Practical Approaches to the Verification of a Telecom Megacell using FormalCheck

Leila Barakatain Sofiène Tahar
ECE Department, Concordia University
1455 deMaisonneuve West, Montreal
Quebec, H3G 1M8 Canada
+1 (514) 848-3100
{l_baraka, tahar}@ece.concordia.ca

Jean Lamarche Jean-Marc Gendreau
PMC-Sierra Inc.
3333 Boul. Graham, Ville Mont-Royal
Quebec, H3R 3L5 Canada
+1 (514) 734-3700
{lamarche, gendreau}@pmc-sierra.com

ABSTRACT

In this paper we present practical approaches to formally verify the RTL implementation of a Telecom megacell using model checking techniques based on the FormalCheck tool. We adopted a hierarchical verification method, which relies on the built-in hierarchy of the design as the mechanism to conquer its verification complexity. We then applied a number of tool guided abstraction and reduction techniques within FormalCheck to avoid state space explosion. The case study we considered is the Transmit Master/ Receive Slave (TMRS) Telecom megacell from PMC-Sierra, Inc., implementing a SCI-PHY (Saturn Compatible Interface for ATM-PHY devices) protocol. Using our approaches, we succeeded the model checking of the TMRS design and were able to uncover a number of flaws in the RTL design as well as in the documentation specification.

1. INTRODUCTION

In today's red-hot economy, staying on schedule to hit a narrow widow of market opportunity often means the difference between product dominance and product death. With the volume and complexity of logic required to satisfy function-hungry consumers comes exponential growth in the difficulty of making sure it all works. Verification is on the critical path for today's Integrated Circuit (IC) designers, no matter what type of system they are building. Formal methods have been around in academia for sometime [8]. However it is now becoming prevalent in the Electronic Design Automation (EDA) industry as a means to combat the explosive complexity of current IC designs. As an IC functionality increases linearly, the amount of vectors required to fully testing this functionality increases exponentially. It therefore becomes impossible for a human being to fathom all the vectors required to fully test an IC. By using formal verification in parallel with the design efforts, the overall design cycle can be reduced [9]. Formal verification has problems of its own class too. As the number of reachable states increases rapidly with the size of a system, the underlying verification algorithms are unable to perform an exhaustive check within the limited time and memory that is available. This phenomenon is known as the state space explosion problem [8].

The goal of this paper is to introduce and apply practical model checking approaches, which offer some solutions to alleviate the state space explosion problem for verification of high level descriptions of real systems. We suggest a number of practical verification approaches, which rely on the built-in hierarchy of the design as the mechanism to conquer its complexity during verification. These approaches minimize the state space explosion problems by concentrating the bulk of the verification effort to register transfer level (RTL) modules suitable for model checking. We successfully applied these ideas on an industrial case study, the Transmit Master/Receive Slave (TMRS) Telecom megacell designed by PMC-Sierra, Inc. [12]. It implements a SCI-PHY (Saturn Compatible Interface for ATM-PHY devices) protocol and works in different transmit/receive master/slave modes. Throughout our verification efforts, we uncovered a number of mismatches between the RTL design and the SCI-PHY protocol as well as the TMRS documentation specification. Some of these are pertinent design bugs, which were not caught during the simulation process of the product. Finally, from the lessons learned, we suggest a set of recommendations to improve the design flow by accelerating the verification process.

For the formal verification, we used FormalCheck [5], a model checking tool commercialized by Cadence. It is used to verify synthesizable VHDL or Verilog RTL design models using model checking [8]. FormalCheck embraces a number of patented reduction algorithms which allow the handling of larger designs than any other model checker on the market. In [5] it is reported that properties on designs of up to 5,000 latches and 100,000 combinational variables have been verified with this tool. FormalCheck also provides an intuitive graphical interface to simplify the verification process.

A number of related work on the formal verification of telecommunications hardware is reported in the open literature. Notably the work of Chen et. al. [6] on the verification of an ATM (Asynchronous Transfer Mode) circuit at Fujitsu using the SMV model checker. Lu et. al. [10] performed the model checking of the Cambridge Fairisle ATM switch using VIS. Xu et. al. [15] verified a Frame Multiplexer/Demultiplexer (FMD) Chip from Nortel Semiconductors using FormalCheck. In all these verifications, the authors required a number of abstraction and reduction techniques

in order to combat state space explosion. These, however, were done manually or at most semi-automatically which might not be very practical for a large design. Moreover, the work in [6] and [10] do not provide a formal proof for the soundness of the reduction or composition used during the verification. In this paper, we present our results of formally verifying the TMRS Telecom megacell using FormalCheck where not only the model abstraction and reduction are done automatically, but also the normal operating environment of the design is defined inside the tool.

The rest of the paper is organized as follows. In Section 2, we present a general verification approach based on the built-in hierarchy of a design. In Section 3, we outline a diversity of abstraction and reduction techniques to enable a suitable verification in FormalCheck. In Section 4, we briefly introduce the structure of the TMRS megacell and its behavior in SCY-PHY mode. We also describe sample properties we checked on TMRS using FormalCheck and display related experimental results. In Section 5, we give a set of practical recommendations to improve the design verification flow. Finally, Section 6 concludes the paper

2. HIERARCHICAL VERIFICATION APPROACH

A typical design flow relies on a top-down partitioning phase where the specifications are broken down into focused areas of functionality (cores and design units). This top-down partitioning phase is followed by a bottom-up implementation phase (Figure 1).

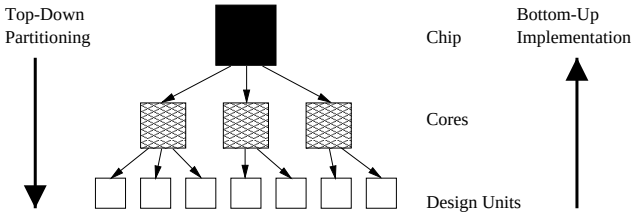

Figure 1: Hierarchy in high-level design process

At the implementation phase, designers mix and match legacy components, third-party IP cores, and newly developed pieces of logic to create their designs. During this phase, designers not only spend most of their time writing RTL code, but simulating and debugging the interfaces between the third-party solutions and their own designs.

Using a model checker (in our case FormalCheck) early in the design cycle addresses the functional verification problem as the implementation phase evolves, that is, in a bottom-up fashion. As each design unit becomes available, a model checker is used to assess their correct functionality. This *horizontal verification* is followed up by a *vertical verification* [14]. The goal of the horizontal verification is to attain a high level of confidence on the correct functionality of each design unit as a stand alone entity. This is verified by ensuring that all properties in a design unit hold true within the *Constraints* set forth by assuming certain environmental conditions. In vertical verification, properties that are already proven at lower levels in the hierarchy of the chip, are assumed to conduct the verification of properties at higher levels. In this work, we have investigated the notions of horizontal and vertical verification in practice by considering different possible structures of a design block and how to verify the properties of each specific structure. We found out that there are some special cases, where the application of vertical verification will lead to a circular behavior as explained below.

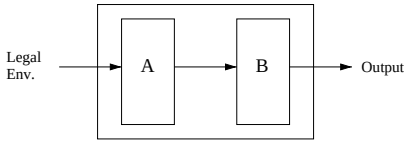

Figure 2: One-way dependency between design blocks

The relationship between the interacting design units of a core can be a one way (linear) or two-way dependency. In the former, the output signals of a block depend only on the values of the signals coming from another block (see example of blocks A and B in Figure 2). In the latter, the output of one block depends on the internal output of the other block, and vice versa (see example of blocks C and D in Figure 3). In this case, we have a circular behavior in a compositional verification [1]. This circularity can for example be broken through induction over time [11].

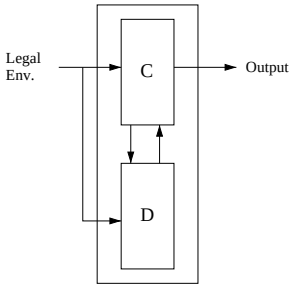

Figure 3: Two-way dependency betwee design blocks

During the verification of properties on the TMRS design, we needed (in some cases) to decompose a global property in subproperties related to specific blocks in the design. These subproperties once verified can be changed to Constraints to verify the other global properties (cf. Section 4.2)

3. ABSTRACTION AND REDUCTION IN FORMALCHECK

State space explosion is a well-known problem in FSM-based verification approaches [9]. In digital hardware designs, the state space increases exponentially with the number of latches of a design. Because model-checking algorithms are based on state space exploration, their efficiency is also limited by this phenomenon. The more extensive the reachable states, the more time and memory it takes to verify a system. The biggest obstacle of model checking is often the unmanageably huge number of reachable states. FormalCheck relies on the localization of a property into a portion of the design that is relevant to what is being verified. However, such localization by itself may not succeed in performing the verification unless assisted by the user. As the complexity of functional building blocks increases, the default setting used by most formal verification tools may be insufficient. One needs some reduction and abstraction methods in order to avoid the state space explosion problem. For instance, to make the proper verifi-

cation environment for certain design blocks, we used the following tool guided techniques within FormalCheck: model abstraction, model reduction, and environment constraints.

3.1. Tool Guided Abstraction

In FormalCheck, model abstraction is possible using *Electronic Scissors* [5]. For example, assume in the target model there is a test block in addition to the main block which controls the main functionality of the model. Also assume only the verification of the main block is of interest. In that case, the test block can be removed using the Electronic Scissors. The inputs of the main block which are fed by the test block, can be set to neutral constant values using Constraints in FormalCheck.

To reduce the state space and speed up the verification of our case study, we tried to trim the TMRS design by eliminating the blocks that did not have any or very little effect on the control blocks (cf. Section 4.1).

3.2. Tool Guided Reduction

In FormalCheck there are two reduction algorithms, the *1-Step algorithm* and *Iterated algorithm* [5]. The 1-Step algorithm performs a single reduction and then verifies the query (a query consists of a property/properties and Constraints). This algorithm looks at the dependency graph and removes any portion of the design from the verification proof that cannot affect the outcome of the query. The Iterated algorithm iteratively reduces the design model during the verification process. This algorithm takes an attempt to find a small portion of the design that can be used to verify the current query. This technique guarantees that queries proven to be true on the small portion of the design would also be true for the entire design.

To reduce the memory usage even more, we also introduced *Reduction Seeds* with the *Start as Input* option [4]. By defining some of the signals as Reduction Seeds, we are reducing the explored state space for model checking and hence improving performance. Assigning Start as Input reduction value to an element will initially free it. It starts as a primary input but FormalCheck can make it active again if it is determined that the logic driving the design element cannot be pruned [4].

Examples of reduction techniques and options we applied for the verification of TMRS are given as part of the description of sample properties presented in Section 4.2 below.

3.3. Tool Guided Constraints

In FormalCheck a query is made up of both properties and *Constraints* [5], where the Constraints establish the operating environment for the design. In formal verification all reachable states are explored. By adding Constraints to the properties, we reduce the state space explored in verification and therefore improve the overall performance, which means less CPU time and/or memory usage [5].

Correct design behavior requires certain input behavior. In FormalCheck, primary signals are assumed to be non-deterministic, meaning they could acquire any value within their range on any edge of the clock. However, in most cases correct design operation is allowed on a single edge of the clock. For this reason, properties should be observed using the appropriate clock edge.

A full list of the Constraints we defined for properties checking on the TMRS design can be found in [3].

4. TMRS VERIFICATION CASE STUDY

4.1. The TMRS Telecom Megacell

The TMRS (Transmit Master/Receive Slave) is a Telecom megacell designed by PMC-Sierra, Inc. It implements the output port of a cell interface and can output the cell data either in 8-bit or 16-bit wide format at clock rates up to 52 MHz [12]. Data transfers are cell-based, that is an entire cell is transferred to one physical layer (PHY) device, e.g. for ATM (Asynchronous Transfer Mode), before another is selected. The TMRS supports SCI-PHY (Saturn Compatible Interface for ATM-PHY devices) Level 2 [13] and ANY-PHY (proprietary protocol of PMC-Sierra) [12] protocols. SCI-PHY Level 2 protocol is a superset of UTOPIA (Universal Test & Operations PHY Interface for ATM) Level 2 protocol [2]. The TMRS hence outputs cells on SCI-PHY/ANY-PHY compatible interface.

The TMRS block is designed to interface directly to a Multi-channel Cell FIFO (MCF). It directly supports up to 32 logical channels each corresponding to a physical layer ATM device. Each logical channel corresponds to a FIFO channel in the external FIFO. When the TMRS is operated as a bus slave, it autonomously multiplexes the traffic from up to 32 logical channels and presents them as a single cell stream. The logical channel is identified by the first word of the data cell received from the FIFO.

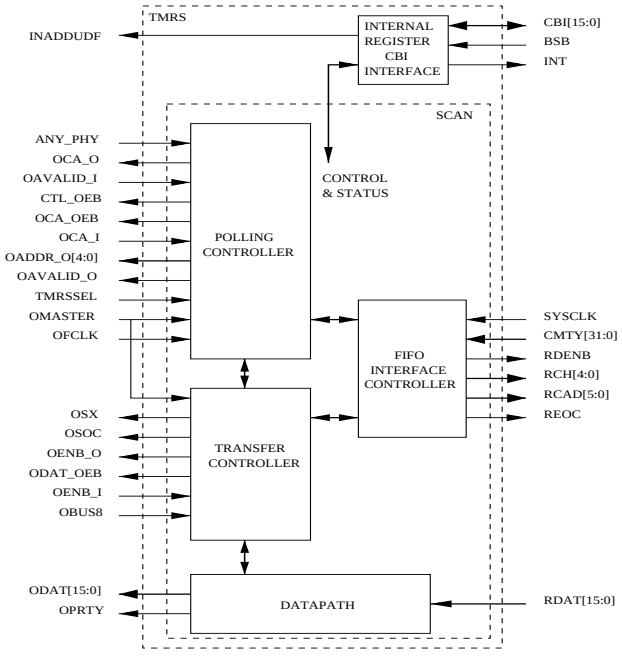

Figure 4: Block diagram of the TMRS

The TMRS consists of two major blocks, the CBI block and SCAN block (Figure 4). The CBI block is used only for the configuration and the test interface and the SCAN block drives the main functionality of the TMRS. The SCAN block of TMRS consists of four main blocks, namely, Datapath, Polling controller, FIFO interface controller and Transfer controller. The Polling controller block handles the control signals related to the polling of the TMRS in

the Receive-Slave mode and polling the physical layer devices in Transmit-Master mode. The FIFO interface controller block decides if any of the 32 FIFOs have a cell available and also decides which FIFO channel should be selected. The Transfer controller block determines when the transmission of a cell starts, and the Datapath block actually handles the transmission of data.

To reduce the state space and speed up the verification, we eliminated those blocks in the TMRS that did not have any or very little effect on the three control blocks. Some of the model abstractions we adopted are as follows:

- Isolated the SCAN block of the TMRS. Hence, we eliminated the CBI block used for test purposes.
- Removed the DATAPATH module from the SCAN block.
- Removed the INPUT_MUX module from the SCAN block. We modeled the outputs of this block which were used by the control blocks, inside the SCAN block.

4.2. Verification Properties

After establishing a proper environment in FormalCheck through the abstraction and reduction of blocks as well as the definition of Constraints, we considered a number of relevant properties of the TMRS, including liveness and safety properties. These properties have been defined based on either the SCI-PHY Level 2 protocol, the documentation specification of the TMRS, or the test benches already written to perform functional simulation. In this section, we discuss three sample properties. We first give an informal description of each property, then its equivalent expression in FormalCheck including details on the particular reduction technique and options used. The experimental results of these sample properties are summarized in Table 1 (cf. Section 4.3). A complete and more detailed description of all properties we checked on TMRS is reported in [3].

Property_A: According to the SCI-PHY protocol: "When the Output port Enable (OENB_I) input signal is sampled low by the PHY layer device, the Output Start Of Cell (OSOC) signal will be accepted by the ATM layer device on the next rising edge of Output FIFO clock (OFCLK)" [13]. In FormalCheck, this property is expressed as follows.

```
Property: property_A
Type: Always
After: (@TMRSselected) and (@WaitSelected)and
       (@CLKrising)
Always: OSOC = 1 and @StartTxState
Options: Fulfill Delay: 0
         Duration: 1 Count of @CLKrising
Reduction Technique: Iterated
Reduction Seed: New (User defined seed)
Start as input: Fifo_Ctl_Inst
```

Table 1 reports that the verification of Property_A was "Terminated", indicating that the verification attempt was stopped by an abnormal condition. Based on the specification of the TMRS "the START_TRANSF state starts the transfer by asserting OSOC signal" [12], therefore, Property_A expects the FSM to move to START_TRANSF state and the OSOC signal to be set to high in the same clock cycle. The Transfer_Slave state machine is actually made of two parallel processes (state machines). One of them generates the OSOC as well as other control signals, while the other one determines the state of the Transfer_Slave state machine in the next clock cycle. To discover the cause of termination, this property was decomposed according to the vertical verification approach described above to two properties, Property_A1, which checks for the state transition to START_TRANSF state, and Property_A2, which checks for the assertion of the OSOC signal as follows.

```
Property: property_A1
Type: Always
After:(@TMRSselectedLastCC) and
      (@WaitSelectState) and (@CLKrising)
Always: @StartTxState
Options: Fulfill Delay: 0
         Duration: 1 Count of @CLKrising
Reduction Technique: Iterated
Reduction Seed: New (User defined seed)
Start as input: Fifo_Ctl_Inst
```

```
Property: property_A2
Type: Always
After:  (@TMRSselected) and (@Latch4State or
        @WaitSelectState) and (@CLKrising)
Always: Osoc = 1 and @WaitSelectState
Options: Fulfill Delay: 0
         Duration: 1 Count of @CLKrising
Reduction Technique: Iterated
Reduction Seed: New (User defined seed)
Start as input: Fifo_Ctl_Inst
```

The verification of Property_A1 and Property_A2 revealed the origin of the problem, which caused the termination of the verification of Property_A. Namely, the "Transfer_Slave" state machine proceeds to "START_TRANSF" state one clock cycle after the OSOC signal (Output Start Of Cell) is asserted. Therefore, the OSOC signal and START_TRANSF state are not synchronized.

Property_B: According to the TMRS specification: "In Receive Slave SCI-PHY back to back transfer mode, when the external master device does not deassert OENB_I at the end of a transfer, the TMRS remains selected for another cell transfer. If all FIFOs are empty, the TMRS will deassert ODAT_OEB (set to 1) and TMRS will wait to be reselected" [12]. In FormalCheck, this property is expressed as follows.

```
Property: property_B
Type: Always
After: (@TransferDone) and (Oenb_I = 0) and
       (Polling_Sm_Inst:Allfifoempty = 1)
       and (@RstDone)
Always: Odat_Oeb = 1
Options: Fulfill Delay: 0
         Duration: 1 Count of @CLKrising
Reduction Technique: 1-Step
Reduction Seed: Empty
```

It is observed from Table 1, that the verification of Property_B failed. This means even after a cell transfer is done and there are no more cells to transfer, ODAT_OEB will still be asserted. This problem was taken into consideration by the designer of the TMRS

and was fixed. In the new version of the design, when in the back-to-back transfer mode, after completing a cell transfer the ODAT_OEB will stay asserted and the data on the ODAT bus will be the byte/word which was transferred last. The master is capable of determining the end of a cell (by counting the number of bytes/ words it has received), therefore, the master waits for the assertion of the OSOC signal (from the TMRS).

Property_C: According to the SCI-PHY protocol: "Each PHY link shall have a unique address corresponding to a value between 0 and 31. Upon sampling its address with the rising edge of the RFCLK signal, a PHY must drive the RCA signal to indicate whether it has an entire cell in its buffer" [13]. In FormalCheck, this property is expressed for TMRS as follows.

```
Property: property_C
Type: Always
After: (@TMRSpolled) and (@FIFOnotEmpty) and
       (@PrefetchDone) and (@CLKrising) and
       (@RstDone)
Always: @RCAhigh
Options: Fulfill Delay: 0
         Duration: 1 Count of @CLKrising
Reduction Technique: Iterated
Reduction seed: New (User defined seed)
Start as input: Fifo_Ctl_Inst
```

In Property_C, the Polling and Transfer controllers are interacting blocks. The Transfer controller uses the internal output of the Polling controller to determine whether a cell is available, and asserts its internal output to the Polling controller when the prefetch cycle of that cell is done. When the TMRS is polled by the master, the Polling controller uses the internal output of the Transfer controller to assert/deassert the RCA signal. While this property was successfully checked (see Table 1), the vertical verification described in Section 2 could be applicable for this type of property in case of state explosion.

4.3. Experimental Results

The experimental results of the sample properties described in Section 4.2 are shown in Table 1, including the reduction technique used, the status of the property verification, the number of reached states, the number of state variables in the model, the verification CPU time (real time) in seconds, and the memory usage in megabytes. All verifications were executed on a HP9000 (440MHz) with 6144 MB RAM and HP-UX11 operating system.

During the verification process, we found several errors in the document specification of the TMRS which is showing mismatches with respect to the SCI-PHY protocol. We also found a number of mismatches between the RTL implementation of TMRS and the TMRS specification. We hence suggested some modifications to the design and specification of TMRS which were considered acceptable by the designer of the TMRS, and the specification was revised to reflect these corrections. A full list of the uncovered errors can be found in [3].

5. RECOMMENDATIONS TO IMPROVE THE VERIFICATION PROCESS

Designers of today's massive circuits face a pair of conflicting goals. They must increase verification coverage and quality to ensure single-pass success. They also must find ways to contain the expansion of verification time, effort, and cost. One of the biggest concerns of companies is "What kind of changes in the design flow can accelerate the verification process?". To find a way to reduce the time spent for verification, including simulation and formal verification, first we have to answer this question: Where do bugs come from? Bugs can be introduced to the design through several channels such as [7]:

- Incorrect specifications
- Misinterpretation of specifications
- Misunderstanding between designers
- Missed cases
- Protocol non-conformance
- Resource conflicts, etc.

The only way to detect bugs inside a design is to define the right queries (for model checking) and/or to write the proper test benches (for simulation) to uncover them. Test processes and methodologies can considerably reduce the risk of releasing systems which could fail in the field. From the lessons learned through out our study, we propose following ways to improve the design flow, in order to ease and accelerate the query definition and verification process:

- The first and probably most important improvement in the design process is having a good quality (clear, complete, and up-to-date) specification for each product life cycle step. Applying the hierarchical verification methodology explained in this paper will contribute to this improvement. As mentioned before, the model checking of the design units starts as soon as they are ready. At that time, the specifications for the embedded design unit is still fresh in designer's mind.

Table 1: Verification results of model checking of TMRS using FormalCheck

Properties	Reduction Algorithm	Verification Status	States Reached	State Variables	Real Time (seconds)	Memory Usage (MB)
Property_A	Iterated	Terminated	N/A	N/A	N/A	N/A
Property_A1	Iterated	Verified	2.93e+18	195	29	7.22
Property_A2	Iterated	Verified	2.94e+18	195	31	7.22
Property_B	1-Step	Failed	1.76e+18	251	10370	126.79
Property_C	Iterated	Verified	2.93e+18	197	73	38.87

- It is a well known fact that the detail available in the specification of the targeted design is usually not enough to fully verify the functionality of it, therefore, a test plan should be created early in the product design and verification cycle. A test plan identifies all the features whose functionality must conform to the specification. Taking this approach will notably ease defining suitable queries for verification of the design model.
- As designs usually are modified in some point in time, properties verified on the earlier version of the design should be reused (with minor modifications) to verify the behavior of the new design. Since the properties usually do not contain the cycle-by-cycle implementation details found in test vectors or test benches, they are more easily reused when verifying a modified version of the previous work.

6. CONCLUSIONS

In this paper, we investigate several practical verification approaches which we applied in the model checking of an industrial Telecom megacell, TMRS from PMC-Sierra, Inc. The main contributions of this work are (1) the suggestion and application of a hierarchical verification approach, (2) the establishment of suitable abstraction and reduction techniques to reduce the state space within FormalCheck, (3) the verification of the VHDL model of the TMRS as a case study, (4) the discovery of several mismatches between the TMRS design, its specification, and the SCI-PHY protocol, and (5) the elaboration of a set of recommendations to accelerate the design verification flow.

Many designers hope that model checking will be a solution to the problems involved with simulation, such as increasing test coverage of design, and shortening the time to market. Model checking will help (indeed, it will be essential) in niches where it is particularly effective, however, it does not replace traditional functional simulation. Model checking is used to expand these verification techniques. Since datapaths increase the number of selection variables per state, datapaths and computational intensive designs tend to rapidly cause state space explosion. Also, in a lot of instances the data values do not affect the control of the design. Hence, model checking is most effective when used for the verification of control-oriented designs such as control logic, finite state machines and protocols. On the other hand simulation is better suited for datapath analysis and computation results analysis.

ACKNOWLEDGMENTS

This work was partially supported by an industrial research grant from PMC-Sierra Inc. The authors are grateful to PMC-Sierra colleagues whose cooperation contributed to the maturity of this work.

REFERENCES

[1] M. Abadi and L. Lamport: Conjoining Specifications; *ACM Transactions on Programming Languages and Systems*, Vol. 17, No. 4, pp. 1-27, September 1995.

[2] The ATM Forum Technical Committee: *UTOPIA Level 2*; Version 1.0, June 1995.

[3] L. Barakatain: Practical Approaches to Model Checking using FormalCheck; MaSc Thesis, Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada, April 2000.

[4] Bell Labs Design Automation, Lucent Technologies: *FormalCheck Users Guide*; V2.1, July 1998.

[5] Cadence: *Formal Verification Using Affirma FormalCheck*; Manual, Version 2.3, October 1999.

[6] B. Chen, M. Yamazaki, and M. Fujita: Bug Identification of a Real Chip Design by Symbolic Model Checking; *Proc. International Conference on Circuits And Systems (ISCAS'94)*, London, U.K., pp. 132-136, June 1994.

[7] D. Dill: What's Between Simulation and Formal Verification?; *Invited Lecture in Design Automation Conference (DAC'98)*, San Francisco, CA, USA, June 1998.

[8] C. Kern and M. Greenstreet: Formal Verification in Hardware Design: A Survey; *ACM Transactions on Design Automation of E. Systems*, Vol. 4, April 1999, pp. 123-193.

[9] R. P. Kurshan: Formal Verification in a Commercial Setting, *Proc. Design Automation Conference (DAC'97)*, Anaheim, California, June 1997, pp 258-262.

[10] J. Lu, S. Tahar, D. Voicu and X. Song: Model Checking of a real ATM Switch; *Proc. IEEE International Conference on Computer Design (ICCD'98)*, Austin, Texas, USA, pp. 195-198, October 1998.

[11] K. L. McMillan: Verification of an Implementation of Tomasulo's Algorithm by Compositional Model Checking; *Proc. Computer Aided Verification (CAV'98)*, Vancouver, BC, Canada, pp. 110-121, June/July 1998.

[12] PMC-Sierra, Inc.: *SCI-PHY Transmit Master and Receive Slave TSB Specification*; Issue 2, May 10, 1999.

[13] PMC-Sierra, Inc.: *Saturn Compatible Interface Specification for PHY Layer and ATM Layer DEVICES, Level 2*; Application Note, Issue 4: August 1997.

[14] C.M. Roman: A Hierarchical Verification Methodology; *International Cadence User Conference*, Orlando, Florida, USA, September 1999.

[15] Y. Xu, E. Cerny, A. Silburt, A. Coady, Y. Liu and P. Pownall: Practical Application of Formal Verification Techniques on a Frame Mux/Demux Chip from Nortel Semiconductors; *Proc. Correct Hardware Design and Verification Methods (CHARME'99)*, Bad Herrenalb, Germany, pp. 110-124, September 1999.