Design for Verification of the PCI-X Bus

Haja Moinudeen, Ali Habibi and Sofiène Tahar
Electrical and Computer Engineering Department
Concordia University, Montreal, Canada
Email: {haja_m,habibi,tahar}@ece.concordia.ca

Abstract— **The importance of re-usable Intellectual Properties (IPs) cores is increasing due to the growing complexity of today's system-on-chip and the need for rapid prototyping. In this paper, we provide a design for verification approach of a PCI-X bus model, which is the fastest and latest extension of PCI technologies. We use two different modeling levels, namely UML and AsmL. We integrate the verification within the design phases where we use model checking and model based testing, respectively at the AsmL and SystemC levels. This case study presents an illustration of the integration of formal methods and simulations for the purpose of providing better verification results of SystemC IPs.**

I. Motivation and Proposed Methodology

With the advent of high technology applications, an increasingly evident need has been that of incorporating the traditional microprocessor, memories and peripherals on a single silicon. This is what has marked the beginning of the System-on-Chip (SoC) era. An SoC can be viewed as a collection of various Intellectual Property (IP) cores, with interconnecting buses running among them. There is a dire need for standard buses to connect IPs obtained from different vendors. One such and latest bus standards is PCI-X [8], which is a high performance bus for interconnecting chips, expansion boards, and processor/memory subsystems. It has the performance to feed the most bandwidth-hungry applications and helps to alleviate the I/O bottleneck problem while at the same time maintaining complete hardware and software backward compatibility to previous generations of PCI [8].

In this paper, we present a design for verification effort done for the PCI-X bus. We start with an informal specification of PCI-X and model it with the Unified Modeling Language (UML) in order to have a clear view of the design modules and their interactions. Then, we construct an Abstract Machine Language (AsmL) [3] model from the UML representation.

We define a set of properties of the PCI-X in the Property Specification Language (PSL) [1] that we verify using model checking. Finally, we translate the AsmL model to SystemC [5]. Unfortunately, not all bus properties can be verified using model checking, that is why we use model based testing (MBT) [6] to perform a guided simulation of the IP.

Related work to ours in the context of PCI technologies design and verification environment concerns, in particular, the work of Shimizu *et al.* [7] who presented a specification of the PCI bus as a Verilog monitor. Any modification or refinement of the model provided [7] is complex due to the low level of specification of the bus. Furthermore, the PCI-X standard includes very complex transaction rules in comparison to PCI which cannot be handled only using model checking.

II. PCI-X Bus

PCI-X provides backward compatibility by allowing devices to operate at conventional PCI frequencies and modes. The bus structure includes an arbiter that performs the bus access arbitration among multiple initiators and targets (see Figure 1). Unlike the conventional PCI bus, the arbiter in PCI-X systems monitors the bus in order to ensure good functioning of the bus. PCI-X supports two modes of operations: Mode 1 and Mode 2. In Mode 1 operation, data transfers always use common clock. Mode 2 operation of PCI-X also supports 16 bit bus interface which facilitates low cost interface.

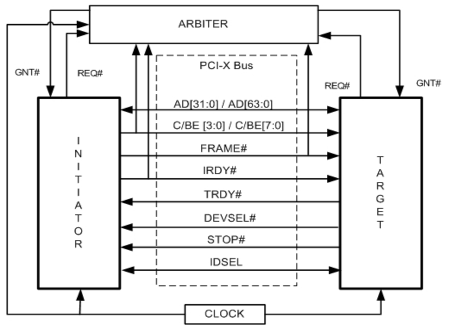

Fig. 1. General Architecture of PCI-X.

III. Design Phases

UML Level: From the specification of PCI-X, we identify the core components of the bus viz, initiators, targets, arbiters, PCI-X bus, which will be represented as classes, where specific instances of the components are called as objects. In addition to these four components, we also added another component, the Simulation Manager (SimManager), in order to have a notion of updates. We modeled different modes and types of operations of PCI-X using sequence diagrams which enable us to model the bus in AsmL easily and efficiently.

AsmL Level: Class diagrams in UML help implementing the classes in AsmL. Each of the five core components of PCI-X has its own data members (signals) and methods (behavior) in addition to a constructor. We also use enumeration features (*enum*) of AsmL to model different modes of PCI-X, different types of transaction phases, the state of the system and the clock. In Figure 2, we show how a target can signal its readiness using the TRDY# signal. We call this method as *PCIX_Target_TRDY_Assert()*. The pre-conditions are the following: TRDY# is *false*, FRAME# and DEVSEL# are *true*, CLK is *CLK_UP*, Phase is *DATA_PHASE_FIRST* and the *AD* of the Bus should be the *ID* of the target. If the pre-conditions are true, then TRDY# will be asserted.

```
public PCIX_Target_TRDY_Assert()
    require me.TRDY = false and Bus.FRAME = true
        and Bus.AD = me.ID and Bus.DEVSEL = true
        and Smanager.CLK = CLK_UP
        and Phase = DATA_PHASE_FIRST
    me.TRDY := true
    me.AD := Bus.AD
    Bus.TRDY := true
    Phase := DATA_PHASE
```

Fig. 2. Target Assert AsmL Method.

SystemC Level: After the AsmL model of a PCI-X is verified against the properties, we translate it to SystemC using a sound syntactical transformation developed by Habibi *et al.* in [4].

IV. Verification Approach

Model Checking: The AsmL model is validated using a set of user-defined PSL properties. Any incorrect property detection halts the reachability algorithms and outputs a trace for counter-examples. Table I provides the results of model checking of PCI-X model with 5 initiators and 5 targets [1]. We show the CPU time, number of states and transitions for the PCI-X model with various properties that we defined. As can be seen from the table, all properties have been verified except Property 6 and Property 7 due to a state explosion problem. For instance, these properties are related to the successful completion of a data transfer which typically takes more cycles than other transactions.

TABLE I
Model Checking Results

Property	CPU Time (s)	States	Transitions
P1	385.24	2169	3250
P2	194.23	1800	2563
P3	150.52	1578	2156
P4	130.45	1489	2096
P5	156.35	1478	2265
P6	–	–	–
P7	–	–	–
P8	173.50	1925	2439
P9	174.47	2013	2698
P10	178.42	1873	2359
P11	256.63	2192	2980
P12	143.52	1356	1923

Model Based Testing: In MBT the behavior of a system is defined in terms of actions that change the state of the system. Such a model of the system results in a well-defined Finite State Machine (FSM) which helps understand and predict the system's behavior. We first generate the FSM of the PCI-X model in AsmL based on the algorithm developed by Grieskamp *et al.* in [2]. Then, using existing graph traversing techniques, test cases are obtained from the generated FSMs to validate the PCI-X model. Table II shows the CPU time, states and number of transitions in the generated FSM, for several combinations of initiators and targets. Using the generated FSM, we apply various techniques to choose the tests. We took advantage of several efficient graph traversing techniques in the open literature, in particular, the Chinese Postman Tour (CPT) and Random walk methods. MBT was of a great help in identifying several bugs in the SystemC PCI-X model we developed.

[1]Experimentation platform: Pentium IV(2.4 GHz) / 768 MB of memory.

TABLE II
FSM Generation: Direct Algorithm.

Number of		CPU	States	Transitions
Initiators	Targets	Time (s)		
1	5	27.95	234	253
2	5	59.50	466	505
3	5	108.04	698	757
4	5	171.73	930	1009
5	2	69.89	472	511
5	3	118.20	702	761
5	4	204.93	932	1011
5	5	254.82	1162	1261
10	10	2925.31	4622	5021

V. Conclusion

We presented a design for verification approach applied on the latest high speed PCI-X standard bus. Starting with a UML formal specification, we derived an AsmL model which we model checked against a set of PSL properties. We then translated the AsmL model to SystemC on which we investigated the potential of model based testing approach for SystemC designs. The traversal of the FSM was performed to generate test cases. The final PCI-X SystemC IP was thoroughly and formally verified, which makes it very suitable for use as external monitor to validate existent PCI-X compatible IPs. We believe that our approach shows how one can improve the verification of SystemC models by integrating formal methods and guided simulation in a single design flow.

References

[1] Accellera Organization. Accellera Property Specification Language Reference Manual, version 1.1. www.accellera.org, 2005.

[2] W. Grieskamp, Y. Gurevich, W. Schulte, and M. Veanes. Generating Finite State Machines from Abstract State Machines. *Software Engineering Notes*, 27(4):112–122, 2002.

[3] Y. Gurevich, B. Rossman, and W. Schulte. Semantic Essence of AsmL. Technical report, Microsoft Research, MSR-TR-2004-27, March 2004.

[4] A. Habibi and S. Tahar. On the Transformation of SystemC to AsmL using Abstract Interpretation. *Electronic Notes in Theoretical Computer Science*, 131:39–49, May 2005.

[5] Open SystemC Initiative. www.systemc.org, 2006.

[6] H. Robinson. Model-based testing. website: http://www.geocities.com/model_based_testing/, 2006.

[7] K. Shimizu, D. L. Dill, and A. J. Hu. Monitor-based formal specification of PCI. In *Formal Methods in Computer-Aided Design*, pages 335–353. LNCS 1954, Springer-Verlag, 2000.

[8] PCI Special Interest Group. Website: www.pcisig.com, 2006.