Technology Mapping by Binate Covering

Michal Z. SERVÍT [1] and Kang YI [2]

[1] Dept. of Computer Science and Engineering, Czech Technical University, Karlovo nám. 13, Prague, CZ 121 35 Czech Republic, e-mail: servit@cs.felk.cvut.cz

[2] Dept. of Computer Engineering, Seoul National University, Shillim-dong Kwanak-Gu, Seoul, 151-742 Korea, e-mail: yk@riact.snu.ac.kr

Abstract. Technology mapping can be viewed as the optimization problem of finding a minimum cost cover of the given Boolean network by choosing from given library of logic cells. The core of this problem in turn can be formulated as the *binate covering problem* that is NP-hard. A number of heuristics solving the binate covering problem has been proposed. However, no experimental comparison of efficiency of such techniques with respect to this specific domain has been published according to our knowledge. The aim of this paper is to analyze specific features of the binate covering formulation of the technology mapping, to propose and test a collection of heuristics using MCNC benchmarks and to select the most efficient heuristic algorithm.

1 Introduction

Logic synthesis takes the circuit specification at the functionality level and generates an implementation in terms of interconnection of logic cells from a given library. Since synthesis is a difficult task, it is often separated into two phases [3], [4]: *technology–independent optimization phase* (logic optimization), followed by a *technology mapping phase*. The optimization phase attempts to generate an optimum abstract representation of the circuit. The technology mapping phase inputs an interconnection of abstract operators - *the subject Boolean network* - and generates an interconnection of logic cells selected from a given library. Further we will assume that any abstract operator can be implemented by a single logic cell (i.e. we assume that the given network is *feasible*), and that no modifications of the given subject network are allowed while it may improve mapping results [1].

Under the above assumption we can easily decompose the technology mapping into two subsequent phases without loss of generality:

i. Construction of all (feasible) clusters: a cluster (also called match [4] or supernode [3]) is a subnetwork of the given subject network implementable by a single logic cell from the given library.
ii. The selection of clusters for a functionally correct implementation of the subject network while optimizing area, delay, or power.

The enumeration of all clusters is relatively simple problem; the hard part is selecting the optimum subset that satisfies all constraints. This is why we will focus our attention to the second phase. It is well known [3], [4] that it can be formulated as the *binate covering problem* where all constraints related to functional correctness are

expressed as a product-of-sums (POS) Boolean formula while the optimization criteria are expressed as a cost function.

Because the binate covering problem is NP-hard (for formal proof see e.g. [8]) we resort to heuristic methods. A number of such methods has been proposed recently [2], [6], [13] (for review see [5]) with the aim of general applicability. On the other hand, domain specific procedures are used in many technology mappers [1], [4], [10] (for review see [3]) to solve essentially the same task. Upon to our knowledge, no exhaustive comparison of methods of both types using a technology mapping experimental data has been published so far while such an experiment has been reported in [13] for state minimization of incomplete FSMs. This is why we decided to propose a collection of heuristics and compare their efficiency using MCNC benchmarks.

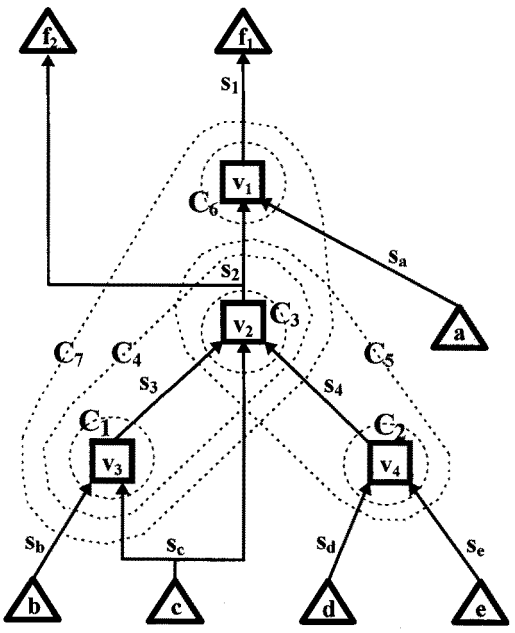

Fig. 1 Feasible subject network

The rest of the paper is organized as follows: Section 2 gives the problem formulation, Section 3 describes the proposed collection of heuristics. Section 4 shows some experimental results. In Section 5 we finally draw conclusions.

2 Binate Covering Problem Formulation

Let a feasible subject network and a set of corresponding clusters be given. The subject network consists of primary input nodes, primary output nodes and internal nodes representing abstract operators (abstract gates/blocks) interconnected via (generally multiterminal) nets (Fig. 1). *Clusters* are subnetworks of the given network that can be mapped into a single-output library cell. We will say that a node or cluster *produces* a signal: for example node v_4 produces the signal s_4 and cluster C_3 produces the signal s_2. By $C(s)$ we will denote the set of all clusters producing the signal s : for example $C(s_2) = \{C_3, C_4, C_5\}$. Similarly we will say that a node or cluster *consumes* a signal: for example node v_2 consumes signals s_3, s_4, s_c and cluster C_5 consumes signals s_3, s_c, s_d, s_e.

Our task is to select an optimum subset S from the set of all clusters C that satisfies the following constraints [3, p. 114]:

- all signals consumed by primary output nodes have to be produced by clusters from S (*output constraints*), and

- if a cluster C_i is in S, each signal consumed by C_i should be produced either by a primary input node or by some cluster(s) in S *(implication constraints).*

Output constraints ensure that each primary output is realized by a selected cluster. Implication constraints ensure that the network obtained in terms of the chosen clusters is a physically realizable circuit, i.e. no cluster selected has a dangling input.

It is well known that the above constraints can be expressed in terms of a product-of-sums (POS) Boolean formula. For each cluster C_i, a Boolean variable x_i is introduced: $x_i = TRUE \Leftrightarrow C_i \in S$, $x_i = FALSE \Leftrightarrow C_i \notin S$. The constraints produce clauses (sum terms) as follows:

- Given a primary output node v consuming a signal s, let $C(s) = \{C_{s1}, C_{s2} \ldots C_{sk}\}$ be the set of clusters generating the signal s. Than at least one of them has to be selected, i.e. the *output constraint* for v can be written as follows:

$$(x_{s1} + x_{s2} + \ldots + x_{sk})$$

- Given a signal s consumed by a selected cluster C_0, let $C(s) = \{C_{s1}, C_{s2} \ldots C_{sk}\}$ be the set of clusters producing the signal s. Than at least one of then has to be selected, i.e. the implication constraint with respect to s and C_0 can be written as follows:

$$x_0 \Rightarrow (x_{s1} + x_{s2} + \ldots + x_{sk}) \Leftrightarrow (\bar{x}_0 + x_{s1} + x_{s2} + \ldots + x_{sk})$$

Take the product of all the sum clauses generated above to form a product-of-sums Boolean expression T. Any satisfying assignment to T is a solution to the binate covering problem, and, consequentially, to the original technology mapping problem.

For our example (Fig. 1), we get $T = T_1 \cdot T_2 \cdot \ldots \cdot T_8$ where:

$$\begin{aligned}
T_1 &= (x_6 + x_7) && \cdots \text{ output constraint for } f_1 \\
T_2 &= (x_3 + x_4 + x_5) && \cdots \text{ output constraint for } f_2 \\
T_3 &= (\bar{x}_3 + x_1) && \cdots \text{ implication constraint for } C_3, s_3 \\
T_4 &= (\bar{x}_3 + x_2) && \cdots \text{ implication constraint for } C_3, s_4 \\
T_5 &= (\bar{x}_4 + x_2) && \cdots \text{ implication constraint for } C_4, s_4 \\
T_6 &= (\bar{x}_5 + x_1) && \cdots \text{ implication constraint for } C_5, s_3 \\
T_7 &= (\bar{x}_6 + x_3 + x_4 + x_5) && \cdots \text{ implication constraint for } C_6, s_2 \\
T_8 &= (\bar{x}_7 + x_2) && \cdots \text{ implication constraint for } C_7, s_4
\end{aligned} \qquad (1)$$

The set $S = \{C_6, C_5, C_1\}$ is a solution to our technology mapping problem because the assignment $x_6, x_5, x_1 = TRUE$ and $x_7, x_4, x_3, x_2 = FALSE$ satisfies T.

Let a positive cost be associated with each cluster expressing area needed for the implementation of the cluster. Finding a satisfying assignment with the least sum of costs of all selected clusters - the least *total cost* - is the same as finding an optimum solution to the original technology mapping problem when the total area of cells is to be minimized. To simplify the argumentation we will assume further that all library cells have the same cost, i.e. just the number of selected clusters is to be minimized. Let us note that generalizations of this cost measure are quite straightforward [2], [5].

For our example, the solution set $S = \{C_6, C_5, C_1\}$ has the total cost equal to 3 while the solution set $S_1 = \{C_6, C_5, C_1, C_4\}$ has the total cost equal to 4. Notice that S_1 is a *redundant* solution because there exists a proper subset of S_1 that is a solution. On the other hand, S is an *irredundant* solution because no cluster can be deleted from S. It is convenient to distinguish between two types of redundancy: either S contains one or more clusters producing a signal that is not consumed by any other cluster in S (*ALPHA redundancy*) or S contains two or more clusters producing

the same signal (*BETA redundancy*). Notice that C_4 is BETA redundant with respect to S_1.

A set of all output and implication constraints will be called the *basic set of constraints* because it expresses *all* conditions that any solution must obey. However, as Murgai et.al. [3, p.115] pointed out, a basic set can be *enhanced* by adding covering constraints in order to help heuristic procedures in obtaining somewhat better approximate solutions. Of course, their presence does not affect the optimum solution.

Covering constraints [1], [3] are constructed as follows. If an internal node v is covered by clusters $C_{v1}, C_{v2} \ldots C_{vk}$, write a clause $(x_{v1} + x_{v2} + \ldots + x_{vk})$ indicating that at least one of the clusters $C_{v1}, C_{v2} \ldots C_{vk}$ must be selected. Repeat it for each internal node of the network. Four covering constraints are added in our example: $(x_6 + x_7), (x_3 + x_4 + x_5 + x_7), (x_1 + x_4 + x_7), (x_2 + x_5)$.

Binate covering problems that originate in technology mapping have some domain specific features that can easily be identified (for formal proofs see [12]):

- Let PL_j denote a set of *positive* literals from T_j where T_j is from the *basic* set of constraints, i.e. PL_j represents a set of clusters producing the same signal. It holds: if $PL_j \cap PL_k \neq \Phi$ then $PL_j = PL_k$. See for example $PL_2 = PL_7 = \{x_3, x_4, x_5\}$. Notice that clusters C_3, C_4 and C_5 produce s_2.
- Let $x_i = TRUE$, $x_i \in PL_j$ and the formula T is satisfied, then all variables from $PL_j \setminus \{x_i\}$ can be set *FALSE* and T remains satisfied. Notice that it is sufficient to select just one of the clusters producing the same signal. The application of this rule allows to avoid generation of BETA redundant solutions (see Section 3.3).
- An irredundant solution can always be found by traversing a subject network from primary outputs to primary inputs provided that the network is acyclic. The application of this rule allows to avoid generation of redundant solutions (see Section 3.1).

To conclude this section: we will investigate both basic and enhanced set of constraints while minimizing the number of clusters selected.

3 Heuristic Algorithm Proposal

According to our experience [3], [5], [6], [7], a greedy algorithm combined with backtracking is appropriate to solve the binate covering problem. In this section, we will propose the core of such a procedure with several options that seems to be reasonable to investigate experimentally bearing in mind domain-specific knowledge.

The basic operation of our algorithm is reduction. The *reduction* consists of two steps: the value *TRUE* or *FALSE* is assigned to a variable x_i, and the formula T is simplified accordingly. For our example, we get by assignment $x_7 = FALSE$:

$$T = (x_6) \cdot T_2 \cdot T_3 \cdot T_4 \cdot T_5 \cdot T_6 \cdot T_7 \qquad (2)$$

Notice that the above formula (2) can be satisfied only if $x_6 = TRUE$ because it contains the clause (x_6). This is why we denote the reduction that is based on existence of a clause in T that contains only one literal as *exact*. A reduction that is not an exact one is denoted as a *heuristic* reduction. For example:

$x_6 = TRUE$, $T = T_2 \cdot T_3 \cdot T_4 \cdot T_5 \cdot T_6 \cdot (x_3 + x_4 + x_5)$ is an exact reduction with respect to (2) and $x_7 = FALSE, T = (x_6) \cdot T_2 \cdot T_3 \cdot T_4 \cdot T_5 \cdot T_6 \cdot T_7$ is a heuristic reduction with respect to (1).

Using the notion of exact and heuristic reduction, we can formulate the core of the proposed algorithm as follows:

Step1: Perform all possible exact reductions.

Step2: If a solution was found – remove redundant clusters and STOP. If the reduced formula cannot be satisfied – backtrack to the last heuristic decision and assign the opposite value to the variable involved.

Step3: Perform a heuristic reduction and repeat Step 1.

The crucial part of this algorithm is Step 3: select a variable x_i from T (select a cluster C_i) and assign it a value (accept or reject cluster C_i). There are several possibilities: how to define a *candidate set* for variable selection, how to *select a variable* from the candidate set, and how to *assign a value* to the variable selected.

3.1 Candidate set

Two options are to be investigated: either to use the whole set of variables from T as the candidate set, or to consider only such variables that appear in positively unate clauses (i.e. clauses that do not contain an inverted literal) in the basic set of constraints (i.e. covering clauses are not taken into account!). We will call these variants as *complete* and *restricted* candidate set respectively. In our example (1), the complete candidate set is $\{x_1, x_2, \ldots, x_7\}$, and the restricted candidate set is $\{x_3, x_4, \ldots, x_7\}$.

Notice that the restriction of the candidate set has the same effect as selecting candidate clusters by a traversal from primary outputs to primary inputs. The traversal has been used in [1] successfully. This motivated us to explore this option in our experiments.

When the restricted candidate set is used, the solution found is always irredundant provided that the subject network is acyclic [12]. Moreover, it ensures that a solution will be found without backtracking [12].

3.2 Variable selection

For variable selection, we propose to use slight modifications of score functions as defined in [2]. The score functions used are based on the following simple observation: the probability that a cluster C_i will be included in at least one minimal solution is directly proportional to the number of terms containing literal x_i, while the contribution from clause T_j containing literal x_i is inversely proportional to the number of literals contained in it. We define *weight* w_j of T_j as inversely proportional to the number of literals in T_j. It holds for our example (1): $w_1 = 1/2$, $w_2 = 1/3$, $w_3 = 1/2$ ... etc.

Direct score DS_i approximates the probability that a cluster C_i is a part of at least one minimal solution:

$$DS_i = \sum_{j=1}^{m} dw_j$$

where $aw_j = w_j$ if T_j contains literal x_i, otherwise $dw_j = 0$, and m is the number of variables in T. It holds for our example: $DS_1 = w_3 + w_6 = 1/2 + 1/2$, $DS_2 = w_4 + w_5 + w_8 = 1/2 + 1/2 + 1/2$, $DS_3 = w_2 + w_7 = 1/3 + 1/4$, ... etc.

Similar statement as above can be drawn for the probability that a cluster C_i will not be included in at least one minimal solution. *Indirect score* IS_i approximates this probability:

$$IS_i = \sum_{j=1}^{m} iw_j$$

where $iw_j = w_j$ if T_j contains literal $\bar{x}_i$, otherwise $iw_j = 0$. It holds for our example: $IS_1 = 0$, $IS_2 = 0$, $IS_3 = w_3 + w_4 = 1/2 + 1/2 = 1$, $IS_4 = w_5 = 1/2$, ... etc.

Obviously, a variable x_i with the highest value of DS_i or IS_i is a candidate for the value assignment. The difference $DIF_i = DS_i - IS_i$ can be used for the same purpose as well.

3.3 Value assignment

Three strategies of value assignment are to be investigated: Accept [2], Reject [2] and Accept-or-Reject.

Accept strategy assigns $x_i = TRUE$ to the variable x_i having the highest value of DS_i. Ties are resolved with respect to IS_i. For our example (1), the first assignment made is: $x_2 = TRUE$.

Reject strategy assigns $x_i = FALSE$ to the variable x_i having the highest value of IS_i. Ties are resolved with respect to DS_i. For our example (1), the first assignment made is: $x_3 = FALSE$.

Accept-or-Reject strategy assigns a value to the variable having the highest value of $|DIF_i|$: $x_i = TRUE$ if $DIF_i \geq 0$, otherwise $x_i = FALSE$. For our example (1), the first assignment made is: $x_2 = TRUE$.

Notice that the reject strategy provides irredundant solutions only, while the remaining ones may produce a redundant solution [2]. However, the notion of (heuristic) reduction can be *enhanced* by using the domain-specific features of technology mapping. If $x_i \in PL_j$ is set $TRUE$ (i.e. C_i is accepted), then all the remaining variables from PL_j are set $FALSE$ and the constraint formula is simplified accordingly. For our example (1), we get by assignment $x_3 = TRUE$:

$$x_4 = FALSE \text{ and } x_5 = FALSE \Rightarrow T = T_1 \cdot T_3 \cdot T_4 \cdot (x_2) \cdot (x_1) \cdot T_7 \cdot T_8$$

This allows us to avoid generation of BETA redundant solutions and to speed-up the whole process.

3.4 Orthogonal set of heuristics

It is possible to derive an orthogonal set of heuristics by combining three types of options: *basic* or *enhanced* set of constraints, *complete* or *restricted* candidate set, and

accept or *reject* or *accept-or-reject* strategy. For obvious reasons, the *enhanced definition of reduction* is applied in all cases.

4 Experiments

MCNC benchmarks listed in Table 1 were used for experimentation. A feasible acyclic subject network was generated for each benchmark using the same technique as in [1]: a modified *misII* provided an optimized network that was decomposed by *ite* in order to obtain a feasible network. The set of clusters was derived for ACTEL ACT1 [9] library cell using the front-end of the mapper described in [1]. All computations were done on HP9000/735 (99 MHz, 41 MFLOPS, 80 Mbyte memory).

4.1 Problem size

Table 1 lists the number of nodes and clusters (*#clu*) for each benchmark as well as

					basic set of constr.			*enh. set of constr.*		
	# of nodes					*after exact red.*			*after exact red.*	
name	*input*	*output*	*internal*	*#clu*	*#cons*	*#clu*	*#cons*	*#cons*	*#clu*	*#cons*
5xp1	7	10	40	60	119	69	43	159	43	89
alu2	10	6	171	208	429	118	117	600	117	198
alu4	14	8	88	92	156	5	8	244	8	9
apex2	39	3	104	117	241	28	25	345	25	40
apex6	135	99	310	450	646	372	317	956	317	512
apex7	49	37	95	113	191	42	40	286	40	60
b9	41	21	59	80	117	61	47	176	47	83
bw	5	28	65	141	380	309	128	445	128	359
c1908	33	25	91	263	526	379	186	700	186	472
c499	41	32	166	180	320	24	24	484	24	180
c5315	178	123	539	624	1247	239	177	1786	177	624
c880	60	26	173	235	441	329	176	614	176	443
clip	9	5	48	62	103	55	43	151	43	80
count	35	16	39	39	52	0	0	91	0	0
des	256	245	1308	1908	5275	2386	1136	6583	1136	2919
duke2	22	29	164	254	535	268	184	669	184	358
e64	65	65	95	125	184	31	61	279	61	125
f51m	8	8	40	43	74	5	7	114	7	9
misex1	8	7	18	31	65	31	25	83	25	43
misex2	25	18	38	47	70	13	17	108	17	21
rd73	7	3	28	37	64	14	14	92	14	19
rd84	8	4	48	53	118	29	25	166	25	49
rot	135	107	248	289	516	97	87	764	87	289
sao2	10	4	47	50	68	21	19	115	19	37
vg2	25	8	33	34	47	8	8	80	8	15
z4ml	7	4	12	14	24	6	5	36	5	14

Table 1 Sizes and complexity of benchmarks

numbers of constraints (*#cons*) for both basic and enhanced set of constraints. Additionally, problem sizes after exact reduction (after the first application of the Step 1) are listed as well.

4.2 Efficiency of heuristics

set of constr. / *cand. set* / *strategy*	*enhanced restricted accept*		*enhanced restricted acc-or-rej.*		*enhanced restricted reject*		*enhanced complete accept*		*enhanced complete acc-or-rej.*		*enhanced complete reject*	
	cost	*time*	*cost*	*time*	*cost*	*time*	*cost*	*time*	*cost*	*time*	*cost*	*time*
SUM	3918	2.29	3921	2.12	3953	4.29	3929	14.46	3927	2.60	3933	4.14

set of constr. / *cand. set* / *strategy*	*basic restricted accept*		*basic restricted acc-or-rej.*		*basic restricted reject*		*basic complete accept*		*basic complete acc-or-rej.*		*basic complete reject*	
	cost	*time*	*cost*	*time*	*cost*	*time*	*cost*	*time*	*cost*	*time*	*cost*	*time*
SUM	3933	1.83	3941	1.94	3971	3.90	3930	2.69	3932	2.94	3955	3.88

Table 2 Comparison of heuristics

Table 2 lists summarized costs of solutions (in the number of clusters used) and run times (in seconds) for all cluster selection heuristics proposed in this paper. Results of mis-pga(new)[1] (*mis-pga*) [3, p. 59], [10] and of the algorithm of Kang Yi and Chu-Shik Jhon (*ky*) [1] are presented in Table 3 in order to enable comparison between our algorithms and the best representatives of domain-specific mappers. Table 3 also contains costs of minimal solutions (*min cost*) provided by the commercial ILP solver CPLEX (for the formulation of our problem in terms of 0-1 LP see [11], [12]).

Mis-pga(new) is based on a different philosophy than the other mappers under comparison: it does not contain explicitly expressed phases of cluster enumeration and selection and it allows to modify the given subject network. This is why it is not possible to measure mis-pga cluster selection time. However, the comparison of run times of the whole technology mapping procedure is possible (see Table 3, columns *full time*). Additionally, results provided by mis-pga might be better than the minimum results provided under our assumption that no modifications of the given subject network are allowed. In reality, this happened for *vg2* and *duke2* only.

5 Conclusions and Future Work

Thorough analysis of the data collected can be summarized as follows:

- In average, the *enhanced* set of constraints provides better results than the *basic* one.
- In average, the *restricted* candidate set provides better results than the *complete* one.

[1] Standard options: ite_map -cn 1 -F 50 -d 4 -f 7 -rNU ; ite_map -cn 1 -d 4 -f 8 -r

- In average, the *accept* strategy provides slightly better results than the *accept-or-reject* strategy that, in turn, provides better results than the *reject* one.
- ALPHA type of redundancy appeared only if the *complete* candidate set and *accept* or *accept-or-reject* strategy is used.
- BETA type of redundancy was not observed.
- Backtracking was not employed.

The majority of above observations is in agreement with the results of theoretical analysis [12]. However, two phenomena seems to be rather confusing: we expected, based on results from [2], that the *reject* strategy will provide better quality solutions than the remaining strategies, and that a backtracking will be needed when the complete candidate set is employed. The explanation of both phenomena lies probably in specific properties of technology mapping binate covering problems.

The main achievement of our work is that the best heuristic from the orthogonal collection of heuristics under test was identified: *Enhanced* set of constraints / *Restricted* candidate set / *Accept* strategy. This heuristic provides better results (by 4% in average) than those provided by *mis-pga(new)* [10] and practically the same results as *ky* [1] in much shorter time. It should be noted that our approach is better

	enh. / res. / accept			*mis-pga*		*ky*			*min*
name	*cost*	*time*	*full time*	*cost*	*full time*	*cost*	*time*	*full time*	*cost*
5xp1	34	0.00	3.20	36	23.70	33	0.15	3.35	33
alu2	162	0.04	13.64	169	140.10	162	0.66	14.26	162
alu4	88	0.01	3.41	109	19.70	88	0.11	3.51	88
apex2	103	0.03	5.63	103	81.20	103	0.12	5.72	103
apex6	230	0.11	15.11	275	145.80	230	1.68	16.68	229
apex7	90	0.03	2.53	92	41.30	90	0.10	2.60	90
b9	56	0.01	1.21	62	36.20	55	0.06	1.26	55
bw	56	0.04	45.34	55	12.60	55	0.56	45.86	54
c1908	154	0.05	19.25	160	141.30	154	2.47	21.67	151
c499	164	0.02	8.62	164	60.90	164	1.94	10.54	164
c5315	520	0.19	49.09	558	448.40	521	4.54	53.44	519
c880	154	0.05	8.45	161	94.50	153	0.29	8.69	153
clip	42	0.02	4.32	43	46.00	41	0.04	4.34	41
count	39	0.02	0.42	39	8.60	39	0.01	0.41	39
des	1278	1.47	227.47	1285	1520.30	1280	38.76	264.76	1278
duke2	152	0.07	13.57	148	65.00	154	0.89	14.39	151
e64	94	0.03	0.63	94	1.20	94	0.31	0.91	94
f51m	39	0.01	3.11	40	31.30	39	0.05	3.15	39
misex1	16	0.01	1.11	16	8.40	16	0.04	1.14	16
misex2	38	0.00	0.40	38	3.30	38	0.06	0.46	38
rd73	28	0.01	1.51	29	13.50	28	0.03	1.53	28
rd84	47	0.01	4.31	49	59.90	47	0.04	4.34	47
rot	243	0.05	8.75	251	163.20	244	1.26	9.96	243
sao2	46	0.00	1.10	46	24.50	46	0.03	1.13	46
vg2	33	0.00	0.40	32	9.10	33	0.02	0.42	33
z4ml	12	0.01	0.52	14	13.10	12	0.01	0.52	12
SUM	3918	2.29	443.10	4068	3213.10	3919	54.23	495.04	3906

Table 3 Comparison of our best heuristic to *mis-pga(new)* and *ky*

especially in case of large and complex circuits. Moreover, no substantial improvement in the cost of solution can be expected in future because the results of the best of our heuristics are just about 3.7% above the minimum in the worst case (*bw*) and 0.3 % in average!

We are studying enhancements of the proposed techniques for the case when a collection of library cells with *different* costs is given (see e.g. Xilinx XC4000 family). Finally, we are analyzing the possibility to adopt the proposed techniques when *delay* or *power* minimization is the goal, i.e. to use them for performance oriented mapping. Preliminary results can be found in [12].

Acknowledgment

The authors would like to thank Dr. Rajeev Murgai for providing the code of mis-pga(new) and hints how to run it in the most efficient way.

The research of one of the authors (Michal Servít) was supported by COPERNICUS Coprodes CP 940453 grant.

References

[1] Kang Yi, Chu-Shik Jhon: A New FPGA Technology Mapping Approach by Cluster Merging. In: R.W. Hartenstein, M. Glesner (Eds.): Field-Programmable Logic. Springer 1996, pp 366-370.

[2] M. Servít, J. Zamazal: Heuristic Approach to Binate Covering Problem. EDAC'92 Proc., 1992, pp. 123-129.

[3] R. Murgai, R. Brayton, A. Sangiovanni-Vincentelli: Logic Synthesis for Field-Programmable Gate Arrays. Kluver, 1995.

[4] L Stok *et al.*: BooleDozer - Logic Synthesis for ASICs. IBM J. of Res. and Dev., 1995, Vol. 40, No. 4, pp. 407-430.

[5] J. Zamazal: Boolean Satisfiability and Covering Problems - Design and Evaluation of Efficient Algorithms. PhD Dissertation, Czech Technical University, 1995.

[6] O. Coudert, J. Madre: New Ideas for Solving Covering Problems. 31st DAC Proc., 1995.

[7] M. Servít, J Zamazal: Decomposition and Reduction - General Problem-Solving Paradigms. VLSI Design J., 1995, Vol. 3, Nos. 3-4, pp. 359-371.

[8] C. Papadimitriou, K. Steglitz: Combinatorial Optimization - Algorithms and Complexity. Prentice-Hall, 1982, pp. 406-409.

[9] The Actel FPGA Data Book, Actel Inc., 1993.

[10] R. Murgai, K. Brayton, A. Sangiovanni-Vincentelli: An Improved Synthesis Algorithm for Multiplexor-based PGA's. 28th DAC Proc., 1992, pp. 380-386.

[11] Kang Yi, Soeng-Yong Ohm, Chu-Shik Jhon: An Efficient FPGA Technology Mapping Tightly Coupled with Logic Minimization. IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, to appear in September 1997.

[12] M. Servít, Kang Yi: Binate Covering Approach to FPGA Technology Mapping Problem. CTU Research Report under preparation.

[13] T. Kam *et al.*: Synthesis of Finite State Machines - Functional Optimization. Kluver, 1997.