A New Paradigm for Dichotomy-based Constrained Encoding

Olivier Coudert
Synopsys, Inc., 700 East Middlefield Rd.
Mountain View, CA 94043

Abstract

One essential step in sequential logic synthesis consists of finding a state encoding that meets some requirements, such as optimal implementation, or correctness in the case of asynchronous FSMs. Dichotomy-based constrained encoding is more general than other constrained encoding frameworks, but it is also more difficult to solve. This paper introduces a new formalization of this problem, which leads to original exact and heuristic algorithms. Experimental results show that the resulting exact solver outperforms the previous approaches.

1 Introduction

A k-bit encoding of a set of states S is a mapping α from S into $\{0,1\}^k$. A *dichotomy* is an unordered pair $\{S_0, S_1\}$ of disjoint subsets of S. An encoding satisfies a dichotomy $\{S_0, S_1\}$ iff there is a bit of the encoding that distinguishes the states of S_0 from the states of S_1, i.e., this bit has the value 0 for the states of S_0 and the value 1 for the states of S_1, or vice versa.

Definition 1 *Given a set S of states and a set D of dichotomies, the* constrained encoding problem *consists of finding a minimum-length encoding that satisfies all the dichotomies.*

Fig. 1 shows a minimum-length encoding satisfying the three dichotomies $\{\{1,3\},\{2,4\}\}$, $\{\{3,4\},\{1\}\}$, $\{\{2,3\},\{4\}\}$, and $\{\{1,2,3\},\{\}\}$, on the set of states $S = \{1,\ldots,4\}$

	$\alpha(s)$		
1	0	0	0
2	1	$-$	0
3	0	1	0
4	1	1	1

Figure 1: A three-bit encoding.

Originally introduced by Tracey [12], dichotomy-based constrained encoding can be used to solve the following problems: generate an asynchronous implementation that is critical race free [12], or independent from the gate and wire delays [13]; generate a minimum-area PLA implementation [7, 8, 9, 15]; generate optimal PLA implementations of Boolean expressions [3, 4]; produce a state assignment for an event-based specification [6]; and solve hazard-free minimization and asynchronous FSM encoding for multiple-input changes [5].

Fig. 2 illustrates a unified framework [11, 15] for constrained encoding in sequential logic synthesis: dichotomies can be used to express the correctness of the implementation (race-free, hazard-free, or speed-independent implementation), or to express some optimization criterions (area and speed). It is more general than some other frameworks, e.g. [14], which cannot cope with the state assigment of asynchronous circuits.

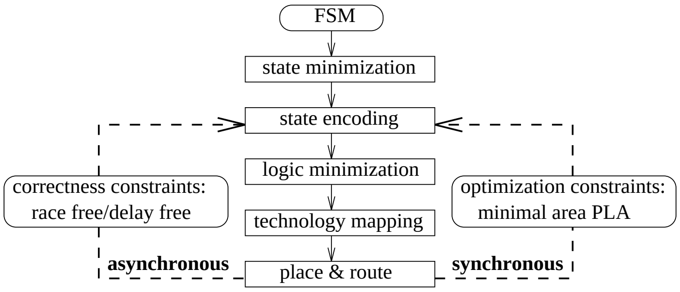

Figure 2: A constrained encoding framework.

This paper addresses the exact resolution of the dichotomy-based constrained encoding problem. Section 2 outlines the state-of-the-art. Section 3 gives a new formulation of the problem and explains how it can be solved. Section 4 discusses some experimental results.

2 Terminology & Previous Work

A *compatible set* is a set of dichotomies that can be simultaneously satisfied by a single bit encoding. Two dichotomies are *compatible* iff they form a compatible set. A compatible set is necessarily made of pair-wise compatible dichotomies, but the converse is false. For example, the three dichotomies $\{\{1,2\},\{\}\}$, $\{\{1,3\},\{\}\}$, and $\{\{2\},\{3\}\}$ are pair-wise compatible, but there is no single bit encoding that satisfies the three of them at the same time.

Tracey introduced the first exact algorithm for constrained encoding [12]. His method, which laid down the basis of most of the subsequent works, is as follows:

(1) Build the set M of all the maximal compatible sets from the given set of dichotomies D;
(2) Find a minimum number of elements of M that cover all the dichotomies.

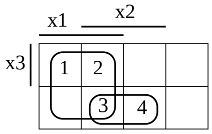

Figure 3: Face hypercube embedding.

Step (2) is NP-complete, and the cardinality of M can be exponential w.r.t $|D|$.

A compatible set can be represented by a dichotomy, e.g., the compatibility of $\{\{1\},\{2,3\}\}$ and $\{\{1,4\},\{2\}\}$ is denoted with the new dichotomy $\{\{1,4\},\{2,3\}\}$. One can solve (1) by computing pair-wise compatible dichotomies, adding the resulting compatible sets to the set of dichotomies, and iterating this process until all the compatible sets are maximal [12, 15, 9]. However this method is applicable to small size problems because one needs to examine a huge number of dichotomy pairs. The reason is that in practice, dichotomies involve a small number of states, and consequently most of them are pair-wise compatible. For example, n dichotomies whose subsets S_0's and S_1's are all mutually disjoint produce $n(n-1)$ new dichotomies denoting the compatibility between any two dichotomies. Thus the number of dichotomies after k (k small) iterations is about n^{2^k}.

An *ordered* dichotomy (S_0, S_1) is an ordered couple of disjoint subsets of S. One says that two ordered dichotomies (S_0, S_1) and (S'_0, S'_1) are compatible iff $S_0 \cup S'_0$ and $S_1 \cup S'_1$ are disjoint. Two dichotomies $\{S_0, S_1\}$ and $\{S'_0, S'_1\}$ are compatible iff (S_0, S_1) is compatible with (S'_0, S'_1) or (S'_1, S'_0). The concept of compatibility becomes then more manageable, since a set of ordered dichotomies is compatible iff they are pair-wise compatible. One can solve (1) by associating two ordered dichotomies to each (unordered) dichotomy, computing all the maximal sets of compatible ordered dichotomies, and then converting them back to unordered dichotomies. Based on this idea, [11] presents an algorithm that computes M in $O(|M|)$. However this method is still limited by the size of M, which can be exponential w.r.t. $|D|$.

In [2] is presented two ZBDD based algorithms to solve the constrained encoding problem. Both algorithms are based on set covering with compatible sets. They avoid the bottleneck the other methods face because ZBDDs allow to manipulate very large sets of compatible sets implicitly. However they suffer from the irreducibility of the resulting set covering problems.

Face hypercube embedding is another constrained state encoding framework [11]. A *facet* is a subset f of states of S that are constrained to be encoded within one face (cube) of the hypercube $\{0,1\}^k$, without having any other state intersecting this face. Fig. 3 shows an optimal face embedding of the two facets $f_1 = \{1,2,3\}$ and $f_2 = \{3,4\}$. The faces spanned by f_1 and f_2 are 1−− and −10 respectively.

A set of facets can be easily translated into a set of dichotomies: for every facet f, generate the dichotomies $\{f, \{s\}\}$ for $s \in S - f$, which asserts that no other state intersects the face spanned by f; also add the dichotomies $\{\{s\},\{s'\}\}$ for $s, s' \in S, s \neq s'$, which asserts that every state is distinguishable. However, there is no systematic way to translate a set of dichotomies into a set of facets.

3 Twin Graphs and Dichotomies

The methods that have been proposed in the past rely on the explicit notion of compatible sets and on covering. This section introduces a new formalization which strongly differs from this paradigm. It reduces the original problem to a *twin graph coloring*. It then explains how to solve this new problem.

3.1 Formalization

A simple (i.e., undirected and self-loop free) graph G is denoted with (V, E), where V is its set of vertices, and E its set of edges. Given a set of vertices V', we will use the notation $G - V'$ to denote the subgraph induced by $(V - V', E)$. Coloring a graph consists of assigning a color to every vertex such that there is no two vertices linked by an edge that have the same color.

Definition 2 *A* twin graph *is a pair (G, T), where $G = (V, E)$ is a simple graph, and T is a matching*[1] *on V. Each pair $\{v, v'\}$ of T is called a twin couple, and we say that v is the twin of v'. A vertex which belongs to some pair of T is a twin vertex.*

An *instance graph* of a twin graph (G, T) is a graph $G - V'$, where V' is a set of twin vertices that does not contain any twin couple. In other words, it is obtained by removing from G no more than one vertex for every twin couple.

Let us define a coloring of a twin graph (G, T) as a coloring of one of its instance graphs. The minimum coloring of a twin graph is the minimum cardinality coloring that can be obtained over all its instance graphs. In other words, it is a minimum coloring of G such that only one vertex of every twin couple needs to be colored, the other vertex being removed from the graph.

The twin graph (G, T) derived from a set of (unordered) dichotomies D is such that V is the set of ordered dichotomies, $G = (V, E)$ is the uncompatibility graph of the ordered dichotomies (obtained by linking with an edge two uncompatible dichotomies of V), and T is made of the couples $\{v, v'\}$ such that v and v' are the two ordered dichotomies resulting from a unique unordered dichotomy. From now on, we identify vertex and ordered dichotomy, edge and uncompatibility.

Theorem 1 *The minimum coloring of the twin graph derived from a set of dichotomies D gives the minimum-length encoding satisfying D.*

Proof. (sketch) An independent set of the uncompatibility graph G is a compatible set. Thus any k-coloring

[1] A matching on a set of vertices is a set of edges that has no self-loop and such that no two edges have a common endpoint.

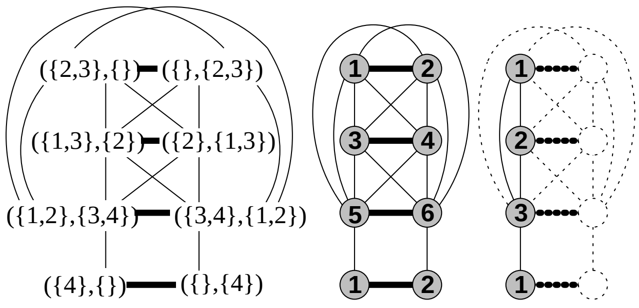

Figure 4: Coloring the twin graph of some dichotomies.

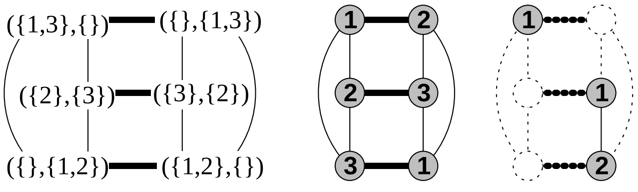

Figure 5: Twin graph coloring and symmetry breaking.

of the uncompatibility graph, which partitions V into k independent sets, yields a k-encoding. However a minimum coloring of the uncompatibility graph does not necessarily produce a minimum encoding. This is because each dichotomy generates two ordered dichotomies, and that only one of those needs to be put in the partition, i.e., needs to be colored. This is precisely the definition of the coloring of a twin graph. □

Consider the set of four dichotomies $\{\{2,3\},\{\}\}$, $\{\{1,3\},\{2\}\}$, $\{\{1,2\},\{3,4\}\}$, and $\{\{4\},\{\}\}$. Fig. 4 shows on the left the uncompatibility graph G of the ordered dichotomies. Edges between twin vertices are shown with a bold line. The middle graph is the same graph optimaly colored with 6 colors, assuming it is a "normal" graph. The graph on the right shows the optimal coloring of the *twin* graph with 3 colors. The twin vertices and their incident edges that have been removed are shown with dotted lines.

The reader may think that finding a "normal" minimum coloring of the graph resulting from removing every edge between twin vertices gives also a minimum coloring of the twin graph derived from a set of dichotomies. One reason for this intuition is the strong symmetry of such twin graphs. Indeed, a twin graph derived from a set of dichotomies can be drawn as a graph whose pairs of twin vertices are organized in rows, and the graph is completely symmetric w.r.t. to a vertical axis. In the example of Fig. 4, the minimum coloring of the graph in the middle, when ignoring the bold edges, is made of 3 colors, as is the minimum coloring of the twin graph. But this is not true in general. The reason is that ordering the dichotomies creates new edges that encode the *uncompatibility* of sets otherwise made of *pairwise compatible unordered* dichotomies.

For instance, consider the three dichotomies $\{\{1,3\},\{\}\}$, $\{\{2\},\{3\}\}$, and $\{\{\},\{1,2\}\}$. They are pairwise compatible but they do not form a compatible set. Fig. 5 shows the corresponding twin graph. The middle graph shows the same uncompatibility graph, and its "normal" minimum coloring is made of 3 colors, with or without keeping the edges between twin vertices. However, the minimum coloring of the twin graph, shown on the right, is made of 2 colors only.

3.2 Algorithm

Let D be a set of dichotomies over a set of states S. The algorithm that solves the encoding problem with twin graph coloring is as follows:

(1) Compute a set of irredundant, reduced, ordered dichotomies V from D (in $O(|S| \times |D|^2)$).

(2) Build the uncompatibility relation on V to obtain the twin graph (in $O(|S| \times |D|^2)$).

(3) Color the twin graph (NP-complete).

(4) Build an encoding from the coloring (in $O(|D| \times |S|)$).

This algorithm avoids the bottlenecks of the previous methods: it does not need to build the set of maximal compatible sets M, whose potential exponential size is an obvious limiting factor; and it can be ran within a polynomial size memory (in $O(|D|^3)$), which is not the case for BDD/ZBDD based methods.

Steps (2) and (4) are straightforward. We address some details of steps (1) and (3) in the sequel.

3.2.1 Reduced Set of Ordered Dichotomies

If one generates all ordered dichotomies from D, one obtains a complete symmetric twin graph of $2|D|$ nodes. However, one can break the symmetry of this graph by generating a reduced set of ordered dichotomies.

Since complementing any bit of the encoding still preserves the dichotomy constraints, one can force one state to have an all-zero encoding. This means that from a completely symmetrized set of ordered dichotomies, one can select one state s of S, and remove all ordered dichotomies (S_0, S_1) such that $s \in S_1$. Choosing a state s that minimizes the number of resulting ordered dichotomies can be done in time linear w.r.t. the size of all dichotomies.

3.2.2 Coloring a Twin Graph

A "normal" sequential graph coloring algorithm picks a vertex v, colors it with a color non conflicting with v's neighbors, and iterates until all vertices are colored. Backtracking is forced to find better solutions, or when one cannot color a vertex without conflict [1].

One can sequentially color a twin graph by having two possible actions: one can pick a twin couple $\{v, v'\}$

and keep v as the *twin representative*, thus removing v' from the graph; or one can pick a *twin-free* vertex (i.e., a vertex whose associated twin vertex, if any, has been removed from the graph) and color it. The performance of this algorithm is very dependent on the action taken at each step. We now discuss the possible strategies and heuristics to decide which action should be carry out at each recursion.

Selecting a Vertex to Color

A good heuristic to select a vertex to be colored is the DSATUR algorithm [1]. It consists of picking the vertex that has the largest saturation number (i.e., the number of forbidden colors, which are the colors used by its neighbors), and in breaking ties with the largest degree in the uncolored graph. The idea is to choose the vertex that is the most "difficult" to color, and that propagates as many constraints as possible.

Selecting a Twin Representative

Let g be a function that estimates how difficult the coloring of a vertex v is (the greater the value of $g(v)$, the more difficult v is to color). Let *Represent* be the heuristic that select a twin representative for a twin couple $t = \{v, v'\}$. A natural choice for the twin representative is the vertex that is the easiest to color:

$$\mathit{Represent}(t) \quad = \quad \arg\min_{v \in t} g(v). \tag{1}$$

We want to pick the representative of the most difficult twin couple to color, which is:

$$\arg\max_{t \in T} g(\mathit{Represent}(t)).$$

Strategy

There are two extreme strategies:

(1) "color as late as possible": first choose all the twin representatives, then color the resulting graph.
(2) "color as soon as possible": whenever a vertex is twin-free, it is immediately colored.

Strategy (1) performs poorly for the following reasons. If one selects the "wrong" twin representative v from a twin couple $\{v, v'\}$ in the early stages of the recursions, one has to exhaust the search for an optimal coloring on a suboptimal instance graph before backtracking and considering v' as the twin representative. Moreover, there is no color constraint information to help the twin representative selection.

Strategy (2) performs reasonably well. However, since it colors all twin-free vertices before selecting any twin representative, it can underperform when the symmetry of the twin graph has been broken and twin couples are far harder to color than the initial twin-free vertices. This situation can occur since in practice a reduced sets of ordered dichotomies is not symmetric.

The general strategy, which can alternate twin representative selection and twin-free vertex coloring in any order, opens many alternative heuristics. The simplest of them merely merges the twin representative heuristic with the coloring vertex selection heuristic. It consists of computing the vertex v maximizing *select*:

$$\mathit{select}(v) = \begin{cases} g(v) \text{ if } v \text{ is twin free} \\ g(v) \text{ if } v = \mathit{Represent}(t) \text{ for some } t \in T \\ -\infty \text{ otherwise} \end{cases}$$

If the resulting vertex is twin free, we color it, else we select it as the twin representative of the twin couple it belongs to. We use this strategy, where g captures the DSATUR heuristic. In practice, this strategy closely mimics strategy (2) but overcomes the problem mentioned above.

4 Experimental Results

The benchmark consists of MCNC industrial examples representing a wide range of FSMs. Face-embedding constraints are generated with ESPRESSO-MV [10]. We compared the algorithm presented in this paper with the best known previous exact solvers, [11] and [2]. The results are summarized in Table 1.

The twin graph coloring paradigm consistently beats the two other methods, and it can solve problems that fail to terminate otherwise. Note that our method finds the first known exact solution for the long-standing *tbk* example (the minimum solution is 17, while the best known upper bound was 18). When limiting the number of backtracks to 5000, one obtains a heuristic solver, which finds the optimum solution in all but 4 cases.

5 Conclusion

This paper introduced a new approach to solve dichotomy-based constrained encoding. It showed how the later reduces to a *twin graph* coloring problem, and it explained how to solve this new problem. Experimental results show that the resulting algorithm outperforms the other best known exact solvers.

The twin graph coloring formalization opens new opportunities to tackle the constrained encoding problem, exactly and heuristically. In particular, sophisticated vertex and action selection during the twin graph coloring can lead to better and faster constrained encoding heuristics. Also, symmetry breaking and dynamic lower bound computation are two challenging problems that can help the resolution.

References

[1] D. Brélaz, "New Methods to Color Vertices of a Graph", *Comm. of the ACM*, **22**-4, pp. 251–256, 1979.
[2] O. Coudert, C.-J. Richard Shi, "Exact Dichotomy-based Constrained Encoding", *ICCD'96*, Oct. 1996.
[3] S. Devadas, A. R. Newton, "Exact Algorithms for Output Encoding, State Assignment, and Four-level Boolean Minimization", *IEEE Trans. CAD*, **1**-10, pp. 13–27, Jan. 1991.
[4] S. Devadas, A. R. Wang, A. R. Newton, A. L. Sangiovanni-Vincentelli, "Boolean Decomposition of Programmable Logic Arrays", *CICC'88*, 1988.

Example				Twin graph				CPU		
FSM	**|S|**	**|D|**	**min/k**	**|V|**	**|E|**	**|T|**	**#back**	**[2]**	**[11]**	**twin**
bbsse	16	60	4/6	75	1082	27	28681	9.50	2.38	1.66
bbtas	6	9	3/3	13	26	4	0	0.01	0.02	0.03
beecount	7	18	3/4	22	167	6	19	0.02	0.03	0.03
cse	16	59	4/5	81	1612	28	425	1.84	15.46	0.07
$dk14$	7	25	3/4	28	321	7	0	0.01	15.43	0.03
$dk15$	4	10	2/3	12	43	2	0	0.01	0.01	0.03
$dk16$	27	368	5/6	632	54730	264	–	–	∗	–
$dk17$	8	31	3/4	46	446	15	5	0.01	0.10	0.01
$dk27$	7	21	2/3	30	178	9	56	0.02	0.04	0.02
$dk512$	15	106	4/5	180	5013	74	112638	–	238.72	27.77
donfile	24	480	5/6	860	136207	380	–	–	∗	–
$ex1$	20	71	5/7	114	1909	43	11	128.63	∗	0.05
$ex2$	19	106	5/6	175	5108	70	7088	–	∗	1.23
$ex3$	10	38	4/5	57	701	20	186	0.05	0.31	0.04
$ex4$	14	91	4/4	169	1872	78	0	–	–	0.07
$ex5$	9	31	4/5	29	408	5	11	0.02	0.08	0.02
$ex6$	8	22	3/4	8	196	0	0	0.02	0.04	0.03
$ex7$	10	36	4/5	40	643	11	12	0.02	0.13	0.04
keyb	19	99	5/7	150	5062	51	441	14.43	125.2	0.23
lion	4	6	2/2	3	11	0	0	0.01	0.01	0.01
$lion9$	9	44	4/4	65	1073	21	0	0.01	0.24	0.03
$mark1$	15	77	4/5	117	1780	48	4324	1.00	23.43	0.42
mc	4	6	2/2	9	12	3	0	0.01	0.01	0.01
$modulo12$	12	66	4/4	121	1100	55	0	28679.3	21.05	0.04
opus	10	33	4/4	56	326	23	0	1.59	0.31	0.03
planet	48	767	6/6	1435	79751	668	4722	–	∗	1.66
$s1$	20	131	5/5	240	3678	109	15466	–	–	3.53
$s8$	5	7	3/3	10	15	3	0	0.01	0.01	0.01
sand	32	363	5/6	693	16928	330	–	–	∗	–
scf	121	5683	7/7	11003	1371017	5320	–	–	∗	–
shiftreg	8	28	3/3	43	375	15	0	0.01	0.09	0.03
sse	16	60	4/6	75	1082	27	28681	19.21	2.34	1.70
styr	30	193	5/6	326	9039	133	1168	–	–	0.80
tav	4	6	2/2	12	24	6	0	0.01	0.01	0.01
tbk	32	994	5/**17**	1695	666442	701	635	–	∗	22.75
$train11$	11	96	4/5	157	4383	61	7814	84.22	5.67	2.20
$train4$	4	8	2/2	10	27	2	0	0.01	0.01	0.01

$|\mathbf{S}|$ is the number of states of the FSM, and $|\mathbf{D}|$ the number of irredundant dichotomy constraints. **min** is $\lceil \log_2(|\mathbf{S}|) \rceil$, i.e., the length of the minimum encoding without constraint. **k** is the length of the minimum constrained encoding. $|\mathbf{V}|$ is the number of nodes of the twin graph (i.e., #reduced ordered dichotomies), $|\mathbf{E}|$ its number of edges (i.e., #uncompatible pairs of ordered dichotomies), and $|\mathbf{T}|$ its number of twin couples. **#back** is the number of backtracks to optimaly color the twin graph. The **CPU** time is in seconds on a 167 MHZ UltraSparc Workstation with 96MB ("–" is more than 2h, "∗" is out of memory).

Table 1: Experimental results.

[5] R. M. Fuhrer, B. Lin, S. M. Nowick, "Symbolic Hazard-free Minimization and Encoding of Asynchronous Finite State Machines", *ICCAD'95*, Nov. 1995

[6] L. Lavagno, C. W. Moon, R. K. Brayton, A. L. Sangiovanni-Vincentelli, "An Efficient Heuristic Procedure for Solving the State Assignment Problem for Event-based Specification", *IEEE Trans. CAD*, **14**-1, pp. 45–60, Jan. 1995.

[7] G. D. Micheli, R. K. Brayton, A. L. Sangiovanni-Vincentelli, "Optimal State Assignment for Finite State Machines", *IEEE Trans. CAD*, **4**-3, July 1985.

[8] G. D. Micheli, "Symbolic Design of Combinational and Sequential Logic Circuits Implemented by Two-level Logic Macros", *IEEE Trans. CAD*, **5**-1, Oct. 1986.

[9] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

[10] R. L. Rudell, A. L. Sangiovanni-Vincentelli, "Multiple-Valued Minimization for PLA Optimization", *IEEE Trans. CAD*, **6**-5, pp. 727–750, Sept. 1987.

[11] A. Saldanha, T. Villa, R. K. Brayton, A. L. Sangiovanni-Vincentelli, "Satisfaction of Input and Output Encoding Constraints", *IEEE Trans. CAD*, **13**-5, pp. 589–602, May 1994.

[12] J. H. Tracey, "Internal State Assignment for Asynchronous Sequential Machines" *IEEE Trans. Elec. Comp.*, pp. 551–560, Aug. 1966.

[13] S. H. Unger, *Asynchronous Sequential Switching Circuits*, John Wiley & Sons, Inc., 1969.

[14] T. Villa, A. L. Sangiovanni-Vincentelli, "NOVA: State Assignment of Finite State Machines for Optimal Two-level Logic Implementation", *IEEE Trans. CAD*, **9**-9, pp. 905–924, Sept. 1990.

[15] S. Yang, M. J. Ciesielski, "Optimum and Suboptimum Algorithms for Input Encoding and its Relationship to Logic Minimization", *IEEE Trans. CAD*, **10**-1, pp. 4–12, Jan. 1991.