

Exact Dichotomy-based Constrained Encoding

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

C.-J. Richard Shi
University of Iowa, ECE Dept.
Iowa City, Iowa 52242

Abstract

Constrained encoding has several applications in the synthesis of finite state machines (FSMs). E.g., it can be used to generate asynchronous FSM state assignment that guarantees a critical hazard-free implementation, or to generate synchronous FSM state assignment with minimum PLA implementation. This paper presents ZEDICHO, an original zero-suppressed binary decision diagram (ZBDD) based algorithm that solves exactly the *dichotomy*-based constrained encoding problem.

1 Introduction

Let $S = \{s_1, \dots, s_n\}$ be a set of n states. An *encoding* is a mapping¹ α from S into $\{0, 1\}^k$, where k is the length of encoding. One essential step in sequential logic synthesis consists of finding a state encoding that meets some requirements. One can be interested in some optimality criterions of the implementation, e.g., area and speed, or in some correctness criterions in the case of asynchronous FSM, e.g., race-free implementation.

A *dichotomy* is an unordered pair $\{P, Q\}$ of disjoint subsets of S . A dichotomy constraint expresses that the states of P must be distinguished from the states of Q by at least one bit, i.e., this bit must have the value 0 for all the states in P and 1 for all the states in Q , or vice versa. An encoding satisfies a dichotomy if and only if (iff) there is a bit of the encoding that meets the requirement expressed above.

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 of S that satisfies all the dichotomy constraints.*

For example, consider the four dichotomy constraints $\{\{s_1, s_3\}, \{s_2\}\}$, $\{\{s_1, s_3\}, \{s_4\}\}$, $\{\{s_3, s_4\}, \{s_1\}\}$, and $\{\{s_3, s_4\}, \{s_2\}\}$, on the set of states $S = \{s_1, \dots, s_4\}$. Fig. 1 shows a minimum-length encoding satisfying all these constraints.

¹It does not have to be injective.

	$\alpha(s)$		
s_1	1	0	0
s_2	1	0	1
s_3	0	1	0
s_4	1	1	1

Figure 1: A three-bit encoding.

Tracey introduced this problem 30 years ago [14]. He showed that if the state assignment satisfies some set of dichotomy constraints, then the resulting asynchronous implementation is critical race free. In [15, 16], it is shown that finding an asynchronous implementation that is independent from the gate and wire delays comes down to constrained encoding. De Micheli showed that state encoding targeting minimum-area PLA implementation is related to the constrained encoding problem [7, 8, 9, 18]. This applies also to optimal PLA implementation of Boolean expressions [4]. State assignment for event-based specifications is expressed as a constrained encoding problem in [6]. In [5] the problem of hazard-free minimization and asynchronous FSM encoding for multiple input changes is reduced to constrained encoding.

A unified framework [12, 18] for constrained encoding in sequential logic synthesis is illustrated in Fig. 2. Dichotomy constraints can be used to express the correctness of the implementation, or to express some optimization criterions. Dichotomy-based constrained encoding is more general than some other frameworks [17] that cannot cope with the state assignment of asynchronous FSMs.

This paper addresses the *exact* resolution of the dichotomy-based constrained encoding problem. Section 2 discusses the state-of-the-art in this field. Section 3 formalizes two approaches to solve dichotomy-based constrained encoding, and Section 4 presents their ZBDD based implementations. Section 5 discusses some experimental results.

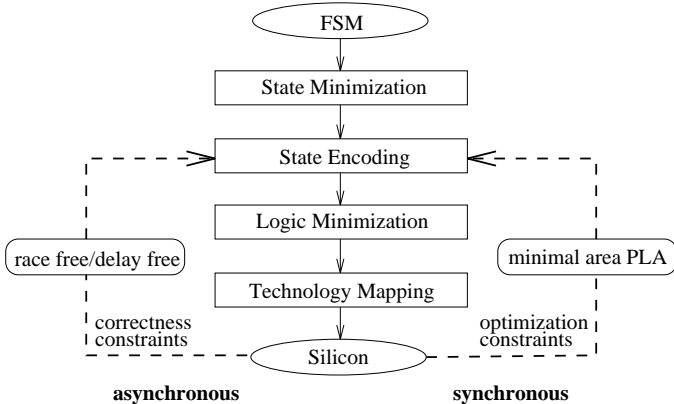


Figure 2: Sequential logic synthesis framework.

2 Previous Work

Tracey’s approach was the first exact² algorithm for constrained encoding [14]. This approach, called *prime-covering* because of its similarity with the Quine-McCluskey procedure for Boolean minimization [1], laid down the basis of all the subsequent works.

Two dichotomies are *compatible* iff they can be simultaneously satisfied by a single bit encoding. A compatible set is a set of dichotomies that can be simultaneously satisfied by a single bit encoding. The *prime-covering* procedure [14] is as follows:

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

A procedure that solves step (1) has been first suggested in [14], and is described in [18, 9]. Clearly, two dichotomies $\{P_1, Q_1\}$ and $\{P_2, Q_2\}$ are compatible iff there is no couple of states that are both in P_1 (or Q_1) and that are splitted between P_2 and Q_2 (and conversely, exchanging the indices). For example, $\{\{s_1\}, \{s_2, s_3\}\}$ is compatible with $\{\{s_1, s_4\}, \{s_2\}\}$, but not with $\{\{s_2\}, \{s_3\}\}$. The compatible sets can be represented by new dichotomies, e.g., the compatibility of $\{\{s_1\}, \{s_2, s_3\}\}$ and $\{\{s_1, s_4\}, \{s_2\}\}$ is denoted with the new dichotomy $\{\{s_1, s_4\}, \{s_2, s_3\}\}$. Solving (1) can be done by computing pair-wise compatible dichotomies (starting with D), and by iterating this process until all the compatible sets are maximal.

The difficulty of the prime-covering approach is that not only step (2) is NP-complete (it is a set covering

²Heuristic algorithms have been proposed in [14, 16, 18, 13].

problem), but also the number of maximal compatible sets can be exponential w.r.t $|D|$. But what primarily limits this approach to small size problems is that one needs to examine a huge number of dichotomy pairs in order to find all the maximal compatible sets. In practice, dichotomies only involve a small number of states. As a result, a lot of dichotomies are compatible, which makes finding the maximal compatible sets very expensive. For example, m dichotomies whose subsets P ’s and Q ’s are all mutually disjoint produce $m(m - 1)$ new dichotomies denoting the compatibility between any two dichotomies. Thus the number of dichotomies after k (k small) iterations is about m^{2^k} . For example, one obtains about 10^8 dichotomies after 3 iterations from an initial set of 100 such dichotomies.

The dichotomies of a compatible set are necessarily pair-wise compatible, but the converse is false. For example, the three dichotomies $\{\{s_1, s_2\}, \{\}\}$, $\{\{s_1, s_3\}, \{\}\}$, and $\{\{s_2\}, \{s_3\}\}$, are pair-wise compatible, but there is no single bit encoding that can satisfy the three of them at the same time. The problem is that the first two dichotomies force s_2 and s_3 to have the same encoding bit, while the third dichotomy requires these two states to have a distinguishing bit encoding.

An *ordered* dichotomy (P, Q) is an ordered couple of disjoint subsets of S . One says that two ordered dichotomies (P_1, Q_1) and (P_2, Q_2) are compatible iff $P_1 \cup P_2$ and $Q_1 \cup Q_2$ are disjoint. Two (unordered) dichotomies $\{P_1, Q_1\}$ and $\{P_2, Q_2\}$ are compatible iff (P_1, Q_1) is compatible with (P_2, Q_2) or (Q_2, P_2) . The concept of compatibility becomes then more manageable, since a set of ordered dichotomies is compatible iff they are pair-wise compatible. One associate two ordered dichotomies to each (unordered) dichotomy $\{P, Q\}$, and obtain all the maximal compatible sets by first computing the maximal sets of compatible ordered dichotomies, and then converting them to unordered dichotomies. The *compatibility graph* of a set C of ordered dichotomies is the graph obtained by linking with an edge two compatible dichotomies of C .

In [12] is presented an effective exact algorithm. It is based on the observation that if the compatibility graph is dense, then its complementary (the conflict graph) is sparse. The method consists of computing the set of all maximal compatible sets on the conflict graph instead of the compatibility graph. Let us say that two dichotomies a and b are conflicting when they are not compatible. Then all compatible sets are the “1” assignments that evaluate the following Boolean function f to 1.

$$f = \bigwedge_{a,b \text{ conflicting}} \neg(a \wedge b)$$

A maximal compatible set is the set of variables missing in some prime implicant of f . This leads to an explicit computation algorithm of all maximal compatible sets, whose complexity is linear w.r.t. the number of maximal compatible sets. However this method suffers from two problems. First, because of the duality between the conflict and compatibility graphs, the number of product terms can become unmanageable. Second, the method still needs the explicit construction of all the maximal compatible sets, which can be exponential w.r.t. $|D|$.

The approach presented in this paper has several improvements over the previous works. First, the dichotomies are represented and manipulated *implicitly* with ZBDDs. Second, the constrained encoding problem is directly expressed as a “modified” set covering problem, and the latter is reduced *implicitly* using the algorithmic developed in [3]. Section 5 shows that this approach can produce the cyclic core of all the problem instances, and that some of the problems that cannot be solved with the method of [12] are eventually solved here.

3 Formalization

This section gives two formulations of the dichotomy-based constrained encoding problem.

Formulation 1: Minimum Clique Partition

A compatible set is nothing but a clique of the compatibility graph. Thus the constrained encoding problem is a *minimum clique partition problem* that can be solved as follows. First, symmetrize the set of unordered dichotomies D to yield C , the set of ordered dichotomies. Second, build the compatibility graph of C . Third, find a minimum clique partition of this graph.

Formulation 2: Modified Set Covering Problem

Let us call a bi-partition of S a dichotomy that is a partition of S (i.e., a bi-partition is a dichotomy $\{S_1, S_2\}$ such that $S_1 \cup S_2 = S$). Let us say that a bi-partition $\{S_1, S_2\}$ *covers* a dichotomy $\{P, Q\}$ iff S_1 contains P and S_2 contains Q (or S_1 contains Q and S_2 contains P).

Each bit of an encoding defines a bi-partition, made of the states on which the bit is 0, and the states where the bit is 1. An encoding satisfies a dichotomy constraint iff the later is covered by the bi-partition associated with one of its bit. Thus the constrained encoding problem consist of covering the set of (unordered) dichotomies D with a minimum cardinality set of bi-partitions.

These two formulations of the constrained encoding

problems are indeed very different. The first one is a common NP-complete graph problem, and may need to enumerate all the maximal cliques (i.e., maximal compatible sets). On the other hand, the second formulation expresses directly the encoding problem as a set covering problem, where the notion of covering is not belongingness but a sort of set containment. However, the number of bi-partitions is $2^{|S|-1}$. Next section shows that indeed the second formulation yields an original and effective exact constrained encoding problem solver.

4 Two Exact Minimum Constrained Encoding Algorithms

This section shows how two ZBDD-based exact minimum constrained encoding algorithms can be derived from the two formulations given in previous section.

Given a finite set V , ZBDDs is a graph data structure used to represent subsets of 2^V in a compact and canonical way. Set operations can be efficiently implemented with ZBDDs, in such a way their costs (which are quadratic in the worst case) are not related to the sizes of the sets they operate on, but to the sizes of the ZBDDs representing these sets. In practice, very large sets can be represented with small ZBDDs. The reader is referred to [10] for an introduction to ZBDDs.

4.1 Minimum Clique Partition

Let C be the set of ordered dichotomies derived from the set of dichotomies D ($|C| = 2|D|$). The set U of *un-compatible* pairs of ordered dichotomies can be computed in $O(|S| \times |D|^2)$. A clique of the compatibility graph is a set of dichotomies that does not contain any element of U . Thus we can express the set of all maximal cliques (i.e., all maximal compatible sets) as:

$$\begin{aligned} \text{MaxClique} &= \max_{\subseteq} \text{NotSupSet}(2^C, U), \quad \text{where} \\ \max_{\subseteq} X &= \{x \in X \mid \forall x' \in X, x \subseteq x' \Rightarrow x = x'\}, \quad \text{and} \\ \text{NotSupSet}(X, Y) &= \{x \in X \mid \forall y \in Y, y \not\subseteq x\}. \end{aligned}$$

Note that the size of the ZBDD of 2^C is $2|D|$, and the one of U is bound by $2|U|$, thus is $O(|D|^2)$. Both operations *NotSupSet* and \max_{\subseteq} can be implemented with ZBDDs using a divide-and-conquer strategy [3]. We actually use an algorithm that combines both of these operations in one pass.

Definition 2 *Let X and Y be two sets, and R a binary relation on $X \times Y$. An element y of Y covers an element x of X iff $x R y$. The set covering problem $\langle X, Y, R \rangle$ consists of finding a minimum subset E of Y such that any x of X is covered by some y of E .*

Once all the maximal cliques have been computed, we can translate them back to unordered dichotomies, and we need to solve the set covering problem

$$\langle D, \text{MaxClique}, \in \rangle. \quad (1)$$

The Quine–McCluskey procedure [1] cannot be applied, since $|\text{MaxClique}|$ can be exponential w.r.t. $|D|$. To overcome this limitation, we use the results developed for SCHERZO [3, 2]. The method consists of transforming the original set covering problem (1) into an isomorphic set covering problem (2) over the complete lattice $(2^D, \subseteq)$:

$$\langle X, \text{MaxClique}, \subseteq \rangle, \quad \text{where} \quad (2)$$

$$X = \{\{d\} \mid d \in D\}.$$

It is shown in [3, Theorem 8] that one can apply ZBDD based reduction rules on (2) to yield a set covering problem, say (3), that is isomorphic to the cyclic core of (1). From the solutions of (3) one can recover the solutions of the original problem (1) using basic set operations [3]. The interest of such an approach is that the reduction to the isomorphic cyclic core can be done with ZBDDs in an *implicit* way, i.e., with a complexity that does not depend on the number of elements of D and MaxClique .

Sketch of Algorithm 1

- From the set D of unordered dichotomies, build the set C of ordered dichotomies (in $O(|S| \times |D|)$), and the set U of incompatible pairs of ordered dichotomies (in $O(|S| \times |D|^2)$).
- Create $|C|$ ZBDD variables, one for each ordered dichotomy, and note each pair of variables that denote the same unordered dichotomy with two different ordering (in $O(|D|)$).
- Build the ZBDD of U (in $O(|D|^2)$).
- Build the ZBDD $Z1$ of all maximal cliques from the ZBDD of U (non-polynomial).
- Build the ZBDD $Z2$ of $\{\{d\} \mid d \in C\}$ (in $O(|D|)$).
- For each pair of associated variables, substitute one with the other in $Z1$ and $Z2$ (in $O(|Z1| + |Z2|)$). It yields the set of unordered dichotomies and the set of maximal compatible sets.
- Compute implicitly the cyclic core of the set covering problem (2) expressed with these two ZBDDs (non-polynomial).
- Solve explicitly the cyclic core (NP-complete).

The interest of such an approach is that the first drawback of the prime-covering method is avoided, since the size of the ZBDD of all the maximal compatible sets is not related to their number. However, since the number of ZBDD variables is $2|D|$, this approach cannot be applied if the number of dichotomies is too large.

4.2 Modified Set Covering Problem

The second algorithm expresses directly the constrained encoding problem as a set covering problem, where one tries to cover the (unordered) dichotomies with bi-partitions. Here, “cover” is not belongingness, as it was the case with the first algorithm. Let us express that a dichotomy $\{P_1, Q_1\}$ covers a dichotomy $\{P_2, Q_2\}$ by writing $\{P_2, Q_2\} \sqsubseteq \{P_1, Q_1\}$. We have:

$$\{P_2, Q_2\} \sqsubseteq \{P_1, Q_1\} \Leftrightarrow ((P_2 \subseteq P_1) \wedge (Q_2 \subseteq Q_1)) \vee ((P_2 \subseteq Q_1) \wedge (Q_2 \subseteq P_1))$$

The relation \sqsubseteq is a partial order on the set of dichotomies $\mathcal{D} = \{\{P, Q\} \mid P \cap Q = \emptyset\}$. Thus we have the lattice structure $(\mathcal{D}, \sqsubseteq)$ required to apply [3, Theorem 8] on the set covering problem $\langle D, \mathcal{B}, \sqsubseteq \rangle$, where \mathcal{B} is the set of bi-partitions. However, the reduction rules yielded on this lattice are not simple, unlike the standard set covering problem. The problem is due to the logical symmetry in the expression of \sqsubseteq , whose duality prevents any simplification.

To overcome this problem, the logical symmetry in the expression of the partial order \sqsubseteq must be broken, but reported on the ground set, i.e., the set on which the covering relation operates. Let $S = \{s_1, \dots, s_n\}$ be the set of states. We associate a dual element s'_k with each state s_k , and note S' the set $\{s'_1, \dots, s'_n\}$ of dual elements. We note P' the dual of a subset P of S . To denote a dichotomy $\{P, Q\}$, we use the mapping μ defined as:

$$\mu(\{P, Q\}) = \{P \cup Q', Q \cup P'\}.$$

A set of dichotomies D is then denoted by:

$$\mu(D) = \bigcup_{d \in D} \mu(d).$$

For example, the set of dichotomies

$$\{\{\{s_1\}, \{s_2, s_3\}\}, \{\{s_3\}, \{s_4\}\}, \{\{s_2\}, \{\}\}\}$$

is represented by:

$$\{\{s_1, s'_2, s'_3\}, \{s'_1, s_2, s_3\}, \{s_3, s'_4\}, \{s'_3, s_4\}, \{s_2\}, \{s'_2\}\}.$$

The *logical* duality of \sqsubseteq is now hidden in the *structural* duality of the much simpler lattice $(2^{S \cup S'}, \subseteq)$. Each element x of $\mu(D)$ has its dual x' that also belongs to $\mu(D)$, and both of them are linked together in the sense that a bi-partition b covers x iff b' covers x' . In other words, solving the covering problem

$$\langle \mu(D), \mu(\mathcal{B}), \subseteq \rangle \quad (3)$$

produces the solution $\mu(E)$, where E (set of bi-partitions) is the solution of the original minimum constrained encoding problem.

Sketch of Algorithm 2

- Create $2|S|$ ZBDD variables, one and its dual for each state (in $O(|S|)$).
- Build the ZBDD of $\mu(D)$ from the set D of (unordered) dichotomies (in $O(|D| \times |S|)$).
- Build the ZBDD of $\mu(\mathcal{B})$ (in $O(|S|)$ with the variable ordering $s_1 < s'_1 < s_2 < s'_2 < \dots$).
- Compute implicitly the cyclic core of the set covering problem (3) expressed with these two ZBDDs (non-polynomial).
- Unsymmetrize the cyclic core by translating back to unordered dichotomies, i.e., apply μ^{-1} (linear).
- Solve explicitly the cyclic core (NP-complete).

The interest of such an approach is twofolds. First, the ZBDDs are build on $2|S|$ variables, while they need $2|D|$ variables with the first algorithm. Since the number of dichotomies is far greater than the number of states for most of the practical examples, the size of the ZBDDs are much smaller. Second, we do not need additional processes (such as computing the set of incompatible pairs, or computing the maximal cliques), since we expressed constrained encoding with one single set covering where the (implicit) set of rows is directly built from the set of dichotomies D , and the (implicit) set of columns is the linear-size ZBDD of $\mu(\mathcal{B})$.

5 Experimental Results & Conclusion

The first set of examples comes from the literature on the synthesis of asynchronous FSMs. The aim is either a race-free [14], a delay-free [15], or a hazard-free [5] implementation. These small problems are all solved exactly in less than a second.

The second set of tests consists of 37 MCNC industrial examples representing a wide range of FSMs. In our experiments, we used ESPRESSO-MV [1, 11] to generate all the face-embedding constraints, and used a pre-processor to generate all the dichotomy constraints. We compared the two algorithms presented in this paper with the best known previous exact solver [12]. The results are summarized in Table 1. The CPU time does not include the time used by ESPRESSO-MV and the pre-processor, since it is neglectable. Algorithm 1 is the minimum clique partition based method presented in Section 4.1, and ZEDICHO is the bi-partition based method presented in Section 4.2.

Clearly, algorithm 1 is very limited, since it involves a number of variables that is at least the number of unordered dichotomies. Thus when this number is large, algorithm 1 runs out of memory, or takes too long time. Note that algorithm 1 comes down to solve a set cov-

ering problem with $|D|$ rows and $|MC|$ columns, while ZEDICHO ends up with a set covering problem with **row** rows and **col** columns.

ZEDICHO is more interesting. First of all, it succeeds in (quickly) computing the cyclic core of all the examples. Second, it is comparable to [12], and it succeeds in solving *ex1* while [12] failed. An interesting point is that the size of the cyclic core can be amazingly huge. Another point is that even when the cyclic core is small, it can take a long time to solve it because the number of equally minimum solutions is so large that SCHERZO [2] has to explore the whole search space before terminating (e.g., *dk512*, *ex2*, *ex4*, *modulo12*).

References

- [1] R.K. Brayton, G.D. Hachtel, C.T. McMullen, A.L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer, Dordrecht, 1984.
- [2] O. Coudert, “On Solving Covering Problems”, Proc. *33rd DAC*, pp. 197–202, June 1996.
- [3] O. Coudert, “2-Level Logic Minimization: An Overview”, *Integration*, **17-2**, pp. 97–140, Oct. 1994.
- [4] 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.
- [5] R. M. Fuhrer, B. Lin, S. M. Nowick, “Symbolic Hazard-free Minimization and Encoding of Asynchronous Finite State Machines”, Proc. *ICCAD’95*, pp. 604–611, 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**, pp. 269–285, 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**, pp. 597–616, Oct. 1986.
- [9] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.
- [10] S. Minato, “Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems”, Proc. *30th DAC*, pp. 272–277, June 1993.
- [11] R. L. Rudell, A. L. Sangiovanni-Vincentelli, “Multiple-Valued Minimization for PLA Optimization”, *IEEE Trans. CAD*, **6-5**, pp. 727–750, Sept. 1987.

Example FSM	Example			Algo. 1 (Section 4.1)			ZeDicho (Section 4.2)			[12]
	S	D	k	U	MC	CPU	row	col	CPU	CPU
<i>bbsse</i>	16	260	6	1080	902	163.5	44	771	9.50	2.38
<i>bbtas</i>	6	38	3	54	16	0.0	9	25	0.00	0.02
<i>beecount</i>	7	56	4	236	23	0.1	16	22	0.00	0.03
<i>cse</i>	16	352	5	1644	2968	39.19	53	2145	1.84	15.46
<i>dk14</i>	7	61	4	536	30	0.30	22	23	0.00	15.43
<i>dk15</i>	4	22	4	34	6	0.00	4	4	0.00	0.01
<i>dk16</i>	27	886	8?	66423	*	*	374	3.80e + 07	—	*
<i>dk17</i>	8	88	4	461	65	0.65	31	64	0.00	0.10
<i>dk27</i>	7	49	3	273	48	0.25	21	45	0.00	0.04
<i>dk512</i>	15	252	5	5087	9310	—	106	9310	—	238.72
<i>donfile</i>	24	480	6?	136587	—	—	480	3.25e + 06	—	*
<i>ex1</i>	20	484	7	1952	30545	433.85	71	38607	128.63	*
<i>ex2</i>	19	438	6	5178	61025	—	105	60296	—	*
<i>ex3</i>	10	122	5	721	170	2.10	37	336	0.05	0.31
<i>ex4</i>	14	182	4	1950	—	—	91	8191	—	—
<i>ex5</i>	9	96	5	415	38	0.25	24	30	0.00	0.08
<i>ex6</i>	8	89	4	201	23	0.10	0	0	0.00	0.04
<i>ex7</i>	10	121	5	655	47	0.50	28	41	0.00	0.13
<i>keyb</i>	19	606	7	5867	—	—	99	8294	14.43	125.2
<i>lion</i>	4	15	2	12	4	0.05	0	0	0.00	0.01
<i>lion9</i>	9	63	4	1107	88	1.55	44	82	0.00	0.24
<i>mark1</i>	15	146	4	1737	3830	2171.7	66	2298	1.00	23.43
<i>mc</i>	4	12	2	15	7	0.0	6	7	0.00	0.01
<i>modulo12</i>	12	132	4	1155	—	—	66	2047	28679.3	21.05
<i>opus</i>	10	88	4	316	253	8.25	29	253	1.59	0.31
<i>planet</i>	48	2612	6	80419	*	*	767	9.95e + 13	—	*
<i>s1</i>	20	526	5	4263	—	—	131	429945	—	—
<i>s8</i>	5	27	3	26	10	0.05	7	13	0.00	0.01
<i>sand</i>	32	1170	6	17258	*	*	363	8.72e + 08	—	*
<i>scf</i>	121	12691	8	*	*	*	5656	1.07e + 36	—	*
<i>shiftrig</i>	8	28	3	390	70	0.55	28	70	0.00	0.09
<i>sse</i>	16	260	6	1080	902	168.90	44	771	19.21	2.34
<i>styr</i>	30	1376	6	9172	*	*	193	2.33e + 08	—	—
<i>tav</i>	4	15	2	15	7	0.00	6	7	0.00	0.01
<i>tbk</i>	32	2907	23?	817604	*	*	994	3.09e + 08	—	*
<i>train11</i>	11	96	5	4444	836	1744.5	85	836	84.22	5.67
<i>train4</i>	4	8	2	29	6	0.0	8	6	0.00	0.01

|S| : #states of the FSM.
|D| : #dichotomy constraints.
k : minimum constrained encoding length (“?” is only an upper bound).
|U| : #uncompatible pairs of ordered dichotomies.
|MC| : #maximal compatible sets.
row and col: #rows and #columns of the cyclic core yielded by the bi-partition based formulation.
CPU : CPU time in seconds on a 75Mhz SuperSparc Workstation with 96MB
(“-” is more than 2h, “*” is out of memory).

Table 1: Experimental results on MCNC FSMs.

- [12] 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.
- [13] C.-J. Shi, J. A. Brzozowski, “An Efficient Algorithm for Constrained Encoding and its Applications”, *IEEE Trans. CAD*, **12**-12, pp. 1813–1826, Dec. 1993.
- [14] J. H. Tracey, “Internal State Assignment for Asynchronous Sequential Machines”, *IEEE Trans. Elec. Comput.*, pp. 551–560, Aug. 1966.
- [15] S. H. Unger, “A Row Assignment for Delay-free Realizations of Flow Tables Without Essential Hazards”, *IEEE Trans. Elec. Comput.*, **17**-2, pp. 145–158, Feb. 1968.
- [16] S. H. Unger, *Asynchronous Sequential Switching Circuits*, John Wiley & Sons, Inc., 1969.
- [17] 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.
- [18] 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.