Logic Synthesis and Design Verification

Robert K. Brayton

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720

Much work has been done over the last 7-8 years on the automatic synthesis of combinational logic. This work has found its way into the software of successful companies whose programs are found almost everywhere digital circuits are designed. More recently, research has focused on the automatic synthesis of sequential networks.

In combinational logic, the model of computation used is a Boolean network which is a multi-rooted DAG where each node is an arbitrary logic function. An arc is drawn from node i to node j if node j explicitly uses the result of the node i computation, node i's output. For sequential logic, the model of computation is a FSM network, an arbitrary graph where each node is a FSM with symbolic inputs, outputs, and internal states. One nodes outputs are another's inputs.

One of the paradigms for logic synthesis is to focus on a single node and to simplify it as much as allowed. Each node is then iteratively simplified until there is no change at any node. At this point, the logic is at some minimal representation. Such a paradigm works quite well for combinational logic now, although it took many years to work out the theory and practice of using the full capabilities of node minimization at each node. One of the problems was to simply derive and represent all the flexibility that is allowed at any node, i.e. all the "permissible" functions at a node.

Recently, these ideas have been extended to FSM networks. It has been shown that the "permissible" FSM's at a node can be represented with a single nondeterministic FSM, where a permissible FSM is a completely specified deterministic FSM that is simulated by the NDFSM. Then one form of node minimization is to select the permissible machine with the least number of states - a hard problem, but possibly one where good heuristics can be developed. Unfortunately, the derivation of the NDFSM involves a subset construction, so here again heuristics need to be employed.

The FSM network model is exactly the same one that is used in some forms of design verification - e.g. L-processes. Indeed in the FSM network it is allowed to have nodes with nondeterministic behavior, perhaps representing part of the environment. Design verification deals with proving that the FSM network satisfies certain properties, $P_1, \ldots, P_n$. Using L-automata, each of these (if ω-regular) can be represented by a finite set of deterministic L-automata, $P_{i1}, \ldots, P_{ik}$. If "complete", the entire set can be taken as our specification. This is analogous to the "observability relation" used in combinational logic, which specifies what input-output behavior is allowed. Such an observability relation is then used to derive a local environment for a single node in the Boolean network. One can

speculate that a similar theory should hold for the FSM network. Other useful parallels hold between the two areas of logic synthesis and design verification.

The purpose of this talk is to survey these parallels, with the idea that an improved understanding of both areas leads to common ideas and tools and gives guidelines about fruitful avenues of research to pursue.