Temporal Floorplanning Using Probability Solution Space Smoothing

Shuyi Zheng[1] Sheqin Dong[1] Xianlong Hong[1]

[1] Dept. of Computer Science & Technology, Tsinghua University, Beijing [1]100084, P. R. China

Abstract

A temporal floorplanning problem arose from dynamically reconfigurable FPGAs. It can be regarded as 3D placement problem if we regard time as the third dimension. In this paper, we combine the traditional Solution Space Smoothing (SSS) technique with a new local search strategy to design a more effective optimization algorithm named Probability Solution Space Smoothing (P-SSS) to solve such a problem. Experimental results prove that it is a new efficient method for temporal floorplanning problems.

Keywords: Solution Space Smoothing, Probability, Temporal Floorplanning, 3D-subTCG.

1. Introduction

Temporal floorplanning problem comes from dynamically reconfigurable FPGAs. This kind of hardware systems usually consists of *Reconfigurable Functional Unit* (RFU) [1] which can be programmed during the execution of the program with varying configurations at different time [2]. Figure 1 is a simplified example representing a program with three parts of codes mapped to RFU operations (called *RFUOPs* or *modules*).

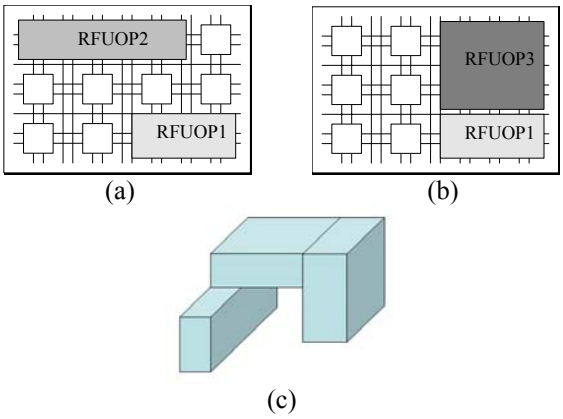

Figure 1: (a) RFU at time t1 (b) RFU at time t2
(c) Corresponding 3-D placement problem

We can see, in the RFU, not all the modules can be placed on the chip at the same time due to area limitation. Therefore, we have to schedule these modules with a proper loading sequence in order to meet the place constraint and fulfill the whole function in less time. Naturally, it becomes a 3-D placement problem as shown in Figure 1-c. And the objective is to allocate modules in the RFU to optimize both the area and execution time.

1.1. Previous Work

Several methods have already been proposed recently to deal with temporal floorplanning problem. Paper [3][4] used a branch-and-bound tree search algorithm based on component graphs to address it. Bazargan et al. dealt with two types of placement (online placement and offline placement) based on Greedy algorithm and Simulated Annealing method respectively [1][2][5]. And Paper [6] first used a topological representation named 3D-subTCG to solve such a problem.

1.2. Our Contribution

In this paper, we use solution space smoothing method to solve temporal floorplanning problem. Unlike previous works on SSS [7][8][9], we discuss the effects of smoothing in a more thorough way and consider the problem from both global and local aspects. The 3-Dimensional sub-Transitive Closure Graph (3D-subTCG) [6] is used as for 3D placement representation.

The remainder of this paper is organized as follows. Section 2 describes the principle of temporal floorplanning problem with traditional SSS. Section 3 designs a new local search algorithm for SSS heuristic. Section 4 reports the experimental results. Finally, a conclusion is given in section 5.

2. Traditional SSS for Temporal Floorplanning Problem

The traditional solution space smoothing is to gradually guide the local search from smoothed solution spaces to rougher ones. Initially, a simplified

[1] This work was supported partly by NSFC 60473126, NSFC and Hong Kong RGC joint Project 60218004 and Hi-Tech Research & Development (863) Program of China 2004AA1Z1050

placement instance with a smooth terrain surface is solved. Then a more complicated placement instance that has a rougher terrain surface is generated. It takes the solution of the previously solved placement as an initial placement and further improves the placement. Eventually, the original placement instance with the most complicated search space structure is solved.

In order to transform the original placement instance into a series of placement instances, we adopt the size changing approach. Suppose PI is the original placement instance with n modules, and PI_0 is a placement instance where all of its modules have the same size and duration. From PI_0, we slightly change the size and duration of all the modules simultaneously and produce PI_1. In the same way, from PI_1 we produce PI_2, and so on. Thus, we obtain a smoothed sequence PI_0, PI_1, PI_2, PI_3, PI_4 … where the solution space changes slightly from smoothness to toughness. The changing stops when all modules restore to their original size and duration.

Figure 2 shows the effect of traditional SSS technique. From the figure, we can see, when a solution space is smoothed, the "whole" terrain is smoothed simultaneously. We call this kind of smoothing effect as "global smoothing".

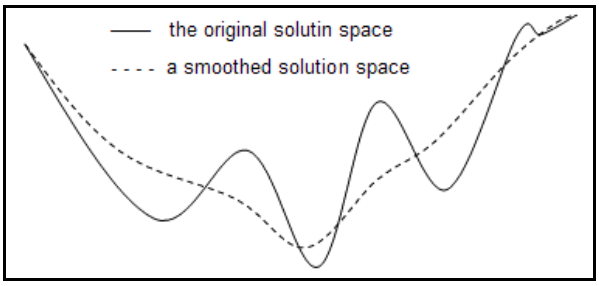

Figure 2: Global Smoothing Effect of Traditional SSS

In traditional SSS, two important functions need to be mentioned: smoothing function S(α) and schedule function f(α).

Smoothing function is used to produce a placement instance under a smoothing factor α which is controlled by schedule function. Thus, it can produce series of placement instances with different solution spaces. The rugged degree of these solution spaces increase slightly from one to another with the schedule of α until reach the original unsmoothed solution space.

In this paper, we adapt the Power-law Smoothing Function(1) proposed by [7] and employ the Exponential Schedule Function (2) decrease α from large value to 0.

$$S(\alpha)\begin{cases} w_i(\alpha) = w_{min} + (w_i - w_{min})^{\alpha+1} \\ h_i(\alpha) = h_{min} + (h_i - h_{min})^{\alpha+1} \\ t_i(\alpha) = t_{min} + (t_i - t_{min})^{\alpha+1} \end{cases} \quad (1)$$

$$w_{min} = \min\{w_1, \dots, w_i, \dots, w_n\}$$
$$h_{min} = \min\{h_1, \dots, h_i, \dots, h_n\}$$
$$t_{min} = \min\{t_1, \dots, t_i, \dots, t_n\}$$

$$F(\alpha): \quad \alpha_{k+1} = \lambda \alpha_k, \ 0<\lambda<1 \quad (2)$$

The algorithm of temporal floorplanning based on traditional SSS could be summarized as follows:

1) Let w_i, h_i and t_i be the width, height and duration of module i. Normalize all w_i, h_i, t_i so that $0 \le w_i, h_i, t_i \le 1$.
2) $\alpha := \alpha_0$; Get a starting solution x using the Greedy heuristic.
3) If a=0, stop. The current solution x is the final solution. Otherwise continue.
4) Create the placement instance according to S(α).
5) Use current solution as a starting solution; Apply the *Greedy Local Search* to produce a new current solution x.
6) Update the value of α by f(α). Go to step 3.

3. New Local Search Strategy for SSS Technique

Though the efficiency of Traditional SSS is already experimentally proved, it could not guarantee elimination of local minima. It can only reduce the rugged degree of the solution space globally and thus lessen the possibility of being trapped in the local valley.

The reason is that in above step 5, Greedy Local Search is applied, which could not jump out of local minima in any solution space. In Greedy Local Search, suppose x_j is x_i's neighborhood solution and function $\Phi^{(\alpha)}(x)$ is the cost function in the smoothed space characterized by the smoothing factor α, $\Delta\Phi_{ij}^{(\alpha)} = \Phi^{(\alpha)}(x_j) - \Phi^{(\alpha)}(x_i)$, then the probability of accepting the transition $x_i \rightarrow x_j$ is

$$A_{ij}\begin{cases} = 1 & \Delta\Phi_{ij}^{(\alpha)} \le 0; \\ = 0 & \Delta\Phi_{ij}^{(\alpha)} > 0; \end{cases} \quad (3)$$

This accepting probability make it impossible to accept worse solution, which restrict the ability to walk through the valley-hill structure of the solution space, and hence restrict its searching ability. It gives us a hint that in an already globally smoothed solution space, we also need to eliminate some local minima. We define this kind of elimination effect as local smoothing.

A local search, which is sensitive to both global and local smoothing effects, would probably lead to a better result. To make the local search have the local

smoothing ability, a smooth transition from 1 to 0 of the accepting probability should be introduced. We wish that the local search algorithm could degenerate to Greedy algorithm on the original unsmoothed solution space for convergence reason. Therefore such a function should be an approximation of (3), and converges to (3) as $\alpha \to 0$. The comparison with Simulated Annealing Algorithm reminds us of using the Metropolis function.

$$A_{ij} \begin{cases} = 1 & \Delta\Phi_{ij}^{(\alpha)} \le 0; \\ = \exp(-\Delta\Phi_{ij}^{(\alpha)}/K\alpha) & \Delta\Phi_{ij}^{(\alpha)} > 0; \end{cases} \quad (K > 0) \qquad (4)$$

Due to the difference on whether accepting worse solutions, our new algorithm using probability equation is named as "Probability SSS" (P-SSS) and the traditional algorithm, which employs Greedy local search, is called "Greedy SSS" (G-SSS).

4. Experimental Results

The following objectives are considered in our experiments.

- **Volume (the minimum bounding box of a placement):** In a temporal floorplanning, we need to consider the area of a device and the total execution time tradeoff.
- **Wirelength (the summation of half bounding box of interconnections):** Due to the special architecture of the reconfigurable device, we have to project all nodes into the same frame before computing their wirelength if those nodes are executed at different times.

Cost function Φ used in our algorithm is given by

$$\Phi = \text{Volume} + \beta * \text{Wirelength}, \qquad (5)$$

where β is user-specified weights for different problems.

To guarantee the correctness of the functions in the reconfigurable architecture, temporal precedence requirements, which describe the temporal ordering among modules, should also be satisfied in our algorithm. In order to satisfy these constraints, we do feasibility detection after each solution perturbation in the local search process and perturbations violating the constraints should be abandoned.

In our experiment, we use the MCNC benchmarks. Like paper [6], we assigned the execution times and precedence constraints to the circuit modules by ourselves. We call the new benchmark suite the 3D-MCNC benchmarks. We have implemented our P-SSS algorithm using C++ programming language on a Sun V880 system.

Using 3D-MCNC benchmarks, two groups of experiments are performed. In each experiment, $\alpha_0=6$, $\alpha_{k+1} = 0.95\times\alpha_k$, $\beta=1$, 5000 local searches for each value of α, 10 times of test for each circuit to get average results.

In the first experiment, our objective is to compare P-SSS with G-SSS as the quantity of precedence constraints varies. Figure 3 shows the average volume for 3D-ami33.

Two conclusions could be drawn from figure 3. First, the increase of the precedence constraints number leads to decrease of the quality of search. Second, combination with Metropolis algorithm makes SSS more powerful than that with Greedy algorithm as local search method.

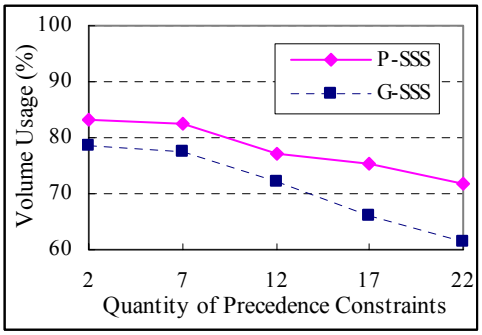

Figure 3: P-SSS vs. G-SSS: Placement volume varies with precedence constraints number for 3D-ami33.

We summarize the experimental results on all circuits of 3D-MCNC in Table 1. From these data, we can see that P-SSS algorithm improve over G-SSS algorithm in both volume and wirelength. That means P-SSS is more robust than G-SSS. Figure 4 is the best results of 3D-ami49 obtained by P-SSS (volume usage is about 84.9 %)

Table 1: P-SSS vs. G-SSS test results in volume usage (%) and wirelength (mm)

	G-SSS		P-SSS	
Circuit	Volume Usage (%)	wirelength (mm)	Volume Usage (%)	wirelength (mm)
3D-apte	92.7	354.2	94.3	332.7
3D-xerox	91.5	603.5	90.8	602.8
3D-hp	87.5	155.8	90.1	144.6
3D-ami33	77.6	76.1	82.4	68.9
3D-ami49	79.2	783.7	84.5	747.7
average	85.7	394.66	88.42	379.34

In the second experiment, using same benchmarks and same constraints, we respectively execute the Simulated Annealing approach and P-SSS algorithm based on two kind of representation: 3D-subTCG and Sequence Triplet (ST). ST is extended from the well-

known Sequence Pair (SP) [10][6]. The comparison results are listed in Table 2. Again, it shows that P-SSS could reach a better searching results with no significant performance lost.

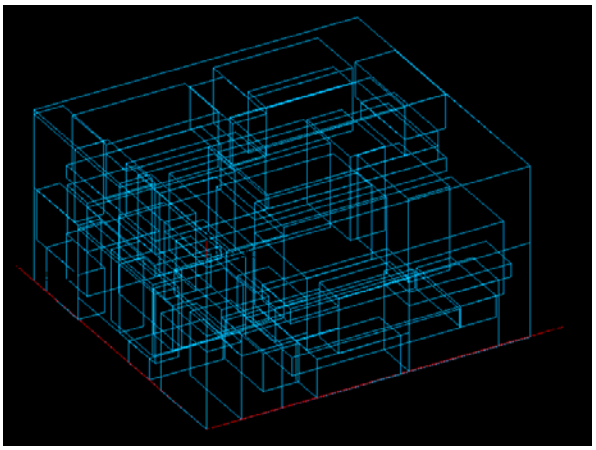

Figure 4: The best results of 3D-ami49 obtained by P-SSS (volume usage is about 84.9 %)

5. Conclusion

An efficient algorithm P-SSS for temporal floorplanning problem is proposed in this paper. By combining the advantage of traditional Greedy SSS and Simulated Annealing algorithm, P-SSS can make use of both global smoothing effect and local smoothing effect. Experimental results on 3D-MCNC benchmarks for both volume and wirelength optimization demonstrated that P-SSS algorithm is better than traditional SSS technique and Simulated Annealing based algorithm.

References

[1] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "Fast Template Placement for Reconfigurable Computing Systems," *IEEE Design & Test of Computers*, vol. 17, no. 1, pp. 68-83, 2000.

[2] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "3-D Floorplanning: Simulated Annealing and Greedy Placement Methods for Reconfigurable Computing Systems," *Design Automation for Embedded Systems – RSP'99 Special Issue*, 2000.

[3] J. Teich, S. P. Fekete, and J. Schepers, "Compile-Time Optimization of Dynamic Hardware Reconfiguration," *Proc. of PDPTA*, pp. 1097-1103, 1999.

[4] S. P. Fekete, E. Kohler, and J. Teich, "optimal FPGA Module Placement with Temporal Precedence Constraints," *Proc. of DATE*, pp. 658-665, 2001

[5] K. Bazargan and M. Sarrafzadeh, "Fast Online Placement for Reconfigurable Computing Systems," *IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 300-302, 1999.

[6] Ping-hung Yuh, Chia-Lin Yang, Yao-Wen Chang and Hsin-Lung Chen, "Temporal Floorplanning Using 3D-subTCG," *Proc. Of ASP-DAC*, 2003

[7] Jun Gu and Xiaofei Huang, "Efficient Local Search with Search Space Smoothing: A Case Study of the Traveling Salesman Problem (TSP)," *IEEE Trans. On Systems. Man. And Cybernetics*, vol. 24, no. 5, pp. 728-735, 1994

[8] Johannes Schneider, Markus Dankesreiter, Werner Fettes, Ingo Morgenstern, Martin Schmid and Johannes Maria Singer, "Search-space smoothing for combinatorial optimization problems," *Physica* A 243, pp. 77-112, 1997

[9] Sheqin Dong, Xianlong Hong, Xin Qi, Ruijie Wang, Song Chen and Jun Gu, "VLSI Module Placement with Pre-placed Modules and Considering Congestion Using Solution Space Smoothing", *ACM/IEEE ASP-DAC*, Japan, 2003, pp.741-744.

[10] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-Packing Based Module Placement," *Proc. Of ICCAD*, pp. 472-479, 1995.

Table 2: Comparison with the solution quality and run-time

		ST						3D-subTCG					
		Volume Usage (%)		Wirelength (mm)		Run Time (sec)		Volume Usage (%)		Wirelength (mm)		Run Time (sec)	
3D-ami33	SA	51.2		87.3		378		74.9		82.1		131	
	P-SSS	63.7		86.6		227		82.4		68.9		119	
	Improve	12.5	24.4%	0.7	0.8%	151	39.4%	7.5	10%	13.2	16.1%	12	9.1%
3D-ami49	SA	53.7		980.2		992		78.1		823.2		590	
	P-SSS	64.9		913.4		714		84.5		747.7		537	
	Improve	11.2	20.9%	66.8	6.8%	278	28%	6.4	8.1%	75.5	9.2%	53	9%