Formal verification of characteristic properties

Michel Allemand

Laboratoire d'Informatique de Marseille, URA CNRS 1787
CMI de l'Université de Provence - UFR MIM
39, rue Joliot Curie - 13453 Marseille Cedex 13 - France
e-mail: amichel@gyptis.univ-mrs.fr

Abstract. In this paper we introduce a verification methodology well adapted to circuits where the specifications are described in terms of characteristic properties instead of algorithmic procedures. This method avoids most of the interpretation mistakes which could invalidate the proof process. In order to describe implementations, we present a formalism, based on sequences, which is close to HDLs. Then these description and proof methodologies are implemented in the Larch Prover which is adequate for this kind of verifications. This work is illustrated by the verification of the correctness of the nontrivial Minmax circuit.

Introduction

As far as verifying the correctness of digital devices is concerned, an alternative to simulation and test methodologies is the formal proof approach. One aspect of the formal proof of hardware consists in verifying the equivalence between an implementation and its specification (expected behaviour). Both the implementation and the specification are then mathematically modeled. The specification can be either a computational definition (an algorithm or an inductive definition) or an axiomatic definition (a set of characteristic properties). In the second case, a usual approach consists of translating the initial properties into an algorithm which is considered as the new specification. This approach could introduce some mistakes and could invalidate the proof process. Indeed, the translation of informal or formal properties into a formal algorithm is not straightforward. In this paper we present how to deal with such specifications using the semi-automatic Larch Prover [4].

In a first part, we present a proof methodology and a formalism based on temporal sequences which is well adapted to the proof method as well as to LP and to hardware descriptions. In a second part, we apply this work to the verification of the Minmax circuit. We describe its specification in LP according to our formalism and we show how we deal with the modularity of the implementation in order to verify the circuit.

1 Overview of the methodologies

In the field of formal verification of digital devices, two approaches are possible. We can use a specific system like tautology checker or model checker. On the

other hand we can use general provers like the Boyer-Moore prover Nqthm, HOL, the Larch Prover and so on. In order to implement our description and proof methodologies, we chose to use LP. The Larch prover is a rewrite-rule based tool which allows induction and works on a subset of multisorted first order logic.

1.1 Description methodology

Our description formalism is based on the approach of [7], of [1] and on the P-Calculus [5]. In [7] the formalism is based on an "execution history" of a circuit which consists of a sequence of values for each data path in the circuit. The approach of [1] is based on streams of values. In a similar way, in the P-calculus, synchronous circuits are modeled by letting the input and output signals be temporal sequences of values (time functions) and by means of functions on such sequences. In order to express temporal features of digital devices, only one operator is needed: a Past functional $\mathcal{P}$ delay operator. In such a formalism the behaviour of a register R (with an input R_i and an output R_o) will be expressed by $R_o = \mathcal{P}(R_i)$. Thus we describe each signal by a sequence of values. Like in the P-calculus, these sequences are associated with a temporal scale. Sequential components are driven by a global clock which corresponds to this temporal scale. Like in [7] we consider an operator "." which takes as inputs a sequence S and an integer t and returns the t^{th} value of S : " $S.t$ " (i.e: the value of S at the time step t).

The behaviour of each combinational component C is described by the following first order logic formula: $\forall t, OUT.t = \mathcal{C}(IN.t)$ where $\mathcal{C}$ is a function which expresses the combinational behaviour of C and t the time, and the behaviour of each register R is formalized by the formula: $\forall t, R_o.(t+1) = R_i.t$

The composition of two components $C1$ and $C2$ is performed by a formula which expresses that the input sequence of $C2$, $C2_i$ follows the output sequence $C1_o$ of $C1$: $\forall t, C2_i.t = C1_o.t$

This methodology is well adapted to a hierarchical approach. Indeed, we can describe parts of a circuit as modules with their own inputs and outputs which can be easily connected. The behaviour of a module can be seen at the bit vector level or at the arithmetic level. Descriptions in this formalism can be obtained easily from VHDL ones.

1.2 Proof methodology

In this paper, we only consider the specification of circuits which are described by informal characteristic properties. In order to solve such a problem, several approaches are feasible. The first one consists in deducing a high level algorithm from the various points of such a specification. This algorithm becomes the new specification with respect to which we prove the implementation. The main drawback of such an approach is that the deduced specification can be very far from the initial one. Thus mistakes are likely and the proof process can be falsified.

Our approach consists in transforming the informal properties of the circuit into first order formulas. Such a process cannot prevent all mistakes. Indeed, we are not able to deal with informal specifications and, as soon as we translate informal properties into formal properties, we face mistake risks. However, in case where some ambiguities remain, this transformation can be achieved with designer interactions, thus the safety of this process is sufficient. These formulas are translated into LP descriptions as follows:

- All external universally quantified free variables are translated into LP free variables which are implicitly universally quantified.
- In the same way all external existentially quantified variables could be translated into LP formal constants (i.e. 0-ary operators).
- We translate all expressions such as “ $\forall t' > t P$ ” by introducing a new free LP variable h and by substituting in P the term “ $t + succ(h)$ ” to t' , where $succ$ is the constructor operator for the natural numbers. We use this constructor to refer to a natural number which cannot equal zero.
- We express an expression such as “ $(C_1 \wedge \dots \wedge C_n) \Rightarrow E$ ”, by the LP deduction rule “when $C_1, \dots, C_n$ yield E ” if none of the C_i is universally quantified.
- The expression “ $(C_1 \wedge \dots \wedge C_n) \Rightarrow E$ ” is translated into the LP deduction rule

$$\text{when } C_1, \dots, C_{i-1}, (\text{forall } t_i) C'_i, C_{i+1}, \dots, C_n \text{ yield } E$$

if C_i is universally quantified.

Finally the proof process consists in checking with the Larch Prover that the implementation verifies all properties described in the specification. This is done by asserting the description of the implementation and by proving each point of the specification, mainly by induction on time.

The main difficulty of such a verification lies in the strategy of proof. It is independent from the abstraction level of the component descriptions. Thus we chose to describe implementation at the arithmetical level. In case a preverified library of components is not provided, the first step of the proof process will be the verification of the correctness of all of these components.

2 Application to the MinMax circuit

In order to illustrate the application of our description and proof methodologies using the Larch Prover, we will verify the MinMax circuit. This circuit which has been proposed by IMEC [2] has already been verified using symbolic manipulations in [3] and using Nqthm in [6]. The main drawback of these approaches is that the necessity of formalizing the characteristic properties of the circuit as an algorithm takes us far from the initial informal specification. Furthermore, another disadvantage of [3] is that the size of the bit vectors must be fixed.

The Minmax circuit, the implementation of which is given in figure 1, has an input signal IN which consists of a sequence of integers and three boolean

Fig. 1. Implementation of the MinMax circuit

control signals *clear*, *enable* and *reset*. The unit produces an output sequence *OUT* at the same rate as *IN* in the following way:

- *OUT* is zero if *clear* is true, independent of the other control signals
- if *clear* is false and *enable* is false then *OUT* equals the last value of *IN* before *enable* became false.
- if *clear* is false and *enable* is true and *reset* is true then *OUT* follows *IN*.
- if *reset* becomes false, then *OUT* equals, on each time point *t*, the mean value of the maximum and the minimum value of *IN* until that time point.

These informal characteristic properties are rewritten into first order logic. Thus, the formal specification of the second property is:

$$\forall t_0, \forall t_1 > t_0 [((clear.t_1 = F) \wedge (enable.t_1 = F) \wedge (enable.t_0 = T) \wedge (\forall t' > t_0, t' < t_1 \Rightarrow enable.t' = F)) \Rightarrow (OUT.t_1 = IN.t_0)]$$

Although mistakes are possible, it is obvious that the translation into first order formulas is rather straightforward. Then, according to the rules given in section 1.2, we obtain the following LP conjecture:

when (forall *l*) $l < h \Rightarrow enable.(t_0 + succ(l)) = false,$
 $clear.(t_0 + succ(h)) == false,$
 $enable.(t_0 + succ(h)) == false,$
 $enable.t_0 == true$ yield
 $OUT.(t_0 + succ(h)) == IN.t_0$

2.1 Verification of the implementation

We describe only the verification of the second property. In order to establish this conjecture, we have to prove some intermediate lemmas. These lemmas are deduced from the modular description of the implementation. We generate two subgoals:

- The output *last* of the module "Last" always produces the last value of *IN* before *enable* became false.

when (forall l) $l \leq h \Rightarrow \text{enable}.(t0 + \text{succ}(l)) = \text{false},$
 $\text{enable}.t0 == \text{true} \text{ yield}$
 $\text{last}.(t0 + \text{succ}(h)) == IN.t0$

- If *clear* is false and *enable* is false then *OUT* follows the output *last* of the module "Last".

$((\text{clear}.t = \text{false}) \& (\text{enable}.t = \text{false})) \Rightarrow ((OUT.t) = (\text{last}.t))$

The proof of the second subgoal is as straightforward as the validation of points 1 and 3 of the specification.

On the other hand, we need to introduce a new intermediate lemma to achieve the verification of the first one. This lemma expresses that, as soon as *enable* becomes false the value of *last* is the immediate preceding value of *IN*, i.e. when *enable* was true. It is formalized by the following LP conjecture:

$((\text{enable}.h = \text{true}) \& (\text{enable}.succ(h) = \text{false})) \Rightarrow \text{last}.succ(h) = IN.h$

Then, we don't prove directly the first subgoal, but a little more generalized conjecture:

when (forall l) $l \leq h \Rightarrow \text{enable}.(t0 + \text{succ}(l)) = \text{false},$
 $\text{enable}.t0 == \text{true} \text{ yield}$
 $(k \leq h) \Rightarrow \text{last}.(t0 + \text{succ}(k)) == IN.t0$

The validation of this conjecture is performed by induction on *k*. It is obvious that the first subgoal is directly deduced from this conjecture where *k* equals *h*. Finally we achieve the proof of the second property using the two subgoals, the verification of which we have achieved.

Conclusion

The main contribution of this work is a formal verification methodology, devoted to circuits whose specification is given in term of characteristic properties. This method has been implemented using the semi-automatic Larch Prover which works on first order logic. Furthermore we introduce a description formalism which is adapted to this prover as well as to the methodology and to hardware descriptions.

As far as the description of the implementations is concerned, the formalism that we have chosen is very legible. It is well adapted to hierarchical implementations. Moreover the LP descriptions in this formalism are very close to the corresponding VHDL descriptions. All of that is a great advantage for the designers.

The separated validation of each point of a specification described in term of characteristic properties is feasible in a relatively short time. Although the mistakes are not completely excluded, this approach preserves from most of them: even in the cases where the proof fails, this approach allows to see the properties which are not verified. This could be the starting point of an error diagnostic procedure.

In order to verify some properties of the circuit we have to introduce several intermediate lemmas according to the modular structure of the implementation. However, due to the readability of LP descriptions and LP conjectures, this work can be achieved by a designer.

The proof methodology concerns only circuits, the specifications of which are described in terms of characteristic properties. On the other hand, our description formalism could also be applied to the description of invariant specifications where we have to state some assumptions on time. Thus, future works will aim at developing this approach.

References

1. A. Bronstein and C. Talcott. Formal verification of synchronous circuits based on String-Functional Semantics: The seven Paillet circuits in Boyer-Moore. In *Workshop on automatic verification methods for finite state systems*, Grenoble, June 1989.
2. L. Claesen, editor. *Internal Workshop on applied Formal Methods for VLSI Design*, Leuven, Belgium, 1990. North-Holland.
3. O. Coudert, C. Berthet, and C. Madre. Verification of sequential machines using boolean functional vectors. In L. Claesen, editor, *Formal VLSI Correctness Verification*. North Holland, 1990.
4. S. J. Garland and J. V. Guttag. A guide to LP, the Larch Prover. Report 82, DEC Systems Research Center, Palo Alto, CA, December 1991.
5. J.-I. Paillet. A functional model for descriptions and specifications of digital devices. In D. Borriore, editor, *IFIP WG 10.2 Workshop From HDL descriptions de guaranteed correct circuit designs*. North-Holland, 1987.
6. L. Pierre. The formal proof of the "Min-max" sequential benchmark described in CASCADE using the Boyer-Moore theorem prover. In L. Claesen, editor, *Formal VLSI Correctness Verification*. North Holland, 1990.
7. J. B. Saxe, S. J. Garland, J. V. Guttag, and J. J. Horning. Using transformations and verification in circuit design. In J. Staunstrup and R. Sharp, editors, *International Workshop on Designing Correct Circuits*. North-Holland, IFIP Transactions A-5, 1992. Also published as DEC Systems Research Center Report 78.