FPART: A Multi-way FPGA Partitioning Procedure Based on the Improved FM Algorithm*

Zhou Rongzheng, Tong Jiarong and Tang Pushan

Department of Electronic Engineering, Fudan University, Shanghai 200433, P.R.C
Tel/Fax: 86-21-65640850, Email: rzzhou@fudan.edu.cn

Abstract— In this paper, a multi-way FPGA partitioning procedure FPART is introduced. The objective function of this procedure is to reduce the number of FPGA devices and the IOB utilization. An improved min-span FM bi-partitioning algorithm on the basis of an advanced gain model is adopted as the fundamental method, and three modules: *init-part, optimize*, **and** *merge* **are combined in FPART to approach better results. After initial partitioning, the procedure optimizes the subsets to reduce the total span of cutset and then merges some subsets by removing the cells in them. Experimental results with MCNC'93 benchmarks show that FPART is fast and efficient.**

I. Introduction

To partition a large circuit into several smaller sub-circuits which can be implemented with FPGA devices is proved an efficient way in designing VLSI. Partitioning problem is NP-hard [1] and only heuristic methods are used to achieve a sub-optimal solution [2]. The FPGA partitioning problem is much harder than common VLSI partitioning ones because of two constraints: size constraint (CLB number, for LUT-based FPGA) and terminal constraint (IOB number) – and the latter is exactly the bottleneck of FPGA partitioning.

Recent researches [3, 4, 5, 6, 7, 8, 9] show that two kinds of objectives are studied in FPGA partitioning problem due to the different applications:

Cost Driven. Considering the high cost of FPGA devices, the objective *to minimize the number of FPGA devices* is the primary one if only homogeneous FPGAs can be implemented with [6, 7]; and when heterogeneous FPGAs can be selected it is better *to minimize the total device cost.* [4, 5, 8, 9] The objectives of *min-cut* and *min-span* are also studied but they are usually the secondary ones.

To achieve the above objectives, the technique of cell-replication is proposed. [4, 5] The approach of functional-replication reported by Kuznar [4] reduces the number of cuts dramatically so that the number of sub-partitions can be decreased. But it is obvious that the replicated cells are hard to be tested so it is not applied in this paper. Chou *et.al* [6, 7] reports another approach to partition the circuits by set-covering and it is also able to reduce the number of sub-partitions.

Performance Driven. While the performance of sub-partitions is considered, the objective *to minimize the critical path delay* is studied in [4, 7, 8]. Chan *et.al* [8] also presents a theory to predict the *routability* of partitioned sub-circuits before partitioning.

In this paper, the major objective is reducing the number of FPGA devices. To achieve it, the secondary objective of minimizing the total span of the cut set is utilized.

II. Problem Definition

Hypergraph. The FPGA circuit is usually described as a hypergraph $H = (V, T, E)$ where $V = \{v_1, ..., v_m\}$ as the vertex or cell set, $T = \{t_1, ..., t_r\}$ as the terminal set and $E = \{e_1, ..., e_n\}$ as the edge or net set. The multi-way partitioning is to divide a hypergraph into k non-empty sub-hypergraphs $H_i = (V_i, T_i, E_i)$ where $V = V_1 \bigcup V_2 \bigcup ... V_k$. $|e|$ and $span(e)$ denote the number of cells and sub-partitions a net e connects with. The net e with $span(e) \geq 2$ is called a *cut.*

Device Library. The circuit is partitioned into and implemented with several FPGA devices such as XC3000. The device library is described as a structure $L = (V_l, T_l, \mu_v, \mu_t)$, representing the CLB and IOB number of the device and the highest utilization respectively. (μ_v and μ_t are two parameters defined by user to control the routability of the sub-circuits.) Then the size and terminal constraints can be described as:

$$|V_i| \leq V_l \times \mu_v, i = 1, ..., k \qquad (1)$$

$$|T_i| \leq T_l \times \mu_t, i = 1, ..., k \qquad (2)$$

where $|V_i|$ and $|T_i|$ represent the number of CLBs and IOBs contained in a sub-circuit H_i. We define LB as the lower bound of the sub-partition number:

$$LB = \lceil max(\frac{|V|}{V_l \times \mu_v}, \frac{|T|}{T_l \times \mu_t})\rceil$$

*Supported by National Key Project of "9.5" Plan for Technique Research and Development of P.R.C [Grant No. 96-738-01-09-01].

TABLE I
XC3000 Device Library

Device	CLB	IOB
XC3020	64	64
XC3030	100	80
XC3042	144	96
XC3064	224	120
XC3090	320	144

where $|V|$ and $|T|$ represent the number of CLBs and IOBs contained in the original circuit. The XC3000 device library used in this paper is listed in Table I.

Objective Function. The main objective of FPGA partitioning is *to reduce the number of sub-partitions* and can be described as:

$$min\{k\}, V = V_1 \bigcup V_2 \bigcup ...V_k \tag{3}$$

and the secondary objective is *min-span*:

$$min \sum_{j=1}^{n} span(e_j) \tag{4}$$

III. Improved Min-span FM Algorithm

One of the efficient heuristic methods in resolving the VLSI partitioning problem is KL-FM algorithm [10, 11]. But it is easy to be trapped into some local minima, and this shortcoming restricts it to reach the global optimization. In [12] LA algorithm enhances FM by adding look-ahead multi-level gains and improves the results of small circuits, while a cluster-removal method is proposed in [13] to improve the results of larger circuits by assigning a large weight to the nets connected to moved cells.

In this paper, an improved min-span FM algorithm is proposed. The potential gains that a moved cell adds on its neighbors are considered in this algorithm and when cells in two sub-circuits are in exchanging, the effect of the other sub-circuits (in multi-way partitioning) must be considered too.

A. Gain Calculation

Basic Definition. The cell is called *free* when it is not moved, and *locked* after moving. The *neighbor cells* of a cell c are the cells connect to c directly through one or more nets. The *incident number* of a net e with respect to a cellset A (that is, the number of cells in set A that are on net e) is defined as:

$$\alpha_A(e) = |\{c | c \in A \text{and} c \in C_e\}| = |A \bigcap C_e|$$

where C_e denotes the cells on the net e. The *binding force* of a net e with respect to the set A(denoted as $\beta_A(e)$) is defined as:

$$\beta_A(e) = \begin{cases} \alpha_{A_F}(e) & \text{if} \alpha_{A_L} = 0 \\ \infty & \text{if} \alpha_{A_L} > 0 \end{cases}$$

where $A_F(A_L)$ denotes the subset that contains all the free cells of A (all locked cells of A). The binding force can be intuitively viewed as an indicator of how a net is bound to a cell set.

Initial Gain. The *initial gain* is defined as the same as that of [11]. Define $g(c)$ as the initial gain of a cell c and $g(e)$ as the initial gain of the net e that is on c when c moves from set A to set B:

$$g(e) = \begin{cases} 2 & \text{if} \alpha_A(e) = 1 \text{and} \alpha_B(e) > 0 \text{and} \alpha_C(e) = 0 \\ 1 & \text{if} \alpha_A(e) = 1 \text{and} \alpha_B(e) > 0 \text{and} \alpha_C(e) > 0 \\ 0 & \text{if} \alpha_A(e) = 1 \text{and} \alpha_B(e) = 0 \text{and} \alpha_C(e) > 0 \\ 0 & \text{if} \alpha_A(e) > 1 \text{and} \alpha_B(e) > 0 \\ -1 & \text{if} \alpha_A(e) > 1 \text{and} \alpha_B(e) = 0 \text{and} \alpha_C(e) > 0 \\ -2 & \text{if} \alpha_A(e) > 1 \text{and} \alpha_B(e) = 0 \text{and} \alpha_C(e) = 0 \end{cases} \tag{5}$$

where C is the complement set of $A \bigcup B$. Let E_c denote the net set on the cell c, then $g(c)$ can be described as:

$$g(c) = \sum_{e \in E_c} g(e) \tag{6}$$

Potential Gain. When a cell is moved and locked, the gains of its neighbors must be updated. We define *potential gain* with the locked cell to its neighbors. Let $g_p(e)$ denote the potential gain of net e on cell c when c moves from set A to set B:

$$g_p(e) = \begin{cases} 0 & \text{if} \beta_A(e) = |e| \\ 0 & \text{if} \beta_A(e) = \infty \text{and} \beta_B(e) = \infty \\ -R & \text{if} \beta_A(e) = \infty \text{and} \beta_B(e) < |e| \\ R & \text{if} \beta_A(e) < |e| \text{and} \beta_B(e) = \infty \end{cases} \tag{7}$$

where R can be considered as a *penalty factor*. The potential gain can be considered as the force that the locked cell adds on its neighbors. When the neighbor cell c is in the same subset with locked cell c_l, the force is negative then c is enforced to stay in the original subset and when c is in a different subset from c_l, the force is positive then c is pulled into the subset where c is in. So the tightly connected cells (called a *cluster*) can be converged to one subset.

R is specified as $2 \times \lfloor \bar{c} \rfloor$ where $\bar{c}$ denotes the *average degree* of the nets. Then the force of a locked cell is large enough to compare with that of all the free cells. Then $g_p(c)$ can be defined as:

$$g_p(c) = \sum_{e \in E_c} g_p(e) \tag{8}$$

Cost Function. The total gain with respect to the min-span objective function can be defined as:

$$g_{min-span}(c) = g(c) + g_p(c) \tag{9}$$

B. Two-way Min-span FM Algorithm

When the number of subsets k is 2, a min-span partitioning is actually the same as a min-span one; when k

is more than 2, the effect of the other subsets must be considered.

Define C as the complement set of $A \bigcup B$ when the two subsets A and B are in exchanging. The two-way min-span FM algorithm called *minspan-FM* can be illustrated as Algorithm 1:

Algorithm 1:

```
$g = \infty$; ($g$ is the maximum exchange gain)
while($g > 0$)begin
    Free all the cells in $A$ and $B$;
    Calculate the gain of cells in $A$ and $B$;
    while(there is free cells in $A$ and $B$)begin
        Select cell $v$ with maximum gain in $A$ and $B$;
        Push $v$ into stack, lock it;
        Calculate the gain of cells in $A$ and $B$;
        Calculate $g$;
    end while
    Select cells $v_0, v_1, \ldots, v_k$ with maximum exchange gain $g$;
    if($g \leq 0$)then
        break;
    end if
    Exchange $v_0, v_1, \ldots, v_k$ between $A$ and $B$;
end while
```

IV. Multi-way FPGA Partitioning Algorithm FPART

FPART is a heuristic algorithm combined by *init-part, optimize* and *merge*. We give a survey about the procedure at first, then the details of the three modules. And at last, an example about the partitioning of a large circuit is presented to illustrate the algorithm imaginably.

A. Overview

FPART utilizes the FM algorithm proposed in the last section. After the whole circuit is divided into k subsets by step-by-step two-way min-span FM algorithm in the procedure of *init-part*, the k subsets are optimized by the procedure of *optimize* to reduce the total span of the cutset, and the procedure of *merge* is applied to decrease the number of subsets k. The procedure of *optimize* and *merge* are applied repeatedly to approach a better result. Algorithm 2 illustrates the overview of FPART:

Algorithm 2:

```
$k = 0$;
init-part();
for($i = 0; i < r; i++$)begin
    if($k = LB$)then
        break; (End of optimization and merge)
    end if
    optimize();
    merge();
end for
optimize();
```

B. Initial partition

In conventional FM algorithm, the initial sub-partitions are usually generated by random cell-allocating on the basis of the relationship between the cells. With this method the size and terminal constraints are not always satisfied and it is hard to make it so afterward. In FPART, k subsets that satisfy both the size and terminal constraints can be generated and k closes to the lower bound LB as much as possible.

In *init-part*, each sub-partition is cut from the original circuit step-by-step. A candidate subset S with as many cells as possible(but no more than the size constraint) is created by expanding from a seed which is randomly selected from the present cell set G(G is the original cell set V at first and it is the remainder cell set after some subsets are generated); then the improved FM algorithm is applied to reduce the cut-size of S; if the cut-size of S still violates the terminal constraints some cells are removed from S to reduce it until S can be implemented with a selected FPGA device.(Algorithm 3).

Algorithm 3:

```
$G = V, k = 0$ ($V$ is the original cell set)
while(size of $G$ >size constraint)begin
    Randomly select a seed cell $v$ in $G$;
    $S = \{v\}, G = G \backslash \{v\}$, lock $v$;
    while(size of $S$ <size constraint)begin
        Calculate the gain of neighbors of $v$ in $G$;
        Select cell $v$ with maximum gain in $G$;
        $S = S \bigcup \{v\}, G = G \backslash \{v\}$, lock $v$;
    end while
    minspan-FM($S$,$G$);
    while(cut-size of $S$ >terminal constraint)begin
        Select cell $v$ with maximum gain in $S$;
        $S = S \backslash \{v\}, G = G \bigcup \{v\}$, lock $v$;
        Calculate the gain of neighbors of $v$ in $S$;
    end while
    Free all the cells in $S$ and $G$;
    $k++$;
end while
```

C. Optimization

The procedure of *init-part* is able to generate k subsets that can be implemented with k FPGA devices but these subsets are still able to be optimized for two reasons: (1) the seed is randomly selected so the results are not constant; (2) the subsets are generated via step-by-step cutting so not all the subsets are min-span.

The best method to optimize the subsets is exchanging cells between all the subsets simultaneously, that is, to utilize a multi-way min-span FM algorithm. It is most possible to reach the best result but is very hard to be realized because of the terminal constraint and the high space/time complexity(to each cell, $k-1$ gains must be calculated simultaneously). A faster method is exchanging cells between every two subsets step-by-step. The space complexity is much lower but the time complexity is still high(total $\frac{k(k-1)}{2}$ times of FM exchanging will be processed).

We alternate with an efficient procedure based on the second method but improve the time complexity. For every two subsets V_i and V_j, define $A = V_i, B = V_j$ and $C = V - V_i \bigcup V_j$, then the cell gains are calculated(cost function (9)) and the maximum gains of V_i and V_j are defined as $g_i(j)$ and $g_j(i)$. Only when $g(i,j) = g_i(j) + g_j(i) > 0$ that is the two-way min-span FM algorithm applied between A and B to reduce the total span of the cutset. This operation is illustrated in Algorithm 4:

```
Algorithm 4:
  for($i = 1; i < k-1; i++$)begin
    for($j = i+1; j < k; j++$)begin
      Calculate cell gains of $V_i$ and $V_j$;
      Define maximum gain of $V_i$ and $V_j$ as $g_i(j)$ and $g_j(i)$;
      $g(i,j) = g_i(j) + g_j(i)$;
      if($g(i,j) > 0$)begin
        minspan-FM($V_i$,$V_j$);
        Free cells in $V_i$ and $V_j$;
      end if
    end for
  end for
```

In this kind of neglecting the effect is a little worse because sometimes the maximum exchanging gain could be more than 0 after more cells are moved, though $g(i,j) \leq 0$. But it is worthy to be adopted for its lower time complexity.

D. Merge

The goal of procedure *merge* is decrease the number of subsets by emptying some of them. After optimization all the subsets are min-span and there are some vacant for more cells to be added in. If k is equal to the lower bound LB, merging is not necessary; if not, the procedure of *merge* tries to empty the subset which owns the least cells by moving cells out. When a subset is successfully nullified, update k and try another round of merging; if not, end the procedure of *merge*.

This procedure is empirical but efficient. The cells moved to a new subset are somewhat less tight with the original one but the number of subsets is reduced so it is worthy to try.

Running times of *merge* r is defined by user. In our experiments, r is 3. We find that if more times is applied little gains are got. It is possible that a round of merging fails and the total span of the cutset increases. To obtain new chances to succeed, FPART does not return back to the formal min-span status but goes on to apply *optimize* on the basis of the changed subsets.

TABLE II
PARTITION s15850 INTO XC3042 DEVICES($\mu_v = 0.9, \mu_t = 1$)

Step	k	*sec*	CLB%	IOB%	IOB
init-part	8	22	.73	.93	716
optimize(1)	8	27	.73	.86	658
merge(1)	8	28	.73	.88	675
optimize(2)	8	32	.73	.85	654
merge(2)	7	33	.84	.94	629
optimize	7	38	.84	.87	584

E. Illustrative Example

We illustrate our algorithm by partitioning the MCNC'93 benchmark circuit $s15850$ into XC3042 devices. The CLB(size) and IOB(terminal) constraints of XC3042 are 144 and 96. The maximum CLB utilization is defined as 0.9 while the maximum IOB utilization is 1. The CLB number of $s15850$ is 842, and IOB number is 102. Then the lower bound LB is:

$$LB = \lceil max(\frac{842}{144 \times 0.9}, \frac{102}{96 \times 1}) \rceil = 7$$

The example is tested in a workstation of SUN-sparc20. Table II illustrates the results of the partitioning for $s15850$ with FPART.

After the procedure of *init-part* is utilized, the circuit of $s15850$ is divided into 8 sub-partitions with each of them satisfies both the size and terminal constraints. Then *optimize* is operated and the IOB number decreases from 716 to 658, about 8%. The procedure of *merge* fails to reduce the number of sub-sets with the number of IOB increasing unfortunately , but it is decreased again by the second *optimize* operation. Then *merge* is operated again and successes to empty one sub-circuit. Now k equals to LB so no more rounds of *optimize-merge* operation are needed and the last *optimize* reduces the IOB number from 629 to 584, about 7%. At last, 7 sub-partitions of which the average CLB and IOB utilization are 84% and 87% are generated.

V. Data Structure and Time Complexity

A. Data Structure

As in [11], a bucket-sorting structure is defined for each subset to achieve a linear time complexity. Let $p$max denote the maximum number of nets on a cell, then the total gain of each cell is bounded between a range according to the cost function (9):

$$-(R+1) \times p\max \leq g(c) \leq (R+1)p\max$$

So the number of buckets in a structure is $2 \times (R+1) \times p\max + 1$.

In the LUT-based FPGA circuits, the number of cells on a net(denoted as $c$max) could be very large(see Table III). when one of the cells on this net is moved, the time consumed in gain-updating is unbearable . We define a constant MAXPIN to solve this problem. The nets with more than MAXPIN cells are celled HUGE nets(usually the Reset/Set and Clock signals of DFF). The cells connect with a HUGE net are not exactly intimate to each other(if they are, more nets surely exist between them) So when a cell is moved, the gain recalculation of the cells on the HUGE nets can be neglected to reduce the time complexity.

B. Time Complexity

Though the gain calculation is a little more complex than min-cut FM algorithm, the time complexity of two-way min-span FM algorithm is also $O(P)$ when the total pin number of the two subsets in exchanging is P.

The time complexity of FPART includes three items: the time of *init-part, optimize* and *merge*. The process of *merge* is very fast and able to be ignored. Now suppose the pin number of the circuit is P and the number of subsets is k, then the pin number in each subset can be approximated as P/k. In *init-part*, the time complexity is determined by FM exchanging. When a new subset is generated, P/k pins are eliminated from the exchanging cell sets until the last two sets. So the time of *optimize* is: $O(\sum_{i=0}^{k-2}(P-\frac{i}{k}P)) = O(\frac{k+1}{2}P)$.

The *optimize* process is $\frac{k(k-1)}{2}$ times in a round (the worst case) and each time it costs the time of $O(\frac{2}{k}P)$. The number of rounds is r(r is about 3 in the most common case), then the total complexity of *optimize* is: $O(r\times\frac{k(k-1)}{2}\times\frac{2}{k}P) = O(r(k-1)P)$.

So the time complexity of FPART is:

$$O(\frac{k+1}{2})P + O(r(k-1)P) \approx O((r+\frac{1}{2})kP)$$

VI. Experimental Results

The algorithm of FPART is realized in UNIX system with C language and MOTIF X-window interface. It can be run on a SUN or HP workstation.

Benchmark circuits in MCNC'93 are tested. The largest nine circuits of it are listed in Table III. The device library is selected from XC3000 listed in Table I. Firstly all the circuits are partitioned into XC3020 and XC3042 devices for 10 times each on a workstation of SUN-sparc20. The results are listed in Table IV and they are compared with results copied from [4]. The results of *init-part* are generated by initial partitioning and *(p,o,m)* by the whole procedure.

The results of *init-part* and *(p,o,m)* are quite close to the lower bound LB, with the excess of 11.6% and 5.8%(XC3020), 9.7% and 4.2%(XC3042). They are better than or very near to that of *(p,r,o,p)* (which are replicated and re-mapped). The number of sub-partitions decreases from *init-part* to *(p,o,m)* by 5% which proves that the procedure of *merge* is quite efficient. The IOB utilization

TABLE III
MCNC'93 Benchmark Circuits

Circuit	CLB	IOB	NET	PIN	$c$max	$p$max	$\bar{c}$
c3540	283	72	489	1573	27	7	3.22
c5315	377	301	699	2106	23	7	3.01
c6288	833	64	1472	3775	16	6	2.56
c7552	489	313	921	2619	77	7	2.84
s5378	381	86	628	2246	125	9	3.58
s9234	454	43	716	2630	156	9	3.67
s15850	842	102	1265	4893	395	9	3.87
s38417	2221	136	3216	13132	1026	9	4.08
s38584	2901	292	3884	17201	1231	9	4.43

TABLE IV
Partitioned into XC3020 and XC3042 devices(times=10,$\mu_v = 0.9, \mu_t = 1$)

Partitioned into XC3020 devices							
	Best of PROP		Best of FPART				
	(p,r,o,p)		*init-part*		*(p,o,m)*		LB
Circuit	IOB	k	IOB	k	IOB	k	k
c3540	.80	6	.95	6	.92	6	5
c5315	.90	8	.85	9	.90	8	7
c6288	.55	12	.70	15	.70	15	15
c7552	.77	9	.89	9	.88	9	9
s5378	.78	9	.95	10	.94	9	7
s9234	.65	9	.85	9	.78	9	8
s15850	.69	16	.80	18	.92	16	15
s38417	.53	44	.77	42	.79	40	39
s38584	.61	56	.74	56	.81	53	51
Total	-	169	-	174	-	165	156
Partitioned into XC3042 devices							
	Best of PROP		Best of FPART				
	(p,r,o,p)		*init-part*		*(p,o,m)*		LB
Circuit	IOB	k	IOB	k	IOB	k	k
c3540	.89	2	.91	3	.73	3	3
c5315	.83	4	.95	4	.94	4	3
c6288	.55	5	.71	7	.67	7	7
c7552	.90	4	.89	5	.93	4	4
s5378	.89	4	.88	5	.94	4	3
s9234	.65	4	.82	4	.77	4	4
s15850	.73	7	.83	8	.88	7	7
s38417	.41	19	.70	19	.73	18	18
s38584	.56	25	.81	24	.77	24	23
Total	-	74	-	79	-	75	72

of *(p,o,m)* is better than that of *init-part*(when k is the same) which proves that the procedure of *optimize* is also efficient.

The largest three circuits are partitioned into XC3090 then. The CLB and IOB utilization are both 1. The best results of 10 times running are listed in Table V. Note that RFM(Recursive FM) and LRSC are copied from [7], KPF is from [8](it is the marginal results) and Kim is the result of [9]. It is very clear that FPART is more efficient when the constraints are less tight.

The run time is also tested. Table VI shows the run time of some circuits when partitioned into different FPGA devices. The CLB and IOB utilization are 0.9 and 1 when partitioned into XC3020 and XC3042 and they are both 1 when partitioned into XC3090. The results of RFM and LRSC copied from [7] and that of Kim copied from [9] are partitioned into XC3090. The run

TABLE V
Partitioned into XC3090 devices(Times=10,$\mu_v = 1, \mu_t = 1$)

Circuit	RFM	LRSC	KPF	Kim	FPART	LB
s15850	4	3	4	3	3	3
s38417	12	10	9	8	7	7
s38584	17	14	11	14	10	10

TABLE VI
Partitioning time of FPART

				Avg. of FPART		
Circuit	RFM	LRSC	Kim	XC3090	XC3042	XC3020
c3540					4	7
c5315					6	9
c6288					8	12
c7552					10	10
s5378					7	8
s9234					7	8
s15850	10	20	66	25	40	32
s38417	72	199	976	90	114	159
s38584	135	309	1181	135	180	256

time of FPART is the average of 10 times running. We can conclude from Table VI that FPART is quite fast and is efficient for partitioning large circuits.

VII. Conclusions

Based on an advanced gain model, an improved min-span FM partitioning algorithm is proposed. It is utilized in the FPGA partitioning procedure of FPART which is combined by three modules: *init-part, optimize* and *merge*. *optimize* can reduce the total span of the cutset and *merge* can reduce the number of sub-partitions. The whole procedure of FPART is fast and efficient to decrease the number of CLB devices and the utilization of IOB.

From the experimental results we can see that FPART is more efficient when constraints of the device library are looser. So if more advanced FPGA chips such as XC4000 series are applied, good results can also be obtained.

There are some aspects can be improved. The results are not stable due to the random selection of the seeds when new subsets are created. The procedure of *optimize* maybe can be enhanced by some more efficient algorithms such as a multi-way min-span FM algorithm.

References

[1] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Company, 1979

[2] W. Donath, "Logic partitioning", in *Physics Design Automation of VLSI System*, Menlo Park, California, Benjamin/Cummings, 1988

[3] F. Johannes, "Partitioning of VLSI circuits and system", *33rd ACM/IEEE Design Automation Conf.*, 1996

[4] R. Kuznar and F. Brglez, "PROP: a recursive paradigm for area-efficient and performance oriented partitioning for large netlist", *Int. Conf. on CAD*, pp. 644-649, 1995

[5] R. Kuznar, F. Brglez and B. Zajc, "Multi-way netlist partitioning into heterogeneous FPGAs and minimization of total device cost and interconnect", *31st ACM/IEEE Design Automation Conf.*, pp. 238-243, 1994

[6] N-C. Chou, L-T. Liu, C-K.Cheng, W-J. Dai, R. Lindelof, "Local ratio cut and set covering partitioning for huge logic emulation system", *IEEE Trans. on CAD*, Vol. 14, No. 11, pp. 1342-1357, 1995

[7] N-C. Chou, L-T. Liu, C-K.Cheng, W-J. Dai, R. Lindelof, "Circuit partitioning for huge logic emulation system", *31st ACM/IEEE Design Automation Conf.*, pp. 244-249, 1994

[8] P. K. Chan, M. D. Schlag and J. Y. Zien, "Spectral-based multiway FPGA partitioning", *IEEE Trans. on CAD*, Vol. 15, No. 5, pp. 554-560, 1996

[9] C. Kim and H. Shin, "A performance-driven logic emulation system: FPGA network design and performance-driven partitioning", *IEEE Trans. on CAD*, Vol. 15, No. 5, pp. 560-568, 1996

[10] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs", *The Bell Sys. Tech. Journal*, pp. 291-307, Feb. 1970

[11] C. Fiduccia and R. Mattheyses, "A linear-time heuristic for improving network partitioning", *19th ACM/IEEE Design Automation Conf.*, pp. 175-181, 1982

[12] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI networks", *IEEE Trans. on Computers*, Vol. C-33, No. 5, pp. 438-446, 1984

[13] S. Dutt and W. Deng, "VLSI circuit partitioning by cluster-removal using iterative improvement techniques", *Int. Conf. on CAD*, 1996

[14] X. Tan, J. Tong and P. Tang, "A multiple-optimizing algorithm for multiple-way VLSI network partitioning", *Chinese J. of Computers*(in Chinese), Vol. 10, No. 5, pp. 321-328, 1996

[15] R. Zhou, J. Tong and P. Tang, "A fast and efficient multi-way FPGA partitioning procedure via two-way min-cut algorithm", *Int. Conf. on CAD/Graphics'97*, P.R.C, 1997