A system for modelling and proving circuits

Michel Allemand, Solange Coupet-Grimal, Line Jakubiec and Jean-Luc Paillet
Laboratoire d'Informatique de Marseille – URA CNRS 1787
39, rue F. Joliot-Curie 13453 Marseille France
e-mail: {amichel,solange,jakubiec,jluc}@gyptis.univ-mrs.fr

We present the architecture of FORMATH: a system for modelling and proving circuits. Its core is a formal system, the P-calculus which allows to accurately describe structures and behaviours of circuits. In order to perform verifications, the P-calculus is implemented into proof assistants.

Formal verification of digital circuits requires to abstract devices into mathematical objects, to carry out algebraic transformations on them, and finally to prove properties. We develop a CAD verification oriented system for synchronous circuits (FORMATH), which is composed of four parts: a formal system, called the P-calculus, user input interfaces, theorem provers (the Larch Prover and the Coq system), and an interface for implementing the formal system into the provers.

The P-calculus is not a simple stream based functional HDL, but it is an actual formal system which allows automated algebraic transformations, thanks to rewrite rules. Indeed, many hardware functional languages have been already defined such as LCF-LSM, LUSTRE, mupp, HML and many authors use stream based functional modelling of hardware. These languages allow to describe and to simulate circuits, and sometimes to synthesize FSM, but they do not permit to directly process automated formal transformations. We modify an already defined functional algebra that we called P- Calculus and enrich it with new operators. In addition we couple it with a formal system involving a typed formal language and a rewriting system.

In order to get reliable devices it is crucial to have proof processes validated by proof-assistants. Various provers are already used for the formal hardware verification such as Nqthm, HOL, Nuprl. In fact, it is of interest to use several complementary provers in order to face the problems raised by such or such devices. We chose to explore the capabilities of two provers LP [3] and Coq [2] presenting rather different features. An interface executes the P-calculus expressions processing and translates them into the syntax of the provers.

We carried out several proofs of sequential circuits in both provers LP and Coq [1], investigating their potentials and taking advantage of their particular features in order to get processes as general, as easy to drive, as neat, and as understandable as possible. In fact, we have to take essentially into account the effort and the time required from the user for establishing mathematical strategies. In the case of control dominated circuits, the superiority of LP seems undeniable. This is due to the fact that in the Coq underlying intuitionist logic, a proposition is not interpreted by a boolean value: it is a type and proving it amounts to finding out a lambda term inhabiting this type. Therefore, boolean operators must be defined by hand and the proof requires user intervention whereas in LP, so the boolean calculus is built-in, the proof is almost automatic. On the other hand, in the case of computation dominated circuit, we can take advantage from Coq higher order for proving universal theorems leading to more general and complete proofs.

It remains work for developing input interfaces which could offer the users the possibility of taking advantage of all the algebraic features of the P-calculus. Two kinds of interfaces can be considered: the ones for entering specifications and descriptions which are already formalized in a commonly used HDL, such as VHDL, the others, allowing direct specifications in the P-calculus by means of user-friendly menus.

We envisage to carry on these investigations by taking advantage of recent works on semantics of VHDL. In fact the translation tool should be bidirectional: from P-calculus expressions obtained after validated transformations and refinements, we could obtain VHDL programs that can be given as inputs into current synthesis tools.

Two reasons encourage us to undertake the study of an assistant tool for formal specifications in the P-calculus: first, there is a direct correspondence between the structure of circuits and the P-calculus expressions, second, it is very easy to express most of temporal properties given in an informal description. This interface should display possible implementations in the form of netlists and to detect early some errors. Moreover, partial modifications of initial specifications should be performed without requiring a complete analysis of the whole object.

References