A Simple Calculus for Synthesizing Circuits

David Basin and Stefan Friedrich
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123, Saarbrücken, Germany
Email: { basin, fried }@mpi-sb.mpg.de
Phone: (49) (681) 302-5435
Fax: (49) (681) 302-5401

The task of developing correct hardware consists in designing an implementation of a circuit and after proving it to be correct with respect to its specification. In the discipline of hardware synthesis one combines these two steps by developing a correct circuit from its specification. This is done either by developing a circuit simultaneously with its correctness proof or by using only those development techniques that preserve correctness.

The Veritas group at Kent [3, 2] proposed a novel approach to synthesizing circuits based on the first approach. Their technique, which they call *Formal Synthesis*, is to refine a "design goal" (*spec*) interactively, using *methods*, eventually concluding with a theorem that some circuit specification *circ* achieves *spec*; i.e., that $circ \rightarrow spec$. Each method consists of a pair of functions: a subgoal function and a validation function. The former decomposes a specification *spec* into subspecifications $spec_i$. The latter constructs from a proof that the $circ_i$ achieve $spec_i$, an implementation *circ* with a proof that *circ* that achieves *spec*.

In our work we reconstruct the Veritas development methodology as a conservative extension of Isabelle's higher order logic. That is we derive appropriate rules and develop tactics that implement the methods of Veritas. Under our approach, a circuit is instead synthesized by carrying out a correctness proof, where at first the implementation of the circuit is represented by a placeholder (i.e. a metavariable) and is incrementally instantiated by higher-order resolution as the proof proceeds. The resulting approach is rather different from the original Veritas development methodology based on "validation functions" and "achievement theorems" but it yields a conceptually and implementationally simpler calculus than the original presentation that also is more powerful and flexible in many respects. Moreover, since our calculus is developed using derived rules in HOL, we have a formal guarantee of the correctness of the development rules.

An brief example should help clarify the nature of our recasting. Consider the simplest of their methods, *Split*, that takes a specification which is the conjunction of specifications and returns the individual specifications as subgoals.

$$\frac{spec_1 \quad spec_2}{spec_1 \wedge spec_2} \qquad (Split)$$

The validation theorem for this is the following inference rule[1]

$$\frac{circ_1 \rightarrow spec_1 \quad circ_2 \rightarrow spec_2}{(circ_1 \wedge circ_2) \rightarrow (spec_1 \wedge spec_2)}$$

Looking at the above validation theorem we immediately see that this is a derivable rule in Isabelle (in a first or higher-order theory). Moreover, if we apply this rule to prove a goal of the form

$$?C \rightarrow (spec_1 \wedge spec_2)$$

then the substitution instance will partially construct a circuit; the resulting subgoals tell how to elaborate the design of the circuit while establishing its correctness.

Part of our work has also addressed the design of new methods in our setting and designing our calculus so that it is independent of particular hardware models (voltage driven circuits, tri-state, etc.) and therefore of the underlying logic that is used (FOL, HOL). Our theory also allows synthesis of parameterized circuits based on structural and well-founded induction over natural numbers. We have used this to synthesize a variety of circuits including parameterized ripple-carry and carry-lookahead adders, comparators, and the like.

References

[1] Albert Camilleri, Mike Gordon, and Tom Melham. Hardware verification using higher-order logic. In *From HDL Descriptions to Guaranteed Correct Circuit Designs*, pages 43–67. Elsevier Science Publishers B. V. (North-Holland), 1987.

[2] Hanna, F. K., Daeche, N., and Longley, M. Formal synthesis of digital systems. In Brown, G. Leeser, M., editor, *Hardware Specification, Verification and Synthesis: Mathematical Aspects; Mathematical Sciences Institute Workshop; LNCS Vol. 408*, pages 532–547. Springer-Verlag, July 1989.

[3] Hanna, F. K., Daeche, N., and Longley, M. Veritas$^+$: A specification language based on type theory. In Brown, G. Leeser, M., editor, *Hardware Specification, Verification and Synthesis: Mathematical Aspects; Mathematical Sciences Institute Workshop; LNCS Vol. 408*, pages 358–379. Springer-Verlag, July 1989.

[1] In the hardware formal reasoning community, circuits are often represented as relations expressing constraints (on port values) [1]. These constraints may be joined by logical $\wedge$ and this also represents an operator that builds a circuit from two subcircuits by parallel composition. So the $\wedge$ used to construct the implementation in the validation theorem represents both a circuit constructor (so we see we are building a circuit) and a logical operator (so we may prove facts about it).