Importing MDG Verification Results into HOL

Haiyan Xiong[1], Paul Curzon[1], and Sofiène Tahar[2]

[1] School of Computing Science, Middlesex University, London, UK
{h.xiong, p.curzon}@mdx.ac.uk
[2] ECE Department, Concordia University, Montreal, Canada.
tahar@ece.concordia.ca

Abstract. Formal hardware verification systems can be split into two categories: theorem proving systems and automatic finite state machine based systems. Each approach has its own complementary advantages and disadvantages. In this paper, we consider the combination of two such systems: HOL (a theorem proving system) and MDG (an automatic system). As HOL hardware verification proofs are based on the hierarchical structure of the design, submodules can be verified using other systems such as MDG. However, the results of MDG are not in the appropriate form for this. We have proved a set of theorems that express how results proved using MDG can be converted into the form used in traditional HOL hardware verification.

1 Introduction

In general, machine-assisted hardware verification methods can be classified into two categories: interactive verification using a theorem prover and automated finite state machine (FSM) verification based on state enumeration. This study investigates the combination of two such systems: the HOL and MDG systems. The former is an interactive theorem proving system based on higher-order logic [8]. The latter is an automatic system based on Multiway Decision Graphs [2]. Tahar and Curzon [12] compared these two systems based on using both to independently verify real hardware: the Fairisle 4 by 4 switch fabric. Their results indicate that both systems are very effective and they are complementary. By combining them, it is hoped that the advantages of both can be obtained.

The MDG system is a hardware verification system based on Multiway Decision Graphs (MDGs). MDGs subsume the class of Bryant's Reduced Ordered Binary Decision Diagrams (ROBDD) [1] while accommodating abstract sorts and uninterpreted function symbols. The system combines a variety of different hardware verification applications implemented using MDGs [15]. The applications developed include: combinational verification, sequential verification, invariant checking and model checking.

The MDG verification approach is a black-box approach. During the verification, the user does not need to understand the internal structure of the design being verified. The strength of MDG is its speed and ease of use. However, it does not scale well to complex designs. In general, BDD based systems cannot cope with designs that combine datapaths and control hardware. MDG overcomes some of these problems. However, the largest example verified to date is the Fairisle 4 by 4 fabric [13].

In HOL, the specification language is higher-order logic. It allows functions and relations to be passed as arguments to other functions and relations. Higher-order logic is very flexible and has a well-defined and well-understood semantics. It also allows us to use a hierarchical verification methodology that effectively deals with the overall functionality of designs with complex datapaths. Designs that combine control hardware and datapaths can be verified. HOL scales better than MDG as illustrated by the fact that a 16 by 16 switch fabric constructed from elements similar to the 4 by 4 fabric has been verified in HOL [3] [4]. This is beyond the capabilities of MDG on its own. To complete a verification, however, a very deep understanding of the internal structure of the design is required, as it is a white-box approach. This enables the designer to gain greater insight into the system and thus achieve better designs. However, the learning curve is very steep and modeling and verifying a system is very time-consuming. The HOL system is generally better for higher-level reasoning in a more abstract domain.

Can we combine the two systems to reap the advantages of both? If we could, the problem size and complexity limits that can be handled in practice would be increased. We cannot, however, just accept that a piece of hardware verified using an automated verification tool such as the MDG system can be assumed correct in a HOL proof. In this paper, we focus on the theoretical underpinning of how to convert MDG results into HOL. In particular, we consider how to convert MDG results to appropriate HOL theorems as used in a traditional HOL hardware verification in the style of Gordon [6]. We give formalizations of MDG results in HOL based on the semantics of the MDG input language. We then suggest versions of these results that are of the form needed in a HOL hardware verification. Finally, we derive theorems that show that we can convert between these two forms. Thus, these theorems provide the specification for how MDG results can be imported into the HOL system in a useful form. This work is one step of a larger project to verify aspects of the MDG system in HOL so that MDG results can be trusted in the HOL system. The work presented here thus integrates with previous work to verify the MDG components library in HOL [5] and work to verify the MDG-HDL compiler.

Whilst this work concentrates on the MDG and HOL systems, the work has a much wider applicability. The theorems proved could be applicable for other verification systems with similar architectures based on reachability analysis or equivalence checking. Furthermore, the general approach taken is likely to be applicable to verification systems with different architectures.

The structure of this paper is as follows: in Section 2, we review related work. In Section 3, we overview the hierarchical hardware verification approach in HOL and motivate the need for MDG results to be in a particular form when importing them into the HOL system. In Section 4, we give the formal theorems that convert the MDG results into useful HOL theorems. These theorems have been verified using HOL. Our conclusions are presented in Section 5. Finally, ideas for further work are presented in Section 6.

2 Related Work

In 1993, Joyce and Seger [10] presented a hybrid verification system: HOL-Voss. In their system, several predicates were defined in the HOL system, which presents a mathematical link between the specification language of the Voss system (symbolic trajectory evaluation) and the specification language of the HOL system. A tactic VOSS_TAC was implemented as a remote function. It calls the Voss system that is then run as a child process of the HOL system. The Voss assertion can be expressed as a term of higher-order logic. Symbolic trajectory evaluation is used to decide whether or not the assertion is true. If it is true, then the assertion will be transformed into a HOL theorem and this theorem can be used by the HOL system to derive additional verification results. Zhu et al. [16] successfully applied HOL-Voss for the verification of the Tamarack-3 microprocessor.

Rajan et al. [11] proposed an approach for the integration of model checking with PVS: an automated proof checking system. The mu-calculus was used as a medium for communicating between PVS and a model checker. It was formalized by using the higher-order logic of PVS. The temporal operators that apply to arbitrary state spaces are given the customary fixpoint definitions using the mu-calculus. The mu-calculus expression was translated to an input that is acceptable by the model checker. This model checker was then used to verify the subgoals. In [9], a complicated communication protocol was verified by means of abstraction and model checking.

More recently, HOL98 has been integrated with the BuDDy BDD package [7]. HOL was used to formalize the QBF (Quantified Boolean Formulae) of BDDs. The formulae can be interactively simplified by using a higher-order rewriting tool such as the HOL simplifier to get simplified BDDs. A table was used to map the simplified formulae to BDDs. The BDD algorithms can also strengthen its deductive ability in this system.

In the work presented in this paper, we are not using the MDG system as an oracle to then prove results, already determined, by primitive inference in HOL, nor are we using HOL to improve the way MDG works. Furthermore, we are not just farming out general lemmas (e.g., propositional tautologies) that arise whilst verifying a particular hardware module and that can be proved more easily elsewhere. Our work is perhaps closer in spirit to that of the HOL-VOSS system than to other work in this sense. We are concerned with linking HOL to a dedicated hardware verification system that is in direct competition with it. It produces similar results about similar descriptions of circuits. We utilize this fact to allow MDG to be used when it would be easier than obtaining the result directly in HOL. The main contribution of this paper is that we present a methodology by which this can be done formally. We do not simply assume that the results proved by MDG are directly equivalent to the result that would have been proved in HOL.

3 Hierarchical Verification in a Combined System

In this section, we motivate the need for the results from a system such as MDG to be in a specific form by outlining the traditional HOL hierarchical hardware verification methodology. We also look at how an MDG result might be incorporated into such a verification approach.

Generally, when we use HOL to verify a design, the design is modeled as a hierarchy structure with modules divided into submodules as shown in Figure 1. The submodules are repeatedly subdivided until eventually the logic gate level is reached. Both the structure and behavior specifications of each module are given as relations in higher-order logic. The verification of each module is carried out by proving a theorem asserting that the implementation (its structure) implements (implies) the specification (its behavior). That is:

$$\vdash \textit{implementation} \supset \textit{specification} \tag{1}$$

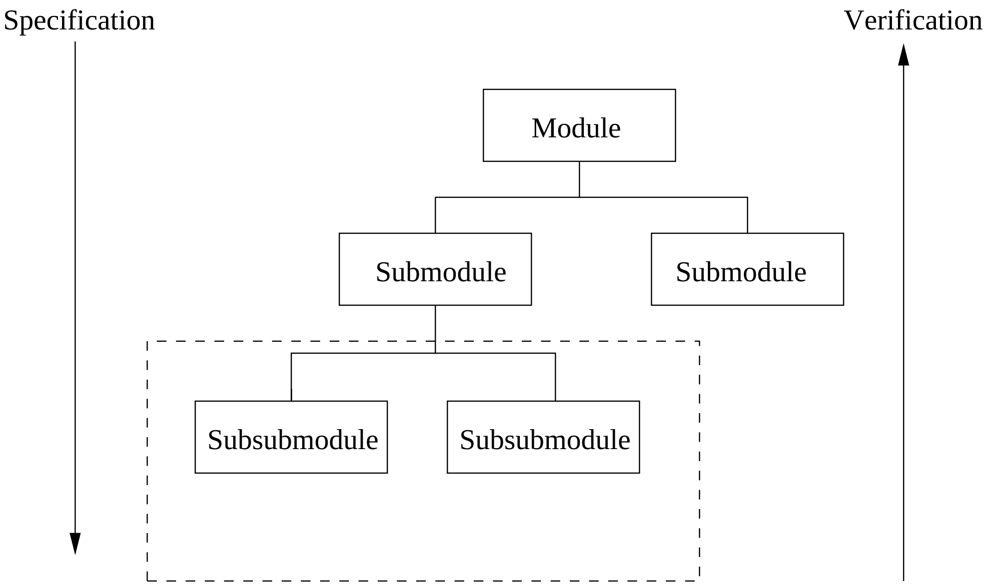

Fig. 1. Hierarchical Verification

The correctness theorem for each module states that its implementation down to the logic gate level satisfies the specification. The correctness theorem for each module can be established using the correctness theorems of its submodules. In this sense the submodule is treated as a black-box. A consequence of this is that different technologies can be used to address the correctness theorem for the submodules. In particular, we can use the MDG system instead of HOL to prove the correctness of submodules.

In order to do this, we need to formalize the results of the MDG verification applications in HOL. These formalizations have different forms for the different verification applications, i.e., combinational verification gives a theorem of one form, sequential verification gives a different form and so on. However, the most natural and obvious way to formalize the MDG results does not give theorems of the form that HOL needs if we are to use traditional HOL hardware verification techniques. We therefore need to be able to convert the MDG results into a form that can be used. In other words, we need to prove a series of translation theorems (one for combinational verification, one for sequential verification, etc.) that state how an MDG result can be converted to the traditional HOL form[1]:

$$\vdash \textit{Formalized MDG result} \supset (\textit{implementation} \supset \textit{specification}) \tag{2}$$

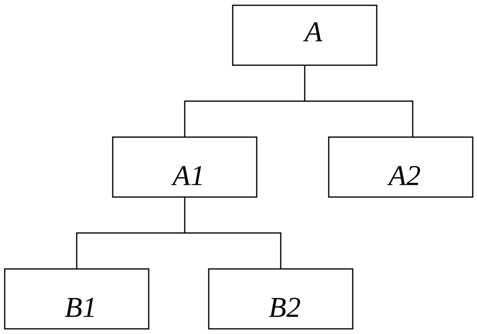

Fig. 2. The Hierarchy of Module A

To illustrate why we need a particular form of result in HOL consider the HOL verification of a system A. A theorem that the implementation satisfies its specification needs to be proved, i.e.

$$\vdash A\_imp \supset A\_spec \tag{3}$$

where A_imp and A_spec express the implementation and specification of system A, respectively. Suppose system A consists of two subsystems $A1$ and $A2$ and

[1] In this discussion, we have simplified the presentation for the purposes of exposition. In particular details of inputs and outputs are omitted.

$A1$ is further subdivided as shown in Figure 2. The structural specification of A will be defined by the equation:

$$\vdash A\_imp = A1\_imp \wedge A2\_imp \tag{4}$$

where $A1\_imp$ is defined in a similar way. Thus (3) can be rewritten to

$$\vdash A1\_imp \wedge A2\_imp \supset A\_spec \tag{5}$$

The correctness theorem of the system A can be proved using the correctness statements about its subsystems. In other words, we independently prove the correctness theorems:

$$\vdash A1\_imp \supset A1\_spec \tag{6}$$

$$\vdash A2\_imp \supset A2\_spec \tag{7}$$

As these are implications, to prove (5) it is then sufficient to prove

$$\vdash A1\_spec \wedge A2\_spec \supset A\_spec \tag{8}$$

Thus we verify A by independently verifying its submodules, then treating them as black-boxes using the more abstract specification of $A1$ and $A2$ to verify A.

Suppose now that $A1$ was verified using MDG instead of HOL, but that we still wish to use the result in the verification of A. To make use of the result, we need MDG to also prove results of the form

$$\vdash A1\_imp \supset A1\_spec \tag{9}$$

so that the implementation can be substituted for a specification. However, results from MDG are not of this form[2]. For example, with sequential verification MDG proves a result about "reachable states" of a product machine. We need to show how such a result can be expressed as an implication about the actual hardware under consideration as above. If $A1\_MDG\_RESULT$ is such a statement about a product machine, then we need to prove

$$\vdash A1\_MDG\_RESULT \supset (A1\_imp \supset A1\_spec) \tag{10}$$

Theorems such as this convert MDG results to the appropriate form to make the step between (5) and (8).

Ideally, we want a general theorem of this form that applies to any hardware verified using MDG's sequential verification tool. We also want similar results for the other MDG verification applications. In this paper, we prove such translation theorems for a series of MDG applications. This is described in the next section.

4 The Translation Theorems

In this section, we consider each of the verification applications of the MDG system in turn, describing the conversion theorem required to convert results to a form useful within a HOL proof. Each of these theorems has been proved within the HOL system.

[2] We give details of the form of theorems that MDG does prove in the next section.

4.1 Combinational Verification of Logic Circuits

The simplest verification application of MDG is the checking of equivalence of input-output for two combinational circuits. A combinational circuit is a digital circuit without state-holding elements or feedback loops, so the output is a function of the current input. The MDGs representing the input-output relation of each circuit are computed by a relational product algorithm to form the MDGs of the components of the circuit. Because an MDG is a canonical representation, we can check whether the two MDGs are isomorphic and so the circuits are equivalent. It is simple to formalize this in HOL. We use $M(ip, op)$ and $M'(ip, op)$ to represent the circuits (machines) being compared. M is a relation on input traces (given by ip) and output traces (given by op). The relation is true if op represents a possible output trace for the given input trace ip and is false otherwise. M' is a similar relation on inputs (ip) and outputs (op). An MDG combinational verification result can be formalized as:

$$\vdash \forall\ ip\ op.\ M\ (ip,\ op)\ =\ M'\ (ip,\ op) \tag{11}$$

It verifies that the two circuits are identical in behavior for all inputs and outputs. If ip and op are possible input and output traces for M, then they are also possible traces for M', and vice versa. This is not in the form of an implication as described above. However, the MDG result does not need to be converted to a different form for it to be useful in a HOL hardware verification, since an equality can be used just as well as an implication.

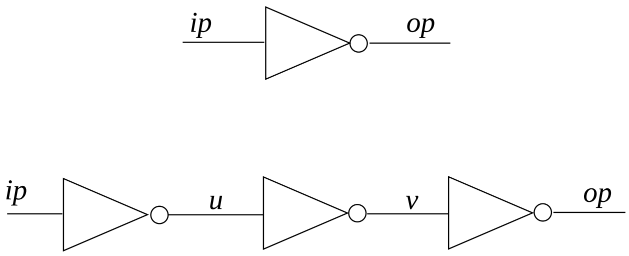

Fig. 3. Two Equivalent Combinational Circuits

Example 1. Consider the two circuits shown in Figure 3. Assume they have been verified to be equivalent using MDG combinational equivalence checking. We will show in the following how to convert the MDG result to a useful HOL theorem.

The first circuit is a single NOT gate that can be specified as:

$$\vdash \forall\ in\ op.\ \mathrm{NOT}\ (ip,\ op)\ =\ (\forall\ t.\ op\ t\ = \sim ip\ t)$$

The second circuit consists of three NOT gates in series and can be formalized as:

$$\vdash \forall\ ip\ op.\ \ \mathrm{NOT3}\ (ip,\ op)\ \ = \\ \exists\ u\ v.\ \mathrm{NOT}\ (ip,\ u) \wedge \mathrm{NOT}\ (u,\ v) \wedge \mathrm{NOT}\ (v,\ op)$$

The MDG verification result can be stated as

$$\vdash \forall\ ip\ op.\ \mathrm{NOT3}\ (ip,\ op)\ = \mathrm{NOT}\ (ip,\ op)$$

This theorem has the form that we need in the HOL verification system, so no translation theorem is required

4.2 Combinational Verification of Sequential Circuits

Combinational verification can also be used to compare two sequential circuits when a one-to-one correspondence between their registers exists and is known. In this situation M and M' are relations on inputs (ip), outputs (op) and states (s). The result of the MDG proof can then be stated as:

$$\vdash \forall\ ip\ op\ s.\ M\ (ip,\ op,\ s)\ \ =\ \ M'\ (ip,\ op,\ s) \tag{12}$$

This is explicitly concerned with state in the form of the variable s. In a HOL verification the way we model hardware by a relation between input and output traces means that we do not, in general, need to model the state explicitly. Traces are described as history functions giving the value output at each time instance. A register can then, for example, be specified as

$$\vdash \mathrm{REGH}\ (ip,\ op) = (op\ 0 = \mathrm{F}) \wedge (\forall\ t.\ (op\ (t+1) = ip\ t)) \tag{13}$$

There is no explicit notion of state in this definition—we just refer to values at an earlier time instance.

MDG descriptions in MDG-HDL on the other hand explicitly include state: state variables are declared and state transition functions given. The MDG version could be formalized in HOL by

$$\vdash \mathrm{REGM}\ (ip, op, s) = \mathrm{INIT}\ s \wedge \mathrm{DELTA}\ (ip, s) \wedge \mathrm{OUT}\ (ip, op, s) \tag{14}$$

where

$$\begin{array}{l} \mathrm{INIT}\ s = (s\ 0 = \mathrm{F}) \\ \mathrm{DELTA}\ (ip,\ s) = (\forall\ t.\ s\ (t+1) = ip\ t) \\ \mathrm{OUT}\ (ip,\ op,\ s) = (\forall\ t.\ op\ t = s\ t) \end{array}$$

We therefore need a way of abstracting away this state when converting to the HOL form. As was explained in Section 3, ultimately the correctness theorem that HOL wants should have the form of (1). We can hide the state using existential quantification and obtain:

$$\vdash \forall\ ip\ op.\ (\exists\ s.\ M\ (ip,\ op,\ s))\ \supset (\exists\ s.\ M'\ (ip,\ op,\ s)) \tag{15}$$

This is of the required form:

$$M\_imp\ (ip,\ op) \supset M\_spec\ (ip,\ op)$$

where

$$\begin{aligned} M\_imp\ (ip,\ op)\ &= (\exists\ s.\ M\ (ip,\ op,\ s)) \\ M\_spec\ (ip,\ op)\ &= (\exists\ s.\ M'\ (ip,\ op,\ s)) \end{aligned}$$

In this situation the converting theorem is:

$$\begin{aligned} &\vdash \forall\ M\ M'. \\ &\quad (\forall\ ip\ op\ s.\ M\ (ip,\ op,\ s) = M'\ (ip,\ op,\ s)) \supset \\ &\qquad (\forall\ ip\ op.\ (\exists\ s.\ M\ (ip,\ op,\ s)) \supset (\exists\ s.\ M'\ (ip,\ op,\ s))) \end{aligned} \tag{16}$$

We have proved this theorem in HOL. Note that the relations M and M' are universally quantified variables. The theorem thus applies to any hardware for which an MDG result is verified.

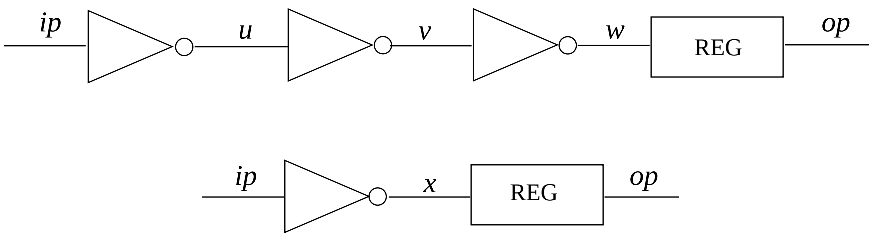

Fig. 4. Two Equivalent Sequential Circuits

Example 2. Consider verifying the sequential circuits in Figure 4 using combinational equivalence. We check that three not gates and a register are equivalent to a single not gate and register. We use REGNOT3M to formalize the first circuit,

$$\begin{aligned} &\vdash\ \mathrm{REGNOT3M}(ip,\ op,\ s)\ = \\ &\quad \exists\ u\ v\ w. \\ &\quad\ \mathrm{NOT}\ (ip,\ u) \wedge \mathrm{NOT}\ (u,\ v) \wedge \mathrm{NOT}\ (v,\ w) \wedge \mathrm{REGM}\ (w,\ op,\ s) \end{aligned}$$

We use REGNOTM to formalize the second circuit,

$$\vdash \mathrm{REGNOTM}\ (ip,\ op,\ s)\ =\ \exists\ x.\ \mathrm{NOT}\ (ip,\ x) \wedge \mathrm{REGM}\ (x,\ op,\ s)$$

Suppose we have verified that these two circuits are equivalent using the MDG system. The MDG verification result can be stated as:

$$\vdash \forall\, ip\ op\ s.\ \mathrm{REGNOT3M}\ (ip,\ op,\ s)\ =\ \mathrm{REGNOTM}\ (ip,\ op,\ s)$$

Combining this with our conversion theorem (16), we obtain

$$\begin{aligned}&\vdash\ \forall\, ip\ op.\\ &\quad (\exists\, s.\ \mathrm{REGNOT3M}\ (ip,\ op,\ s))\ \supset\ (\exists\, s.\ \mathrm{REGNOTM}\ (ip,\ op,\ s))\end{aligned} \tag{17}$$

This is not quite the theorem we would have proved if the verification was done directly in HOL. However, it can be obtained if we first prove the theorems:

$$\vdash \mathrm{REGNOT3H}\ (ip,\ op)\ =\ (\exists\, s.\ \mathrm{REGNOT3M}\ (ip,\ op,\ s)) \tag{18}$$
$$\vdash \mathrm{REGNOTH}\ (ip,\ op)\ =\ (\exists\, s.\ \mathrm{REGNOTM}\ (ip,\ op,\ s)) \tag{19}$$

where REGNOT3H and REGNOTH are stateless HOL descriptions of the corresponding circuits[3]. They are defined as follows:

$$\begin{aligned}&\vdash \mathrm{REGNOT3H}\ (ip,\ op)\ =\\ &\qquad \exists\, u\ v\ w.\\ &\qquad\quad \mathrm{NOT}\ (ip,\ u) \land \mathrm{NOT}\ (u,\ v) \land \mathrm{NOT}\ (v,\ w) \land \mathrm{REGH}\ (w,\ op)\\ &\vdash \mathrm{REGNOTH}\ (ip,\ op)\ =\\ &\qquad\qquad \exists\, x.\ \mathrm{NOT}\ (ip,\ x) \land \mathrm{REGH}\ (x,\ op)\end{aligned}$$

Finally, using (18) and (19) to rewrite (17), we obtain the theorem which is needed in a traditional HOL verification.

$$\vdash \forall\, ip\ op.\ \mathrm{REGNOT3H}\ (ip,\ op)\ \supset\ \mathrm{REGNOTH}\ (ip,\ op)$$

It should be noted that the actual verification application of MDG does not do state traversal so the state is not actually used in the MDG verification process. However the MDG hardware description language (HDL) is still used as the description language. Therefore the explicit introduction of state is required if the relations are to represent semantic objects of MDG-HDL. This is of importance in our work since we ultimately intend to link these theorems with ones that explicitly refer to the semantics of MDG-HDL.

4.3 Sequential Verification

The behavioral equivalence of two abstract state machines (Figure 5) is verified by checking that the machines produce the same sequence of outputs for every sequence of inputs. The same inputs are fed to the two machines M and M' and then reachability analysis is performed on their product machine using an invariant asserting the equality of the corresponding outputs in all reachable

[3] The need for such theorems will be discussed further in Section 6.

states. This effectively introduces new "hardware" (see Figure 5) which we refer to here as PSEQ (the Product machine for SEQuential verification). PSEQ has the same inputs as M and M', but has as output a single Boolean signal ($flag$). The outputs *op* and *op*$'$ of M and M' are input into an equality checker. On each cycle, PSEQ outputs true if *op* and *op*$'$ are identical at that time, and false otherwise. PSEQ can be formalized as

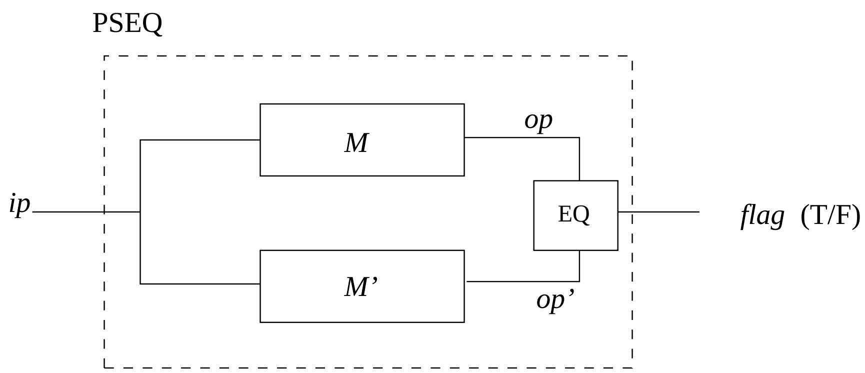

Fig. 5. The Product Machine used in MDG Sequential Verification

$$\begin{aligned}&\vdash \ \text{PSEQ} \ (ip,\ flag,\ op,\ op',\ s,\ s',\ M,\ M') = \\ &\qquad\qquad M\ (ip,\ op,\ s) \wedge M'\ (ip,\ op',\ s') \wedge \text{EQ}\ (op,\ op',\ flag) \qquad (20)\end{aligned}$$

where EQ is the equality checker defined as:

$$\vdash \text{EQ}\ (op,\ op',\ flag) = (\forall\ t.\ flag\ t\ =\ (op\ t\ =\ op'\ t)) \qquad (21)$$

The result that MDG proves about PSEQ is that the flag output is always true, i.e., the outputs are equal for all inputs. This can be formalized as

$$\begin{aligned}&\vdash \forall\ \ s\ s'\ ip\ op\ op'. \\ &\qquad \text{PSEQ}\ (ip,\ flag,\ op,\ op',\ s,\ s',\ M,\ M') \supset (\forall\ t.\ flag\ t\ = \text{T}) \qquad (22)\end{aligned}$$

Note that this is not of the form $P\_imp \supset P\_spec$, (i.e., implementation implies specification) for M and M' but is of that form for the fictitious hardware PSEQ. To make use of such a result in a HOL hardware verification, we need to convert it to that form for M and M'. This can be done in a series of steps starting from (22). Expanding the definitions and rewriting with the value of flag, we obtain

$$\begin{aligned}&\vdash \forall\ \ s\ s'\ ip\ op\ op'. \\ &\qquad M\ (ip,\ op,\ s) \wedge M'\ (ip,\ op',\ s') \supset (\forall\ t.\ op\ t\ =\ op'\ t) \qquad (23)\end{aligned}$$

i.e., we have proved a lemma:

$$
\begin{array}{l}
\vdash \ \forall\, M\ M'. \\
\quad (\forall\ s\ s'\ ip\ op\ op'. \\
\qquad \text{PSEQ}\ (ip,\ flag,\ op,\ op',\ s,\ s',\ M,\ M') \supset \forall\, t.\ flag\ t\ =\ \text{T}) \supset \\
\quad (\forall\ s\ s'\ ip\ op\ op'.\ M\ (ip,\ op,\ s) \wedge M'\ (ip,\ op',\ s') \supset (\forall\, t.\ op\ t\ =\ op'\ t))
\end{array} \qquad (24)
$$

This is still not in an appropriate form, however. We need to abstract away from the states as with combinational verification. The theorem should also be in the form of (1). The machine M can be considered as the structure specification (implementation) and machine M' the behavior specification (specification). Based on this consideration, the theorem that HOL needs is as follows:

$$\vdash \forall\ ip\ op.\ (\exists\ s.\ M\ (ip,\ op,\ s)) \supset (\exists\ s'.\ M'\ (ip,\ op,\ s')) \qquad (25)$$

i.e., for all input and output traces if there exists a reachable sequence of states s that satisfy the relation $M\ (ip,\ op,\ s)$, then must exist a reachable state s' that satisfies the relation $M'\ (ip,\ op,\ s')$. As mentioned above, the converting theorem from MDG to HOL should be in the form of (2). For sequential verification the conversion theorem should be

$$(22) \supset (25).$$

To prove this, given (24) it is sufficient to prove

$$(23) \supset (25).$$

However, this can only be proved with an additional assumption. Namely, for all possible input traces, the behavior specification M' can be satisfied for some output and state traces (i.e., there exists at least one output and state trace for which the relation is true):

$$\vdash \forall\ ip.\ \exists\ op'\ s'.\ M'\ (ip,\ op',\ s') \qquad (26)$$

This means that the machine must be able to respond whatever inputs are given. This should always be true for reasonable hardware. You should not be able to give inputs which break it. For any input sequence given to this machine, at least one output and state sequence will correspond. Therefore, we can actually only prove $\vdash (22) \wedge (26) \supset (25)$,

$$
\begin{array}{l}
\vdash \forall\ \ MM'. \\
\quad ((\forall\ s\ s'\ ip\ op\ op'. \\
\qquad \text{PSEQ}\ (ip,\ flag,\ op,\ op',\ s,\ s',\ M,\ M') \supset \forall\, t.\ flag\ t\ =\ \text{T}) \wedge \\
\quad (\forall\ ip.\ \exists\ op'\ s'.\ M'\ (ip,\ op',\ s'))) \supset \\
\qquad\quad (\forall\ ip\ op.\ (\exists\ s.\ M\ (ip,\ op,\ s)) \supset (\exists\ s'.\ M'\ (ip,\ op,\ s')))
\end{array} \qquad (27)
$$

With the same reasoning, the machine M' could have been considered as the structural specification and machine M could have been considered as the behavioral specification. We would then need the assumption

$$\vdash \forall\ ip.\ \exists\ op\ s.\ M\ (ip,\ op,\ s) \tag{28}$$

We would obtain the alternative conversion theorem (29)

$$\begin{aligned}
&\vdash \forall\ M M'.\\
&\quad ((\forall\ s\ s'\ ip\ op\ op'.\\
&\qquad \text{PSEQ}\ (ip,\ flag,\ op,\ op',\ s,\ s',\ M,\ M') \supset \forall\, t.\ flag\ t\ =\ \text{T}) \wedge\\
&\quad (\forall\ ip.\ \exists\ op\ s.\ M\ (ip,\ op,\ s))) \supset\\
&\qquad\quad (\forall\ ip\ op.\ (\exists\ s'.\ M'\ (ip,\ op,\ s')) \supset (\exists\ s.\ M\ (ip,\ op,\ s)))
\end{aligned} \tag{29}$$

Both these theorems have been verified in HOL. As with combinational verification, the universal quantification of M and M' means the theorems can be instantiated for any hardware under consideration. The symmetry in these equations is as might be expected given the symmetry of PSEQ.

Example 3. The circuits given in Figure 4 can also be verified using sequential verification. We shall show how to convert the result obtained to form a useful HOL theorem.

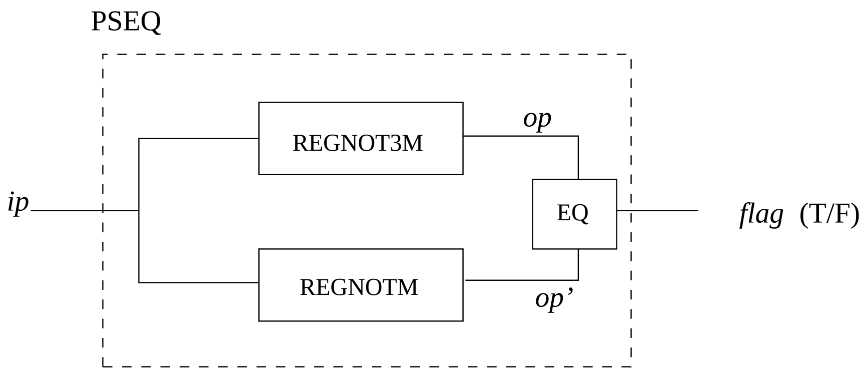

Fig. 6. The Machine used for Sequential Verification of the REGNOT3M Circuit

The MDG verification result can be stated as

$$\begin{aligned}
&\vdash \forall\ ip\ op\ s.\\
&\quad \text{REGNOT3M}\ (ip,\ op,\ s) \wedge \text{REGNOTM}\ (ip,\ op,\ s) \wedge \text{EQ}\ (op,\ op',\ flag) \supset\\
&\qquad\qquad (\forall\ t.\ flag\ t\ =\ \text{T})
\end{aligned}$$

We have proved the required theorem that states that the REGNOTM unit responds whatever inputs are given.

$$\vdash \forall\ ip.\ (\exists\ op'\ s'.\ \text{REGNOTM}\ (ip,\ op',\ s'))$$

Combining the above two theorems with our conversion theorem (27), we obtain:

$$\vdash \ \forall\, ip\ op. \\ \exists\, s.\ \text{REGNOT3M}\ (ip,\ op,\ s) \supset \exists s'.\ \text{REGNOTM}\ (ip,\ op,\ s') \tag{30}$$

Finally, after using (18) and (19) to rewrite (30), we obtain a theorem in a form that can be used in a HOL verification.

$$\vdash \ \forall\, ip\ op.\ \text{REGNOT3H}\ (ip,\ op) \supset\ \text{REGNOTH}\ (ip,\ op)$$

4.4 Invariant Checking.

Systems such as MDG also provide property/invariant checking. Invariant checking is used for verifying that a design satisfies some specific requirements. This is useful since it gives the designer confidence at low verification cost. In MDG, reachability analysis is used to explore and check that a given invariant (property) holds in all the reachable states of the sequential circuit under consideration, M. We consider one general form of property checking here.

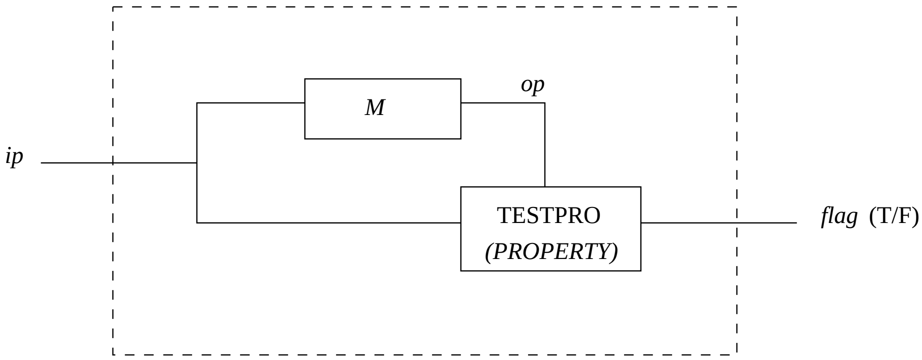

Fig. 7. The Machine Verified in Invariant Checking

As was the case for sequential verification, we introduce new "hardware" (see Figure 7) which we refer to as PINV (Product machine for INVariant checking). It consists of the original hardware and hardware representing the test property[4] wired together so that the property circuit has access to both the inputs and outputs of the circuit under test. PINV checks whether the outputs of the machine M satisfy the specific property or not. It is formalized as follows:

$$\vdash \text{PINV}\ \ (ip,\ flag,\ op,\ s,\ M,\ PROPERTY)\ = \\ M\ (ip,\ op,\ s) \wedge \text{TESTPRO}\ (ip,\ op,\ flag,\ PROPERTY) \tag{31}$$

[4] Invariants in MDG must be written in or converted to the same hardware description language as the actual hardware.

where

$$\vdash \text{TESTPRO}\ (ip,\ op,\ flag,\ PROPERTY) = \\ (\forall\, t.\ flag\ t\ =\ PROPERTY\ (ip\ t,\ op\ t)) \tag{32}$$

i.e., TESTPRO is a piece of hardware which tests if its inputs and outputs satisfy some specific requirements given at each time instance by *PROPERTY*. *PROPERTY* is a relation on input and output values. Again in discussing correctness it is actually a result about this different hardware that we obtain from the property checking. The result that the property checking proves about PINV can be stated as:

$$\vdash\ \forall\ ip\ s\ op\ M\ PROPERTY. \\ \text{PINV}\ (ip,\ flag,\ op,\ s,\ M,\ PROPERTY) \supset \forall\, t.\ flag\ t = \text{T} \tag{33}$$

i.e., its specification is that the $flag$ output should always be true. Note that this is not of the form (1) (i.e., implementation implies specification) for M but in that form for the fictitious hardware PINV. To make use of such a result in a HOL hardware verification we need to convert it to the form:

$$\vdash \forall\, ip\ op.\ \exists\, s.\ M\ (ip,\ op,\ s) \supset \forall\, t.\ PROPERTY\ (ip\ t,\ op\ t) \tag{34}$$

i.e., for all input and output sequences, if there exists a reachable state trace, s, satisfying the relation $M\ (ip,\ op,\ s)$ then the relation $PROPERTY$ must be true for the input and output values at all times. In other words, the machine M satisfies the specific requirement $\forall\ t.\ PROPERTY\ (ip\ t,\ op\ t)$. Hence the conversion theorem for invariant checking is:

$$\vdash\ \forall\ M\ PROPERTY. \\ (\forall\, ip\ op\ s. \\ (\text{PINV}\ (ip,\ flag,\ op,\ s,\ M,\ PROPERTY) \supset \forall\, t.\ flag\ t\ =\ \text{T})) \supset \\ (\forall\, ip\ op.\ \exists\, s.\ M\ (ip,\ s,\ op) \supset \forall\, t.\ PROPERTY\ (ip\ t,\ op\ t)) \tag{35}$$

We have proved this general conversion theorem in HOL. Once more the theorems can be instantiated for any hardware and property under consideration.

5 Conclusions

We have formally specified the correctness results produced by four different hardware verification applications using HOL. We have in each case proved a theorem that translates them into a form usable in a traditional HOL hardware verification, i.e., that the structural specification implements the behavioral specification. The first applications considered were the checking of input-output equivalence of two combinational circuits and the similar comparison of two sequential circuits when a one-to-one correspondence between their registers exists and is known. The next application considered was sequential verification, which checks that two abstract state machines produce the same sequence of outputs for every sequence of inputs. Finally, we considered a general form of the checking of invariant properties of a circuit.

The verification applications considered were based on those of the MDG hardware verification system. We have thus given a theoretical basis for converting MDG results into HOL. Furthermore, by proving these theorems in HOL itself we have given practical tools that can be used when verifying hardware using a combined system—MDG results can be initially imported into HOL as theorems in the MDG form and converted to the appropriate HOL form using the conversion theorems. This gives greater security than importing the theorems directly in the HOL form, as mistakes are less likely to be introduced. Alternatively, if theorems of the HOL form are created directly, then the conversion theorems provide the specification of the software that actually creates the imported theorem.

Whilst the verification applications were based on the MDG system and the proof done in HOL, the general approach could be applied to the importing of results between other systems. The results could also be extended to other verification applications. Furthermore, our treatment has been very general. The theorems proved do not explicitly deal with the MDG-HDL semantics or multiway decision graphs. Rather they are given in terms of general relations on inputs and outputs. Thus they are applicable to other verification systems with a similar architecture based on reachability analysis, equivalence checking and/or invariant checking.

The translation theorems are relatively simple to prove. The contribution of the paper is not so much in the proofs of the theorems, but in the methodology of using imported results presented. It is very ease to fall into the trap of assuming that because a result has been obtained in one system, an "obviously" corresponding result can be asserted in another. An example of the dangers is given by the extra assumption needed for sequential verification and invariant checking that the circuit verified can respond to any possible input. It could easily be overlooked. By formalizing the results in the most natural form of the verification application, and proving it equivalent to the desired form, we reduce the chances of such problems occurring.

6 Discussion and Further Work

The reason that the state must be made explicit in the formalism of MDG is that the semantics of MDG-HDL—the input language to the MDG system—has an explicit notion of state. We have used relations M and M' to represent MDG semantic objects. Ultimately we intend to combine the theorems described here with correctness theorems about the MDG-HDL compiler which translates MDG-HDL programs into decision graphs [5] [14]. This will provide a formal link between the low level objects actually manipulated by the verification system (and about which the verification results really refer) and the results used in subsequent HOL proofs. Compiler verification involves proving that the compiler preserves the semantics of all legal source programs. It thus requires the definition of both a syntax and semantics of HDL programs. Using an explicit notion of state in the translation theorems ensures they will be compatible with the semantics of the MDG-HDL language used in the compiler verification.

However, if proving results directly in HOL, as we saw, introducing such an explicit notion of state is unnecessary. We could do so for the convenience of combining systems. However, this would make subsequent specification and verification in HOL more cumbersome. Consequently, we hide the state in the conversion theorems, introducing terms such as:

$$\exists\ s.\ M(ip,\ op,\ s)$$

However, this implies that we also define HOL components in this way. For example, it suggests a register is defined as

$$\vdash \forall\ ip\ op.\ \mathrm{REGH}\ (ip,\ op)\ =\ \exists\ s.\ \mathrm{REGM}\ (ip,\ op,\ s) \tag{36}$$

where REGM is as defined in (14). We actually want to give and use definitions as in (13), however, and derive (36). As a consequence for any component verified in MDG we must prove a theorem that the two versions are equivalent as we did in Examples 2 and 3.

In general, we need to prove theorems of this form for each MDG-HDL basic component. We then need to construct a similar theorem for the whole circuit whose verification result is to be imported. Such proofs can be constructed from the theorems about individual components. For instance, we must prove for any network of components (n) that

$$\begin{aligned}\vdash \forall\ n.\ \mathrm{HOL\_DESCRIPTION}\ \ n\ (ip,\ op)\ =& \\ &\exists\ s.\ \mathrm{MDG\_DESCRIPTION}\ n\ (ip,\ op,\ s).\end{aligned}$$

If we prove this theorem, we can then convert our importing theorems into ones without state automatically. To do such a proof requires a syntax of circuits: precisely what is needed in the verification of the MDG compiler as noted above.

Acknowledgments

We are grateful to Prof. Ann Blandford for her helpful comments and suggestions. This work is funded by EPSRC grant GR/M4522, and a studentship from the School of Computing Science, Middlesex University. Travel funding was provided by the British Council, Canada.

References

1. R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computer Surveys*, 24(3), September 1992.

2. F. Corella, Z. Zhou, X. Song, M. Langevin, and E. Cerny. Multiway decision graphs for automated hardware verification. *Formal Methods in System Design*, 10(1):7–46, 1997.
3. P. Curzon. The formal verification of the Fairisle ATM switching element. Technical Report 329, University of Cambridge, Computer Laboratory, March 1994.
4. P. Curzon. Tracking design changes with formal machine-checked proof. *The Computer Journal*, 38(2), July 1995.
5. P. Curzon, S. Tahar, and O. Aït-Mohamed. The verification of aspects of a decision diagram based verification system. In Jim Grundy and Malcolm Newey, editors, *Theorem Proving in Higher-Order Logics: Emerging Trends*, pages 31–46. Department of Computer Science, The Australian National University, 1998.
6. M. J. C. Gordon. Why higher-order logic is a good formalism for specifying and verifying hardware. In G. J. Milne and P. A. Subrahmanyam, editors, *Formal Aspects of VLSI Design: the 1985 Edinburgh Workshop on VLSI*, pages 153–177. North-Holland, 1986.
7. M. J. C. Gordon. Combining deductive theorem proving with symbolic state enumeration. Presented at *21 Years of Hardware Verification*, Royal Society Workshop to mark 21 years of BCS FACS, http://www.cl.cam.ac.uk/users/mjcg/BDD, December 1998.
8. M. J. C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-order Logic*. Cambridge University Press, 1993.
9. K. Havelund and N. Shankar. Experiments in theorem proving and model checking for protocol verification. In *Formal methods Europe FME '96*, number 1051 in Lecture Notes in Computer Science, pages 662–682, March 1996.
10. J. Joyce and C. Seger. Linking BDD-based symbolic evaluation to interactive theorem-proving. In *the 30th Design Automation Conference*, 1993.
11. S. Rajan, N. Shankar, and M.K. Srivas. An integration of model-checking with automated proof checking. In Pierre Wolper, editor, *Computer-Aided Verification*, number 939 in Lecture Notes in Computer Science, pages 84–97. Springer-Verlag, 1995.
12. S. Tahar and P. Curzon. Comparing HOL and MDG: A case study on the verification of an ATM switch fabric. To appear in the *Nordic Journal of Computing.*
13. S. Tahar, X. Song, E. Cerny, Z. Zhou, M. Langevin, and O. Aït-Mohamed. Modeling and automatic formal verification of the Fairisle ATM switch fabric using MDGs. To appear in *IEEE Transactions on CAD of Integrated Circuits and Systems.*
14. H. Xiong and P. Curzon. The verification of a translator for MDG's components in HOL. In *MUCORT98, Third Middlesex University Conference on Research in Technology*, pages 55–59, April 1998.
15. Z. Zhou and N. Boulerice. *MDG Tools (V1.0) User Manual*. University of Montreal, Dept. D'IRO, 1996.
16. Z. Zhu, J. Joyce, and C. Seger. Verification of the Tamarack-3 microprocessor in a hybrid verification environment. In *Higher-Order Logic theorem proving and Its Applications, The 6th International Workshop*, number 780 in Lecture Notes in Computer Science, pages 252–266. B. C., Canada, August 1993.