A Dynamic Model for the State Assignment Problem

M. Martínez, M.J. Avedillo, J.M. Quintana, and J. L. Huertas

Instituto de Microelectrónica de Sevilla (IMSE-CNM). Universidad de Sevilla.
Edificio CICA. Av. Reina Mercedes s/n. 41012 Sevilla. Spain. E-mail: avedillo@imse.cnm.es

Abstract

Traditionally, state assignment algorithms follow the two-step strategy of first constraint generation and secondly constraint-guided encoding. There are well known drawbacks in both currently used models for constraint generation. Approaches following the input model generate face constraints without taking into account the sharing of logic among next state lines. Approaches following the input-output model generate face constraints for a priori determined set of dominance/disjunctive relations among the codes of the states which may not hold in final encoding. To overcome these limitations, we propose a dynamic input model which implements both above cited steps concurrently. The dynamic constraints are of the face type but they are generated during the encoding process and so take advantage of actual relations among partial codes. A general algorithm based on this model and which can target two-level as well as multiple-level implementations is described. Results obtained with the algorithm on the IWLS'93 machines are shown and they compare favorably with standard tools .

1.- Introduction

Frequently, when synthesizing logic integrated circuits, there are symbolic variables in the specification of a design. The binary encoding of such symbols should be chosen to optimize the final implementation. This task is known as the encoding problem. In particular, if the symbolic variables are input (output) variables of the function being synthesized, it is called an input (output) encoding problem. A very interesting encoding example is the state assignment of finite state machines which arises during the design of sequential circuits. Classically, this problem is formulated as that of obtaining a binary code for each state of the FSM so that given design criteria are optimized.

* This research was supported in part by CICYT project TIC95-0094

Most of the state assignment algorithms aim to minimize the area of an implementation of the combinational component assuming a predetermined design style (two level, multilevel). This is, codes should be given such that the combinational minimizer produces good results. The difficulty resides in the prediction of the subsequent combinational step [1]. In general, based on this prediction on combinational synthesis effects, a set of constraints on the relationship between codes for different states are derived. Then, symbols are encoded such that constraints are satisfied.

Many algorithms approach the state assignment as an input encoding problem. This is, present states are considered as symbolic input variables while next states are given a 1-hot code during the generation of constraints. Other algorithms treat state assignment as a combined input-output problem as symbolic states appear in both the present state field and the next state field.

In this paper we investigate the limitations of the input and the input-output models and propose a new one for the state assignment problem. The new model can be described as a dynamic input model because only present states are considered symbolic, being the constraints modified concurrently with the encoding step. Information on partial codes generated so far is used to take into account the effect of the next state field. The rest of the paper is organized as follows. Section 2 reviews concepts and notation. Section 3 describes both previous models and their limitations. Section 4 introduces the new model. Section 5 describes the algorithm developed. Section 6 summarizes experimental results. Finally some conclusions are given in Section 7.

2.- Definitions

In this section we review some basic concepts in encoding theory and introduce the notation we will use.

Definition 1.- Binary encoding: given a set of symbols $S=\{S_1, S_2, ..., S_n\}$ and an integer k, a binary encoding of S is a one-to-one mapping $S \rightarrow \{0,1\}^k$

We can think of the encoding as a code matrix $C \in \{0,1\}^{nxk}$ where the *i*th row represents the code assigned to symbol S_i, and the *j*th column represents bit *j* of the encoding.

Definition 2.- **Face constraint**: a group constraint gc on a set of symbols $S = \{ S_1, S_2, ..., S_n\}$ is a subset S' of symbols from S which must be assigned such that the minimum boolean cube containing their codes does not intersect the codes of the symbols absent from S'.

Definition 3.- **(seed) Dichotomy**: a dichotomy d is a disjoint two block partition, $(B_1:B_2)$, associated with a group constraint gc_1, such that the block B_1 contains all symbols that belongs to gc_1 and B_2 contains exactly one of the symbols that does not belong to gc_1. A dichotomy constraint requires that subset B_1 of d be distinguished from subset B_2 of d by at least one encoding bit.

Definition 4.-**Dominance relation**: the code of a symbol S_1, $enc(S_1)$ dominates the code of symbol S_2, $enc(S_1) > enc(S_2)$, if for each bit position in $enc(S_2)$ that contains a 1, the corresponding bit position in $enc(S_1)$ also contains a 1.

Definition 5.- **Disjunctive relation**: the code of symbol S_1, $enc(S_1)$, is the disjuntion of the codes of the symbols S_2, S_3 ... and S_k, $enc(S_1) = enc\ (S_2) \mid enc(S_3) \mid ..\mid enc(S_k)$, if the code of S_1 is the bitwise OR of the codes of S_2, S_3, ., S_k.

3.- Models and Limitations

3.1 State assignment as an input encoding problem

Fundamental developments in the input encoding problem targeting two level implementations were the work in [2] where a tabular representation of the function with symbolic inputs is symbolically minimized and the use of two-level multiple-valued minimization for this task [3]. This minimization step generates face constraints on the relationship between codes for different symbols. In a following step, symbols are encoded such that constraints are satisfied. Satisfaction of the constraints guarantees that the optimization at the symbolic level will be preserved in the boolean domain: a two level implementation with as many product terms as in the minimal symbolic cover can be obtained. Later, this strategy was extended to multi-level implementations with the development of multi-level multiple-valued optimization algorithms [4]. When modeling the state assignment as an input encoding problem a 1-hot code is used for the next states. Symbols appear now only in the input field and so the above input techniques can be applied. Better area results (average reduction of 20%) have been obtained with minimum-length algorithms as compared to others that attempt to satisfy all constraints at the expense of an increased number of state variables for two-level implementations [5]. Thus, practical algorithms maximize constraint satisfaction for a given (user-specified or minimum) encoding length. Examples of algorithms following this input model are *ihybrid_code* and *igreedy_code* in NOVA [6], or ENCORE [7].

Approximating state assignment in this way neglects the effect of the next state field. The number of product terms in the symbolic minimized cover is only an upper bound for the size of the boolean cover when using an encoding which satisfies the complete set of constraints because next states have disjoint ON-sets in the symbolic cover while in the boolean cover several states can have a 1 on the same position [1].

3.2 State assignment as an input-output problem

When dealing with symbolic outputs, two types of constraints are more widely used. They are the dominance and the disjunctive relationship between the codes assigned to different output symbols. Heuristic [8] as well as exact [9] techniques that generate constraints which when satisfied result in maximal (maximum) sharing during the combinational minimization step have been reported. Both input (face) constraints and output (disjunction and/or dominance) constraints arise in input-output problems. An important difference with the input model is that it is not possible to state the unconditional existence of an encoding that satisfies both a set of face relations and a set of dominance and disjunctive relations [1]. Concerning state assignment, algorithms which are extensions to output encoding strategies above referenced have been proposed. In [9] a method that guarantees a two level boolean cover with a minimum number of product terms is reported. However, heuristic approaches are more interesting from a practical point of view. In option *io_hybrid* in NOVA [6] a symbolic minimization loop based on [8] is used to build up a set of face and dominance constraints. During the constraint satisfaction step, and using a minimum length encoding, it attempts to maximally satisfy the set of constraints. However, there are face constraints which are meaningful only if given dominance relations hold and current techniques do not take into account these interferences between both types of constraints.

4.- Dynamic Input Encoding Problem

To overcome limitations of the input model, next state must be taken into account but the relationship between face constraints and dominance and disjunctive relations should be efficiently handled. We propose a new dynamic input model in which only present states are considered symbolic, being the constraints modified concurrently with the encoding step. Information on partial codes generated so far is used to take into account the effect of the next state field.

Next, we report experiments conducted for the IWLS'93 FSM benchmark set [10]. The same encoding algorithm was applied to two different sets of constraints for each machine. Those obtained with 1-hot next states (conventional face constraints) and those without the next state field (output-field face constraints). Results are summa-

rized in Table I. Machines *donfile* and *s1a* have been eliminated because they have equal outputs for every input-present state combination specified. In total, the cost of implementing the benchmark is similar. However, individual results are quite different. In 8 of the 18 machines tp's counts obtained with each approach differ equal or more than 25%. Each one wins in half the cases. The two strategies can be considered as opposite as while the cardinality of minimized symbolic cover with 1-hot next states is an upper bound for the cardinality of the encoded cover, the other one represents a lower bound.

These results suggest that it is worth exploring an intermediate approach which takes into account the implementation of the next state lines but allows the sharing of logic among them. We propose to modify the set of face constraints dealing with during the encoding step by employing a column based assignment algorithm. Thus, before the building of each column, symbolic minimization of the original description substituting next state field with columns generated so far is performed. This aims at deriving new face constraints (dynamic face constraints) which are added to the current set or allow to modify existing ones.

Table I: conventional / output-field constraints.

FSM	*conv. tp*	*output-field tp*	***FSM***	*conv. tp*	*output field tp*
s208	25	20	*pma*	45	57
s420	25	20	*styr*	94	86
dk16	59	77	*tbk*	154	59
ex1	48	46	*s820*	76	81
ex2	29	40	*s832*	72	73
keyb	48	115	*planet*	91	98
s1	80	64	*s1494*	139	139
sand	101	110	*s1488*	133	132
tma	33	35	*scf*	148	159

5.- New Algorithm Description

5.1 The dynamic approach

Figure 1 shows the pseudo-C description of the proposed algorithm for state assignment. The core of the algorithm is the function **Assigns_column** which generates a column of the encoding matrix, CODES. Note that it works with a different set of constraints, derived by function **Generates_constraints**, at each iteration. After having produced one encoding column, function **Selects_phase** chooses between it and its complement. Next, we explain with more detail each function.

Let us start by function **Assigns_column** and suppose the *j*-th column of the encoding is to be generated. Initially all its bits are assigned to 1. The algorithm assigns bits to 0

```
Face_constraints = symbolic_minimization;
/* (description with 1-hot coded next states)*/
Constraint_matrix = Face_constraints;
for(j= 1; j <= number of variables; j++)
{Constraint_matrix =+ Generates_constraints();
Assigns_column(j, Constraint_matrix );
Selects_phase();
}
Assigns_column(j, Constraint_matrix )
{ while(column is invalid)
i = Selects_Symbol_to_assign_0(CODES, Constraint_matrix);
CODES[j][i] = 0;
}
Generates_constraints(j)
{Dynamic_constraints = symbolic_minimization;
/* ( substituying next states by j -1 columns generated));
adds  Dynamic_constraints to Constraint_matrix;
}
```

Figure 1.- Description of the algorithm.

until the resulting column together with the *j* -1 previously built columns forms a valid partial encoding. This is, it is possible to distinguish every state with the columns still not generated. The key of the procedure is the selection of which bit to assign to 0 each time. This task is accomplished by function **Selects_Symbol_to_assign_0**. It evaluates a cost for each bit (symbol) which can be fixed to 0 without avoiding the generation of a valid partial encoding. Then the bit which maximizes this cost is selected and assigned to 0. We propose to use as cost function a weighted sum of the satisfied seed dichotomies. The weight of each seed dichotomy is related to the fraction of associated seed dichotomies from same face constraints satisfied by previously generated columns. This aims at favoring the satisfaction of dichotomies leading to the fulfillment of group constraints.

Function **Generates_constraints** produces the set of constraints that will guide the generation of the *j*-th column using the *j*-1 columns already generated. In the original symbolic description the next state field is substituted by the partial encoding. The *Dynamic_constraints* obtained are added to current constraints, *Constraint_matrix*. Note that conventional face constraints are also included in *Constraint_matrix* at the beginning of the algorithm.

Function **Selects_phase** takes advantage of the fact that a valid partial encoding is preserved under complementation of columns. The column produced by **Assigns_column** or its complement will be used for the final encoding depending on which one results in a smaller minimized symbolic cover.

The above algorithm description fits both two level and multi-level implementations: depending on the target style a two-level multiple-valued minimizer or a multi-level multiple-valued one will be used.

5.2 Reshaping the conventional face constraints

The algorithm above described compares favorably with standard state assignment tools as will be shown in next Section. However, simply adding the constraints generated at each iteration is not the only way in which information on partial codes can be exploited. Next, we illustrate that from the minimized covers which dynamic constraints are obtained from, it can be concluded that a given conventional constraint does not require to be satisfied, or that any constraint from a given set can be satisfied instead. This is, information on partial codes can be used so that not only new face constraints are added before the next encoding column is generated, but conventional constraints are modified in order to ease its satisfaction. Let us consider machine *bbara* from IWLS'93 benchmark set. Part of The minimal symbolic cover is shown in Figure 2a with the conventional constraints in bold. Now assume that the strategy described in Section 5.1 is adopted. So, let us suppose that the encoding column in Figure 2b has been obtained by any algorithm. Before generating second encoding column multiple-valued minimization is applied incorporating the encoding column in Figure 2b and, producing the cover in Figure 2c. Note that a constraint consisting of all the states, which always holds, has appeared. Also note that the symbolic implicant in which it is contained (in bold in Figure 2c) covers symbolic implicant with an asterik in Figure 2a. This is, the input field of symbolic implicant in Figure 2c contains the input field of symbolic implicant from Figure 2a, the present state field contains the present state field, the output field dominates the output field and there is a 1 in the next state field of symbolic implicant in Figure 2c, and the state in the next state field in Figure 2a, *st1*, has a 1 in the encoding column we are using (Figure 2b). If the code of *st1* is completed with zeros, the symbolic implicant with the asterik in Fig. 2a can be implemented with only one product term in the boolean domain even if its associated face constraint is violated because certain relations hold among the codes of the states. Thus, such constraint does not need to be satisfied.

In general, minimization incorporating information on partial codes does not lead to the elimination of any conventional constraint but it can simplify its satisfaction. This situation arises when a covering relation holds between a symbolic implicant from an intermediate minimization and one from the conventional, but the present state field of the first does not contain the whole set of states. In this case the conventional constraint can be reshaped. The modification involves the addition of inspecifications. Let us call *D* the dynamic constraint and *G* the conventional one. Those states in *D* but not in *G* are taken as inspecifications in the new *G*. So, it does not matter whether their codes intersect the minimum cube containing the symbols originally in *G*. So, satisfaction of *G* has been simplified. Inclusion of inspecifications can make satisfiable a constraint *G* which is not.

6.- Experimental Results

This Section shows the results obtained with DISA, a C implementation of the algorithm described in Section 5.1. Table II summarizes the results obtained with different state assignment algorithms for a subset of the FSM benchmark and two level implementations. The algorithms included are: 1) NOVA *i-hybrid*, 2) NOVA *io-hybrid*, 3) ENCORE (results taken from [7]), 4) COL, a column based on function **Assigns_Columm** working with the conventional input constraints at every iteration, and 5) DISA, the dynamic approach described in Section 5.1. Note that NOVA *ihybrid*, ENCORE and COL handle the same set of face constraints. DISA updates these constraints during the encoding step but uses same column generation algorithm than COL. Finally, NOVA *io-hybrid* deals with both face and dominance constraints. The five algorithms generate minimum length codes. The number of product terms in the combinational component for the encodings obtained with each program after logic minimization is shown in Table II. Times are given in seconds in a SparcStation 10.

First we compare NOVA *i-hybrid*, an standard tool, and COL in order to validate the cost function used in the generation of the encoding columns. COL outperforms NOVA in 9 of the 20 machines while NOVA obtains better results than COL also in 9 machines. Implementation of the 20 examples is a 7% more expensive with NOVA. Comparing the *i-hybrid* and the *io-hybrid* options slightly better total results are produced with the *io-hybrid* approach. DISA obtains better results than COL in 13 of the 20 machines. Only in two cases worse results were produced by DISA. In global, implementation of benchmark set is around 20% more expensive with an standard tool like NOVA than with the proposed dynamic approach DISA. A version of DISA incorporating the arguments stated in Section 5.2 is currently being developed.

7.- Conclusions

This papers presents a novel algorithm for the state assignment of sequential circuits. Limitation of commonly used approaches which follow a two step encoding paradigm are overcome by the new approach by modifying the constraints during the encoding phase. Modifications are made on the basis of actual relations among the codes of the states. We have shown that the preliminary version of the new algorithm compares favorably to standard state assignment tools. The ideas supporting a more elaborate version currently being developed have been sketched. We think the work presented in this papers paves the way for a new treatment of input-output encoding problems.

8.- References

[1] P. Ashar, S. Devadas and A. R. Newton: “Sequential Logic Synthesis”, Kluwer Academic Pub., 1992.

[2] G. de Micheli, R. K. Brayton y A. L. Sangiovanni-Vincentelli: “Optimal State Assignment of Finite State Machines”, IEEE Trans. on CAD, Vol. CAD-4, pp. 269-285, July 1985.

[3] R. Rudell and A. L. Sangiovanni-Vincentelli: “Multiple-valued Minimization for PLA Optimization”, IEEE Trans. on CAD, Vol. CAD-6, pp.727-751, September 1987.

[4] S. Malik, L. Lavagno, R.K. Brayton and A. Sangiovanni-Vincentelli, “Symbolic Minimization of Multilevel Logic and the Input Encoding Problem”, IEEE Trans. on CAD, Vol. 11, no. 7, pp. 825-843, July 1992.

[5] J.L. Huertas y J.M. Quintana: “Efficiency of State Assignment Methods for PLA-Based Sequential Circuits”, IEE Proc., Vol. 136, Pt E, no. 4, pp. 247-250, July 1989.

[6] T. Villa y A.L. Sangiovanni-Vincentelli: “NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementation”, IEEE Trans. on CAD, Vol. CAD-9, no. 9, pp. 905-923, Sept. 1990.

[7] C.J. Shi and J.A. Brzozowski, “An efficient Algorithm for Constrained Encoding and Its applications”, IEEE Trans. on CAD, V.12, no.12, pp. 1813-1826, Dec. 1993.

[8] G. de Micheli, “Symbolic design of Combinational and Sequential Logic Circuits Implemented by Two-Level Macros”, IEEE Trans. on CAD, Vol. 5, pp. 597-616, September 1986.

[9] S. Devadas and A. R. Newton, “Exact Algorithms for Output encoding, State Assignment and Four LevelBoolean Minimization”, IEEE Trans. on CAD, Vol. 10, no. 1, pp. 13-27, January 1991.

[10] K. McElvain, “LGSynth93 Benchmark Set: Version 4.0”

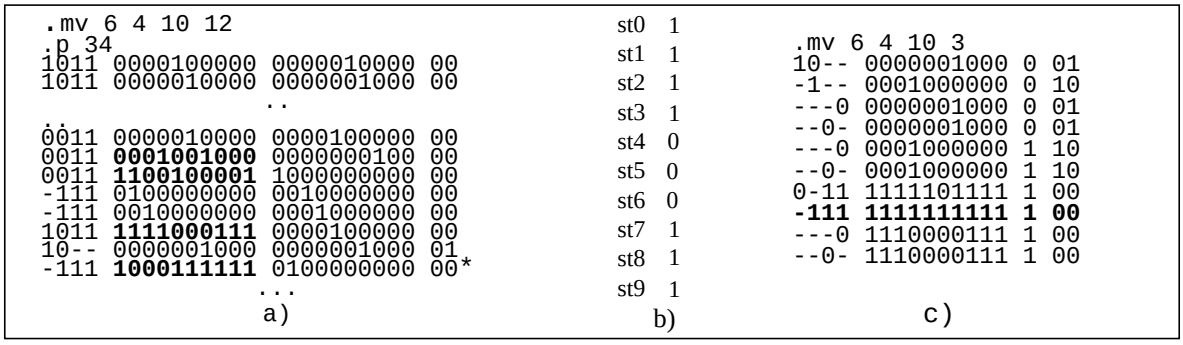

Figure 2.- a) conventional minimized symbolic cover; b) encoding column; c) intermediate minimized symbolic cover.

Table II: Comparison of state assignment algorithms.

total: summ tp’s counts all machines; total$ summ tp’s count machines with ENCORE results available

FSM	*tp ihybrid*	*time ihybrid*	*tp iohybrid*	*time iohybrid*	*tp ENCORE*	*tp COL*	*tp DISA*	*time DISA*
s208	25	8.9	24	62.3	*	26	23	3.2
s420	25	8.7	24	62.9	*	26	23	4.1
dk16	59	69.3	62	385.4	58	63	70	8.2
donfile	35	138.2	47	455.4	18	23	18	5.7
ex1	48	14.6	52	227.6	45	46	42	14.3
ex2	29	1.3	44	74.0	32	32	32	4.1
keyb	48	9.0	102	93.4	51	85	48	18.8
s1	80	2.3	75	239.1	86	87	76	11.0
s1a	76	1.9	73	213.1	73	76	62	3.3
sand	101	15.0	99	585.9	100	98	98	150.5
tma	33	15.8	35	82.3	*	34	34	19.0
pma	45	45.0	51	140.3	*	45	50	24.4
styr	94	27.4	106	762.6	*	93	87	38.1
tbk	154	339.0	94	2993.2	129	52	52	84.7
s820	76	6.7	66	364.8	*	63	58	10.1
s832	72	6.1	64	388.0	*	66	62	10.0
planet	91	21.9	99	1657.6	90	89	89	151.7
s1494	139	150.4	120	2064.3	*	126	104	69.9
s1488	133	154.9	119	1985.1	*	137	101	68.3
scf	148	214.7	143	12242.1	140	151	136	751.2
total	1511		1499			1418	1265	
total $					822	802	723	