Wire Length Prediction in Constraint Driven Placement

Qinghua Liu, Bo Hu
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-5678
{qinghual, hb}@ece.ucsb.edu

Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-2721
mms@ece.ucsb.edu

Abstract

Experiments show that lengths of individual wires are different for different placement algorithms. To achieve accurate wire length prediction, some knowledge of a placer's details is necessary. We postulate that wire length prediction should be coupled with placement flow to obtain accurate results. In this paper, we embed wire length prediction into our constraint-driven placer, developed in Fast Placer Implementation (FPI) framework [15]. We predict individual wire lengths during the clustering step. The predicted wire lengths act as constraints for the simulated annealing refinement stage, which guides the placement towards a solution fulfilling the predictions. Experimental results show that our wire length prediction process yields accurate results without quality loss at a small cost of placement effort. This is the first time that constraints have been used to guide placement and thus increase the accuracy of wire length prediction.

Categories and Subject Descriptors

J.6 Computer-Aided Engineering-Computer-aided Design (CAD)

General Terms

Algorithms

Keywords

Clustering, Wire length prediction

1. Introduction

With the trend toward decrease of feature sizes, the contribution of interconnects to critical path delay increases and becomes comparable to that of active devices. Interconnect estimation is now of critical importance because early wire optimization is needed in the design flow to achieve timing closure. Whether a design can achieve required objectives, such as performance, routability, and power, is largely determined by the placer's quality and its ability to conform to the predicted interconnect lengths. Let's consider timing-driven placement in the context of interconnect length prediction. Timing driven placement algorithms can be grouped into two major categories: path-based [12][17][22] and net-based. The path-based algorithms, try to control critical path delays directly, are usually computationally expensive because of the exponential number of such paths. In net-based algorithms, timing constraints are first translated into physical requirements such as net weights [2][7] or delay budgets[18][21]. Net weights cannot control placement results well since it is hard to force wire lengths by assigning net weights. Constraints in a form of delay budgets or individual wire length predictions allow for more reasonable constraint distribution.

During the past two decades, different approaches have been proposed. The early experimental work analyzing wiring requirements was performed by Rent in the early 1960's. Donath [6] and Feuer [8] showed that the average interconnection wire length can be calculated from the Rent's exponent. Pedram and Preas [19][20] presented an interconnection length and layout area predictor for standard cell layouts. Their approach considers all possible distributions of pins on rows. Hamada and Cheng [11] proposed a topological analysis technique of networks. They characterized the local network structure by a growing sequence of multilevel neighborhoods. They found the wire length distribution by solving equations for the probability density of multilevel neighborhoods. Caldwell and Kahng [4] proposed a net bounding box estimation based on a Uniform Pin Distribution Model. Heineken and Maly [14] proposed an empirical interconnect length model to predict the distribution parameters of net lengths. The model takes as input a standard cell netlist and provides as output the estimates of mean and variance of net lengths on a net-by-net basis. Bodapati and Najm [1] estimated individual wire lengths based on characterization of typical designs.

This paper addresses the problem of individual wire length prediction. Experiments show that individual wire lengths depend on placement algorithms. Wire length predictions accurate for one placement flow may be inaccurate for another. So wire length predictions should be coupled with placement flows. Based on this observation, we embed wire length prediction into the constraint-driven placer that we developed in the Fast Placer Implementation (FPI) framework [15]. Our placement flow includes both structural and physical levels. In the structural level, we reduce the size of an initial netlist by clustering. In the physical level, we apply global placement optimization and simulated-annealing-based refinement. We make wire length predictions based only on clustering information. That is to say, wire length prediction is performed at the structural level of the placement flow, and is referred to as *in-placement*

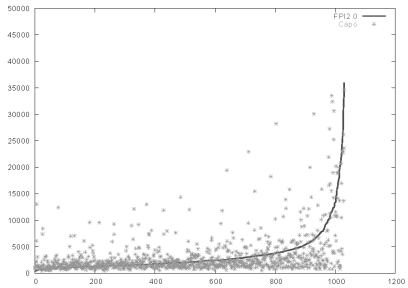
SLIP'03, April 5-6, 2003, Monterey, California, USA

prediction. Experimental results show that wire-length predictions made during the clustering stage of the placement flow can be satisfied in the final placement without quality losses.

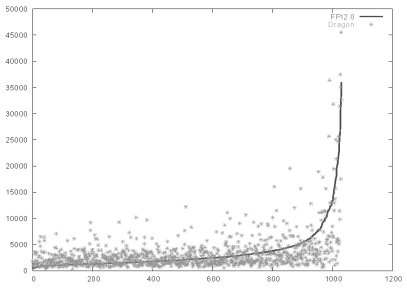
The rest of this paper is organized as follows. In Section 2 we discuss relationship between the individual wire lengths and the placement algorithm, and clarify our motivation to couple wire length prediction with a placement flow. In Section 3 we discuss how appropriate constraints can help prediction. In Section 4 we explain the basic flow of our constraint-driven placement with wire length prediction. In Section 5 we discuss mutual-contraction clustering and constraint generation. In Section 6 we introduce our constraint driven placement. We present experimental results in Section 7. Section 8 concludes the paper.

2. Relationship between individual wire length and placement

Different algorithms may produce different individual wire lengths in their final placements. Take for example the recent placement techniques based on recursive multi-level partitions [3][24]. In those algorithms, clustering strategy plays a key role in determining the individual wire lengths. In some clustering metrics, a particular net may be totally absorbed into one cluster and become an intra-cluster net. The same net may become an inter-cluster net if we apply another clustering metric. So the length of this net could be significantly different in the final placement results.



(a)FPI2.0 VS Capo



(b)FPI2.0 VS Dragon

Figure 1: Net lengths obtained by different placers

Experimental results in figure 1 illustrate this point. In this figure, we compare individual wire lengths for 3 pin nets in different placement flows. The x-axis corresponds to the net id's, and the y-axis to the net lengths. In figure 1(a) we compare the results obtained from the FPI2.0 [15] and Capo [3], and in figure 1(b) we compare FPI2.0 and Dragon [24]. For easy understanding, suppose that FPI2.0 produces "accurate" net lengths and the other algorithms produce "erroneous" lengths. We sort "accurate" lengths and connect them into a solid line as shown in the figure. The "erroneous" lengths are represented by points around the solid line. We define the error of a net's i length as

$$error(i) = \frac{|actual(i) - accurate(i)|}{accurate(i)} \quad \text{(EQ1)}$$

where *actual(i)* is the actual net length produced by the "erroneous" placer (Dragon or Capo) and *accurate(i)* is the net length from the "accurate" placer (FPI2.0). We calculate the average length error(*ave_error*) and length error standard deviation(*error_sd*) for Dragon and Capo, and list them in Table 1.

TABLE 1. Length error for Dragon and Capo

	Dragon	Capo
ave_error	0.72	0.59
error_sd	0.90	1.28

From this experiment we observe that three different placement algorithms result in significantly different lengths of individual wires.

Since individual wire length is a function of the placement algorithm, accurate predictions are possible only if the prediction process is coupled with a placement flow. In our constraint-driven placement flow, we predict individual wire length based on clustering information.

3. Constraints help prediction

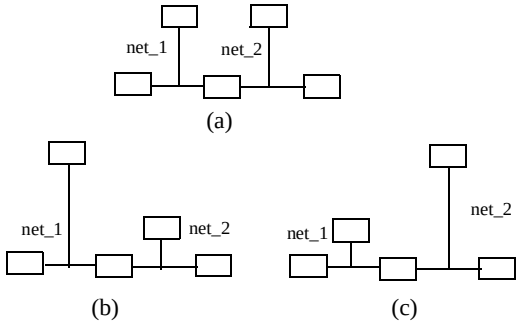


Figure 2: Three placement solutions with the same TWL

A purely total wire length (TWL) driven placement does not take individual wire lengths into account. For example, it cannot distinguish the three placement solutions shown in figure 2, since each of them yields the same total wire length. But if we apply wire length constraints on individual nets, such constraints will guide placer to a solution which will satisfy them. For example, in the case above if we put a larger length-constraint on net_1 and a smaller length-constraint on net_2, a placer will most likely produce a solution similar to that in figure 2(b).

We will view the wire length prediction as a constraint generation process in the following section.

4. Constraint driven placement with wire length prediction

We embed the wire length prediction into our placement flow based on FPI [15] framework. The basic flow is shown in Figure 3. Our constraint-driven placement consists of 3 stages. In stage 1, we reduce the size of the initial large scale netlist by clustering. Several clustering techniques have been proposed in the past years. In this paper, we use the mutual-contraction-based clustering algorithm [16], which has proved to produce better global placement results compared with algorithms based on deterministic connectivity [13] and edge-separability [5]. After clustering information is available, we introduce the wire length prediction step. Since the relative location of cells in the same cluster can be maintained after global placement of a reduced netlist, we can easily control the wire length of those nets totally absorbed in clusters. We estimate individual wire lengths of such nets. The detailed description of the clustering algorithm and the wire length prediction process is detailed in Section 5. In stage 2 of the constraint-driven placement

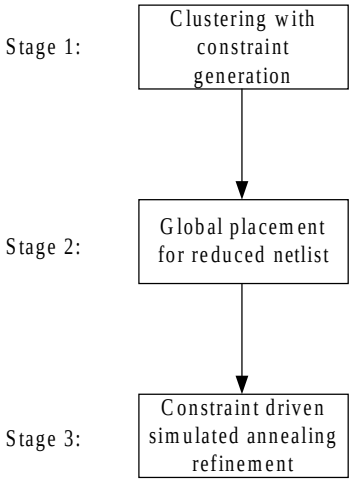


Figure 3: Constraint-driven placement flow

flow, global placement optimization is applied on the reduced netlist. We use Capo8.5 [26] at this stage. Some bad decisions made during the clustering phase may cause placement quality losses, so we need a strategy to recover from such effects. That is the purpose of stage 3, where we refine placement by applying a simulated annealing technique. Unfortunately, such a recovery process may invalidate the wire length predictions made in stage 1. By using the predicted wire lengths as constraints in this stage, our simulated annealing process improves placement quality and guides the placement towards the predictions. The detailed process of constraint driven placement is discussed in section 6.

We distinguish two types of wire length predictions associated with a placement: *a priori prediction* and *a posteriori prediction*. The *a priori prediction* determines wire lengths of a layout design in advance, before placement. The *a posteriori prediction* occurs when we are given a fixed placement and want to estimate the post-routing wire lengths. This is of value whenever routing requires a significant amount of CPU time. In order to get more accurate prediction results, our wire length prediction is embedded into the placement flow when clustering information is available. This strategy is different from the existing prediction types. We label this kind an *in-placement prediction*.

5. Mutual contraction-based clustering with wire length prediction

5.1 Mutual contraction

In [16] we introduced the mutual contraction as a metric to evaluate proximity of connected elements in a netlist. Compared with connectivity[13] and edge separability[5], our metric is capable of predicting short connections more accurately.

To calculate the mutual contraction of a connection(u,v), we first introduce the relative weight of connections incident to a node u as stated in equation 2.

$$w_r(u,v) = \frac{w(u,v)}{\sum_x w(u,x)} \qquad \text{(EQ2)}$$

In (EQ2), $w(u,x)$ is a weight of the connection (u,x). The summation is carried over all nodes x adjacent to u.

Then we use a product of $w_r(u,v)$ and $w_r(v,u)$ to measure the mutual contraction of connection(u,v), as stated in equation 3.

$$c_p(x,y) = w_r(x,y)w_r(y,x) \qquad \text{(EQ3)}$$

We have established a good correlation between the contraction of a connection(u,v), measured by the mutual contraction metric, and the length of (u,v). The connections with strong contraction most likely will end up having short lengths. This observation provides a basis for the deterministic clustering algorithm detailed in the next section.

5.2 Mutual-contraction-based clustering

We apply a pair-wise clustering strategy in stage 1 of our constraint-driven placement flow. The cluster size constraints are given at the beginning of the process. A priority queue is maintained for all connections in the hypergraph. A connection with the largest contraction is on the top of the queue. We pick consecutive connections (x,y) from the queue to see whether grouping x and y violates cluster size constraints. If it does, we discard this connection and continue the process. If grouping x and y does not violate the cluster size constraints, we create a new node z which represents both x and y, and update the connectivity information for z. Meanwhile, we also update incrementally the priority queue. Note that we set cluster size constraints according to average cell area. For example, if we set a cluster size constraint to 5, the overall area of cells inside one cluster should be smaller than 5 times the average cell area. We also set a target cluster number. The clustering process terminates when no more clustering is possible due to the size constraints, or when the cluster number limit has been reached.

5.3 Wire length prediction

Wire length prediction is performed during the clustering process. After each grouping operation as discussed in section 5.2., we check if there are nets totally absorbed in the newly created cluster. If there are, for example, net i in Figure 4, we estimate the length of such a net i as follows:

$$Boundary(i) = \frac{\sum_j W_j + \sum_j H_j}{2} \quad \text{(EQ4)}$$

In (EQ4) Wj and Hj are the width and height of the cells in the cluster. We use the boundaries to predict the actual wire lengths. We do not make wire length predictions on the inter-cluster nets which are difficult to control during the global placement stage.

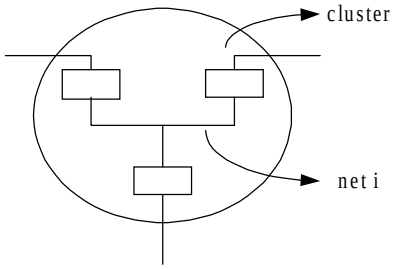


Figure 4: Length prediction for absorbed net

The wire-length predictions will serve as constraints in our simulated annealing refinement stage, which will be discussed in the next section.

6. Constraint-driven placement

The goal of constraint-driven placement is to meet wire length constraints with minimum placement quality loss. Different approaches have been proposed such as net weighting [9][23], linear programming [10] and simulated annealing [25].

In our constraint driven placement with wire length prediction flow, we use a simulated annealing approach to improve the placement and guide it towards the predictions.

We first partition the chip area into global bins. Cells can move between global bins during the annealing. The cost function is:

$$WL + \lambda \sum_{i=1}^{|E|} max(l_i - c_i, 0) \quad \text{(EQ5)}$$

where l_i is the current length and c_i is the constraint we impose on net i. WL is the total bounding-box-based wire length. λ is a parameter balancing between the wire length and constraint violation optimization. The second term in (EQ5), without the parameter λ, measures the constrains violation. We refer to it as *vio_len* in the experimental results section.

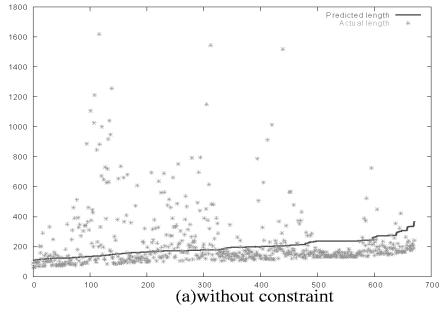
Constraints we put on the nets during the annealing process make our prediction accurate.

7. Experimental results

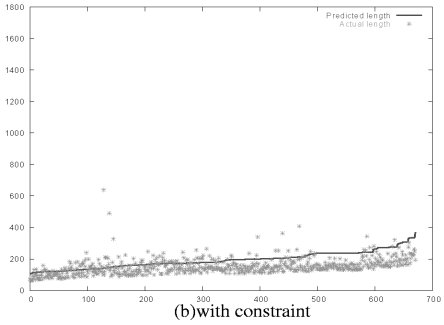
We implemented the constraint-driven placer with wire length prediction in the FPI [15] framework. The experiments were conducted on 2.8Ghz Pentium 4 linux machine with 1 gigabyte memory. We obtained the IBM-place benchmarks from[27]. Sizes of the benchmarks used in our experiments are listed is Table 3.

7.1 Constraints guided prediction

In the first experiment, we take benchmark ibm04 and observe the effect of constraints in our placement flow with wire length predictions. We show wire length prediction results for 3-pin nets in Figure 5. In this figure, on the x-axis are net id's and on the y-axis are net lengths. The solid line represents wire length predicted in stage 1 after sorting them and connecting into a line. The points represent the actual wire lengths in the final placement. If we do not apply constraints in our simulated annealing-based placement refinement stage, we get the result depicted in figure 5(a). When constraints are applied, we get the result depicted in figure 5(b).



(a)without constraint



(b)with constraint

Figure 5: Prediction results for 3-pin nets in ibm04

We define a normalized length violation of a net i as:

$$vio(i) = \frac{max(l_i - c_i, 0)}{c_i} \quad \text{(EQ6)}$$

Then we calculate the average normalized violation (*ave_vio*) and standard deviation of the normalized violation (*vio_sd*). The data for the ibm04 benchmark are shown in Table 2.

TABLE 2. Constraint violation of ibm04

ibm04	non-constraint	constraint
ave_vio	0.36	0.04
vio_sd	0.95	0.17

It is clear that constraints guide our placement process and make the wire length prediction accurate. Note that in most cases, the actual wire lengths are smaller than the constraints imposed on them. The reason is that the constraints serve as upper bounds in the cost function EQ5.

7.2 Penalty for guidance of constraints

In our placement flow, we use constraints to guide the placement towards the predictions, which may cause some extra placement effort or quality losses. In this experiment, we compare the placement results of the non-constraint placement and the constraints-driven placement. We set cluster size constraint to 5 during stage 1.

In table 4, #pre_nets denotes the number of nets on which we make predictions, as a percentage of the total number of nets. We report the total constraint violation length (vio_len) and the total wire length (twl) in meters. We report the corresponding CPU run times in seconds. The constraint-driven placement results are normalized with respect to those of non-constraint driven placement. In the third column we report the total wire length achieved by Capo8.5 to evaluate our placement quality.

We have two conclusions from the data. First, under the cluster size constraint of 5, wire lengths of nearly 50% of nets can be predicted accurately. The results from our constraint-driven placement show that the total constraint violation is very small compared with total wire length. Second, the constraint-driven placement helps in getting accurate predictions without loss of placement quality. The only penalty is a slight increase in the run time. In other words, appropriately determined constraints make our wire length prediction accurate with a small run time penalty.

7.3 Prediction vs. placement effort

During the stage 1 of our placement flow, if we increase the cluster size constraint, we will get fewer clusters of larger size. Then more nets will be absorbed which means we can make length prediction on more nets. On the other side, we have to put more effort in simulated annealing stage to recover placement quality from bad decisions made during clustering. Experiments in this section quantify placement effort as a function of the number of nets on which length predictions are made. In all the experiments we maintain the same placement quality.

We increase the cluster size constraint from 5 to 10 and 15. Using the same simulated annealing schedule as in section 7.2, we compare the results in table 5 and 6. Table 5 shows non-constraint driven placement results and table 6 shows constraint driven placement results. Results of cluster size 10 and 15 have been normalized with respect to those of cluster size 5.

To achieve the same placement quality as in the case of cluster size 5, we adjust the simulated annealing schedule for cluster size 10 and 15. The results are shown in table 7. CPU run times for cluster sizes 10 and 15 have been normalized with respect to those of cluster size 5. Note that the total wire lengths in those three scenarios are almost the same.

From the experiments we draw two conclusions. First, we can make more predictions by increasing cluster size constraints. Second, increasing the number of length-predicted-nets comes together with placement quality losses. We have to put more effort on placement refinement to recover from them.

Another way to make more nets predictable is to apply multi-level clustering and predict wire length in different clustering level. We will address this problem in our future work.

8. Conclusions

In this paper we show that some knowledge of a specific placement flow is necessary for accurate wire length predictions. We also demonstrate that constraints, generated for the target placement flow, can be used to assist wire length predictions. Experimental results indicate that by enforcing constraints in placement, the length of interconnects can be predicted without placement quality losses.

9. Acknowledgements

This work was supported by the DARPA/MARCO Giga Scale Research Center.

References

[1] S.Bodapati and F.N.Najm, “Pre-layout Estimation of Individual Wire Lengths”, ACM Intl. Workshop on System-Level Interconnect Prediction, pp.93-98, Apr 2000.

[2] M.Burstein and M.N.Youssef, “Timing Influenced Layout Design”, In Design Automation Conference, pp.124-130, 1985.

[3] A.E.Caldwell, A.B.Kahng and I.L.Markov, “Can Recursive Bisection alone Produce Routable Placements”, Design Automation Conference, pp.260-263, 2000.

[4] A.E.Caldwell, A.B.Kahng, S.Mantik, I.L.Markov and A.Zelikovsky, “On Wirelength Estimations for Row-based Placement”, IEEE Transactions on Computer-Aided Design, pp.1265-1278, Sep 1999.

[5] J.Cong and S.K.Lim, “Edge Separability Based Circuit Clustering with Application to Circuit Partitioning”, Proc. ASP-DAC, pp.429-434, 2000

[6] W.E.Donath, “Placement and Average Interconnection Lengths of Computer Logic”, IEEE Transactions on Circuits and Systems, vol.CAS-26, pp.272-277, Apr 1979.

[7] A.E.Dunlop, V.D.Agrawal, D.N.Deutsch, M.F.Jukl, P.Kozak and M.Wiesel, “Chip Layout Optimization Using Critical Path Weighting”, In Design Automation Conference, pp.133-136, 1984.

[8] M.Feuer, “Connectivity of Random Logic”, IEEE Transactions on Computers, vol. C-31, no.1, pp.29-33, Jan 1982.

[9] B. Halpin, C.Y.R. Chen and N. Sehgal, “A Sensitivity Based Placer for Standard Cells”, GLS-VLSI, 1999.

[10] B. Halpin, C.Y.Roger Chen and N. Sehgal, “Timing Driven Placement using Physical Net Constraints”, Design Automation Conference, pp.780-783, 2001

[11] T.Hamada, C.K.Cheng and P.M.Chau, “A Wire Length Estimation Technique Utilizing Neighborhood Density Equations”, In Design Automation Conference, pp. 57-61, 1992.

[12] T.Hamada, C.K.Cheng and PlM.Chau, “Prime: A Timing-Driven Placement Tool Using a Piecewise Linear Resistive Network Approach”, In Design Automation Conference, pp.531-536, 1993.

[13] S.Hauck and G.Borriello, “An Evaluation of Bipartitioning Techniques”, IEEE Transactions on Computer-Aided Design, vol 16, No.8, 1997.

[14] H.T.Heineken and W.Maly, “Standard Cell Interconnect Length Prediction From Structural Circuit Attributes”, Proc. Custom Integrated Circuits Conference, pp.167-170, May 1996.

[15] B.Hu and M.Marek-Sadowska, “Fine-granularity Clustering for Large-scale Placement Problems”, to appear in Proc. Intl. Symp. on Physical Design,2003.

[16] B.Hu and M.Marek-Sadowska, “Wire Length Prediction based Clustering and its Application in Placement”, submitted.

[17] M.A.B.Jackson and E.S.Kuh, “Performance-Driven Placement of Cell Based IC’s”, In Design Automation Conference, pp. 370-375, 1989.

[18]R.Nair, C.L.Berman, P.S.Hauge and E.J.Yoffa, “Generation of Performance Constraints for Layout”, IEEE Transactions on Computer Aided Design,8(no.8):860-874, Aug 1989.
[19]M.Pedram and B.Preas, “Interconnection Length Estimation for Optimized Standard Cell Layouts”, Proc. of Int. Conf. on Computer-Aided Design, pp.100-108, Oct. 1989.
[20]M.Pedram and B.Preas, “Accurate Prediction of Physical Design Characteristics for Random Logic”, Proc. of Int. Conf. on Computer Design, pp.390-393, Nov. 1989.
[21]M.Sarrafzadeh, D.A.Knol and G.E.Tellez, “A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement”, IEEE Transactions on Computer Aided Design, 16(11):1332-1341, 1997.
[22]W.Swartz and C.Sechen, “Timing Driven Placement for Large Standard Cell Circuits”, In Design Automation Conference, pp.211-215, 1995.
[23]R.S. Tsay and J. Koehl, “An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement”, Design Automation Conference, pp. 620-624, 1991.
[24]M.Wang, X.Yang and M.Sarrafzadeh, “Dragon2000: Standard-cell Placement Tool For Large Industry Circuits”, Proc. Int. Conf. on Computer-Aided Design, pp.260-264, 2000.
[25]X. Yang, B.K. Choi and M. Sarrafzadeh, “Timing-Driven Placement using Design Hierarchy Guided Constraint Generation”, Proc. Int. Conf. on Computer-Aided Design, 2002
[26]Latest Capo(version 8.5): http://vlsicad.ucsd.edu/Resources/Software Links/PDtools/
[27]IBM-place benchmarks:http://gigascale.org/bookshelf/.

TABLE 3. Benchmark size

bench	ibm01	ibm02	ibm03	ibm04	ibm05	ibm06	ibm07
#cell	12274	19321	22207	26633	29347	32185	45098
#net	11507	18429	21621	26163	28446	33354	44394

TABLE 4. Effect of constraints on placement quality(cluster size constraint: 5)

			non-constraint			constraint		
bench	#pre_nets	Capo twl	vio_len	twl	cpu	vio_len%	twl%	cpu%
ibm01	47.9%	55.8	1.94	54.4	155.65	0.17	1.01	1.07
ibm02	47.4%	159.8	3.44	157.9	324.12	0.26	0.99	1.04
ibm03	48.9%	10.1	0.277	9.59	328.45	0.10	1.00	1.03
ibm04	47.1%	13.1	0.632	12.5	393.11	0.09	1.01	1.06
ibm05	47.0%	35.0	0.995	34.3	488.39	0.10	0.99	1.05
ibm06	46.3%	14.9	0.373	15.4	590.92	0.13	0.98	1.05
ibm07	47.7%	370.0	10.92	358.7	697.04	0.14	1.01	1.05
ave		1.02	1.00	1.00	1.00	0.14	1.00	1.05

TABLE 5. Cluster size VS placement quality(non-constraint)

	5			10			15		
bench	vio_len	twl	cpu	vio_len%	twl%	cpu%	vio_len%	twl%	cpu%
ibm01	1.94	54.4	155.65	0.88	1.08	0.99	0.99	1.11	0.97
ibm02	3.44	157.9	324.12	1.08	1.02	0.96	1.31	1.04	0.93
ibm03	0.277	9.59	328.45	1.67	1.07	0.99	1.55	1.13	0.96
ibm04	0.632	12.5	393.11	1.07	1.04	0.98	0.88	1.07	0.98
ibm05	0.995	34.3	488.39	2.27	1.04	1.03	1.87	1.07	1.02
ibm06	0.373	15.4	590.92	1.26	1.01	0.99	1.13	1.03	1.00
ibm07	10.92	358.7	697.04	1.15	1.07	1.03	1.34	1.07	1.03
ave	1.00	1.00	1.00	1.34	1.05	0.99	1.30	1.07	0.98

TABLE 6. Cluster size VS placement quality(with constraint)

	5			10			15		
bench	vio_len	twl	cpu	vio_len%	twl%	cpu%	vio_len%	twl%	cpu%

TABLE 6. Cluster size VS placement quality(with constraint)

ibm01	0.33	54.9	166.99	1.16	1.06	0.96	1.60	1.09	0.92
ibm02	0.90	156.7	338.00	1.32	1.01	0.93	1.64	1.05	0.95
ibm03	0.03	9.59	337.74	2.16	1.05	0.98	2.40	1.07	0.97
ibm04	0.06	12.6	417.61	1.53	1.04	1.01	1.27	1.05	0.97
ibm05	0.10	33.9	513.86	4.69	1.07	1.01	5.17	1.07	0.98
ibm06	0.05	15.1	620.9	1.46	1.02	0.98	1.36	1.05	1.00
ibm07	1.50	362.9	734.04	1.71	1.05	1.01	2.06	1.07	1.04
ave	1.00	1.00	1.00	2.00	1.05	0.98	2.21	1.06	0.98

TABLE 7. Prediction VS placement quality

bench	5		10		15	
	#pre_nets	cpu	#pre_nets	cpu%	#pre_nets	cpu%
ibm01	47.9%	166.99	59.9%	1.12	63.7%	1.27
ibm02	47.4%	338.00	56.4%	1.02	59.3%	1.15
ibm03	48.9%	337.74	63.6%	1.10	67.0%	1.21
ibm04	47.1%	417.61	61.7%	1.08	65.0%	1.15
ibm05	47.0%	513.86	62.0%	1.14	63.0%	1.21
ibm06	46.3%	620.9	59.7%	1.08	63.4%	1.15
ibm07	47.7%	734.04	60.6%	1.10	63.9%	1.21
ave	47.5%	1.00	60.6%	1.10	63.6%	1.18