Short Papers

Symbolic Cover Minimization of Fully I/O Specified Finite State Machines

DORON DRUSINSKY-YORESH

Abstract—**Symbolic cover minimization is an important step within a well-known state-assignment technique for finite state machines (FSM's) [2]. Currently, multiple-valued-input logic minimization techniques are used to find a minimum symbolic cover. The former problem, however, is computationally intractable, so heuristics are used. We show a simplified technique, based on an extension of the FSM minimization technique, which enables an efficient deterministic solution for fully I/O specified FSM's.**

I. Introduction

Finite state machines (FSM's) are one of the most popular models for VLSI control systems. FSM's have a well-known hardware implementation which consists of two components: a combinational circuit and a memory. The memory stores the (binary) representation of the FSM state throughout computation whereas the combinational circuit generates machine output and next-state representation as a function of the inputs and the present state.

Programmable logic arrays (PLA's) are a regular and structured technique for implementing the combinational circuit. PLA area, however, tends to grow nonlinearly with FSM size. In fact, the number of PLA rows might be as great as the number of edges in the FSM which is $O(n^2)$, where n is the number of FSM states.

It is well known that the binary representation of the FSM state set has tremendous effect on PLA size. The corresponding optimization problem (i.e., to find the binary state representation that minimizes PLA area) is the *state assignment problem*. State assignment has been a subject of extensive research (see [2] for an extensive reference list). A recent approach to state assignment [2] consists of two major steps: *symbolic cover minimization*, and the *constrained encoding problem*. Currently, symbolic cover minimization is computed using *multiple-valued logic* minimization. This problem, however, is computationally intractable, so less accurate heuristics are used. This technique remains computationally intractable even when the input FSM is *fully I/O specified* (but has many states). An alternative approach to computationally intractable problems is to reduce their generality such that the simpler problem has a tractable solution. Accordingly, this paper suggests a deterministic approach for the symbolic cover minimization problem for fully I/O specified FSM's. We prove a uniqueness theorem for a hierarchical extension of FSM's and use this result to derive our algorithm.

II. Preliminaries and Definitions

Following the notation in [2], an FSM is a 6-tuple (y_0, X, Y, Z, δ, λ) where X, Y, and Z are (finite) sets of primary inputs, states and primary outputs, respectively, $\delta: X \times Y \rightarrow Y$ is the next-state function, $\lambda: X \times Y \rightarrow Z$ is the output function,[1] and $y_0 \in Y$ is the *initial* state. Following the definitions in [1] let X and Z consist of tuples of symbolic variables, $X = S_1^I \times \cdots \times S_n^I$ and $Z = S_1^O \times \cdots \times S_m^O$. In the general case $\delta(\lambda)$ is given as a partial function with incomplete specification over X or Y. In this paper however, we assume that these functions are total.[2] Hence, we say our input FSM's are *fully I/O-specified*.

A *symbolic cover* is a set of primitive elements called symbolic implicants. Each symbolic implicant consists of 4 fields of **symbolic** strings, corresponding to the primary inputs, present states, next states, and primary outputs, respectively. A symbolic implicant represents a transition from one or more states to a next state, under some input conditions. Hence, an example of a symbolic cover is [2]

$\neg\alpha$	START	state − 6	α
$\neg\alpha$	state − 2	state − 5	α
$\neg\alpha$	state − 3	state − 5	α
$\neg\alpha$	state − 4	state − 6	α
$\neg\alpha$	state − 5	START	β
$\neg\alpha$	state − 6	START	γ
$\neg\alpha$	state − 7	state − 5	α
α	START	state − 4	γ
α	state − 2	state − 3	β
α	state − 3	state − 7	β
α	state − 4	state − 6	β
α	state − 5	state − 2	α
α	state − 6	state − 2	α
α	state − 7	state − 6	α.

A *minimum symbolic cover* is one of minimum cardinality, i.e., consisting of a minimum number of symbolic implicants. *Symbolic minimization* consists of finding such a minimum symbolic cover.

The state assignment problem consists of choosing a Boolean representation of the internal states of the machines so that PLA size is minimized. The state assignment method suggested in [2] consists of symbolic cover minimization followed by a constrained encoding problem. Symbolic cover minimization is carried out in [2] by transforming the symbolic cover into a *multiple-valued cover* which is a representation of the symbolic cover in multiple-valued logic, followed by a multiple-valued cover minimization procedure. The effect of symbolic minimization is to group together the states that are mapped by some input into the same next state and assert the same output [2].

We define a *hierarchical FSM* (HFSM) as an extension of a FSM in which edges run between subsets of states, called *superstates*. An edge from superstate q_1 to superstate q_2 in an HFSM S, is equivalent to all possible edges with the same label and same output from an element of q_1 to an element of q_2 within the original (*extended*) FSM A. We say S is deterministic iff A is deterministic. Clearly, if S is deterministic, then every such q_2 must be a singleton.

Consider the FSM A in the example above. It has an equivalent

Manuscript received December 15, 1988; revised February 28, 1989 and June 26, 1989. This paper was recommended by Editor M. R. Lightner.

The author is with the CAD Department, Sony Corporation, Atsugi-shi, Kanagawa-ken, 243 Japan.

IEEE Log Number 8934109.

[1] Throughout, we shall refer to deterministic FSM's only, thus δ and λ are functions.

[2] Note that λ's range must be fully specified, otherwise (when λ maps to "subtuples" of elements of Z) a nondeterministic behavior is implied. This also follows from the definitions of λ as a function rather than a relation.

0278-0070/90/0700-0779$01.00 © 1990 IEEE

HFSM S with the superstates {START, 4}, {2, 3, 7}, and {5, 6} as well as all singletons. Hence, for example, S has an edge from {2, 3, 7} to {5} labeled $\neg\alpha$ that asserts output α. This edge "represents" three edges in A.[3] HFSM's are actually a special case of Harels' *statecharts* which are FSM's extended with hierarchy and concurrency. See [3]–[5] for details.

The superstates of an HFSM S together with the set inclusion relation, form a grid which is represented by a directed acyclic graph (DAG) denoted DAG(S). Note that for an HFSM S that extends an FSM A, the set of leaves of DAG(S) (called also *atomic states*) is equal to A's set of states.[4] We call superstates with an outgoing edge labeled α that assert output β, an α; β-superstate, and the corresponding edge an α; β-edge. Hence, in our example {2, 3, 7} is a $\neg\alpha$; α-superstate, and {5, 6} is an α; α-superstate. Clearly, if S is deterministic then every two α; β-superstates are either disjoint or their outgoing α; β-edges lead to the same (atomic) state.

A *finite automaton* (FA) is an acceptor version of an FSM. Formally a FA is a 5-tuple (y_0, X, Y, δ, F), where X, Y, y_0, and δ stand for an FSM, and F is a set of *final* states. A FA M *accepts* an input string x iff M reaches a final state when it finishes scanning x, starting from y_0 (we also say that M has a *run* on x that reaches q) [6]. It accepts a set (language) of strings L, iff every string x in L is individually accepted. FA is an extremely convenient tool for proving theoretical properties of FSM's [6].

A *hierarchical FA* (HFA) is a straightforward hierarchical extension of FA. Hence, an HFA relates to a FA the same way an HFSM relates to an FSM. Clearly, in this case it suffices to consider α-superstates and α-edges. An *edge-minimum* (*state-minimum*) HFA for a language L is an HFA with a minimum amount of edges (atomic states) that accepts L. An edge-minimal HFA is a unique edge-minimum HFA.

III. The Uniqueness of the Minimal Symbolic Cover for Fully I/O-Specified Machines

For simplicity, we shall present the minimality result for acceptors only. Clearly, when all fields of an implicant are symbolic (and assuming fully I/O-specified machines), then the effect of symbolic minimization is to group together the states that are mapped by some input into the same next state. Hence, symbolic minimization corresponds to finding an edge-minimum HFA.

Extending Nerodes [6], [8] definition, we define the following equivalence relation for a language L and an input symbol α: $xR_L^\alpha y$ iff for every $z \in X^*$, $x \cdot \alpha \cdot z \in L$ iff $y \cdot \alpha \cdot z \in L$. Let R_L be Nerodes equivalence relation, that is $xR_L y$ iff for every $z \in X^*$ $x \cdot z \in L$ iff $y \cdot z \in L$. Let $[x]_\alpha$ and $[x]$ denote the R_L^α and R_L equivalence classes that include x, respectively. Clearly, for every α, R_L refines[5] R_L^α. We define S_L as the following HFA. Its set of superstates consists of all R_L^α equivalence classes for all $\alpha \epsilon X$, and all R_L equivalence classes. The HFA grid DAG(S_L) is naturally defined over these classes with the set inclusion relation. S_L includes an α-edge between $[x]_\alpha$ and $[x \cdot \alpha]$ for every input symbol α. Clearly, this is a consistent definition (namely, if does not depend on the choice of x). S_L's initial atomic state is $[\epsilon]$, and its set of (atomic) final states is the set of all states $[x]$ such that x is in L. Clearly, S_L's run on x leads to $[x]$, thus x is accepted by S_L iff it is in L.

Let S be a deterministic HFA. We define the relation: $xR_S^\alpha y$ iff the run of S on x and the run of S on y lead to the same α-superstate. Clearly, R_S^α is reflexive and symmetric. It is not necessarily transitive because x and y might lead to an α-superstate q_1 whereas y and z lead to an α-superstate q_2, where q_1 and q_2 are nondisjoint α-superstates that have an outgoing edge labeled α that leads to a common (atomic) state q_3.[6] Clearly, when R_S^α is an equivalence relation, its index is equal to the amount of α-edges in S.[6]

The following theorem is a "hierarchical" extension of the famous Myhill–Nerode theorem [6], [8].

Theorem 1: The edge-minimum deterministic HFA that accepts a regular set L is unique up to superstate renaming and is given by S_L. Moreover, S_L is state-minimum.

Proof: We prove edge-minimality first. Assume by negation that a different HFA S is edge-minimum. Clearly, for every α, R_S^α must be an equivalence relation. Otherwise, R_S^α is not transitive, which implies that either S is nondeterministic, or there are two nondisjoint α-superstates whose outgoing α-edges lead to the same (atomic) state. By unifying these superstates we can reduce one edge, hence, S is not edge-minimum. A contradiction.

As stated earlier, the number of edges within S is equal to the sum of indexes of all R_S^α, and similarly for S_L and R_L^α. R_S^α, however refines R_L^α. Therefore, R_S^α's index and R_L^α's index must be equal. Hence, each of the superstates of S can be identified with a superstate of S_L. Let q be a superstate of S. There must be some string x such that S's run on x reaches q, otherwise we can remove q. We identify q with superstate q' of S_L for which the run of S_L on x reaches q'. This 1-1 identification is consistent, and defines superstate renaming. Note that this renaming preserves hierarchy, namely, whenever q and q' are mapped together, so are their ancestors. That S_L is state-minimum follows from the fact that it extends the minimal Nerode automaton.[7]

Note that Theorem 1 is actually a minimization theorem for sequential statecharts.

An interesting observation is the fact that the depth of hierarchy in the minimal deterministic HFA is limited by the size of Y. More specifically, the size of Y is always at least the maximum, over all S_L leaves s, of the sum of lengths of all paths from the root to s in DAG(S_L). This is because for every leaf s and every input α, there do not exist two α-superstates on any of these paths, otherwise the HFA is nondeterministic.

IV. A Symbolic Cover Minimization Technique for Fully I/O-Specified Machines[8]

It follows from Theorem 1 (extended to HFSM's), that a straightforward algorithm for symbolic cover minimization is to minimize the (unique) minimal FSM (see [7], or [6] for FA minimization), and then to find the (unique) minimal HFSM. For the later step we define, for every input α, output β, and all states p, q: $p \equiv_{\alpha;\beta} q$ iff $\delta(p, \alpha) = \delta(q, \alpha)$ and $\lambda(p, \alpha) = \lambda(q, \alpha) = \beta$. Clearly, $\equiv_{\alpha;\beta}$ is an equivalence relation and each equivalence class corresponds to an α; β-superstate and to its corresponding α; β-edge. Each such α; β-edge, however, corresponds to a symbolic implicant in the minimal symbolic cover. Hence, the minimal symbolic-cover for our example is

$\neg\alpha$	{START,state 4}	state − 6	α
$\neg\alpha$	{state − 2, state − 3, state − 7}	state − 5	α
$\neg\alpha$	state − 5	START	β
$\neg\alpha$	state − 6	START	γ
α	START	state − 4	γ
α	state − 2	state − 3	β
α	state − 3	state − 7	β
α	state − 4	state − 6	β
α	{state − 5, state − 6}	state − 2	α
α	state − 7	state − 6	α.

[3]Note how each superstate actually corresponds to a *face* in [2].
[4]A DAG(S) node is a leaf iff it precedes the empty-set node in the grid.
[5]Formally, R_L refiles R_L^α iff $xR_L y$ implies $xR_L^\alpha y$.
[6]q_3 is common, otherwise the HFA is nondeterministic.
[7]Therefore, it is not state-minimal.
[8]The method presented herein is part of a pending U.S patent.

V. Conclusion

We have presented an efficient deterministic solution for symbolic-cover minimization for fully I/O specified FSM's. The effect of this algorithm on symbolic-cover minimization is described as the following competition between two state-assignment algorithms. Consider a large fully specified FSM (say with more than 100 states). First we carry out symbolic-cover minimization using the ideas presented in this paper, thus finding the (unique) cover in reasonable time. In contrast, a straightforward application of the symbolic-cover minimization algorithm of [2] for such a large FSM must be heuristic, thus most probably generating an inferior solution. It is this part of the state-assignment algorithm that determines the number of terms in the resulting PLA. Thus we can expect a PLA generated by the first algorithm to have fewer terms than a PLA generated by the second. Next, for both competitors, we continue with the constrained encoding algorithm of [2] which determines the number of columns in the resulting PLA. Clearly, we can now expect the first competitor to do at least as well as the second.

Acknowledgment

The author would like to thank the anonymous referees for their helpful suggestions and guidance.

References

[1] G. De Micheli, "Symbolic design of combinational and sequential logic circuits implemented by two-level logic macros," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 597-616, Oct. 1986.
[2] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal state assignment for finite state machines," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 269-285, July 1985.
[3] D. Drusinsky and D. Harel, "Using statecharts for hardware description," in *Proc. IEEE Conf. on CAD*, Santa Clara, pp. 162-165, 1987.
[4] D. Drusinsky and D. Harel, "Using statecharts for hardware description and synthesis," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 798-807, July 1989.
[5] D. Harel, "Statecharts: A visual approach to complex systems," *Sci. Comput. Programming*, vol. 8, pp. 231-274, 1987.
[6] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley, 1979.
[7] Z. Kohavi, *Switching and Finite Automata Theory.* New York: McGraw Hill, 1978.
[8] A. Nerode, "Linear automaton transformations," in *Proc. AMS*, vol. 9, pp. 541-544.

A Fast Transistor-Chaining Algorithm for CMOS Cell Layout

CHI-YI HWANG, YUNG-CHING HSIEH, YOUN-LONG LIN, AND YU-CHIN HSU

***Abstract*—We propose a fast algorithm for the transistor-chaining problem in CMOS functional cell layout based on Uehara and van Cleemput's layout style [12]. Our algorithm takes a transistor-level circuit schematic and outputs a minimum set of transistor chains. Possible diffusion abutments between the transistor pairs are modeled as a bipartite graph. A depth-first search algorithm is used to search for the optimal chaining. Theorems on the set of branches needed to be explored at each node of the search tree are derived. A theoretical lower bound on the size of the chain set is derived. This bound enables us to prune the search tree efficiently. The algorithm has been implemented and tested. It is able to find optimal solutions almost instantly for all the cases available to us from the literature.**

***Keywords*—CMOS cell layout, optimal chaining, transistor placement, depth-first search.**

Manuscript received January 4, 1989; revised May 31, 1989. This work was supported in part by ERSO under Contract SF-C-010-1 and by the National Science Council, Republic of China, under Contract NSC79-0404-1007-25. This paper was recommended by Associate Editor A. E. Dunlop.

C.-T. Hwang, Y.-L. Lin, and Y.-C. Hsu are with the Department of Computer Science, Tsing Hua University, Hsin-Chu, Taiwan.

Y.-C. Hsieh is with the Electronic Research and Service Organization, Industry and Technology Research Institute, Hsin-Chu, Taiwan.

IEEE Log Number 8934112.

I. Introduction

As CMOS VLSI technology [13] and cell-based layout methodology [1], [4] gain popularity, the automatic layout generation of CMOS functional cells becomes very important and attracts attention from many VLSI/CAD researchers. In [12], Uehara and van Cleemput proposed a paradigm for CMOS functional cell layout, which has inspired much research.

In [12]'s layout style, the transistors are placed in two parallel rows, where all the P-type transistors are in one row while all the N-type transistors are in the other. Power rails are routed along the rows on the outside and intracell routing runs between the rows. Since the height of a cell is usually fixed, the primary concern is to place transistors in such a way that gate signals are aligned and the drain/source diffusions of adjacent transistors are abutted as much as possible, thereby minimizing the number of separations between diffusion strips, which in turn minimizes the layout area. Much research has been done to improve the original proposal [2], [6]–[10], [14].

In this paper, we propose a fast algorithm for the problem of chaining the transistor pairs using a minimum number of chains. The input of our algorithm is a CMOS circuit schematic at the transistor level. The output from the algorithm is a minimum set of chains, where each chain can be realized using only one P-type diffusion strip and one N-type diffusion strip.

We group transistors into pairs with each pair consisting of a P-type and an N-type transistor and then model the possible abutments between the pairs as a bipartite graph. On the graph, a depth-first search algorithm is used to find a maximum set of edges which correspond to a maximum number of realizable abutments. There is a tight upper bound on the number of realizable abutments, and hence, the lower bound on the number of chains needed for an optimal solution, is derived. Theorems are proven to help to reduce the size of the search tree.

In the next section, we will survey some previous work. In Section III, we will present the bipartite graph model. Section IV defines some terminology and derives a number of theorems which will speedup the search process. A theoretical lower bound on the number of chains in an optimal solution is derived in Section V. Section VI describes the algorithm. Section VII presents our implementation and some experimental results. Concluding remarks and future work are discussed in Section VIII.

II. Previous Work

A heuristic method for finding a good, but not necessary optimum, chaining based on the Euler path algorithm was proposed in [12]. A CMOS gate is represented by two multigraphs (one for the P-network and the other for the N-network), where each vertex corresponds to a source/drain connection and each edge represents a transistor. The objective is to minimize the number of dual Euler

0278-0070/90/0700-0781$01.00 © 1990 IEEE