*VLSI DESIGN*
2000, Vol. 11, No. 3, pp. 237–248
Reprints available directly from the publisher
Photocopying permitted by license only



# Circuit Partitioning for FPGAs by the Optimal Circuit Reduction Method

R. P. BAZYLEVYCH*, R. A. MELNYK and O. G. RYBAK

*Software Engineering Department Lviv Polytechnic State University, 12 Bandera Street, Lviv, 79013, Ukraine*

*(Received 1 March 1999; In final form 1 December 1999)*

Mathematically the most difficult partitioning problem – packaging – is being considered. Its purpose is to minimize a number of partitions and to satisfy the constraints on the number of constituent elements and external nets. To solve the problem, the Optimal Circuit Reduction Method, suggested by R. Bazylevych is being used. The optimal reduction tree to reflect the hierarchical entrance of smaller clusters into bigger ones is being built for the first step. At the second step we select one or more tree vertices which better meet the given constraints and are the first partitions generated from. After creating every new partition we eliminate its elements from the circuit and repeat the procedure to complete all partitions. During the last stage optimization strategies to exchange some elements between the partitions are being used. Better or equivalent results among known tests confirm the effectiveness of this method.

*Keywords*: Partitioning; FPGA packaging; Combinatorial optimization

## 1. INTRODUCTION

Packaging of complex circuits to a minimal number of FPGAs is one of the most important tasks in the field of contemporary partitioning problems. Due to a high price, one can easily see the benefit in reducing the number of FPGAs to satisfy the functionality of a given circuit. The challenge involves multiway partitioning with constraints, the most difficult being the number of constituent elements and the number of external IO terminals. Packaging belongs to intractable NP-complete combinatorial optimization problems.

FPGAs are widely used in rapid prototyping, and also in place of ASICs. These problems have been the focus of a number of authors [1–3], however, at present, the solutions produced by current techniques are of inadequate quality. In this study we are proposing a new approach, which utilizes the Optimal (Parallel) Circuit Reduction (OCR) Method [4]. Fundamentally the method solves the problem in three stages. In the first stage, we build an Optimal Reduction Tree (ORT),

*Corresponding author. Tel.: +380322 398877 or +380322 398578, Fax: +380322 744300, e-mail: rbaz@polynet.lviv.ua

which represents the hierarchical entry of smaller circuit clusters into larger ones. In the second stage, the method splits the circuit into the individual FPGAs by using ORT. This type of approach essentially reduces the computational complexity, while simultaneously assuring a high quality solution. In the third stage, new optimizing procedures are used to reduce the number of partitions. The OCR Method provides the high quality solutions to a wide class of partitioning problems with various constraints.

## 2. PREVIOUS WORKS

We concentrate our discussion on studies dedicated to the circuit division to the strongly connected groups of elements, since the Optimal Circuit Reduction Method solves this problem at the first stage. Most likely the first investigation which foresees a division of such groups is in [5]. The authors identify all minimal groups, which have the basic distinction that for each of its subgroups the number of external nets is greater than for the group itself. This approach is interesting, however, it has not been applied to complex circuits with a high degree of complexity. In [6] individual blocks are partitioned at the logic level. This approach assumes previous determination of all initial nets for each group, and every nonprimary block must have exactly one output net. This restriction essentially reduces the number of problems for which the method could be applied, since it requires manual entry of additional information. In [7] an algorithm was presented for cluster recognition, which efficiently identifies the "core" of functional modules. The algorithm has $O(n^{k-l})$ complexity where $k$ is the number of edge-disjoint paths connecting two vertices such that each of these paths has at most length $l$. Such time complexity may be of prohibitive magnitude for large circuits. It is difficult to choose $k$ and $l$ for a given netlist. The authors in [8] presented a circuit clustering method based on a random walk in the netlist graph. They proposed a method for extracting clusters from the random walk *via* the concept of a cycle. The bottom-up algorithm based on recursive collapsing of small cliques in graph was suggested in [9]. This approach requires considerable computational time. In [10] the algorithm constructs an ordering by adding a vertex and then splitting it into clusters. Authors in [3] generated an initial partitioning by using the gain principle of the Fiduccia–Mattheyses heuristic for selected vertices to be moved. With the initial partition they used the same heuristic for improvement of every pair of devices. Very good results of recursive combining clustering with iterative improvement method were presented in [11]. In [12] the authors proposed a new cluster metric called the cluster ratio. A two-phase algorithm that attempts to combine the merits of the bottom-up and top-down approaches was proposed in [1]. The authors devised a local ratio-cut clustering scheme to reduce the circuit complexity, and then a set covering partitioning is used to reduce the number of FPGAs. Some additional materials can be found in [13–17].

## 3. PROBLEM FORMULATION

The initial data to solve the problem are:

- a set of elements

$$P = \{p_1, \ldots, p_n\}; \tag{1}$$

- a set of nets

$$E = \{e_1, \ldots, e_m\}; \tag{2}$$

- a system of sets of elements incident to every net

$$\begin{aligned} &P(E) = \{P(e_1), \ldots, P(e_m)\}; \\ &(\forall P(e_i) \in P(E), i = 1, \ldots, m) \\ &[P(e_i) = \{p \in P | \, p \textit{ is incident to } e_i\}]. \end{aligned} \tag{3}$$

Every set $P(e_i)$ contain those elements of $P$ that are incident to $e_i \in E$.

It is easy to receive a system of sets of nets incident to every element from the previous system:

$$E(P) = \{E(p_1), \ldots, E(p_n)\};$$
$$(\forall E(p_i) \in E(P), i = 1, \ldots, n) \quad (4)$$
$$[E(p_i) = \{e \in E | e \text{ is incident to } p_i\}].$$

Every set $E(p_i)$ contain those elements of $E$ that are incident to $p_i \in P$.

Systems (3) and (4) could be presented as a bipartite graph $G(P \leftrightarrow E)$, where one set of vertices corresponds to the set of circuit elements, and another – to the set of circuit nets.

Initial parameters usually include certain characteristics of the constituent elements

$$W(P) = \{W(p_1), \ldots, W(p_n)\};$$
$$W(p_i) = \{w_{1i}, \ldots, w_{si}\}, \quad i = 1, \ldots, n,$$

and certain characteristics of the electrical nets to be considered:

$$V(E) = \{V(e_1), \ldots, V(e_m)\};$$
$$V(e_i) = \{v_{1i}, \ldots, v_{si}\}, \quad i = 1, \ldots, m.$$

Usually they include the element physical area, power dissipation *etc.*, while for electrical nets they include a signal propagation delay and others. The most significant contributor to the propagation delay is a break caused by a signal passes from one FPGA to another. The propagation delay is large in this case. It is very important to take into account the constraints, which absolutely must be satisfied, otherwise the desired circuit will either not be feasible or will not function according to the desired specifications.

We will formulate the problem statement assuming the following:

It is necessary to obtain a partitioning $\tilde{P}$ of the set of elements $P$:

$$\tilde{P} = \{P_1, \ldots, P_k\} \quad (5)$$

so that a total number of partitions is minimized:

$$k \rightarrow \min,$$

while satisfying the given constraints:

$$(\forall P_i \in \tilde{P})[(n_i \leq n_{i\max}) \ \& \ (m_i^{\text{ex}} \leq m_{i\max}^{\text{ex}})].$$

Here $n_i$ and $m_i^{\text{ex}}$ are the numbers of elements and external nets (IO terminals) in each partition $P_i$ that cannot exceed the upper bounds $n_{i\max}$ and $m_{i\max}^{\text{ex}}$.

It should be also satisfied the following conditions:

$$(\forall P_i \in \tilde{P})[(P_i \subset P) \ \& \ (P_i \neq \varnothing)]; \quad (6)$$

$$(\forall (P_i, P_j) \in \tilde{P})[P_i \cap P_j = \varnothing]; \quad (7)$$

$$\bigcup_{i=1}^{k} \bigcup_{j=1}^{n_i} p_{ij} = P. \quad (8)$$

In certain cases, Eq. (7) could not be satisfied. It allows to replicate elements that may result to improve some partitioning characteristics.

To solve the problem we utilize the OCR Method [4]. The main idea of the approach is to build an Optimal Reduction Tree by bottom-up strategy and then to partition the circuit by applying top-down strategies. The combinatorinal analysis of hierarchically built clusters is being done instead of analysis of original elements. It drastically reduces the computational complexity and improves the quality of solutions. Based on this method, the problem is being solved in three stages:

1. Optimal Reduction Tree Generation,
2. Initial Packaging,
3. Packaging Optimization.

## 4. OPTIMAL REDUCTION TREE GENERATION

### 4.1. The Optimal Reduction Tree

The Optimal Reduction Tree $T^R$ is a binary rooted tree which leaves (level 1) correspond to the elements of set $P$ and a root (level $H$) – to all

grouped circuit (Fig. 1). The reasoning behind this approach is to create the tree, in which intermediate vertices correspond to circuit clusters (elements groups) with more internal nets than others. In addition, it is desirable to build such the tree, where the individual clusters are treated independently and in parallel, meaning to proceed in contrast to the well-known greedy approaches. In the last ones they are being treated in a serial manner, and therefore the first clusters are being created under more favorable conditions, and the last created clusters are much worse than earlier created ones. Our approach is to remove this disadvantage by developing the conditions for parallel formations of groups. It favors to the creation the natural circuit clusters. To remove completely the greedy approach is almost impossible, therefore we attempt to weaken it. We develop a bottom-up parallel-serial approach, where in the initial steps the smallest clusters are being created and they grow to the next steps by the cost of other elements or clusters. It is possible to create the new clusters from the initial elements or to enlarge those formed earlier in the later steps. A physical analogy is the crystallization of a liquid, where there are only certain centers of crystallization, which later grow at the cost of free liquid or other crystals.

Our approach tends to identify natural clusters in the circuit. This means that the tree $T^R$ could be considered as the Hierarchical Cluster Tree. It illustrates a natural hierarchy for the smaller clusters entrance into the greater ones. During the tree generation the most important aspect is the question of the element groupings into clusters, *i.e.*, to find out the criteria for elements and clusters merging. A natural electrical circuit functional tree of is good for this purpose. Obviously it is helpful to take advantage of this information; however, in this case we build the cluster tree based on certain formal criteria.

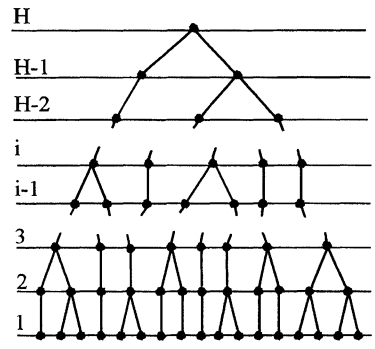


FIGURE 1 The Optimal Reduction Tree.

We have already mentioned the similarities between the cluster tree generation and the freezing process (transformation from the liquid to the solid state). It is need to note that there are significant differences between our processes and simulated annealing processes, while the laster is being widely used for combinatorial problems. The starting temperature is very important for simulated annealing since the result strongly depends on it. Our process has no such dependence. The temperature of the freezing process is constant. The process begins when all elements are free and not grouped with others (liquid state). This state corresponds to an unlimited high temperature with simulated annealing. Secondly, in simulated annealing the cooling schedule is also important. Here the group formation process has some elements of analogy with the cooling schedule. However, the principal distinction between the two techniques is that proposed one could be considered as a constructive method, while the simulated annealing process is iterative. In our case, the final state is being constructed from the smallest original elements by the parallel-serial grouping of clusters. At every level pairs of clusters merge only when they match the best grouping criteria. All other clusters and free elements rise from one level to the next without modification. The passage from one level to the next means discharge of the energy and creation of some new crystals (clusters) and (or) growing old one, which have appeared at the previous levels. The process terminates when all elements are grouped (solid state).

### 4.2. Cluster Definition

Every cluster $C$ at any level could be defined as a pair of a set of elements and a set of nets:

$$C = \langle P(C), E(C) \rangle.$$

A set of cluster nets consists of a set of internal and a set of external nets (Fig. 2):

$$E(C) = E^{\text{in}}(C) \cup E^{\text{ex}}(C).$$

Every internal net has at least two nodes in a cluster. Every external net has at least one node out of the cluster. Some nets could be simultaneously internal and external. We mention these nets as combinational:

$$E^{\text{cmb}}(C) = E^{\text{in}}(C) \cap E^{\text{ex}}(C).$$

We note also a set of pure internal $E^{\text{in}^*}$ nets, all nodes of which are inside a cluster, and a set of pure external $E^{\text{ex}^*}$ nets, with only one node inside a cluster:

$$E^{\text{in}^*}(C) = E^{\text{in}}(C)/E^{\text{cmb}}(C);$$
$$E^{\text{ex}^*}(C) = E^{\text{ex}}(C)/E^{\text{cmb}}(C).$$

Now we propose to use three mutually independent sets $E^{\text{in}^*}(C)$, $E^{\text{ex}^*}(C)$ and $E^{\text{cmb}}$ $(C)$ for a cluster's nets definition:

$$E(C) = E^{\text{in}^*}(C) \cup E^{\text{ex}^*}(C) \cup E^{\text{cmb}}(C),$$

where

$$E^{\text{in}^*}(C) \cap E^{\text{ex}^*}(C) = \oslash, E^{\text{in}^*}(C) \cap E^{\text{cmb}}(C) = \oslash,$$
$$E^{\text{ex}^*}(C) \cap E^{\text{cmb}}(C) = \oslash;$$

and

$$E^{\text{in}}(C) = E^{\text{in}^*}(C) \cup E^{\text{cmb}}(C),$$
$$E^{\text{ex}}(C) = E^{\text{ex}^*}(C) \cup E^{\text{cmb}}(C).$$

### 4.3. Clusters Merging

Every new cluster $C_{st}$ (Fig. 3) at any level of the tree $T^R$ is created by merging some two clusters $C_s$ and $C_t$ of the previous level:

$$C_{st} = \text{Merg}\ \{C_s, C_t\}.$$

Here $C_s$ and $C_t$ are children of the cluster $C_{st}$. We note that the original element of the set $P$ (first level) we consider as initial clusters for simplicity. A set of elements of the new cluster is an union of sets of children's cluster's elements:

$$P(C_{st}) = P(C_s) \cup P(C_t); \quad P(C_s) \cap P(C_t) = \oslash.$$

A set of nets of the new cluster is an union of sets of children's cluster's nets:

$$E(C_{st}) = E(C_s) \cup E(C_t); \quad E(C_s) \cap E(C_t) \neq \oslash$$

in general.

We have for the cluster $C_s$, three mutually independent sets $E^{\text{in}^*}(C_s)$, $E^{\text{ex}^*}(C_s)$, $E^{\text{cmb}}$ $(C_s)$ and

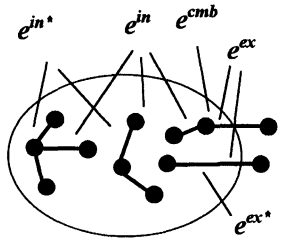


FIGURE 2 Clusters nets.

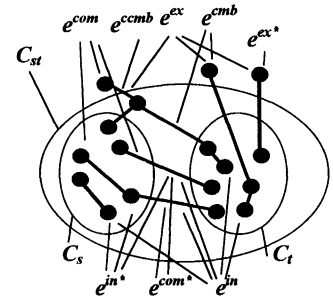


FIGURE 3 Clusters merging.

for the cluster $C_t$ also three mutually independent sets $E^{\text{in}^*}(C_t)$, $E^{\text{ex}^*}(C_t)$, $E^{\text{cmb}}(C_t)$. Now we describe an algorithm to form all sets of the new cluster $C_{st}$ with minimal expenses.

We need to receive three mutually independent sets: $E^{\text{in}^*}(C_{st})$, $E^{\text{ex}^*}(C_{st})$, and $E^{\text{cmb}}(C_{st})$ for the new cluster. First of all we have to determine some auxiliary sets of nets (Fig. 4):

$$E^{\text{com}^*}_{se,te} = \{e \in (E^{\text{ex}^*}(C_s) \cap E^{\text{ex}^*}(C_t)) \mid P(e) \subseteq P(C_{st})\};$$
$$E^{\text{ccmb}}_{se,te} = \{e \in (E^{\text{ex}^*}(C_s) \cap E^{\text{ex}^*}(C_t)\} \mid P(e)/P(C_{st}) \neq \oslash\};$$
$$E^{\text{com}^*}_{se,tc} = \{e \in (E^{\text{ex}^*}(C_s) \cap E^{\text{cmb}}(C_t)) \mid P(e) \subseteq P(C_{st})\};$$

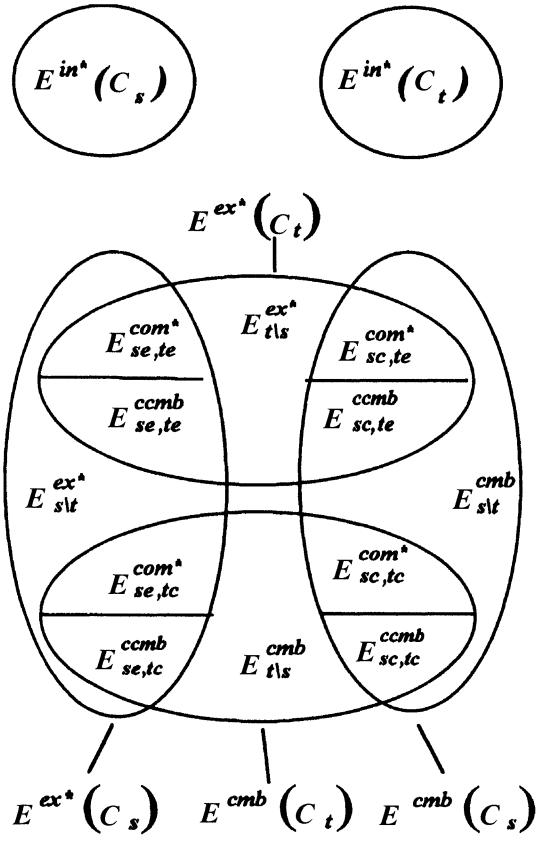


FIGURE 4 Sets of nets of the new and constituent clusters.

$$E^{\text{ccmb}}_{se,tc} = \{e \in (E^{\text{ex}^*}(C_s) \cap E^{\text{cmb}}(C_t)) \mid P(e)/P(C_{st}) \neq \oslash\};$$
$$E^{\text{com}^*}_{sc,te} = \{e \in (E^{\text{cmb}}(C_s) \cap E^{\text{ex}^*}(C_t)) \mid P(e) \subseteq P(C_{st})\};$$
$$E^{\text{ccmb}}_{sc,te} = \{e \in (E^{\text{cmb}}(C_s) \cap E^{\text{ex}^*}(C_t)) \mid P(e)/P(C_{st}) \neq \oslash\};$$
$$E^{\text{com}^*}_{sc,tc} = \{e \in (E^{\text{cmb}}(C_s) \cap E^{\text{cmb}}(C_t)) \mid P(e) \subseteq P(C_{st})\};$$
$$E^{\text{ccmb}}_{sc,tc} = \{e \in (E^{\text{cmb}}(C_s) \cap E^{\text{cmb}}(C_t)) \mid P(e)/P(C_{st}) \neq \oslash\};$$
$$E^{\text{ex}^*}_{s/t} = E^{\text{ex}^*}(C_s)/((E^{\text{ex}^*}(C_s) \cap E^{\text{ex}^*}(C_t)) \cup (E^{\text{ex}^*}(C_s) \cap E^{\text{cmb}}(C_t)));$$
$$E^{\text{cmb}}_{s/t} = E^{\text{cmb}}(C_s)/((E^{\text{cmb}}(C_s) \cap E^{\text{ex}^*}(C_t)) \cup (E^{\text{cmb}}(C_s) \cap E^{\text{cmb}}(C_t)));$$
$$E^{\text{ex}^*}_{t/s} = E^{\text{ex}^*}(C_t)/((E^{\text{ex}^*}(C_s) \cap E^{\text{ex}^*}(C_t)) \cup (E^{\text{cmb}}(C_s) \cap E^{\text{ex}^*}(C_t)));$$
$$E^{\text{cmb}}_{t/s} = E^{\text{cmb}}(C_s)/((E^{\text{ex}^*}(C_s) \cap E^{\text{cmb}}(C_t)) \cup (E^{\text{cmb}}(C_s) \cap E^{\text{cmb}}(C_t))).$$

Here $P(e)$ is a set of elements incidental to the net $e$. All of above sets are mutually independent. Mutual intersections of them are empty. Now it is easy to receive all sets of nets needed. We determine a set of pure common for both children nets:

$$E^{\text{com}^*}(C_{st}) = \{e \in (E^{\text{ex}}(C_s) \cap E^{\text{ex}}(C_t)) \mid P(e) \subseteq P(C_{st})\}.$$

All elements of these nets wholly enter into the new cluster. This set can be formed by the way:

$$E^{\text{com}^*}(C_{st}) = E^{\text{com}^*}_{se,te} \cup E^{\text{com}^*}_{se,tc} \cup E^{\text{com}^*}_{sc,te} \cup E^{\text{com}^*}_{sc,tc}.$$

We determine a set of common combinational for both children nets as follows:

$$E^{\text{ccmb}}(C_{st}) = \{e \in (E^{\text{ex}}(C_s) \cap E^{\text{ex}}(C_t)) \mid P(e)/P(C_{st}) \neq \oslash\}.$$

Pure common nets have no incidental elements outside the cluster $C_{st}$, while every common combinational net should has at least one incidental element outside the new cluster. This set can be created by the way:

$$E^{\text{ccmb}}(C_{st}) = E^{\text{ccmb}}_{se,te} \cup E^{\text{ccmb}}_{se,tc} \cup E^{\text{ccmb}}_{sc,te} \cup E^{\text{ccmb}}_{sc,tc}.$$

We can find a set of all common for both children nets of the new cluster:

$$E^{\text{com}}(C_{st}) = E^{\text{com}^*}(C_{st}) \cup E^{\text{ccmb}}(C_{st}).$$

Now it is easy to form all mutually independent sets of nets for the new cluster:

$$E^{\text{in}^*}(C_{st}) = E^{\text{in}^*}(C_s) \cup E^{\text{in}^*}(C_t) \cup E^{\text{com}^*}(C_{st});$$
$$E^{\text{ex}^*}(C_{st}) = E^{\text{ex}^*}_{s/t} \cup E^{\text{ex}^*}_{t/s};$$
$$E^{\text{cmb}}(C_{st}) = E^{\text{ccmb}}(C_{st}) \cup E^{\text{cmb}}_{s/t} \cup E^{\text{cmb}}_{t/s}.$$

It is obvious that

$$\begin{aligned} E^{\text{ccmb}}(C_{st}) &= E^{\text{cmb}}(C_{st}) \cap E^{\text{com}}(C_{st}) \\ &= E^{\text{com}}(C_{st}) / E^{\text{com}^*}(C_{st}) \\ &= E^{\text{com}}(C_{st}) \cap E^{\text{ex}}(C_{st}). \end{aligned}$$

While merging the clusters $C_s$ and $C_t$ some combinational nets of sets $E^{\text{cmb}}(C_s)$ and $E^{\text{cmb}}(C_t)$ become pure common and are not combinational in the new cluster.

We also find the sets of all internal and all external nets:

$$E^{\text{in}}(C_{st}) = E^{\text{in}^*}(C_{st}) \cup E^{\text{cmb}}(C_{st});$$
$$E^{\text{ex}}(C_{st}) = E^{\text{ex}^*}(C_{st}) \cup E^{\text{cmb}}(C_{st}).$$

### 4.4. Criteria for Clusters Merging

Relations mentioned in the previous paragraph allow to define all sets of the new clusters without any redundancy. It minimizes the calculation expenses for determination of criteria values. The choice of merging criteria is the most important for the ORT generation. Here we consider formal parameters only. We choose the criterion for clusters merging from some possible ones, for example:

$$\begin{aligned} \eta_1 &= m_c^{\text{com}}, \\ \eta_2 &= m_c^{\text{com}^*}, \\ \eta_3 &= m_c^{\text{com}^*} + \chi_1, \\ \eta_4 &= -m_c^{\text{ex}}, \\ \eta_5 &= -m_c^{\text{ex}^*}, \\ \eta_6 &= -(m_c^{\text{ex}^*} + \chi_2), \\ \eta_7 &= m_c^{\text{com}}, -m_c^{\text{ex}}, \\ \eta_8 &= m_c^{\text{com}} + m_c^{\text{ccmb}} - m_c^{\text{ex}}, \\ \eta_9 &= m_c^{\text{com}} + \chi_1 - m_c^{\text{ex}}, \\ \eta_{10} &= m_c^{\text{com}^*} - m_c^{\text{ex}^*}, \\ \eta_{11} &= m_c^{\text{com}^*} - m_c^{\text{ex}^*} + \chi_3, \\ \eta_{12} &= \xi_1 m_c^{\text{com}^*} + \xi_2 m_c^{\text{ccm}} - \xi_3 m_c^{\text{ex}^*}, \end{aligned}$$

where:

$m_c^{\text{com}} = |E^{\text{com}}(C_{st})|$ – the number of common for both children nets;
$m_c^{\text{com}^*} = |E^{\text{com}^*}(C_{st})|$ – the number of pure common for both children nets;
$m_c^{\text{ccmb}} = |E^{\text{ccmb}}(C_{st})|$ – the number of common combinational for both children nets;
$m_c^{\text{ex}} = |E^{\text{ex}}(C_{st})|$ – the number of cluster's all external nets;
$m_c^{\text{ex}^*} = |E^{\text{ex}^*}(C_{st})$ – the number of cluster's pure external nets;

$$\chi_1 = \sum_j \chi_{1j}(e_j^{\text{ccmb}}),$$
$$\chi_{1j}(e_j^{\text{ccmb}} = (\upsilon_{js} + \upsilon_{jt})/\upsilon_j;$$
$$e_j^{\text{ccmb}} \in E^{\text{ccmb}}(C_{st});$$
$$\chi_2 = \sum_j \chi_{2j}(e_j^{\text{ccmb}}),$$
$$\chi_{2j}(e_j^{\text{ccmb}}) = (\upsilon_j - \upsilon_{js} - \upsilon_{jt})/\upsilon_j;$$
$$e_j^{\text{ccmb}} \in E^{\text{ccmb}}(C_{st});$$
$$\chi_3 = \sum_j \chi_{3j}(e_j^{\text{ccmb}}),$$
$$\chi_{3j}(e_j^{\text{ccmb}}) = (2\upsilon_{js} + 2\upsilon_{jt} - \upsilon_j)/\upsilon_j;$$
$$e_j^{\text{ccmb}} \in E^{\text{ccmb}}(C_{st});$$

$v_{js}$ – the number of incidental elements of the net $e_j^{\text{ccmb}}$ in the cluster $C_s$;
$v_{jt}$ – the number of incidental elements of the net $e_j^{\text{ccmb}}$ in the cluster $C_t$;
$v_j$ – the number of all incidental elements of the net $e_j^{\text{ccmb}}$,
$\xi_1$, $\xi_2$, $\xi_3$ – weight coefficients set by the user.

All of above mentioned criteria are to be maximized. We could also consider the number of internal and all combinational nets as well as other parameters in some criteria. There are possibilities of exploiting various mixed criteria with weight coefficients for the individual nets, the element numbers, a size of clusters, time delay, *etc.*

### 4.5. The Algorithm for the Optimal Reduction Tree Generation

The main steps of the algorithm for the arbitrary $(i+1)$-th level of the Optimal Reduction Tree $T^R$ generation are as follows:

1. Consider the set $\{C_i\}$ of all clusters on level $i$. At the first level $\{C_i\} \equiv P$.
2. Form a set of adjacent clusters for every cluster of the set $\{C_i\}$.
3. Calculate the criterion values for all pairs of adjacent clusters.
4. Select one of the mentioned above criterion and create the list $L(\eta)$ of criterion values by the decreasing order.
5. Form the new set of the $(i+1)$-th level clusters. There are several possibilities. In the best case we merge only the maximum number of independent pairs with the best value of chosen criterion. It could cause a large tree height and consequently takes a lot of CPU time. The one possible way to reduce the running time is to take all independent pairs with $\varepsilon$ given decreasing of the best criterion value. The second way is to merge the first $\lambda$ of all possible independent pairs, where $\lambda(0 < \lambda \leq 1)$ is a reduction parameter. In the last case $(\lambda = 1)$ we can take the maximal number of all possible independent pairs of the list $L(\eta)$. Here a height of the ORT is at a minimum and therefore it takes the minimal CPU time but results could be worse. This case corresponds to the forced circuit reduction that might not generate good natural clusters.
6. Form the new $(i+1)$-th level of the tree $T^R$ by including a set of the new clusters, defined by merging at the previous step, and a set of rest remaining clusters from level $i$.
7. Form sets and all clusters parameters of the $(i+1)$-th level.

## 5. PACKAGING ALGORITHMS

There are a number of strategies to solve the problem. The technique depends on the solution quality, CPU time, and memory available.

For the first technique, apportionment of every partition is achieved by generating a unique ORT. The process continues until the first vertex with a violation of the constraint for the maximum number of elements is reached. The violation for the maximum number of external nets could be reached earlier; however, in some cases it is possible to reduce this number by increasing the total number of elements. This could improve the solution. By these following steps we receive from this vertex the largest subcircuit. It has the maximum number of elements non violating the constraints.

These steps are being carried out by a top-down strategy. Here it is possible to utilize a multivariable combinatorial analysis on the reduced number of elements and to consider only clusters of several high levels instead of all initial elements. It drastically reduces the computational complexity. The quality of the solution depends on the breadth and depth of analysis. For example, the initial procedure could involve examining the previous two vertices in the tree and identifying the largest cluster that is assumed to be the basis of the partition. Next, by examination of the remaining vertices of second (smaller) cluster, we try to add the maximum number of them to the first one without violating the constraints.

After the first partition is formed, it is cut from the overall circuit and for the next partition the generation of the tree begins anew. Since the size of the circuit is smaller, the running time for the step is reduced. The process repeats until all partitions are generated.

Other strategies involve forming two or more partitions from one ORT. They reduce the processing time. The tree grows until two or more vertices with violated constraints are reached. Then for each of the subcircuits the algorithm above generates the partitions. It is desirable for every subcircuit to be minimally interconnected *i.e.*, to have a minimal number of common nets. For this reason the ORT generation is to be completed. Then we compare all pairs of initially arisen vertices with violated constraints with respect to the number of common nets and choose the pair with the smallest one.

To reduce the running time it is possible to use an algorithm where a maximum number of partitions are formed from one ORT. In this case the tree is being built until all vertices have violated constraints. If a vertex has reached this point, its further growth is terminated. The rest of the vertices continue to grow until the violation of constraint will appear. Partitions are formed from each such a vertex. Then these partitions are cut away from the overall circuit. To generate the next partitions from the remaining circuit, a new ORT is built and the process is repeated until the entire circuit is divided into the individual partitions.

The minimal running time requires an algorithm similar to the previous one with an exception that the ORT is generated only once. From this tree the partitions are formed utilizing combinatorial analysis from the generated clusters (vertices of tree). There are a number of possible strategies. It is expedient to form the first and subsequent partitions from the largest cluster by adding to them smaller ones as possible.

We implemented the first strategy with some features in our experiments. The ORT is being constructed completely. The cluster exceeding and minimally deviating from constraints is searched for the tree. The following task is to eliminate the largest subcircuit that satisfies both constraints from the selected cluster. For this purpose we estimate all vertices by the ratio of external nets to the number of elements. We consider the vertices with the best criteria value. If the constraints are not met after that, the procedure will be repeated. Otherwise we try to attach the cluster with the largest number of elements and the best net criterion to the removed elements.

The purpose of optimization is to reduce the number of partitions. Some small partitions are being united to receive the number of elements that do not exceed constraints. If the number of external net constraints for these partitions is exceeded, then we realize the optimization technique to exchange some elements between partitions. The process is then terminated and the final results are received if all constraints are eliminated. Otherwise subcircuits with exceeded constraints are being split to the smallest number of subcircuits without violation by the mentioned algorithm.

## 6. PARALLELIZATION APPROACH

The described algorithms are of a parallel-serial type by the nature. It is possible to parallelize some procedures of the algorithms for this reason. Here we consider the main stages. The most computationally expensive step, and the one that must be efficiently parallelized, is the ORT generation. At every level a set of all pairs of connected clusters, for which it is necessary to calculate criterion, is divided to a number of subsets equivalently to processors number and the task is being solved in parallel. The calculation of criterion needs a lot of time and its parallelization could significantly speed up the process.

The criterion calculation of each pair could be additionally divided into four tasks according to Figure 4 and solved on separate processors. For $N$ accessible processors, a set of all connected clusters is divided by $N/4$ parts. For every pair of clusters

four processors are being assigned. These approaches for parallelization are being used in all cases, when it is necessity to build the ORT. The optimization of the initial solution could be carried out in parallel for separate pairs of partitions.

## 7. ALGORITHM COMPLEXITY

We express the total algorithm complexity to receive the partitioning system $\tilde{P}$ as

$$Q(\tilde{P}) = \sum_{j=1}^{j=k} Q(P_j),$$

where $Q(P_j)$ – the complexity for one $j$ partition, which involves the complexity of the ORT generation and the complexity of apportioning it from one tree vertex:

$$Q(P_j) = Q(T_j^R) + Q_A(P_j).$$

The apportioning complexity $Q_A(P_j)$ weakly depends on the size of full circuit and mainly depends on the size of FPGA. This means we can consider it a constant. Some approximation for the algorithm complexity of the ORT generation is equal to the average complexity of the data determination for one level multiplied by the height of the tree. The complexity for one arbitrary $i$ level of the ORT could be expressed as:

$$Q_i(T_j^R) \approx n_{ji}\tilde{\psi}_{ji}\tilde{q}_{ji},$$

where $n_{ji}$ – a number of clusters at level $i$; $\tilde{\psi}_{ji}$ – an average number of adjacent clusters for one cluster; $\tilde{q}_{ji}$ – an average complexity for determination of the criterion.

We can remark that the number of clusters is being decreased from one level to the next higher one according to the above mentioned reduction parameter $\lambda$, but the average number of adjacent clusters increases at every next level. Nevertheless the full complexity decreases from one level to another, since the total number of nets we have to consider at every next level is less than the previous because some nets become pure internal and are not taken more into account: $Q_{i+1}(T_j^R) < Q_i(T_j^R)$. If a number of elements at the first level is equal to $n_1(n_1 = n)$, then a number of elements at the second level is $n_2 = (1-\lambda/2)n$ and consequently a number of elements at the last level is $n_H = (1-\lambda/2)^{H-1}n$. Since at the last level a number is equal to 1, it justifies the expression $1 = (1-\lambda/2)^{H-1}n$. So $H-1 = -\log n/\log(1-\lambda/2)$ and for large height $H \cong -\log n/\log(1-\lambda/2)$. The full complexity of the ORT generation could be expressed by the formula:

$$Q(T_j^R) = \sum_{i=1}^{i=H-1} Q_i(T_j^R).$$

It is obvious that $Q(T_j^R) < Q_1(T_j^R)H$, *i.e.*, $Q(T_j^R) < Q_1(T_j^R)\,|\log n/\log(1-\lambda/2)\,|$. So $Q(T_j^R) = O(n\log n)$ and therefore the same estimation we have for $Q(P_j)$. Our experiments show that the complexity of ORT generation is close to linear by the number of elements. For apportioning of every next partition the size of circuit we have to consider is less that for previous partition, so $Q(P_{j+1}) < Q(P_j)$. For this reason we could conclude that $Q(\tilde{P}) < kQ(P_1)$. It takes place when for every next partition we generate the new ORT. As we have mentioned above it is possible to apportion from one ORT more than one partitions. It reduces the overall complexity.

## 8. EXPERIMENTAL RESULTS

The circuits from [1, 2] were taken for experiments. In our experiments we chosen criterion $\eta_7 = m_c^{\text{com}} - m_c^{\text{ex}}$, and at the every level of the ORT generation we merged only 20% ($\lambda = 0{,}2$) of the better independent pairs of the list $L(\eta)$. The test results (#FPGAs) from [1, 2] and for our OCR methods (initial and optimized) are shown in the Tables I and II. We used 64 CLBs and 58 IOs constraints (FPGA Xilinx XC2064) for the tests with Table I and 320 CLBs and 144 IOs (FPGA

TABLE I Packaging results for FPGAs with 64 CLB and 58 IO (Xilinx XC2064)

| Circuit | #CLBs | #Nets | #FPGAs [2] | #FPGAs [1]–SC | #FPGAs–OCR Initial | #FPGAs–OCR Optimized | Theoretic optimum |
|---|---|---|---|---|---|---|---|
| C499 | 74 | 123 | 2 | – | 3 | 2 | 2 |
| C1355 | 74 | 123 | 2 | – | 2 | 2 | 2 |
| C1908 | 147 | 238 | 3 | – | 4 | 3 | 3 |
| C2670 | 210 | 450 | 6 | – | 6 | 6 | 4 |
| C3540 | 373 | 569 | 6 | 6 | 8 | 6 | 6 |
| C5315 | 531 | 936 | 11 | 12 | 12 | 10 | 9 |
| C7552 | 611 | 1057 | 11 | 11 | 12 | 10 | 10 |
| C6288 | 833 | 1472 | 14 | 14 | 14 | 14 | 14 |

TABLE II Packaging results for FPGAs with 320 CLB and 144 IO (Xilinx XC3090)

| Circuit | #CLBs | #Nets | #FPGAs [1]–RFM | #FPGAs [1]–SC | #FPGAs–OCR Initial | #FPGAs–OCR Optimized | Theoretic optimum |
|---|---|---|---|---|---|---|---|
| s15850 | 842 | 1265 | 4 | 3 | 4 | 3 | 3 |
| s13207 | 915 | 1377 | 7 | 6 | 6 | 4 | 3 |
| s38417 | 2221 | 3216 | 12 | 10 | 10 | 8 | 7 |
| s38584 | 2904 | 3884 | 17 | 14 | 14 | 10 | 10 |

Xilinx XC3090) for the tests with Table II. As one could see from the tables the obtained results are not worse, and in 5 cases of 12 circuits they are the best among known and are optimal. If our results are not being theoretically optimal they are close to the optimal solutions and differ from them minimally, *i.e.*, only by one partition (circuits c5315, s13207 and s38417) or two partitions (circuit c2670). The CPU time for our method is greater then for SC method, but we would like to note that our programs are experimental and our main goal was to study the possibilities of employing OCR method for packaging with the aim to receive the high quality solutions.

## 9. CONCLUSION

The experimental results confirmed the utilization expediency of the OCR method for FPGA packaging. In future our goals are to develop and investigate the new strategies of the method and to work out the industrial programs for very large scale size circuits. The investigations were carried out at the Software Engineering Department of the Lviv Polytechnic State University as well as at the Computer Science and Engineering Department of the University of California at San Diego under support of Dr. C. K. Cheng, for whom the authors are very grateful.

## *References*


[1] Nan-Chi Chou, Lung-Tien Liu, Chung-Kuan Cheng, Wei-Jin Dai and Rodney Lindelof, "Circuit partitioning for huge logic emulation systems", *Proc. of 31st ACM/IEEE Design Automation Conference*, 1994.

[2] Kuznar, R., Brglez, F. and Kozminski, K., "Cost minimization of partitions into multiple devices", *Proc. of IEEE/ACM 30th Design Automation Conference*, 1993.

[3] Ober, U. and Glesner, M., "Multiway netlist partitioning onto FPGA-based board architectures", *Proc. of European Design Automation Conference with EURO-VHDL*, 1995.

[4] Bazylevych, R. P. (1981). *Decomposition and Topological Methods for Physical Design Automation of Electronic Devices*, Lviv: Vyshcha Shkola (In Russian).

[5] Fabrizio Luccio and Mariagiovanna Sami (1969). "On the decomposition of Networks in Minimally Interconnected Subnetworks", *IEEE Trans. on Circuit Theory*, **CT-16**(2).

[6] Roy L. Russo, Peter H. Oden and Peter K. Wolff (1971). "A heuristic procedure for the partitioning and mapping of computer logic graphs", *IEEE Trans. on Computers*, **C-20**(12).

[7] Jorn Garbers, Hans Jurgen Promel and Angelika Steger, "Finding Clusters in VLSI Circuits", *Proc. of IEEE/ACM Intern. Conf on Computer-Aided Design*, Santa Clara, 1990.

[8] Hagen, R. S. and Andrew Kahng, "A new approach to effective circuits clustering", *Proc. of IEEE Intern. Conf. on Computer-Aided Design*, Digest of technical papers, 1992.
[9] Jason Cong and M'Lissa Smith, "A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design", *Proc. of 30th ACM/IEEE Design Automation Conference*, 1993.
[10] Alpert, C. J. and Kahng, A. B., "A General Framework for Vertex Orderings, with Applications to Netlist Clustering", *Proc. of IEEE/ACM Intern. Conf on Computer-Aided Design*, 1994.
[11] Yousseb Saab, "Past-analysis-based clustering dramatically improves the Fiduccio-Mattheyses algorithm", *Proc. of European Design Automation Conference with EURO-VHDL '93*, 1993.
[12] Ching-Wei Yeh, Chung-Kuan Cheng and Ting-Ting Y. Lin (1995). "Circuit clustering using a stochastic flow injection method", *IEEE Trans. on Computer-Aided design of Integrated circuits and systems*, **14**(2).
[13] Alpert, C. J. and Kahng, A. B., "Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning", *Proc. of 30th ACM/IEEE Design Automation Conference*, 1993.
[14] Pak K. Chan, Martine D. F. Schlag and Jason Y. Zien, "Spectral *K*-Way ratio-cut partitioning and clustering", *Proc. of 30th ACM/IEEE Design Automation Conference*, 1993.
[15] Lars Hagen and Andrew Kahng, "Fast spectral Methods for ratio cut partitioning and clustering", *Proc. of IEEE Intern. Conf on Computer-Aided Design*, Digest of Technical Papers, 1991.
[16] Helena Krupnova, Ali Abbara, Gabriele Saucier and Michel Courtoy, "PL-Architect: An innovative prototyping environment", *Designer Track of Conf: Design, Automation and Test in Europe*, 1998.
[17] Frank M. Johannsen, "Partitioning of VLSI circuits and systems", *Proc. of 33rd ACM/IEEE Design Automation Conference*, 1996.

## Authors' Biographies

**Roman P. Bazylevych** received the Ph.D. degree in theoretical electrical engineering from the Lviv Polytechnic Institute in 1964, and the Doctor of Engineering Sciences degree from the Leningrad Electrotechnical Institute in 1984. In 1967 he joined the Lviv Polytechnic State University where he is currently the Chairman of Software Department, Full Professor. His research interests include physical design automation of microelectronic circuits (partitioning, placement and routing problems), combinatorial optimization. He authored the book "Decomposition and Topological Methods for Physical Design Automation of Electronic Devices". He is a Full member of the Shewchenko Scientific Society, Academician of the Academy of Engineering Sciences of Ukraine, Fellow member of the IEE, member of the ACM.

**Roman A. Melnyk** received the Ph.D. degree in theoretical electrical engineering from the Leningrad Electrotechnical Institute in 1984. In 1971 he joined the Lviv Polytechnic State University where he is currently the Associated Professor of Software Department. His research interests include physical design automation of microelectronic circuits (partitioning, placement and routing problems), combinatorial optimization. He authored the book "Algorithms of Hierarchical Modeling for Space and Plate Topology of VLSI". He is a member of the IEE, the ACM the Shewchenko Scientific Society.

**Oleg G. Rybak** received the master. degree in software from the Lviv Polytechnic State University in 1998. He is a Ph.D. student.