N^2(A_3)\&N^3(A_4)\&N^4(A_5)\&N^5(A_6)\&N^6(A_7)$
$\&N^7(A_8)\&N^8(A_9)\&N^9(A_10)\&N^10(A_11)\&N^11(A_12)$
$\rightarrow N^{10}(C_11)\&N^{11}(C_12)$, where
$A_1 = \neg CLK\_K \wedge CLK\_K1 \wedge \neg W\# \wedge R\# \wedge$
$A[5:0] = a0[5:0] \wedge \neg BW0\# \wedge \neg BW1\#$,
$A_2 = CLK\_K \wedge \neg CLK\_K1 \wedge D[15:0] = d1[15:0]$,
$A_3 = \neg CLK\_K \wedge CLK\_K1 \wedge D[15:0] = d2[15:0]$,
$A_4 = A_6 = A_8 = A_{10} = A_{12} = CLK\_K \wedge \neg CLK\_K1$,
$A_5 = A_7 = A_9 = A_{11} = \neg CLK\_K \wedge CLK\_K1$,
$C_{11} = DATA\_OUT[15:0] = d1[15:0]$,
$C_{12} = DATA\_OUT[15:0] = d2[15:0]$.

5 Using RuleBase for Model Checking

RuleBase is a model checking tool [6] that supports PSL [1] as its property specification language. Rulebase engine takes a RTL model of the design usually written in Hardware Description Language (Verilog in our case)composed with an EDL (Environment Description Language) which is the target environment in which the design is expected to run.

PSL is an implementation independent language to define properties. It does not replace, but complements existing verification methodologies like VHDL and Verilog test benches. The syntax of PSL is very declarative and structural which leads to sustainable verification environments. PSL consists of four layers based on the functionality of interest [1]:

The modeling layer is used to model behavior of design inputs for formal verification tools, and to model auxiliary parts of the design that are needed for verification.

The verification layer is used to tell the verification tool what to do with the properties described by the temporal layer.

The temporal layer is used to describe properties of the design, as well as simple general properties. This layer can describe properties that involve complex temporal relations. Temporal expressions are evaluated over a series of evaluation cycles.

The Boolean layer is used to build expressions for the

TABLE I
Model Checking Using STE.

Address Width SRAM Memory	Memory (in MB)	Number of BDD nodes
4	15	6294
6	23	17448
8	55	62682

TABLE II
Model Checking Using Rulebase.

Address Width SRAM Memory	Memory (in MB)	Number of BDD nodes
4	18	8785
6	29	22584
8	65	27695

other layers, specifically the temporal layer. Boolean expressions are evaluated in a single evaluation cycle.

PSL is a hierarchical language, where every layer is built on top of the layer below. This approach allows the expressing of complex properties from simple primitives. A property (also called assertion) is built from three types of building blocks: Boolean expressions, sequences, which are themselves built from Boolean expressions, and finally subordinate properties. Sequences, referred to as SEREs (Sequential Extended Regular Expressions), are used to describe a single– or multi–cycle behavior built from Boolean expressions.

6 Experimental Results

In following, we describe our results on the verification of the LA-1 Interface using the STE based verification technique. The STE and PSL properties were verified on 2 X UltraSPARC-III+ machine with 2 900Mhz processors and 4096M of RAM.

Table I shows the statistics in order to verify all the properties for 4 bits, 6 bits, and 8 bits address width using FORTE. Table II gives a summary of the model checking results using RuleBase for the same properties.

We notice that even though RULEBASE succeeds to verify the property the required memory is relatively big. The result proves that STE offers an alternative to classical symbolic model checking that, within its domain of applicability, often is much easier to use and much less sensitive to state explosion.

Another notable issue which needs to be mentioned is that all the properties which were verified using RuleBase needed important effort in order to define the environment and writing the behavioral model including all the design levels of hierarchy. On the contrary, in Forte, all the initial settings and stimulus patterns for the circuit nodes are defined in the STE antecedent part.

7 Conclusion

In this paper, we addressed a formal verification approach using STE. By proving the writing and reading modes properties using model checking, we also proved the functional sanity of our RTL implementation (for one bank). Besides, experimental results hinted that using partial model checking i.e. STE based technique is more memory efficient when performed at the RTL design phases.

As future work, we consider proving the correctness of the complete RTL implementation of Four Banks of LA-1 Interface. This will allow re-using the verification results that can be proved at any level for the other lower levels; thus, reduce the complexity of verifying RTL.

References

[1] Accellerab Organization. Accellera Property Specification Language reference manual, Version 1.01., 2004.

[2] H. Bhugra. La-1b: Moving the look-aside interface forward. *CommsDesign*, August 2002.

[3] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[4] Intel Corporation. www.intel.com/research/scl/fortefl.htm, 2003.

[5] A. Habibi, A. I. Ahmed, O. Ait Mohamed, and S. Tahar. On the Design and Verification of the Look-Aside Interface. In *Proc. IEEE/ACM Design Automation and Test in Europe (DATE'05)*, pages 649–654, Munich, Germany, March 2005.

[6] IBM Haifa Research Laboratories. *RuleBase Formal Verification Tool (Version 1.5). Users Guide.* May 2003.

[7] Network Processing Forum. *Look-Aside (LA-1) Interface, Implementation Agreement, Revision 1.1.* Kluwer Academic Publishers, April 15, 2004.

[8] M. Pandey, R. Raimi, R. E. Bryant, and M. S. Abadir. Formal verification of content addressable memories using symbolic trajectory evaluation. In *Annual ACM IEEE Design Automation Conference*, pages 649–654, Las Vegas, Nevada, United States, 1996.

[9] M. Pandey, R. Raimi, R. E. Bryant, and M. S. Abadir. Formal verification of content addressable memories using symbolic trajectory evaluation. In *Annual ACM IEEE Design Automation Conference*, pages 167–172, Anaheim, California, United States, 1997.

[10] C.-J. Seger and R. Bryant. Formal verification by symbolic evaluation of partially-ordered trajectories. *Formal Methods in System Design*, 6(2):147–190, March 1995.

[11] J. Yang and A. Goel. Gste through a case study. In *International Conference on Computer-Aided Design*, pages 534–541, San Jose, California, 2002.