An Algorithm for Face-Constrained Encoding of Symbols Using Minimum Code Length

Manuel Martínez[1], María J. Avedillo[1,2], José M. Quintana[1,2] and José L. Huertas[1,2]

[1]Instituto de Microelectrónica de Sevilla. Centro Nacional de Microelectrónica.
[2]Departamento de Electrónica y Electromagnetismo. Universidad de Sevilla.

Abstract

Different logic synthesis tasks have been formulated as input encoding problems but restricted to use a minimum number of binary variables. This paper presents an original column based algorithm to solve this optimization problem. The new algorithm targets economical implementation of face constraints unlike conventional algorithms which do not care about infeasible ones. Experimental results that demonstrate the superiority of the new method versus conventional tools and a previous algorithm specifically developed for the minimum length encoding problem are shown. An state assignment tool which core is the new algorithm is evaluated by implementing an standard benchmark of sequential circuits. It compares very favorably to well known tools like NOVA.

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. The difficulty of encoding problems resides in the modeling of the subsequent optimization step. Many encoding methods are based in optimizing the symbolic representation (using multivalued minimization techniques), to produce a set of constraints on the relationship between the binary codes for different symbols so that the optimization at the symbolic level will be preserved in the boolean domain [1], [2], [3], [4], [5]. Different types of encoding constraints are generated depending on whether symbols appears as input or/and outputs in the symbolic representations to be optimized as well as on the symbolic minimization algorithm used.

The most widely used type are face constraints and so a high number of algorithms which assign binary codes to a set of symbols satisfying face constraints have been developed [6], [7], [8]. These face embedding algorithms have been applied to many different synthesis tasks such as the encoding of mnemonic input fields of the microcode, encoding of symbolic inputs that appear in high level descriptions or the state assignment of sequential circuits However, in some applications, satisfying the complete set of constraints involves such an increase of the length of the codes that gains in terms of area are not usually achieved. Thus, it is a current practice encoding n symbols using $\lceil \log_2 n \rceil$ binary variables, the minimum number required to distinguish every symbol. In this paper we focus on this optimization problem called partial face-constrained encoding and consisting in maximizing a gain function of the face constraints using minimum code length. We propose a new algorithm to solve this problem targeting a two-level design style and show experimental results from an state assignment tool developed on its basis.

The rest of the paper is organized as follows. Section 2 introduces basic definitions and presents the rationale for the work. Section 3 describes the new approach. In Section 4 results obtained for partial face constraint problems with different algorithms including the new one are shown and compared. Also, a tool for state assignment is described and evaluated.

2 Definitions and rationale

In this section, we review several definitions [1], [3], [9] to give a mathematical formulation of the face constrained encoding problem and introduce some new ones to describe developed approach.

A **cube** (product term) in B^{nv} of dimension p (p-cube) is a collection of any 2^p points (minterms) that have exactly $nv - p$ bits all the same. It can be represented by a row vector with nv elements belonging to {0, 1, -}. A **super-cube** of a set of cubes is the smallest cube containing all the minterms contained in the set of cubes. 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 where the i-th row represents the code assigned to S_i, and the j-th column represents bit j of the encoding. A **group constraint** gc on a set of symbolic inputs $S = \{S_1, S_2, ..., S_n\}$ 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 symbolic inputs absent from S'. A (seed) **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

input symbols that belongs to gc_1 and B_2 contains exactly one of the input symbols that does not belong to gc_1. A group constraint is satisfied if and only if the whole set of its associated dichotomies is satisfied. Satisfaction of a dichotomy *d* requires that subset B_1 be distinguished from subset B_2 of *d* by at least one encoding bit. I_k, the **intruder set** of symbols in constraint L_k, are the symbols pertaining to $super^1(L_k)$ not in L_k. Notice that a constraint is satisfied iff $I_k = \varnothing$. Two constraints are ***nv*-compatible** when it is possible to satisfy them simultaneously in B^{nv}.

The following example summarizes the two step encoding strategy described in the introduction and illustrates the identity of the partial problem.

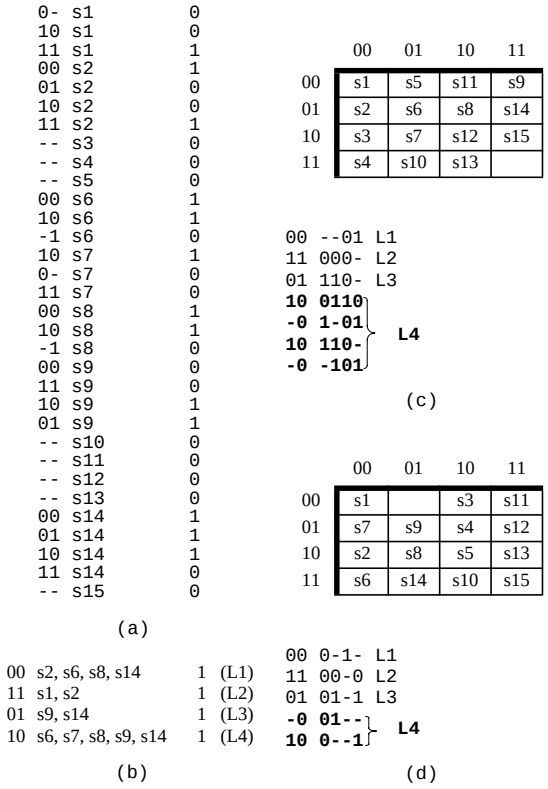
Figure 1a shows a function with a symbolic input. The minimized symbolic representation is shown in Figure 1b. Clearly a boolean representation with the same number of implicants than the symbolic can be obtained if each symbolic implicant with more than one symbol is taken as a face constraint during the encoding process and the complete set is satisfied. If one or more constraints are not satisfied by the encoding, then the boolean cover can have more implicants or product terms. For example encodings depicted in Figure 1c and 1d satisfy face constraints corresponding to rows 1, 2 and 3 and violate row number 4. Both encodings produce two level implementations with more than four product terms which is the cardinality of the minimized symbolic representation. However symbolic implicant corresponding to row number 4 is implemented with four product terms when encoding c is used and with only two when encoding d is used.

This example shows that two encoding which satisfy the same subset of group constraints can result in boolean implementations with different cost. On the other hand when solving the partial face-constrained encoding problem there is not guarantee that the complete set of constraints can be satisfied. This means that conventional approaches to solve this problem which tries to maximize the number of satisfied face constraints, this is, the number of symbolic implicants which can be implemented with a single product term could lead to suboptimal results [10]. Also it can be shown that trying to maximize the number of satisfied seed dichotomies which was claimed to be a better approach to the partial problem does not guarantee cost effective implementation of the complete set of symbolic implicants.

We propose a new algorithm for the partial face constrained encoding problem which aims at minimizing the number of product terms in a sum of product representation of the encoded constraints[2]. This objective has been identified as suitable for this problem [10], [11], [12].

1 Smallest cube which contains the codes (0-cubes) of all the symbols in L_k. This is super-cube of those 0-cubes.

2 A Boolean function is associated with each input constraint. Its on-set contains the codes of the symbols in the constraint and its off-set contains the codes of the symbols not in the constraint. The used codes are in the dc-set.



3 The new algorithm

The new algorithm follows a dichotomy satisfaction strategy while trying to minimize the number of cubes in a SOP form of the encoded constraints. The later is achieved by following main features:

1) the use of a novel constraint matrix notation let us handle concepts, like cubes, cube dimension, etc..., still unused in dichotomy based procedures. This is the key to allow dynamic detection, this is, concurrently with the encoding step, of infeasible constraint.

2) a detailed analysis of the implementation of infeasible constraints has originated the concept of guide-constraint. Satisfying the guide-constraint associated with an infeasible one contributes to an economical implementation of the later.

Figure 2 shows the pseudo-C description of the proposed algorithm. The core of the algorithm is the function *Solve*() which generates a column of the encoding matrix. Note that it works with a different set of constraints, derived by function *Update_constraints*, at each iteration. The update of constraint is based in the identification of constraints which can not be satisfied independently of the code column still not generated (infeasible constraints). This task is accomplished by function *Classify*(). A new constraint called guide-constraint is generated for each identified infeasible constraint and added to the set of constraints handle by the encoding process.

Next we define the notation used in the algorithm. Then, we introduce the concept of guide-constraint and the rationale for using them during the encoding process. Finally, we explain main functions of the algorithm.

```
PICOLA()
{
get_constraint_matrix();
for each column{
Update_constraints();
Solve(); }}

Update_constraints(){
infeasible_constraints = Classify();
Add_guide_constraints(infeasible constraints);
}
```

Figure 2. Pseudocode for the new algorithm.

3.1 New notation for constraint matrix

Usually, a set of face constraints on n symbols denoted $\{S_1, S_2, ..., S_n\}$ is represented by a constraint matrix L_{rxn} where r equals the number of constraints and $L_{ij} = 1$ if the symbolic input S_j belongs to the i-th constraint and 0 otherwise. The new algorithm uses the constraint matrix to store relevant information During the encoding process. After having generated the i-th code-column, the zeros in the constraint matrix corresponding to dichotomies which have been satisfied by this column are changed to $i + 1$. This is, in our procedure the constraint matrix remembers which encoding column satisfies each dichotomy. Let us call nv the number of encoding variables. Through the new notation we have available updated information about $super(L_k)$, the supercube of a group constraint:

$$\dim[super(L_k)] = nv - \begin{array}{l}\textit{number of code columns}\\ \textit{participating in } L_k.\end{array}$$

and the intruder set of a constraint L_k

$I_k = \{ s_i \in S / L_{ki} = 0 \}$

Example2.- Following with example 1c, before third column of code generation, matrix L_{rxn}

$$L_{4x15} = \begin{bmatrix} 012221010222212 \\ 112220033222202 \\ 332223301222212 \\ 002221111222212 \end{bmatrix}$$

At this stage, from third constraint data it can be concluded that the maximum dimension of $super(L_3)$, $\dim[super(L_3)] = 2$ because $nv = 4$ and 1^a and 2^a code columns are participating in L_3. It also informs that $super(L_3)$ also contains s_8. This is $I_3 = \{s_8\}$.

3.2 Guide-constraints

Theorem I.- If the symbols in the intruder set, I_k, of a L_k, forms a cube which does not intersect with any symbol in L_k, then L_k can be implemented with a number of cubes equal to $dim[super(L_k)] - dim[super(I_k)]$.

Proof: Constructive. A collection of cubes covering all minterms (codes) corresponding to the symbols in L_k without intersecting with the codes of the symbols in I_k is built up. Let us call M to the set of literals appearing in $super(I_k)$ and not in super(L_k). For each literal m_j in M a cube is formed by starting with $super(I_k)$, complementing m_j and changing the remaining literals in M to don´t cares. There are $nv - dim[super(L_k)]$ literals in $super(L_k)$ and $nv - dim[super(I_k)]$ literals in $super(I_k)$. Thus the cardinality of M is $nv - dim[super(I_k)] - nv + dim[super(L_k)]$.

Example 3.- Going back to example 1, using the encoding shown in Figure 1c, $super(L_4) = 0---$. The intruders are s_1 and s_2 and $super(I_4) = 00\text{-}0$. Clearly we are in conditions to apply the constructive procedure of Theorem. The collection of cubes is {(01--), (0--1)} which actually implements L_4.

Definition.- A guide-constraint L', associated to a group constraint, L, is the group constraint of the symbols in the intruder set of L.

Satisfying guide-constraints can improve the implementation of infeasible constraints on the basis of previous theorem.

Example 4.- Following with example 1, we can think in encoding 1c as satisfying the guide constraint associated with L_4, namely L_4' :110000000000000 because s_1 has been given 0000 and s_2 0010 and so $super(\{s_1,s_2\}) = 00\text{-}0$ leading to an implementation of L_4 with only two cubes as explained in example 3. Such an implementation is optimum because as L_4 is infeasible in B^4 it can not be implemented with a single cube.

Although a guide-constraint is not completely fulfilled, satisfying a group of its associated dichotomies can contribute to improve the implementation of the original constraint. This is because of every group of such dichotomies marked in same column constitutes a cube for a subgroup of symbols in the original constraint.

3.3 Function *Classify*()

Function *Classify*() identifies infeasible constraints to be substituted by their guide-constraints. Before generating the next encoding column, information on the partial encoding already built is used in order to determine whether each face constraint can still be satisfied using minimum code length. The efficient implementation of this task relies on the way information is stored in constraint matrix and in a novel procedure to determine the nv-compatibility of a pair of constraints. Clearly, once a constraint is satisfied, those ones which are not nv-compatible to it are identified as infeasible

Checking a pair of constraints for nv-compatibility

Some of the following definitions and relations belong to the background of face embedding theory [8].

Definition.- The *son constraint of L_A and L_B*, L_{AB} or L_{son} contains the symbols both in L_A and L_B.

Definition.- The don´t care set of a constraint L_k are the unused codes in its cube implementation. We will refer to the number of elements in this set as $dc(L_k)$.

Checking for *nv*-compatibility is different depending on the constraints involved:

a) If their son constraint is not null or void

First, basics conditions of Boolean algebra are checked:

Conditions I:

if $L_{son} \neq L_{father} \Rightarrow dim(L_{father}) > dim(L_{son})$

if $L_{son} = L_{father} \Rightarrow dim(L_{father}) = dim(L_{son})$

Conditions II: $dc(L_{son}) \leq dc(L_{father})$

Where the cubes satisfying the constraints have to adjust their dimension according to this conditions.

Then, checking for *nv*-compatibility is done on the basis of the following theorem.

Theorem.- If L_A and L_B are *nv*-compatible then

i) $dim[super(L_A,L_B)] \leq nv$, (super containing all the symbols in L_A and L_B)

ii) the expression to calculate this dimension is

$$dim(super(L_A, L_B)) = dim(L_A) + dim(L_B) - dim(L_{AB})$$

Proof: i) Is trivial because of we are encoding in B^{nv}. ii) is derived from the definition

$dim[super(L_A, L_B)]=nv-(col_A+col_B-col_{AB})$, where *col* indicates columns participating of the different constraints, notice that $col_A+col_B-col_{AB}$ are the number of code columns shared by L_A and L_B. Adding $(+nv-nv)$ to the last expression

$dim[super(L_A,L_B)]=(nv-col_A)+(nv-col_B)-(nv-col_{AB})$

and relation is directly demonstrated.

b) If their son constraint is null a sufficient condition to satisfy is: $dc(L_A) + dc(L_B) \leq dc(S)$

3.4 Function solve()

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 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 assign to 0 each time. A cost function is evaluated 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. Concerning the selection of the cost function several comments are in order. We use as cost function a weighted sum of the satisfied seed dichotomies. The weight of each seed dichotomy is related to the size, type (original or guide) of its face constraint and depend on the encoding column generated so far. This aim at favoring the satisfaction of dichotomies leading to the fulfillment of face constraints as well as to the effective implementation of those that are not going to be satisfied using Theorem I.

4 Experimental results

In this section we summarize the results obtained with PICOLA (Partial Input Column based Algorithm), a C implementation of the algorithm described in section 4, on a wide set of standard input encoding problems (derived form IWLS'93 FSM benchmark [13] substituting next state field by an one-hot code) 10]. Table I shows the number of group constraints for each problem; and the number of cubes required to implement the constraints with the minimum length encodings obtained with three different algorithm. The algorithms included are NOVA [6], an conventional tool, ENC [10] and the new one. PICOLA outperforms NOVA in 16 cases while NOVA outperforms PICOLA in only 7 cases. In global, implementation of benchmark is 11% more expensive with an standard tool like NOVA. Comparing ENC and PICOLA, both targeting the partial problem, the quality of the results is similar. We could not carry out a comparison of times with ENC because data is not available in [10]. However, because of the intensive use of logic minimization it involves, it can be concluded that PICOLA is significantly faster than ENC. In fact, ENC is not practical for medium and large examples. Note that It is reported that it fails to solve problem *scf. W*e have included comparison with ENC because it has been successfully applied to different synthesis tasks which are modeled by the problem we are dealing with.

Finally, in order to show an application of the algorithm, we have developed an state assignment tool on the basis of the proposed algorithm. In [14] a column based state assignment algorithm is presented that works with face constraints but takes advantage of the information on the code columns which have been generated so far to overcome limitations of using only face constraints for the state assignment problem. The new tool approach the problem using this model. Table II compares the results obtained with those from NOVA *i_hybrid and* NOVA *io_hybrid*. The size of a two-level implementation of the combinational component of IWLS'93 FSMs, *size*, is shown in this Table. Also, execution times, normalized to those of NOVA, *i_hybrid* are given. New algorithm compares favorably to all of them.

5 Conclusions

A new column-based algorithm for the partial face-constrained encoding problem has been presented. The concept of guide constraint has been introduced and its usefulness in achieving economical implementations of infeasible constraints has been demonstrated. An effective method to detect

infeasible constraints to be substituted by guide constraints has been developed. It takes into account the code columns generated so far as well as the number of binary variables to be used so that the detection is dynamically done during the encoding process. This original concept is easily portable to any algorithm attempting to solve the partial problem. We have shown that the quality of the results obtained with a preliminary version of the new algorithm is comparable to existing approaches but time performance is superior. An state assignment tool have been developed on its basis which clearly outperforms well known standard tools.

6 References

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

[2] 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.

[3] 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.

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

[5] T. Villa, A. Saldanha, R.K. Brayton and A.L. Sangiovanni-Viincentelli, “Symbolic Two-Level Minimization”, IEEE Trans. on CAD, Vol. 16, No. 7, pp. 692-708, July 1997.

[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. Brozowski, “An efficient Algorithm for Constrained Encoding and Its applications”, IEEE Trans. on CAD, Vol. 12, no. 12, pp.1813-1826, December 1993.

[8] E. I. Goldberg, T. Villa, R.K. Brayton and A.L. Sangiovanni-Viincentelli, “Theory and Algorithms for Face Hypercube Embedding”, IEEE Trans. on CAD, Vol. 17, No. 6, pp. 472-488, June 1998.

[9] S. Yang and M.J. Cieselski, “Optimum and Suboptimum Algorithms for Input Encoding and Its Relationship to logic Minimization”, IEEE Trans. on CAD, Vol. 10, no. 1, pp. 4-12, Jan, 1991.

[10] A. Saldanha, T. Villa, R.K. Brayton and A.L. Sangiovanni-Viincentelli, “Satisfaction of Input and Output Encoding Constraints”, IEEE Trans. on CAD, Vol. 13, no. 5, pp. 589602, May 1994.

[11] S. Devadas, A.R. Wang, A. R. Newton and A. Sangiovanni-Vincentelli, “Boolean Descomposition in Multilevel Logic Optimization”, IEEE Journal of Solid State Circuits, Vol. 24, no.2 April 1989.

[12] B. Lin and A.R. Newton, “A Generalized Approach to the Constrained Cubical Embedding Problem”, International Conference on Computer Design: VLSI computers and Processors, pp. 400-404, 1989.

[13] R. Lisanke, “Introduction of Synthesis Benchmarks,” Int. Workshop on Logic Synthesis, North Carolina, 1991.

[14] M. Martinez, M.J. Avedillo , J.M. Quintana and J.L. Huertas, “A Dynamic Model for the State Assignment Problem”, DATE98, Proceedings.

FSM	const	NOVA	ENC	PICOLA
		cubes	cubes	cubes
bbara	4	8	*	5
bbsse	5	12	8	8
cse	12	24	18	17
dk512	10	12	11	12
ex3	6	8	*	8
ex5	7	11	*	10
ex7	6	10	*	9
kirkman	25	58	58	57
lion9	10	10	*	10
mark1	4	6	*	5
opus	2	2	*	2
train11	11	13	*	12
s208	5	8	*	7
s420	5	8	*	7
dk16	34	43	48	55
donfile	24	48	39	43
ex1	11	19	19	19
ex2	8	10	*	12
keyb	33	41	*	41
s1	14	14	14	14
s1a	14	14	14	14
sand	7	8	8	8
tma	11	19	*	16
pma	18	30	*	30
styr	18	29	26	23
tbk	98	284	237	208
s820	15	17	*	17
s832	15	17	*	17
planet	12	12	12	13
s1494	29	81	*	66
s1488	29	70	*	63
scf	14	21	fails	21

Table I.- Comparison between different approaches.

FSM	NOVA -ih *size*	time ratio	NOVA -ioh *size*	time ratio	NEW *size*	time ratio
s208	1375	1	1320	7.00	1265	0.36
s420	1375	1	1320	7.23	1265	0.47
dk16	1298	1	1364	5.56	1364	0.19
donfile	700	1	940	3.40	660	0.04
ex1	2496	1	2704	15.59	2392	0.98
ex2	609	1	924	56.92	609	3.15
keyb	1488	1	3162	10.38	1488	2.08
s1	2960	1	2775	103.91	2738	4.78
s1a	2812	1	2701	112.30	2738	1.73
sand	4646	1	4554	39.06	4094	10.03
tma	1155	1	1225	5.21	1190	1.20
pma	1755	1	1989	3.12	1833	0.54
styr	4042	1	4558	27.83	3913	1.39
tbk	4620	1	2820	8.83	1620	0.25
s820	5320	1	4620	54.45	4550	1.50
s832	5040	1	4480	63.61	4550	1.64
planet	4641	1	5049	75.66	4743	6.92
s1494	7367	1	6320	13.76	5141	0.46
s1488	7049	1	6307	12.81	4982	0.44
scf	19388	1	18733	56.41	18078	3.50
Total	1		0.97		0.86	

Table II.- State Asignment results: size and time performance.