Concurrent Two-Dimensional State Minimization and State Assignment of Finite State Machines

M.A. Perkowski, W. Zhao, D. Hall
Department of Electrical Engineering
Portland State University, Portland, Oregon, 97207, USA

Abstract

A generalization of the classical state minimization of a Finite State Machine (FSM) is described. The FSM is minimized here in two dimensions: numbers of both input symbols and internal states are minimized in an iterative sequence of input minimizatio. and state minimization procedures. For each machine in the sequence of FSMs created by the algorithm an equivalent FSM is found that attempts to minimize the state assignment by selecting in each cell of the transition map one successor state from the set of successors. This approach leads to a partitioned realization of the FSM with an input encoder. Our efficient branch-and-bound program, FMINI, produces an exact minimum result for each component minimization process and a globally quasi-minimum solution to the entire two-dimensional (2D) FSM process of state minimization and state assignment.

1 Introduction

Several problems related to synthesis, verification and testability of networks of FSMs require checking the compatibility of states, detecting unreachable states, and making maximum use of sequential don't cares. One of the processes used to verify and optimize hardware-realized machines with compatible states and don't cares is *state minimization.* State minimization has recently received considerable attention [1, 4, 6, 11], but only a few FSM minimizing programs have been written [1, 4, 6, 11]. State minimizers are also included in the new U.C. Berkeley system, SIS.

The idea of minimizing states and columns of an FSM concurrently was described by Grasselli and Luccio [3, 8]. They show additional minimization capabilities of this approach with respect to standard state minimization. While [8] formulates the problem and gives examples but does not give the solution method, [3] gives a complete exact, nearly exhaustive algorithm which has been, however, not programmed. It is our feeling that if programmed, their method would be very inefficient. The approach from [3] also does not use the input encoder design method, presented below, which is derived from two-dimensional minimization. They are also not concerned with state assignment.

The idea of combining stages of state minimization and state assignment was originally proposed for approximate design of asynchronous machines by Hallbauer [5]. The first exact algorithm for synchronous machines was presented in [7] but was limited to 8 × 8 tables. The state minimization was considered together with state assignment by Kannan and Sarma [6], but they just used separate programs in a sequence. The fastest dedicated method for concurrent state minimization and assignment from Hachtel et al [4], is based on a special cost function and mapping. However, their methods do not use 2D minimization.

No other papers on generalizations of the FSM minimization problem and especially their computer realizations are known to the authors.

The original contribution of this paper is the extension, combination, and implementation of four ideas that have been little studied until now: (1) *Two-dimensional minimization;* (2) *The problem of selection of next states* from a group of equivalent next states, while a final machine is created from the closed and complete set of compatibles [10, 4]. (3) *Concurrent state minimization and state assignment.* (4) *Flip-flop type choice.* An Assignment process optimized for other than D flip-flops and selection of the optimum type of flip-flop for each excitation function [7].

2 The Two-Dimensional Minimization Method for Finite State Machines

The 2D minimization process combines the minimization/assignment of the internal states with that of the input states. Each state or input minimization/assignment process is performed independently, but such processes are executed in an iterative sequence until no better machine is found. Such an approach leads to a partitioned realization of the FSM with an input encoder (Fig. 1).

Next states from columns of a state table are *combinable* when all of the corresponding next states and outputs are *consistent.* Such states can be merged. If all pairs of states from two columns can be merged, those entire columns can be combined to a single column. We attempt to combine all columns to as few groups as possible, where the groups do not overlap. The merging occurs more frequently when there are many don't care terms in internal state transitions and outputs. The process of creating a partition of the minimum number of non-overlapping cliques of columns we call the *input minimization process.* Note that input minimization tries to combine the *columns* of the FSM table, while *state minimization* tries to combine the rows of this table. The possibility of input minimization is also considered for each of the new state tables created by a state minimization process. Following each input or state minimization procedure, the program tests whether further minimization is possible. The optimal solution actually includes two parts: the *optimal input column partitioning (OMCCI)* and the *optimal closed and complete covering (OMCCP)*. These are the last input column partitioning and the minimal closed and complete state covering created when further minimization is impossible.

FMINI,our program which implements these algorithms, produces an exact minimum result for each component minimization process, and a globally quasi-minimum solution to the two-dimensional problem.

Input minimization in **FMINI** is equivalent to the *clique partitioning problem,* which is NP-hard but has several very efficient algorithms. It can be also reduced to graph coloring [12]. State minimization in **FMINI** is reduced to a sequence of two NP-hard problems: the *generation of all cliques (all compatibles),* and the *covering/closure problem of internal states with compatibles* [11]. The generation of compatibles is done by a modified algorithm of Stoffers [16], which generates at the same time all compatibles implied by those groups [11]. The exact method from [4] cannot be used for machines with very large sets of compatibles, while the approach based on graph-coloring and closed graph-coloring can. Because of limited space, the branch-and-bound search algorithm of FMINI is not presented here (see [7, 11, 13, 14]).

FMINI uses a weighted cost function to decide if the search on a certain branch (set of compatible groups) should be retained or not [11]. The *weighted cost function CF(S)* of a solution set S of *compatible groups (CG)* in the FSM state minimization is:

$$CF(S) = \{a_1 * CARD(A)$$

$$a_2 * \sum_{S_j \in S} CARD(S_j)\} \qquad (1)$$

In formula (1), A represents a set of all the internal states of a certain FSM. CARD is the number of elements in the set. a_1 and a_2 are coefficients. The first component of (1) corresponds to the *minimization of the number of states* and the second component to the *minimization of the transition functions,* which is in turn expressed by the minimization of the number of states that occur in more than one CG. This leads to the *maximization of the don't care terms* in Boolean functions when the machine is minimized, its internal states are assigned with codes, and the excitation functions are finally found. If coefficient $a_1 >> a_2$, the solution is selected which has the minimum second component among all machines with the minimum number of states. When set SMAX can be calculated, coefficients a_1 and a_2 from formula (1) are described by the following formulas:

$$a_1 = \frac{1}{CARD(A)} \quad and \quad a_2 \qquad (2)$$

In the above formula, set *SMAX* denotes the *set of all compatible groups.*

This formulation is lacking in other programs (except for [1]) and leads to improved quality of assignments in FMINI. It has special advantages in two-dimensional (2D) minimization. It can be easily verified on examples from [3, 8] that while a machine state-minimized according to Grasselli/Luccio's method cannot be input-minimized (the case of row reduction preventing column reduction), the same machine state-minimized according to our approach can be still input-minimized leading to the global minimum solution.

A set of compatible groups is said to *overlap* if the same elements appear in different groups of this set. The idea of our state minimization method is to minimize the overlap at every stage. This leaves more don't cares for state/input minimizations. Moreover, this retains more don't cares for subsequent processes, which allows for better results during the state assignment and logic minimization processes.

A feasible solution that has the lowest cost, CF, is called the *exact solution.* The final state table with the exact minimum number of states, equivalent to the table before this state minimization, can be easily found from it, but better results are obtained if the *next-state selection* is taken into account (see [10] and section 3 of this paper. In [4]) this is called *mapping.*

Table 1 shows an initial state table of a Mealy machine. After the first Input Minimization, the new state table, Table 2, is produced from it. Next, State Minimization produces Table 3 from Table 2. Finally, the next Input Minimization produces Table 4, which cannot be further minimized. More details of this process can be found in [14, 11, 13].

3 The Next State Selection and State Assignment Process

For each transition with more than one successor (the so-called "non-univocal transitions" [7]), one of the "non-unique" next states is selected to simplify the state assignment. The method presented here is quite similar to those from [10, 7], but it sacrifices quality of the solution for the speed of creation. For every "non-univocal state table" (called also a relational transition graph), only some subset of proper partitions is calculated, as in [10], and their excitation functions individually tested.

For simplification, we will explain the method on an example from [10]. Fig. 2a presents an FSM table after combining in the initial table the rows from closed compatibles found in search. For simplification, the role of outputs (which is taken into account in [7, 10]) is not discussed here. The corresponding non-univocal state table is shown in Fig. 2b. as an example of how this is derived, state 2 is included in groups A and C, so ii is replaced with set {A,C} as next states in the new table. This new table is now evaluated for a subset of partitions. Suppose that partition $\tau_1 = \{\overline{AB}, \overline{CD}\}$ is tested and it is assumed that block $\overline{AB}$ is encoded with 0 and block $\overline{CD}$ with 1. If a single state exists in a cell of the table, then it is replaced with corresponding code of this state from this partition. For instance, state A with 0 and state D with 1. If the cell includes states from one block then these states are replaced with the code of this block, for instance states A,B are replaced with 0 and states C,D with 1. If the group includes states from various blocks, it is replaced by a don't care. For each selected τ_i the individual excitation functions are calculated, given the type of the flip-flop Q_i corresponding to this partition. This may be a D, T or JK flip-flop, which type is just assumed, or selected for the least cost of excitation functions for above types of flip-flops [7]. The excitation function $f(\tau_i)$ is approximately minimized with Espresso-mv fast logic minimizer and its cost is calculated. Fig. 2c,d,e presents calculations for D type flip-flops. For τ_1 (Fig. 2c) the minimized function is $S^C\overline{x} + S^{A,B}xy + S^A\overline{y}$ of cost 2 + 3 + 2 = 7. (We use here the notation of multiple-valued input switching functions [15]). For τ_2 (Fig. 2d) the minimized function is $\overline{x} + S^B\overline{y}$ of cost 1 + 2 = 3. For τ_3 (Fig. 2e) the minimized function is $S^B + S^{A,B}y$ of cost 1 + 2 = 3. A set of partitions is selected that satisfies two conditions: (1) the product of the partitions is a zero-partition ([10], (2) their total cost is minimum [7]). In our example, the partitions τ_2 and τ_3 with a total cost of 6 are selected. Assignment with such partitions (Fig. 2f) allows us to create a Boolean relation that has many don't cares in transitions and only few non-univocal transitions other than don't cares. For our example, the only such non-univocal transitions are $\delta(C,\ 10) = \{00,\ 11\}$, and $\delta(B,\ 00) = \{10,\ 01\}$. ($\delta$ denotes the transition relation). A special approximate fast Boolean relation minimizer is now applied to this function, which leads to the selection of 01 in $\delta(B,\ 00)$ and of 00 in $\delta(C,\ 10)$. This produces the solution: $D_1 = \overline{Q_1}\,\overline{x} + \overline{x}\,y,\quad D_2 = Q_2\,\overline{x} + \overline{Q_1}\,y$ of cost 8. This entire process is executed whenever a new OMCCP S is found with a better (or in another variant, not worse) value of CF(S). Several heuristics are added to make the process more efficient.

4 The Input Logic Encoding Process

As discussed in sect.2, the product implicants (represented as cubes) of primary input variables are accumulated in lists corresponding to the columns of the state table. For instance, in Table 4, input column X_4 corresponds to cube 001, input column $X_{2,5}$ to cubes 10- and 110, and input column $X_{1,3,6}$ to cubes 000, -11, and 010. Since the primary inputs X_i are divided into three distinguishable groups in the solution set, the outputs of the encoder (the secondary inputs of the FSM) should have at least 2 bits, denoted here by n and m. This encoding could use any kind of code. Some minimizing possibilities result also from the state assignment process to select this code, but it is beyond the topic of this paper. The code generated for our case is shown in Table 5. The Boolean minimizer Espresso-mv is called to find the quasi-minimal logic equations of the input encoder quickly. Assuming the code from Table 5, and primary input signals a, b, c the logic equations of the encoder are as follows:

$$m = \overline{a}\ \overline{c} + b\ c, \qquad n = a\,\overline{c} + a\,\overline{b}.$$

5 Evaluation of FMINI

The **FMINI** program has been tried on more than 40 FSM examples with 50 or less internal states and 20 inputs. The results of these trials show that **FMINI** can handle large scale machines. Some statistical results of using **FMINI** for minimization of real life circuits described initially in a high level parallel behavioral language ADL are presented in Table 6. Two-dimensional minimization gives essential size and area improvements for machines with many (more than 30%) don't cares in their outputs (Table 7).

6 Conclusion

The concept of 2D FSM minimization/assignment, implemented here for the first time, is a useful design alternative for machines with many don't cares and compatible states. It allows us to minimize machines in two dimensions, which for some machines leads to further area minimization with respect to one-dimensional minimization. It should be included in comprehensive design automation systems as one of several, script-selected, design methods. Since it is fast, it can be tried on every designed machine without sacrificing much design time. Although each component minimization process in the sequence of input and state minimization procedures is exact, there exist examples [8], which prove that our final machine is not an exact optimum, since minimization in one dimension can prevent the minimization in the other one. We are, therefore, working on an exact algorithm for the 2D minimization problem formulated by Grasselli and Luccio.

References

[1] M.J. Avedillo, J.M. Quintana, and J.L. Huertas, "State Reduction of Incompletely Specified Finite Sequential Machines," *Proc. Workshop on Logic and Architecture Synthesis,* Paris, May 1990, pp. 107-115.

[2] A. Grasselli, and F. Luccio, " A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks, " *IEEE Trans. on Electron. Comp.,* EC-14, pp. 350-359, 1965.

[3] A. Grasselli, and F. Luccio, "A Method for the Combined Row-Column Reduction of Flow Tables," *Proc. 7th Ann. Symp. on Switching and Automata Th.,* pp. 136-147, New York, 1966.

[4] G.D. Hachtel, J.-K. Rho, F. Somenzi, and R. Jacoby, " Exact and Heuristic Algorithms for the Minimization of Incompletely Specified State Machines," *European Design Automation Conference,* Amsterdam, The Netherlands, pp. 184-191, 1991.

[5] G. Hallbauer, " Procedures of State Reduction and Assignment in One Step in Synthesis of Asynchronous Sequential Circuits," *Proc. of the Intern. IFAC Symp. on Discrete Systems,* pp. 272-282, Riga, September 30-October 4, 1974.

[6] L.N. Kannan, and D. Sarma, " Fast Heuristic Algorithms for Finite State Machine Minimization," *Proc. EDAC 91,* Amsterdam, 192-196, Febr. 1991.

[7] E.B. Lee, and M.A. Perkowski, " Concurrent Minimization and State Assignment of Finite State Machines," *Proc. of Intern. IEEE Conf. on Systems, Man, and Cybernetics,* Halifax, Nova Scotia, pp. 248-260, 1986.

[8] F. Luccio, " Reduction of the Number of Columns in Flow Table Minimization," *IEEE Transactions on Electronic Computers,* Vol. EC-15, pp. 803-805, Oct. 1966.

[9] D. Pager, "Conditions for the Existence of Minimal Closed Covers Composed of Maximal Compatibles," *IEEE Trans. on Comp.,* C-20, pp.450-452, 1971.

[10] M.A. Perkowski, A. Rydzewski, and P.Misiurewicz, " Theory of Logic Circuits. Selected Problems," *Publishers of the Technical University of Warsaw* (book in Polish), 1977.

[11] M.A. Perkowski, and N. Nguyen, " Minimization of Finite State Machines in System Super-Peg," *Proc. of Midwest Symp. on Circ. and Syst.,* Luisville, Kentucky, pp. 139-147, 1985.

[12] M.A. Perkowski, and H. Uong, " Automatic Design of Finite State Machines with Electronically Programmable Devices," *Record of Northcon '87,* Portland 1987, paper 13/4.

[13] M.A. Perkowski, J. Liu, and J.E. Brown, *Rapid Software Prototyping: CAD Design of Digital CAD Algorithms Progress in Computer-Aided VLSI Design. Tools. Vol. 1,* G. W. Zobrist, ed., Ablex, pp. 353-401, 1989.

[14] M.A. Perkowski, W.Zhao, and D. Hall, " Restructuring of Finite State Machines: Two-Dimensional Minimization and State Assignment. Mealy-Moore conversion. State-selected-input Decomposition. Use of Invariants." *PSU Report,* 1991.

[15] T. Sasao, "Multiple-Valued Decomposition of Boolean Functions and the Complexity of Programmable Logic Arrays" *IEEE Trans. on Comp.,* Vol.C-30, No.9, pp. 635-643, Sept. 1981.

[16] K.E. Stoffers, "Sequential Algorithm for the Determination of Maximum Compatibles", *IEEE Trans. on Comp.,* ipp/.95-98, January 1974.