GPE: A New Representation for VLSI Floorplan Problem

Chang-Tzu Lin, De-Sheng Chen, Yi-Wen Wang

Department of Information Engineering and Computer Science

Feng Chia University, Taichung, Taiwan

Abstract

In this paper, we propose a new representation of VLSI floorplan and building block problem. The representation is the generalization of *Polish expression* [1]. By proposing a new relational operator, the representation can efficiently reuse some area that cannot be utilized if only having vertical and horizontal operators defined in Polish expression, and is able to present non-slicing structural floorplan. The experimental results show that the representation achieves promising area utilization in commonly used MCNC benchmark circuits.

1 Introduction

The problem of VLSI floorplan and placement is to determinate whether a given set of modules can be packed in a small chip. There are varieties of representing approaches in these problems but naturally classified into two categories.

One is *slicing* structure first proposed by Otten in [3] and a binary tree is used. In [1], Wong *et al*. propose a mechanism that traverses the binary tree in postorder, called Polish expression, to present a slicing floorplan. The slicing representation has some advantages such as smaller encoding cost and solution space bringing faster runtime for packing. Furthermore, it is flexible to deal with hard, pre-placed, soft, and rectilinear modules. However, in real designs optimal solutions might not be in the solution space of the slicing structure.

The other structure of representation is *nonslicing*, including *Sequence Pair* (SP) [4], *O-tree* [7], *B*-tree* [2], *Corner Block List* (CBL) [8], *Transitive Closure Graph* (TCG) [9]. However, these nonslicing representations need more evaluating runtime for packing than Polish expression.

Our main contribution is that we propose a new and easy representation for VLSI floorplan and building block problem. The representation effectively inherits the useful property of Polish expression [1] and is able to present non-slicing floorplan. We tested the approach using MCNC benchmarks and the experiments give promising results.

The paper is organized as follows. Preliminaries are given in section 2. Section 3 describes the new representation. Algorithmic operators are discussed in section 4. Section 5 gives the experimental results on MCNC benchmarks.

2 Preliminaries

Given a set of modules $\mathbf{B} = \{B_1, B_2, \ldots, B_n\}$. Each module B_i is rectangular and has fixed width and height. The coordinates of modules are the absolute coordinates of the lower left corner of the module. The objective of floorplan area optimization problem is to minimize the area of $\mathbf{B}$ subject to the constraints that no pair of modules overlaps each other. The problem has been proved to be NP-complete [6].

2.1 Polish Expression

This representation can only present slicing structure of a floorplan. Each packing is encoded by a sequence, including module name and two relational operators. As illustrated in Fig. 1, every leaf corresponds to a basic module and is marked by a module name. Every internal node of the tree is labeled by a + or a *, corresponding to a vertical or a horizontal cut respectively. We can obtain a Polish expression [1] of length $2n - 1$ with n modules in the slicing floorplan by traversing the slicing tree.

3 GPE (Generalized Polish Expression)

In this section, we introduce an encoding scheme GPE (the abbreviation for *Generalized Polish Expression*). It is the generalization of Polish expression. GPE can efficiently reuse some area that cannot be utilized anymore if only having vertical and horizontal operators defined in Polish expression, and is able to present non-slicing structural floorplan.

GPE uses a sequence of modules to reflect a physically non-slicing floorplan by proposing a new relational operator. Except the geometrically relational operators + and * in Polish expression, the third kind of operator, @ (corner relation in a packing) is introduced. As illustrated in Fig. 2(a), if there are only geometrically vertical and horizontal operators, the utilization of *dead area* is not achievable. The corner operator @, however, will arrange a module or a super-module in a corner formed by the other modules. In Fig. 2(b), the corner operator will arrange E in the corner, i.e. the dead area constructed by A and B, where A, B and E can be a module or a super-module. Through corner operator, the dead area can be effectively reused by the other modules that have not been arranged yet. Furthermore, with the proposed corner operator, the new encoding scheme GPE can express the structure of wheel, as illustrated in Fig. 2(c).

We now show how to derive a GPE from a floorplan, and derive a floorplan from a GPE, in the following subsections.

3.1 Packing to GPE

Prior to discuss the detail, we shall present some terminologies introduced in [1], so as to easily explain the transformation from packing to GPE.

A binary sequence $b_1b_2\ldots b_m$ is a *balloting sequence* if and only if, for any k, $1 \le k \le m$, the number of 0's in $b_1b_2\ldots b_k$ is less than the number of the 1's in $b_1b_2\ldots b_k$. Let σ be a function $\sigma : \{1,2,\ldots,n,+,*,@\} \rightarrow \{0, 1\}$ defined by $\sigma(i) = 1$, where $1 \le i \le n$, and $\sigma(+) = \sigma(*) = \sigma(@) = 0$.

A sequence $\Psi = \{\lambda_1,\lambda_2,\ldots,\lambda_{2n-1}\}$ of elements from $\{1,2,\ldots,n,+,*,@\}$ is a GPE of length $2n$-1 if and only if,

Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02)
1063-6404/02 $17.00 © 2002 **IEEE**

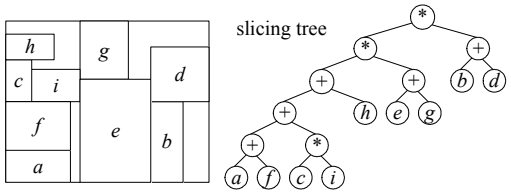


Fig. 1. Slicing tree representation and its corresponding Polish expression of a slicing floorplan.

(1) Each i appears exactly once in the sequence, where $1 \le i \le 2n\text{-}1$.

(2) $\sigma(\lambda_1)\,\sigma(\lambda_2)\ldots\sigma(\lambda_{2n-1})$ is a balloting sequence.

As illustrated in Fig. 3, the non-slicing floorplan can be represented by a GPE or a GPE-tree, where +, * and @ corresponding to the geometrically vertical, horizontal and corner relation of previous sub-packing. GPE-tree can be constructed by scanning GPE from left to right, and each module and relational operator corresponds to leaf and internal node, respectively. We can also obtain the GPE from the GPE-tree by traversing the tree in postorder. Note that the exact position of corner operator is decided by the *corner constraint*, denoted as (R, T), where R is the right boundary left to the packed module and T is the top boundary below the packed module. A corner operator may have several corner constraints, and we will choose the one based on the following consideration, (i) choose the first one that the associated module can be completely filled into the corner, or (ii) choose the first one that putting the module into the corner will not enlarge the floorplan.

3.2 GPE to Packing

According to the postorder of GPE-tree, we can obtain the GPE $\{\lambda_1,\lambda_2,\ldots,+,\ldots,*,\ldots,@,\ldots\lambda_{2n-1}\}$, where +, * and @ corresponding to the geometrically vertical, horizontal and corner relation of previous sub-packing.

To clearly explain the packing procedure, we show an example of GPE $\{a\ b + c * d\ @_{(b,c)} f + e * g\ h\ i + * @_{(f,e)}\}$ in Fig. 3. The GPE will be scanned character by character from left to right. Once a relational operator is scanned, the operator will combine the previous scanned modules or super-modules according to its property. For instance, in terms of $\{a\ b +\}$, we will arrange module b on top of module a when operator + is scanned. For $\{a\ b + c *\}$, module c is packed right of the super-module $\{a\ b +\}$. When a corner operator, $@_{(b,c)}$, is scanned, we will put module d on the corner formed by module b and module c. The rest may be deduced analogically.

4 Algorithmic Operators

We apply the following three kinds of operations to perturb a GPE-tree:

- *Complement*: Complement a chain of nonzero length.
- *Rotate*: Rotate a module.

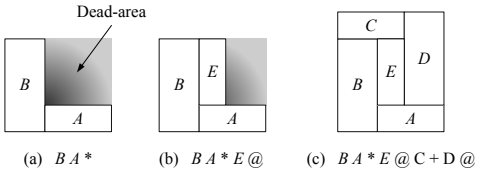


Fig. 2. Relational operators (a) dead area is no longer utilized by only horizontal or vertical operators (b) corner operator, @, can effectively reuse the dead area, and (c) a floorplan of wheel structure, where A, B, C, D and E can be a module or a super-module. Notice that the part of shadow is dead area.

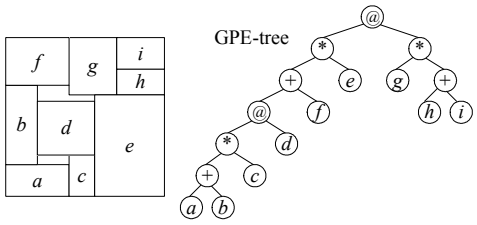


Fig. 3. A GPE-tree and an GPE correspond to its packing.

- *Swap*: Swap two leaves (modules), swap one leaf and one sub-tree (sub-floorplan), or swap two sub-trees.

Complement

A sequence $d_1 d_2 \cdots d_q$ of q operators is called a *chain* of *length* q. Note that $d_i \ne d_{i+1}$ in a GPE for all $1 \le i \le k-1$. The *complement* operation is to change the chain of originally relational operators of nonzero length to the others. Fig. 4(b) shows the resulting GPE, GPE-tree, and floorplan after complementing the chain $\{* + *\}$ to $\{+ * @_{(b,c)}\}$ shown in Fig 4(a). Notice that the corner constraint of the corner operator is decided at packing stage.

Rotate

The operation is to exchange the width and height of i-th leaf (module) in a GPE-tree. The new generated GPE-tree of the first l terms ($l = i - 1$) will be the same with the old one. Fig. 4(c) shows the resulting GPE, GPE-tree and floorplan after rotating the module f shown in Fig 4(b).

Swap

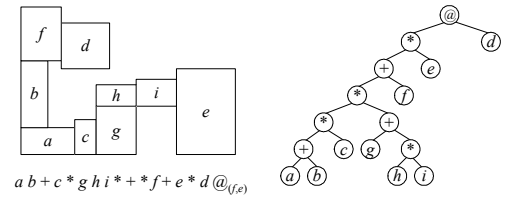
The operation is to randomly swap two leaves (modules) n_i and n_j, swap one leaf n_i and one sub-tree (sub-floorplan) s_j, or swap two sub-trees s_i and s_j in a GPE-tree. Fig. 4(d) shows the resulting GPE, GPE-tree and floorplan after swapping the module d and the subtree $\{g\ h\ i + *\}$ shown in Fig 4(c).

5 Experimental Results

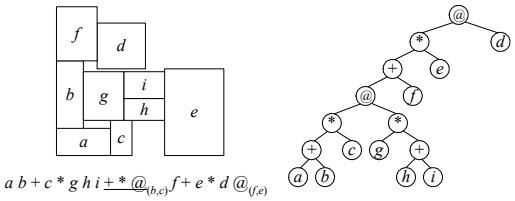
Based on the simulated annealing method [5], we implemented the GPE representation in the C++ programming language on a PC with Intel PIII 800MHz CPU and 128 MB memory. We compared GPE with FAST-SP [6], enhanced O-tree [7], B*-tree [2], CBL [8], and TCG [9], which were recently published, based on the five MCNC benchmark circuits.

TABLE I
COMPARISONS FOR RUNTIME AND AREA REQUIREMENTS AMONG POLISH EXPRESSION, O-TREE, B*-TREE, CBL, SP, AND TCG BASED ON GENETIC AND SIMULATED ANNEALING ALGORITHMS. (NA: NOT AVAILABLE)

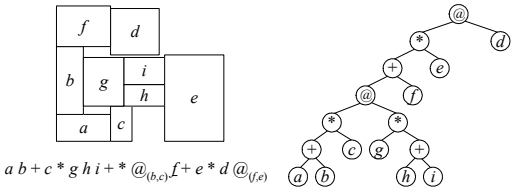
	enhanced O-tree [7]		B*-tree [2]		CBL [8]		FAST-SP [6]		TCG [9]		GPE	
	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)
apte	46.92	11	46.92	7	NA	NA	46.92	1	46.92	1	45.90	1
xerox	20.21	38	19.83	25	20.96	30	19.8	14	19.83	18	20.14	2
hp	9.16	19	8.95	55	NA	NA	8.947	6	8.95	20	9.12	2
ami33	1.24	118	1.27	3417	1.20	36	1.205	20	1.20	306	1.18	81
ami49	37.73	406	36.80	4752	38.58	65	36.5	31	36.77	434	36.45	247



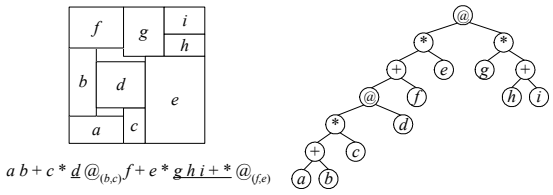
(a) initial configuration of GPE



(b) complement chain {* + *}



(c) rotate module *f*



(d) swap subtree {*g h i* + *} and module *d*

Fig. 4. Three types of perturbation. (a) The initial GPE, GPE-tree and floorplan, (b) The resulting GPE, GPE-tree and floorplan after complementing the chain {* + *}, (c) The resulting GPE, GPE-tree and floorplan after rotating the module *f*, (d) The resulting GPE, GPE-tree and floorplan after swapping the module *d* and the subtree {*g h i* + *}.

The area and runtime comparisons among FAST-SP (on Ultra1), O-tree (on a 200 MHz SUN Sparc Ultra-I workstation with 512 MB memory), enhanced O-tree (on Sun Sparc Ultra-60), B*-tree (on a 200 MHz SUN Sparc Ultra-I workstation with 256 MB memory), CBL (on Sun Sparc 20), and TCG (on a 433 MHz SUN Sparc Ultra-60 workstation with 1 GB memory) are listed in Table II. The area of a placement is measured by that of the minimum bounding box enclosing the placement. As shown in the Table II, GPE achieves promising area utilization among previous works. The final circuit layouts of ami33, ami49 are shown in Fig. 5(a) and Fig. 5(b), respectively.

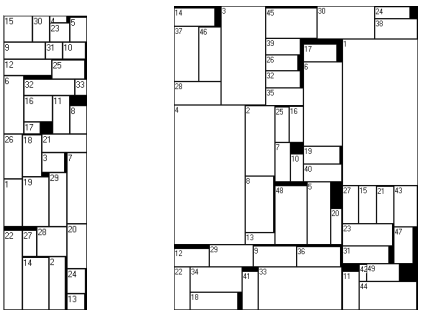


Fig. 5. Final circuit layouts of (a) ami33, (b) ami49.

References

[1] D. F. Wong, and C. L. Liu, "A New Algorithm for Floorplan Design," *Proc. DA*C, pp.101–107, 1986.

[2] Yun-Chih Chang; Yao-Wen Chang; Guang-Ming Wu; Shu-Wei Wu, "B*-trees: A New Representation for Non-slicing Floorplans," *Proc. DA*C, pp. 458 –463, 2000.

[3] R.H.J.M. Otten, "Automatic floorplan design," *Proc. DA*C, pp. 261–267, 1982.

[4] *Nakaya S., Koide T. and Wakabayashi S., "*An adaptive genetic algorithm for VLSI floorplanning based on sequence-pair," *Proc. ISCAS*, pp. 65 -68, 2000.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Scienc*e, pp.671–680, 1983.

[6] X. Tang and D. F. Wong, "FAST-SP: A Fast Algorithm for Block Placement based on Sequence Pair," *Proc. ASP-DAC*, pp. 521-526, 2001.

[7] Y. Pang, C.K. Cheng, and T. Yoshimura, "An enhanced perturbing algorithm for floorplan design using the O-tree representation," *Proc. ISP*D, pp. 168-173, 2000.

[8] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner block list: an effective and efficient topological representation of non-slicing floorplan," *Proc. ICCAD*, pp. 8-12, 2000.

[9] Jai-Ming Lin and Yao-Wen Chang, "TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans," *Proc. DAC*, pp. 764-769, 2001.