A CONGESTION DRIVEN PLACEMENT ALGORITHM FOR FPGA SYNTHESIS

Yue Zhuo, Hao Li and Saraju P. Mohanty

Department of Computer Science and Engineering
University of North Texas
P.O.Box 311366, Denton, Texas 76203
email: {yz0051, hli, smohanty}@unt.edu

ABSTRACT

We introduce a new congestion driven placement algorithm for FPGAs in which the overlapping effect of bounding boxes is taken into consideration. Experimental results show that compared with the *linear congestion* method [1] used in the state-of-the-art FPGA place and route package VPR [2], our algorithm achieves channel width reduction on 70% of the 20 largest MCNC benchmark circuits (10.1% on average) while keeping the channel width of the remaining 30% benchmarks unchanged. A distinct feature of our algorithm is that the critical path delay is not elongated on average, and in most cases reduced.

1. INTRODUCTION

Field programmable gate arrays (FPGAs) have gained rapid commercial acceptance recently, but the area efficiency and performance are still their drawbacks and are needed to be optimized. Area of an FPGA includes routing area and logic area. Unlike an application specific integrated circuit (ASIC), most of the chip area of an FPGA (often 80%-90% [3]) is dedicated to the routing resources. Thus, reducing routing tracks in an FPGA will effectively reduce the whole FPGA area. The research proposed in this paper is a contribution in this direction.

To improve an FPGA's area-efficiency, Marquardt *et al.* [4] tried to simultaneously examine both the area-efficiency and the speed of FPGAs using different logic cluster sizes. In [5], the authors claimed that having wires of 4 to 8 logic blocks long for an FPGA will optimize speed and density. Routability is an issue closely related to FPGA area efficiency. RPack [6] improves the routability by incorporating several routability factors into the packing algorithm. It can be also integrated into timing-based T-VPack to implement a new clustering algorithm called T-RPack. In [7], Schlag *et al.* presented a program, called *Rmap*, which provided the capability to trade off between routability and compactness of a design. Parthasarathy *et al.* [8] utilized Rent's rule to analyze and estimate the routability of a placement.

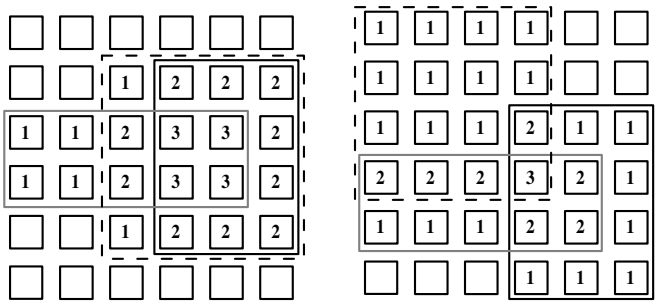
Due to increasing complexities in current VLSI designs, more and more attention has been paid to circuit congestion. In [9], the authors presented an indirect congestion minimization method by correlating wirelength with congestion. The work in [10] tried to reduce congestion and wirelength by utilizing a fast global router and hierarchical area density control. In addition to these approaches, there is a *non-linear congestion* method implemented in Versatile Place and Route (VPR) which adjusts the placement according to the given maximum channel width. Compared to the linear method, on a 4 x 4 grid, the non-linear approach takes 5X CPU time to find a placement [1]. Unlike the linear approach, this method cannot generate the final placement in one pass of the simulated annealing process. Instead it uses a binary search method. If it takes 6 attempts to find the minimum channel width, placement and routing processes will be called 6 times as well. Assume the linear congestion method needs P_l time in placement and R_l time in routing. The total time for the non-linear one is about $30\,P_l + R_l$, which is apparently more time consuming. However, it only achieves 1% reduction in the channel width [1].

Our contribution in this paper is to present a congestion driven placement approach which considers the overlapping among various nets. By introducing a cost function which quantifies this mutual influence, we place the circuit more evenly and hence reduce the routing tracks. Moreover, our approach does not cause any increase in delay (or decrease in performance).

The rest of this paper is organized as follows: In section 2, we review the placement algorithms used by VPR and then propose our placement algorithm. In section 3, we present the experimental results. In section 4, we draw the conclusions and discuss future research.

2. OUR PROPOSED CONGESTION DRIVEN APPROACH

T-VPlace [11] is the simulated annealing based placement algorithm used in VPR, the most widely used academic place-



(a) (b)

Fig. 1: A Circuit with Three Overlapping Bounding Boxes: a) Placement may result in a congested routing. b) Placement leads to a balanced routing. Our goal is to achieve (b).

ment, routing, and architecture exploration system for FPGAs. T-VPlace considers two types of costs, namely timing cost and bounding box cost. The following function [1] is used to estimate the bounding box cost.

$$WiringCost = \sum_{i=1}^{N_{nets}} q[i](bb_x(i) + bb_y(i)), \qquad (1)$$

where N_{nets} is the total number of nets in the circuit. For each net, i, $bb_x(i)$ is its horizontal span, and $bb_y(i)$ is its vertical span. The $q(i)$ factor compensates for the fact that the bounding box wire length model underestimates the wiring necessary to connect nets with more than three terminals [12].

It is clear from the above analysis that the mutual interactions of different nets are not taken into account by VPR's linear congestion method. In the following subsections, we discuss our algorithm which overcomes this drawback.

2.1. Congestion Coefficient

To reduce the routing channel width, a placement algorithm has to pay attention to both the resource consumed by each net, and the interaction (congestion) among different nets. For example, if all nets are restricted to a relatively small fraction of area on the chip, the routing track demand will probably be very high in this region. Although CLBs in other regions on the chip may be easily routed with a small channel width, the overall channel width is determined by the channel that uses the maximum number of tracks if all channels are of the same width.

In our proposed algorithm, we formulated a new bounding box cost function. The final bounding box cost is now computed by multiplying the previous $WiringCost$ with our congestion coefficient CC (defined below).

We first introduce the congestion model used in our algorithm. Assume a circuit consisting of 3 nets is to be placed. An intermediate placement during the simulated annealing process is shown in Figure 1(a). The 3 bounding boxes are shown by different rectangles. The number in each CLB indicates how many bounding boxes are covering this CLB at this moment. A CLB without a label is not covered by any bounding box. For example, a CLB with label 2 means it is covered by the bounding boxes of two nets. Since every net will probably need some routing tracks around the CLBs it covers, the regions covered by more bounding boxes would require more routing resources. In Figure 1(a), the CLBs with label 3 are very likely to be the bottleneck to reduce channel width. Another placement shown in Figure 1(b) provides a better solution even though the dimension of each bounding box remains unchanged.

The following formula is used in our proposed algorithm to compute the congestion coefficient factor for a placement,

$$CC = \left(\frac{\sum_{i,j} U_{i,j}^2}{nx \cdot ny} \Big/ \left(\frac{\sum_{i,j} U_{i,j}}{nx \cdot ny} \right)^2 \right)^k, \quad 1 \le i \le nx, 1 \le j \le ny, \qquad (2)$$

where $U_{i,j}$ is the number of bounding boxes covering $CLB_{i,j}$, k is a small positive integer (explained in section 2.2), and the whole chip consists of nx by ny CLBs. As we know, for n positive numbers $a_1, a_2, \ldots, a_n$, the following inequality $\sum_{i=1}^{n} a_i^2/n \ge (\sum_{i=1}^{n} a_i/n)^2$ is always true. So, CC is always equal to or greater than 1. In Equation (2), if $\sum U_{i,j}$ remains constant, CC is determined only by $\sum U_{i,j}^2$. A balanced set of $U_{i,j}$ will give a small CC. So the value of CC shows how congested the whole placement is expected to be. When it is close to 1, the placement is balanced. If it is much greater than 1, the placement is considered congested.

2.2. Exponent Attenuation Method

During the process of simulated annealing in VPR, the maximum radius to swap two blocks decreases gradually. In the beginning, the radius equals the entire chip size, called $rMax$. For example, if there are N x N CLBs on a chip, the initial radius will be N. As the temperature cools down, the swap radius becomes smaller and finally reaches 1. We can consider the earlier period in simulated annealing as a coarse-grain or global adjustment and the later period as a fine-grain or local adjustment. It is natural to think that a global placement should be paid more attention to.

Based on this assumption, we decide to assign more weight to CC during the earlier periods of simulated annealing. In other words, we will adjust the exponent factor k in Equation (2) according to the swapping radius. When the swapping radius is large, k is set to a large value. When the radius is small, k is set to a small value. Although the swapping

radius is changed dozens of times in a typical simulated annealing process, we only use 3 to 4 levels of integer exponents k in our current implementation for simplicity.

Algorithm 1: Computing Bounding Box Cost

```
procedure compBBCost(r)
  clearBlkUsage(U)
  cost ← 0
  for n ← 0 to num_nets
    getBoundingBox(bb[n])
    for i ← bb[n].xMin to bb[n].xMax
      for j ← bb[n].yMin to bb[n].yMax
        U[i,j] ← U[i,j] + 1
    cost ← cost + getNetCost(n)
  congestion ← congestionFunc(r, U)
  return (cost * congestion)

procedure congestionFunc(r, U)
  sum ← 0
  sos ← 0
  levels ← maxExp − minExp + 1
  for i ← 1 to nx
    for j ← 1 to ny
      sos ← sos + U[i,j] * U[i,j]
      sum ← sum + U[i,j]
  base ← sos * nx * ny/sum^2
  k ← minExp + (int)((r − 1) * levels/rMax)
  return (base^k)
```

Different exponent attenuation schemes are applied to different circuits accordingly. The criterion to choose a scheme is the product of the number of nets in each circuit and $rMax$, which we consider as circuit complexity. If a circuit is simple, small exponents are used. If a circuit is complicated, large exponents are used. Take the benchmark circuit "misex3.net" for example, the product value is 53618, which is computed with the architecture file, "vpr422_arch.txt" [13]. We used the attenuation scheme of (3, 2, 1) in which the exponent k is initialized to 3. When the swapping radius is below $\frac{2}{3}rMax$, k becomes 2. Finally, when the swapping radius is below $\frac{1}{3}rMax$, k is set to 1. Function "congestionFunc" in Algorithm 1 contains the details of computing k. In this case, $maxExp$ is 3 and $minExp$ is 1. The complete table of $maxExp$ and $minExp$ is given in Table 1.

Algorithm 1 is the pseudocode of our proposed algorithm. Function "compBBCost()" calculates the final cost based on the swapping radius r. Function "getBoundingBox()" computes the bounding box for each net and stores its dimension and location in $bb[n]$. Function "getNetCost()" obtains the original bounding box cost computed by VPR, and function "congestionFunc()" calculates the congestion intensity.

3. EXPERIMENTAL RESULTS

We have implemented and integrated our proposed algorithm in the framework of VPR. The experiments were carried out on a Pentium4 2.8GHz PC with 1GB memory running the Centos Linux system. The MCNC benchmark circuits and the VPR source code (version 4.3) were downloaded from [13].

Table 1 shows the results of VPR, VPRb and our algorithm. "CP" stands for "Critical Path Delay" and "CW" stands for "Channel Width". The results of VPR are obtained by running it with all default settings. Note, by default, VPR uses the minimum possible number of routing tracks. The columns labeled "VPRb" show the results when VPR is run with the option "-place_algorithm bounding_box". In this mode, VPR completely turns off the timing-driven feature of its placer and focuses solely on minimizing the overall wirelength. The "Ratio" is the ratio of other results to VPR's. It is evident that, compared with VPR, our algorithm reduces the channel width by 7.1% and reduces the average critical path delay by 0.7%. Circuits placed by our algorithm are routed by VPR's router with default parameters. Note that the routing channel width required by our algorithm never exceeds VPR's which justifies its stability. We can also see that the number of routing tracks required by VPRb is only 1% less than ours, but its critical path delay is 24.5% more than ours. Typically, reducing routing channel width will result in degradation in timing. Thus, it is significant that our algorithm is capable of reducing routing resources without affecting the circuit performance at all.

Most research works minimizing routing resources deal with ASICs, and only a few focus on FPGAs [7, 8]. We compare our work with VPR because other research on FPGAs seldom provide enough statistics on their timing performance. To our best knowledge, the only work which can reduce routing channel width without paying any delay penalty is T-RPack [6], but this algorithm works in the phase of packing LUTs into clusters. Yet, we can still compare our results with T-RPack's. T-RPack works in two modes: in mode 1 (with population factor), it can reduce channel width by 5.3% compared to T-VPack [4], but it can not achieve any performance improvement; in mode 2 (without depopulation), it achieves channel width reduction of 2.67% and delay reduction of 5%. If we consider T-VPack and VPR as a whole package [14], it is very clear that our results outperform T-RPack on both criteria if it is run in mode 1. If T-RPack is run in mode 2, the channel width achieved by our algorithm is still 4.4% lower than T-RPack.

4. CONCLUSION AND FUTURE WORK

In this paper, we presented a congestion driven placement algorithm for FPGAs. The unique characteristic of our ap-

Table 1: Experiment Results: VPR, VPRb and Our Algorithm

Circuit	VPR		VPRb				Ours					
	CP	CW	CP	Ratio	CW	Ratio	CP	Ratio	CW	Ratio	minExp	maxExp
tseng	55.62	8	71.56	1.2866	7	0.875	53.65	0.9646	7	0.875	1	2
apex4	93.25	14	131	1.4048	13	0.9286	92.75	0.9946	13	0.9286	1	3
misex3	95.74	12	107.7	1.1249	11	0.9167	82	0.8565	11	0.9167	1	3
dsip	70.79	7	82.14	1.1603	6	0.8571	66.62	0.9411	6	0.8571	1	4
ex1010	195.3	12	206.1	1.0553	10	0.8333	182.6	0.935	11	0.9167	2	4
clma	228.4	14	243.7	1.067	13	0.9286	213.7	0.9356	13	0.9286	3	5
diffeq	101.5	8	93.38	0.92	7	0.875	63.44	0.625	8	1	1	3
spla	203.1	15	181	0.8912	14	0.9333	158.9	0.7824	15	1	1	4
seq	95.72	12	109.7	1.1461	12	1	81.65	0.853	12	1	1	3
elliptic	137.2	12	187.1	1.3637	11	0.9167	126.8	0.9242	11	0.9167	1	4
pdc	193.5	19	211.3	1.092	17	0.8947	198.9	1.028	17	0.8947	2	4
frisc	135	14	176	1.3037	12	0.8571	142.7	1.057	13	0.9286	1	4
bigkey	78.56	7	88.7	1.1291	7	1	99.46	1.266	6	0.8571	1	4
des	121.2	8	114.7	0.9464	8	1	124.2	1.0248	7	0.875	1	4
alu4	82.1	11	121.7	1.4823	10	0.9091	98.93	1.205	10	0.9091	1	3
apex2	90.02	12	134.8	1.4974	11	0.9167	111.8	1.242	11	0.9167	1	3
ex5p	84.06	15	129.4	1.5394	13	0.8667	87.1	1.0362	13	0.8667	1	1
s298	135.7	8	204	1.5033	8	1	148.5	1.0943	8	1	1	4
s38417	103.3	8	156.9	1.5189	7	0.874	112.3	1.087	8	1	2	5
s38584.1	97.92	8	126.7	1.2939	8	1	98.21	1.003	8	1	2	5
Ave				1.2363		0.9192		0.9928		0.9294		

proach is its ability to reduce the routing channel width without elongating the critical path delay. In our current work, the congestion is treated globally. However, it may become more precise if each bounding box has its own congestion value. We would like to explore this idea in our future research.

5. REFERENCES

[1] V. Betz and J. Rose and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[2] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," in *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, 1997, pp. 213–222.

[3] A. DeHon, "Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization)," in *Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays*. ACM Press, 1999, pp. 69–78.

[4] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," in *Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays*, 1999, pp. 37–46.

[5] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density," in *Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays*, 1999, pp. 59–68.

[6] E. Bozorgzadeh, S. Ogrenci-Memik, X. Yang, and M. Sarrafzadeh, "Routability-Driven Packing: Metrics and Algorithms for Cluster-Based FPGAs," *Journal of Circuits Systems and Computers*, vol. 13, no. 1, pp. 77–100, 2004.

[7] M. Schlag, J. Kong, and P. K. Chan, "Routability-Driven Technology Mapping for Lookup Table-Based FPGA's," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 1, pp. 13–26, Jan. 1994.

[8] G. Parthasarathy, M. Marek-Sadowska, A. Mukherjee, and A. Singh, "Interconnect Complexity-Aware FPGA Placement Using Rent's Rule," in *Proceedings of the 2001 International Workshop on System-Level Interconnect Prediction*. ACM Press, 2001, pp. 115–121.

[9] M. Wang and M. Sarrafzadeh, "On the Behavior of Congestion Minimization during Placement," in *Proceedings of the 1999 International Symposium on Physical Design*. ACM Press, 1999, pp. 145–150.

[10] C. Chang, J. Cong, and Z. Pan, "Physical Hierarchy Generation with Routing Congestion Control," in *Proceedings of the 2002 International Symposium on Physical Design*. ACM Press, 2002, pp. 36–41.

[11] A. Marquardt, V. Betz, and J. Rose, "Timing-Driven Placement for FPGAs," in *Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays*. ACM Press, 2000, pp. 203–213.

[12] C. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling," in *Proceedings of the 1994 IEEE/ACM International Conference on Computer-Aided Design*, 1994, pp. 690–695.

[13] http://www.eecg.toronto.edu/˜vaughn/challenge/challenge.html.

[14] http://www.eecg.toronto.edu/ vaughn/vpr/vpr.html.