Hybrid Tool Integrating HOL Theorem Proving with MDG Model Checking

Rabeb Mizouni and So ene Tahar
Dept. of ECE, Concordia University, Montreal, Quebec
Email: {mizouni,tahar}@ece.concordia.ca

Paul Curzon
Dept. of CS, Queen Mary University of London, UK
Email: pc@dcs.qmul.ac.uk

Abstract— **We describe a hybrid tool for hardware formal verification that links the HOL theorem prover and the MDG (Multiway Decision Graphs) model checker. Our tool supports abstract datatypes and uninterpreted function symbols available in MDG, allowing the verification of high level specifications. The hybrid tool, HOL-MDG, is based on an embedding in HOL of the grammar of the hardware modeling language, MDG-HDL, as well as an embedding of the first-order temporal logic $\mathcal{L}_{mdg}$ used to express properties for the MDG model checker. Verification with the hybrid tool is faster and more tractable than using either tools separately. We hence obtain the advantages of both verification paradigms.**

I. Introduction

Hybrid verification approaches that link interactive proof tools with automated (e.g. BDD based) proof tools are now common. Such links gain the automation of the BDD tools whilst, for example, using the interactive tool to manage the proof. Whilst abstraction can be dealt with by the interactive tool, it is advantageous if it could also be dealt with by the automated tool. In this paper, we describe a hybrid tool that does this. It combines the HOL theorem prover [11] and the MDG model checker [17]. HOL (Higher-Order Logic) is an interactive theorem prover based on higher-order logic. The MDG (Multiway Decision Graphs) system is a decision diagram based verification tool for Abstract State Machines (ASM) verification encoded by multiway decision graphs [5]. The latter extend Reduced-Ordered Binary Decision Diagrams (ROBDD) [3] with abstract datatypes and uninterpreted function symbols. It is this feature that allows abstract designs to be verified automatically using MDG, rather than needing to do such proof wholly in the theorem prover HOL. The down side of this abstraction facility is that in some cases the state reachability algorithm may not terminate [2]. This is due to the fact that the terms that label the edges can be arbitrary large and hence arbitrarily many. In a pure system for this rare case, the user would have to use one of many heuristics provided in [2], [18]. The proposed hybrid tool gives ways to overcome the problem.

There has been a great deal of effort combining model checking tools with proof systems. Similar work to ours, though based on binary decision diagrams rather than multiway ones, includes Rajan *et al.*'s [14] integration of a propositional μ-calculus model checker with PVS, and Schneider and Hoffmann [13] who linked the CTL model checker SMV to HOL. Gordon [6] took a different approach with the BuDDy BDD package, providing a secure and general programming infrastructure to allow users to implement their own BDD-based verification algorithms integrated within the HOL system rather than tools being linked externally. Forte [10], based on the work of Aagarad *et al.* [1] is one of the maturest formal verification environments based on tool integration including simulation. It has been used in large-scale industrial verification projects at Intel. Its power comes from the very tight integration of the two provers, using a single functional language, as both the theorem prover's meta-language and its object language.

The tool described here extends the capabilities of an earlier HOL-MDG tool and methodology [15], [9] for hierarchical hardware verification. The main contribution of the current work is that our hybrid tool supports the *abstract* datatypes of MDG in addition to concrete (enumeration/Boolean) sorts in [9], [15]. This allows abstract designs to be passed from HOL to MDG for verification. This allows, for example, larger data paths to be dealt with automatically than with a BDD based linkage. In particular, we extended a previous HOL formalization of the MDG modeling language, MDG-HDL [12]. We also implemented an interface that automatically supports the communication between the MDG and HOL tools. It generates the necessary MDG files from the HOL files, passing them to the model checker, takes back the MDG results, interprets them, and finally submits them to HOL in an appropriate form (see Figure 1). The tool supports both equivalence checking and *model checking* of abstract designs: a further extension of the original hybrid tool. This involved embedding the MDG temporal property specification language, $\mathcal{L}_{mdg}$ in HOL. An additional novel aspect is the explicit support of model reduction in HOL based on the natural design hierarchy and the specification being verified.

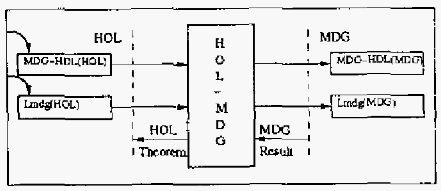

Fig. 1. The Hybrid Tool Overview

The rest of the paper is organized as follows. In Section 2, we present the proposed hybrid verification procedure. Section 3 describes the internal structure of the hybrid tool. In Section 4, we display some sample experimental results. Finally, Section 5 concludes the paper.

II. Hybrid Verification with HOL-MDG

The hybrid tool developed consists of an interface integrating the HOL theorem prover and the MDG model checker. During the verification procedure, the user deals mainly with HOL. As shown in Figure 2, the user starts by giving the HOL design model, property specification, and the goal to be proven. The respective MDG files (property specification, design model, symbol order, algebraic specification, and fairness constraints) are generated automatically and sent to the MDG tool for model checking. If the property holds, a HOL theorem is created. This could be used in higher HOL proofs, for example proving theorems about the consequences of the properties. If the verification within the MDG tool fails (due to the property checking to false, non-termination or state explosion), we have to perform the proof interactively using the theorem prover.

The tool does not accept any arbitrary HOL specification: only MDG-style models and properties using the embedded HOL theories presented. The HOL goal should also be an implication, where the MDG model checking result is converted to a form that can be used in HOL to infer the properties from the design model [16].

Our hybrid tool also supports hierarchical verification, where it is able to extract in HOL the block about which we want to check a property, then generating files of the specific block only. This is achieved by defining the structure "block" in a recursive manner. So, for each block, we are able to determine its subblocks. Hence, the model checker deals with the verification of the considered block only, not the whole design. As a result, we save on model size without constraining the user to write another specification for the appropriate block. This idea of program slicing is well-known in the model checking literature [4]. The difference in our work is the fact that the "slices" are extracted while expanding the proof goal by the theorem prover HOL, and based on the definition of the design block. In our approach, it is therefore done formally within HOL rather than informally outside the tool.

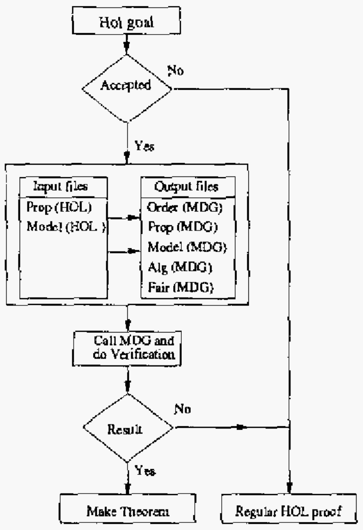

Fig. 2. Verification Procedure with the Hybrid Tool

III. HOL-MDG Hybrid Tool Structure

Our hybrid tool is written in SML. It is composed of five main modules: the *Hybrid Tool Interface*, the *Property Module*, the *Description File Module*, the *HOL Goal Parser Module* and the *MDG Interaction Module* (Figure 3). The user's interface to the hybrid tool is a Java GUI, responsible for getting the HOL goal, the property file and the model description file, passing them to HOL, loading the $\mathcal{L}_{mdg}$ and MDG-HDL theories and at the end of the verification process, communicating the result to the user [7]. The user thus sees the hybrid tool as an integrated system but one that is more powerful than MDG alone. In the second module, the *Property Parser* generates as output a data structure from which the *MDG File Generator* produces the MDG property file, and the *Property Type Generator* provides the property type. The latter contains information about the type of property submitted to the tool, according to which, it calls the appropriate property checking algorithm. The *Description File Module* flattens the specification by removing hierarchy.

When parsing the goal, we obtain the name of the property and the block to check. The latter can be either the main module in the model description or one of its submodules. If the specification is written in a hierarchical way, it is possible to extract the target module, and its submodules, discarding the others. The *Block Extraction Module* achieves this task. In the next step, the corresponding MDG files are generated, including *MDG model* and *MDG property* files, an *algebraic* file containing sorts, functions, and rewriting rules, an *order* file, giving a total order of variables and function symbols, and eventually *fairness* files, each describing an imposed fairness constraint. The MDG file generation is done automatically. The HOL specification file contains two main parts. The first is dedicated to the definition of the different sorts, functions, and MDG terms used. The second is dedicated to the tables definitions. Using a syntactical analysis of the submitted HOL files, our tool extracts the useful information from them to generate the MDG files in the appropriate MDG-HDL syntax.

Before proceeding with the model checking operation, the MDG tool has to encode the MDG-HDL syntax to generate ASMs. Since we wanted the communication between the linked tools to be automatic, we implemented a special module, called the *ASM Generation Interface* that implicitly does the appropriate MDG instructions. The *MDG Interaction Module* does the communication with MDG. It takes all the generated MDG files, the property type and the fairness number. The latter are provided by the property parser module. They indicate respectively the number of fairness constraints in the HOL property, if they exist, and its temporal type. All these files are supplied to the MDG tool, which performs the verification process and passes the result to HOL through

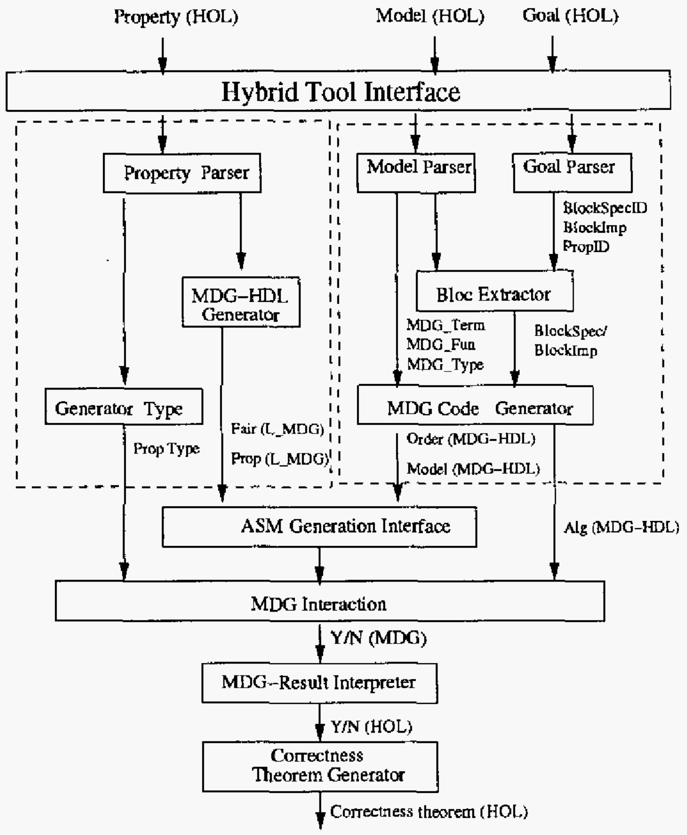

Fig. 3. Hybrid Tool Structure

the *MDG Result Interpreter Module*. If the property holds, a theorem is generated in HOL.

IV. Experimental Results

We have experimented with our hybrid tool using a number of benchmark designs including the Island Tunnel Controller (ITC) [12] (Figure 4), which experimental results we report here. The ITC controls the traffic lights at both ends of a tunnel connecting a mainland and island. It was chosen for two reasons. First, its specification contains abstract sorts and functions. It was not possible to express the specification of this example in the tool in [9]. Second, the same example was verified in [18], where the authors faced a problem of non-termination in the Island Counter module. The hybrid tool offers the solution of doing a hybrid verification, such that the subblocks causing the non-termination problem are verified within the HOL theorem prover interactively, while those which do not are verified within the MDG model checker.

The input specifications for the ITC were written in HOL, using the HOL MDG-HDL theory [12]. It is composed of a term declaration of the MDG part, the different table specifications and the main modules. The specification is written in a hierarchical way. Each component is represented by the conjunction of its tables. The whole system therefore is the conjunction of the five mentioned blocks.

Experimental results on the verification of a set of properties are given in Table I. It gives CPU time, verification memory usage and number of MDG nodes generated as well as the number of components and signals of the reduced (extracted) design model effectively used for model checking in MDG. It is clear that verification is much faster than doing the proof interactively with HOL. At the bottom of Table I, we give the example experimental results of checking Property 1 and Property 3 without block extraction done in the theorem prover side, i.e., on the whole model. We can clearly see that the CPU time and memory consumption were decreased by more than half in the former case, which is due to the block extraction. The results here are similar to those in [17], where only the MDG tool is used on the full model. This fact proves that our hybrid tool achieves the verification without obstructing the model checker.

V. Conclusions

We presented in this paper a hybrid verification tool integrating the HOL theorem prover and the MDG model checker. In an earlier HOL-MDG tool, where HOL and the MDG equivalence checker were linked, neither abstract data sorts nor abstract functions, were supported. The main contribution

TABLE I

EXPERIMENTAL RESULTS ON THE ITC

Property	CPU$_{(s)}$	Memory$_{(MB)}$	MDG Nodes	#Components	#Signals
Property1	0.32	0.66	318	18	32
Property2	0.36	0.77	313	13	31
Property3	0.41	0.73	401	16	34
Property4	1.12	1.91	1266	13	29
Property5	0.91	1.26	1027	10	26
Property6	0.93	1.77	1166	13	29
Property7	1.15	1.39	11002	16	33
Property8	1.15	1.39	11002	16	33
Property1(*)	0.74	1.38	870	26	62
Property3(*)	0.87	1.46	1027	26	62

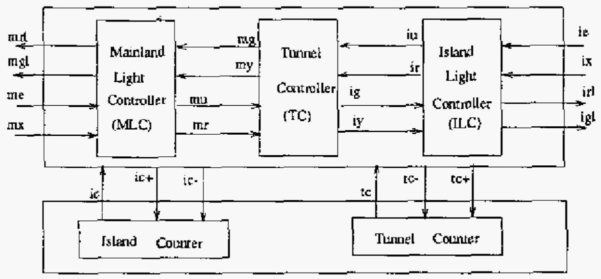

Fig. 4. ITC Structure

of our work is the extension of this tool to handle these main features of MDG compared to BDD based model checkers as with other tools. Our system handles abstraction for model checking and equivalence checking. Furthermore, it directly supports hierarchical proof to be conducted saving verification time and memory usage. It also provides a way for overcoming the non-termination problem of MDG. The tool has been tested on several benchmark examples, including the Island Tunnel Controller reported here. In a future work, we intend to apply our tool on more complex designs as well as looking into ways to render the MDG-HOL specification templates more user-friendly.

REFERENCES

[1] M.D. Aagaard, R. Jones, and C. Seger. Lifted-FL: A Pragmatic Implementation of Combined Model Checking and Theorem Proving. In *Theorem Proving in Higher Order Logics*, LNCS 1690, pages 323-340, Springer Verlag, 1999.

[2] O. Ait Mohamed, X. Song, and E. Cerny. On the Non-termination of MDG-based Abstract State Enumeration. In *Theoretical Computer Science*, 300: 161-179, 2003.

[3] R. Bryant. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams. In *ACM Computing Surveys*, 24(3):293-318, September 1992.

[4] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

[5] F. Corella, Z. Zhou, X. Song, M. Langevin, and E. Cerny. Multiway Decision Graphs for Automated Hardware Verification. In *Formal Methods in System Design*, 10(1):7-46, 1997.

[6] M. Gordon. Combining Deductive Theorem Proving with Symbolic State Enumeration. 21 Years of Hardware Formal Verification. Royal Society Workshop to mark 21 years of BCS FACS, U.K., December 1998.

[7] R. Hun, H. Yip, H. Li, R. Mizouni, and S. Tahar. A GUI for linking HOL to MDG. Technical report, ECE Dept., Concordia University, June 2002.

[8] J. Joyce and C. Seger. The HOL-Voss System: Model-Checking inside a General Purpose Theorem-Prover. In *Higher Order Logic Theorem Proving and Its Applications*, LNCS 780, pages 185-198, Springer Verlag, 1994.

[9] I. Kort, S. Tahar, and P. Curzon. Hierarchical Formal Verification Using a Hybrid Tool. *Software Tools for Technology Transfer*, 4(3):313-322, May 2003, Springer Verlag.

[10] T. Melham. Integrating Model Checking and Theorem Proving in a Reflective Functional Language. In *Integrated Formal Methods*, LNCS 2999, pages 36-39, Springer Verlag, 2004.

[11] T. Melham and M. Gordon. *Introduction to Higher Order Logic, Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[12] R. Mizouni. Linking HOL Theorem Proving and MDG Model Checking. Master's thesis, Electrical and Computer Engineering Dept., Concordia University, 2003.

[13] K. Schneider and D. Hoffmann. A HOL Conversion for Translating Linear Time Temporal Logic to ω-Automata. In *Theorem Proving in Higher Order Logics*, LNCS 1690, pages 255-272, Springer Verlag, 1999.

[14] S. Rajan, N.Shankar, and M.Srivas. An Integration of Model-Checking with Automated Proof Checking. In *Computer Aided Verification*, LNCS 939, pages 84-97, Springer Verlag, 1995.

[15] V.K. Pisini, S. Tahar, O. Ait-Mohamed, P. Curzon, and X. Song. Formal Hardware Verification by Integrating HOL and MDG. In *ACM 10th Great Lakes Symposium on VLSI*, pages 23-28, Chicago, Illinois, USA, 2000.

[16] H. Xiong, P. Curzon, and S. Tahar. Importing MDG Verification Results into HOL. *In Theorem Proving in Higher Order Logics*, LNCS 1690, pages 293-310, Springer Verlag, 1999.

[17] Y. Xu. Model Checking for a First-Order Temporal Logic Using Multiway Decision Graphs. PhD Thesis, University of Montreal, Canada, April 1999.

[18] Z. Zhou, X. Song, S. Tahar, E. Cerny, F. Corella, and M. Langevin. Formal Verification of the Island Tunnel Controller Using Multiway Decision Graphs. In *Formal Methods in Computer-Aided Design*, LNCS 1166, pages 233-247, Springer Verlag, 1996.