An LP-based Methodology for Improved Timing-Driven Placement *

Qingzhou (Ben) Wang, John Lillis and Shubhankar Sanyal
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
{qwang, jlillis, ssanyal}@cs.uic.edu

Abstract— A method for timing driven placement is presented. The core of the approach is optimal timing-driven relaxed placement based on a linear programming (LP) formulation. The formulation captures *all* topological paths in a linear sized LP and thus, heuristic net weights or net budgets are not necessary. Additionally, explicit enumeration of a large number of paths is avoided. The flow begins with a given placement and iteratively extracts timing-critical sub-circuits, optimally places the sub-circuit by LP and applies a timing-driven legalizer. The approach is applied to the FPGA domain and yields an average of 19.6% **reduction in clock period of routed MCNC designs versus [6] (with reductions up to** 39.5%**).**

I. Introduction

With the dominance of interconnect delay in modern technologies, the timing-driven placement problem has become of fundamental importance. While there are many relatively mature techniques available for post-placement optimization such as buffering, gate sizing and timing-driven routing, the capability of such tools is inherently limited by the given placement.

Background and Related Work

Many approaches to the problem have been proposed in the past. Among the more popular are so-called net-weighting methods. In such approaches, usually static timing analysis is applied at intermediate phases and nets are assigned weights according to their criticalities. Thus, such strategies attempt to reduce the timing driven placement problem to the weighted wire-length problem. The underlying optimization technique can vary – e.g., [6] uses a simulated annealing method while [1] uses a quadratic placement approach.

Another net-based method is to impose net constraints at intermediate stages based on static timing analysis. One example of such an approach is [7]. In that work, linear programming is used in concert with partitioning to enforce the net constraints. A weakness of such approaches is that path constraints are necessarily broken down into net constraints and this often over-constrains the problem.

In addition to net-based approaches, there are explicitly path-based approaches. One example is [8], in which critical or near-critical paths are monitored and the effect of perturbations on such paths is put into an evaluation function in a simulated annealing framework. A limitation of such approaches is the fact that the number of near-critical paths may grow exponentially and this limits scalability.

Alternative path-based approaches which have received comparatively little attention *implicitly* optimize *all* topological paths in the timing graph in a mathematical programming framework. Of particular relevance is the work of Jackson and Kuh [5]. In that work, a linear programming (LP) formulation for relaxed placement was presented. The formulation utilized intermediate variables to encode arrival times at cells in the circuit. Such an approach makes possible formulations where wiring resources and timing properties are treated separately. For instance, formulations which minimize wire-length subject to clock period constraints (or vice versa) are natural. Despite these strengths, such techniques have not gained great popularity. We conjecture some reasons for this. First, LP-based techniques have a tendency for excessive cell overlap versus quadratic programming techniques. This implies that in a traditional top-down application of the LP may not provide a good guide for the placer as the cells become too clustered. Second, scalability of LP-based approaches is a concern (however, we note that commercial LP solvers have improved tremendously in recent years – e.g., [10]). A third issue is delay modeling. In [5], interconnect load was incorporated into the gate delay model, but, as was appropriate for the technology of the day, interconnect delay was not included. However, in our view, this is a minor issue in that generalizations of the LP of [5] are possible and utilized in this work. [1]

Overview

In this paper, we develop a framework in which all-path LP-formulations are utilized in an iterative approach. The overall strategy is adapted from the Mongrel placement tool [4]. In that work, an analytical solver – a linear program for wire-length minimization solved by network flow – was applied. However, instead of using the solver in the traditional top-down manner, it was used as the basis for iterative improvement. This mechanism was dubbed Relaxation Based Local Search (RBLS).

The experimental domain addressed is that of FPGAs. As such, since buffered routing switches are prevalent in modern

*This work is supported by SRC under contract 914.001.

[1] In fact, though not pertinent to this work, non-linear delays can be handled by piece-wise linear approximation in an LP framework.

FPGA architectures, a linear delay model is most suitable at the placement level (i.e., pin-to-pin routing delay is approximated by a scaled Manhattan distance between the pins).

While, for experimental reasons, we focus on FPGAs, we note that the approach is applicable to many other placement scenarios. Of particular interest is *pre-buffering placement* for standard cell designs. It is clear that buffer insertion should only occur in the presence of placement information and as such linear models are suitable at this pre-buffering stage.

The subsequent sections give an overview of the approach including the LP-formulation, sub-circuit extraction strategies, and placement legalization. Finally, experimental results are reported. The experimental flow starts with a timing-driven placement from the VPR tool [6] and begins iterative improvement based on RBLS. The resulting placement is evaluated in terms of wire-length and performance at both the pre- and post-routing. The results are very promising with 16.0% reduction of placement level delay estimation and 12.8% reduction for final routed circuits with no more than four tracks more than the minimum needed. When given more available tracks, the reduction can be better, up to average of 19.6%. The increase in the demand for routing resource is also modest, at only a few more tracks for each circuit.

II. Algorithmic Description

The algorithmic approach taken in this work is based on the *Relaxation Based Local Search* (RBLS) mechanism proposed in [3] and [4].

The idea behind RBLS is to take advantage of the more global view obtained by analytical methods in a local search framework. In the context of wire-length driven placement [4], a *Global Placement* problem is solved in which a relatively fine grid is imposed over the placement area and cells are assigned to bins in the grid within row and bin capacity constraints.

With this notion of a global placement, RBLS behaves as follows. Given a feasible placement (i.e., not exceeding bin capacities), repeat the following:

(1) Extract a set of cells $\mathbb{M}$ and declare all such cells to be "mobile" leaving the remainder of the circuit fixed.

(2) Ignoring capacity constraints, *optimally* place the cells in $\mathbb{M}$ in the given context. Here the notion of optimality may depend on the analytical solver being used. In [4], an LP-relaxation of the problem (solved via network flow) was applied. However, in principle quadratic placement or other analytical methods could be used.

(3) Given the result of the analytical placer any capacity violations are resolved by a wire-length driven legalization procedure. This procedure uses the idea of ripple moves from over-congested regions to under-congested regions. The effect on wire-length is captured via the gain-graph abstraction.

(4) If the resulting feasible placement from step (3) is an improvement over that prior to step (1), we accept the new placement; otherwise, the new placement is rejected. Note that even though the relaxed problem was solved optimally, the legalizer introduces noise and therefore, the ultimate solution may degrade.

Such a sequence of operations can be viewed as a single perturbation of the placement. which exploits the more global view afforded by the analytical solver. As the algorithm progresses, the size of extracted sub-circuit is decreased and, normally, the grid is made finer.

We adapt this framework to the timing-driven domain in this work. In order to do this, we have developed a timing-driven sub-circuit extractor for step (1); we utilize a timing-driven LP-relaxation similar to that of [5] for step (2); and we utilize a timing-driven legalizer similar to that of [2] for step (3).

As mentioned in the introduction, the LP-relaxation has some important properties. In particular, through the use of intermediate variables, we can capture *all* topological paths in the design (or sub-circuit) implicitly. Thus, the approach is inherently path-based and eliminates the need for heuristic net weights or budgets.

The following subsections detail these various components.

A. Sub-circuit Extraction

We call a set of mobile cells $\mathbb{M}$ in a circuit a sub-circuit. Sub-circuit extractor decides the sampling scheme of the RBLS framework. To describe the sub-circuit extractor, we first introduce the concept of *criticality*. The *criticality* of a cell is the ratio of the delay of slowest path passing through the cell and the slowest delay of all paths, which is the circuit's clock period .

Our most effective sub-circuit extractor is called "Wavefront by Criticality". In this mode, one of the cells on the critical path is chosen. Then its connected neighbors are put into a candidate queue. Cells in the candidate queue are then selected by a randomized process biased towards cells with higher criticalities. Each time a cell is selected into sub-circuit, its neighbors will be added into the queue. Experimental results have shown that this simple extractor works well.

B. LP Formulation

The LP formulation contains two sets of variables and constraints – Physical and Timing. Physical constraints describe the physical locations of cells and Half-perimeter wire-lengths (HPWL) of nets, while timing variables and constraints capture downstream delays (or arrival times) of cells. In building timing constraints, the net-list is modeled as a directed graph called Timing Graph, $\mathbb{G}^T = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is the set of all cells in the circuit, $\mathbb{E}$ is the set of all *Timing Edges*. In this graph, a net is represented by timing edges in a star model, where edge $e_{(s,t)}$ goes from the driver s to sink t. One assumption about the Timing Graph is that there is no combinational cells that form a cycle. And this is a necessary condition for the LP to behave properly. This condition holds true for all the MCNC FPGA circuits we used in the experiments. The objective function can be wire-length, timing, or a combination of both. Depending upon which one is selected, some extra constraints may be added to the LP to gain better control on the relaxed placement.

B.1 Physical Constraints

For each net, the HPWL is encoded as a set of constraints in the LP formulation:

Let $\mathbb{N}_i$ be the set of cells connected to net i, we have:
$\forall j \in \mathbb{N}_i$,

$$l_i \leq x_j \tag{1}$$
$$r_i \geq x_j \tag{2}$$
$$t_i \leq y_j \tag{3}$$
$$b_i \geq y_j \tag{4}$$
$$Min_x \leq x_j \leq Max_x \tag{5}$$
$$Min_y \leq y_j \leq Max_y \tag{6}$$

where x's and y's encode cell coordinates, Min_x, Min_y, Max_x and Max_y are the boundaries of the placement area, and l_i, r_i, t_i, b_i represent the left, right, top and bottom boundary of net i. Thus, we can write the total wire-length as:

$$WL = \sum_{i=1}^{N} (r_i - l_i + b_i - t_i) \tag{7}$$

where N is the total number of nets.

As for the fixed cells $\mathbb{F}$, we have:
$\forall i \in \mathbb{F}$:

$$x_i = X_i \tag{8}$$
$$y_i = Y_i \tag{9}$$

where X_i and Y_i are known constants.

B.2 Timing Constraints

As mentioned earlier, a linear net delay model is used. We call the delay per unit wire-length D_{unit}. We also model all the delays associated with signal passing through a cell into one value, and call it a cell's internal delay, denoted by *inDelay*.

We start the timing constraints with the timing edges.
$\forall e_{(i,j)} \in \mathbb{E}$:

$$eDelay_{(i,j)} = D_{unit} \cdot (distX_{i,j} + distY_{i,j}) \tag{10}$$
$$distX_{i,j} \geq x_i - x_j \tag{11}$$
$$distX_{i,j} \geq x_j - x_i \tag{12}$$
$$distY_{i,j} \geq y_i - y_j \tag{13}$$
$$distY_{i,j} \geq y_j - y_i \tag{14}$$

where x's and y's are the same as in physical constraints, and $eDelay_{(i,j)}$ represents the delay for the timing edge (i, j).

For each cell i, we introduce tow timing variables, *Downstream Delay* and *Arrival Time*, denoted by $dDelay_i$ and $aTime_i$ respectively. We also define a **fan-out** of a cell i, $\mathbb{O}_i$, as the set of cells for each of which there exists at least one timing edge $e_{(i,k)}$ between i and this cell, say, k. Similarly, we can define the **fan-in** of a cell i, represented by $\mathbb{I}_i$. Then $\forall i \in \mathbb{V}$:
$\forall j \in \mathbb{O}_i$:

$$dDelay_i \geq dDelay_j + inDelay_j + eDelay_{i,j} \tag{15}$$
$$D_{max} \geq dDelay_i \tag{16}$$

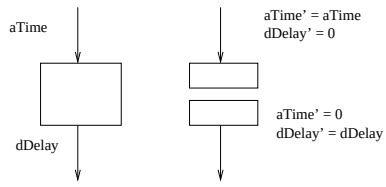

Fig. 1. Delay Modeling for Flip-flops

Similarly, $\forall j \in \mathbb{I}_i$:

$$aTime_i \geq aTime_j + inDelay_j + eDelay_{j,i} \tag{17}$$
$$A_{max} \geq aTime_j \tag{18}$$

Downstream delays for output pads and arrival time for input pads are given constants. As shown in 1 a flip-flop is modeled as two physically coupled cells where the input cell has zero downstream delay and output cell has zero arrival time.

Effectively, minimizing the maximum arrival time is the same as minimizing the maximum downstream delay. Note that a critical path may not start with or end at pads; it can be a path going from one flip-flop to another. Not only for the matter of completeness that we are showing the definitions for both *dDelay* and *aTime* here, both values are needed to calculate the criticality of a net, which is used later in the timing-driven legalization.

B.3 Objective Function

Timing, wire-length or a linear combination of both can be used as the objective of the LP. Since only one of *dDelay* and *aTime* is needed in the LP formulation, we choose *dDelay*, and D_{max} accordingly, for simplicity. Here are the objective functions we used:

1. Minimize D_{max}

2. Minimize D_{max}
   subject to $WL \leq WL_{limit}$, where WL_{limit} is an upper bound on the wire-length of relaxed placement.

The first objective function gives good timing result but may suffer from excessive wire-length increase. The second one is the most commonly used mode in our experiments because it optimizes the critical path delay and limits wire-length increase at the same time.

B.4 Sub-circuit Convexity Extension

After a sub-circuit is extracted, since the rest of the circuit is considered fixed, inequalities (11) through (14) can actually be replaced by the following equalities for edges where *both* ends are fixed: $\forall e_{(i,j)} \in \{i, j | \in \mathbb{F}\}$:

$$distX_{i,j} = |x_i - x_j| \tag{19}$$
$$distY_{i,j} = |y_i - y_j| \tag{20}$$

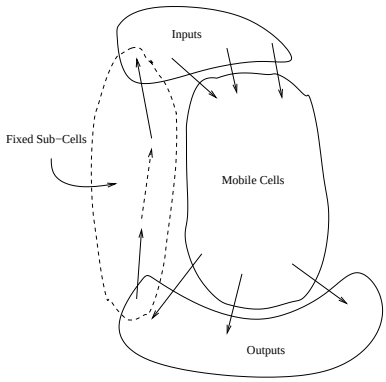

Fig. 2. Sub-circuit Convexity Extension

and the edge delay $eDelay_{(i,j)}$ can be decided according to (10). Also for fixed cells, (15) or (17), may be replaced:

$$dDelay_i = \max_{j \in \mathbb{O}_i} \{dDelay_j + inDelay_j + eDelay_{i,j}\} \quad (21)$$

$$aTime_i = \max_{k \in \mathbb{I}_i} \{aTime_k + inDelay_k + eDelay_{k,i}\} \quad (22)$$

By doing these replacements, the size of LP is reduced by a great deal. But one may find some problems here. As illustrated by Fig.2, consider the situation when there exists a path comes out of the sub-circuit and goes back to it without going through any flip-flop (i.e., a combinational path). Then the LP won't be correct any more. One way to solve this problem is to include these combinational cells in the timing formulation but treat them as physically fixed. We call such added cells *fixed sub-cells*. By doing this, we end up with what we call a *convex*. Effectively, it is the same as that there is no combinational path going from the outputs of the sub-circuit and back to the input of the sub-circuit. The process of finding the fixed sub-cells for a given sub-circuit is called *Sub-circuit Convexity Extension*.

A method has been designed to find a minimum set of such fixed sub-cells. Let the *transitive combinational fan-out* of a set of cells as the set of cells that can be reached through fan-out search from the given cells without passing any flip-flops. Similarly we can define *transitive combinational fan-in*. A minimum set of fixed sub-cells to form a convex sub-circuit is the intersection of the combinational fan-out and combinational fan-in of a given sub-circuit. This algorithm is applied after the sub-circuit extractor as a post process to ensure the correctness of the LP. Experiments have shown the the size of fixed sub-cells set is modest compared to the size of the sub-circuit itself.

B.5 Discussion

LP relaxation can be applied in a *top-down* manner, where the sub-circuit is most of the circuit, if not the whole circuit. Though some additional constraints may be applied, normally excessive cell overlap is still unavoidable. Thus, the "noise" introduced by the legalizer tends to be huge; the good global view on timing captured by LP will get damaged or lost totally in the legalization process. This is not a problem specific to our LP formulation; rather it is a general problem common to methods which use relaxed placement techniques in this *top-down* manner. In our framework though, the legalization noise can be limited to reasonable amount by controlling the size of sub-circuits, combined with using additional constraints, e.g. total relaxed wire-length upper limit.

When HPWL is used in wire-length driven placement, usually the wire-lengths can be decomposed into those of X and Y dimensions. And these decomposed sub-problems can be solved independently (this is the standard procedure in most analytical placers). It is also true for a weighted wire-length, e.g. in a net-weighting based timing driven placement algorithm. But in our case, they are not separable. Only by dealing both dimensions simultaneously can this path-based formulation be described and solved correctly. Similar situation has been pointed out in [7]. We believe this indicates an inherent weakness in weight-based approaches. The LP formulation we adopted handles this "separability" issue very well.

In our experiments, a commercial LP solver CPLEX from iLog [10] is used.

C. Legalization

The timing driven legalization is based on the wire-length driven legalization used in [4]. In order to deal with timing issues in legalization, we adopted a simple net weighing scheme to bias towards moving timingly non-critical cells. It is similar to the timing legalizer used in [2]. A simple weighting scheme is used in our experiments:

Each net is assigned to one of 4 categories based on current slowest path flowing through it; the 4 categories correspond to most critical (weight 1.0) down to least critical (weight 0.25). Then these weights are updated periodically during the legalization process. Using these weights we apply gain-based ripple moves with the goal of discouraging the lengthening of near critical nets. While the strategy is very simple, it seems to be effective. Nevertheless, we believe improvements are possible.

III. Experiments

We implemented the proposed framework in C++. The experiments were carried out on a Pentium4 2.60GHz PC running Linux. The MCNC benchmark files and the VPR source code (version 4.3) were downloaded from [9]. While running VPR placement multiple times did not show any difference in solution quality, we took the results of one run on all the circuits. [2]

The first a few columns of TABLE I show the results of placement level estimation. Column **Delay Est.** is the placement level estimation of the delay values. These values are obtained by running VPR on both versions of the placement files. Column **Run Time** is the run-time for our tool, in terms of VRP run-time. Since we start from a VPR placement, this value should be considered as an addition to the original VPR

[2]The discrepancy between reported delay values in this paper and those from [6] is due to the difference of FPGA architecture files being used. This is described in the header of file `vpr422.arch` which can be downloaded from [9]

Circuit	Size	Placement Level Estimation					CW	Critical Path Delays for Routed Circuits					
		HPWL		Delay Est.		Run		Min		Min + 4		∞	
		VPR	UIC	VPR	UIC	Time		VPR	UIC	VPR	UIC	VPR	UIC
clma	8383	26.42	30.23	227.32	134.33	0.95	15	229.79	144.50	230.99	137.23	230.99	139.71
s38584.1	6447	12.69	13.30	96.55	82.46	0.75	9	97.02	83.60	97.02	83.34	97.02	86.92
s38417	6406	13.96	14.75	90.04	75.10	0.85	9	92.89	112.35	91.35	77.02	91.35	76.44
ex1010	4598	13.58	15.79	180.51	140.06	0.98	12	181.92	191.62	181.25	143.17	181.93	144.94
pdc	4575	18.56	19.68	139.97	111.65	0.92	20	163.46	203.09	141.28	136.24	142.48	114.83
spla	3690	12.24	13.41	110.31	98.43	0.90	16	139.84	152.09	137.79	116.89	113.55	101.01
elliptic	3604	9.56	10.82	110.74	89.34	0.95	12	125.81		113.27	124.79	113.27	93.57
frisc	3556	13.24	13.93	126.44	111.46	0.93	14	132.38		129.04	109.34	129.04	109.34
s298	1931	3.04	3.54	127.46	112.24	1.10	9	129.38		128.80	114.21	128.79	114.21
apex2	1878	6.01	6.44	87.11	76.55	0.91	12		106.44	88.49	80.46	89.09	79.53
seq	1750	5.58	5.97	84.12	66.33	0.90	13	85.58		85.58	69.44	85.58	67.67
bigkey	1707	3.78	4.23	62.77	57.92	0.94	7	67.22		66.62	59.45	66.02	64.23
des	1591	5.83	6.48	119.27	79.28	0.89	7	123.03		122.45	79.73	119.45	79.74
alu4	1522	3.86	4.01	71.17	66.46	0.96	11	86.98		70.91	69.12	70.91	66.72
diffeq	1497	3.19	3.31	69.61	58.53	1.01	9	70.89	60.47	70.89	60.47	70.89	61.67
tseng	1407	2.12	2.13	53.52	53.25	0.92	8	56.04	56.63	56.04	56.63	56.04	56.63
misex3	1397	4.25	4.77	76.15	62.12	1.16	12	98.46		77.42	69.43	77.46	62.43
dsip	1370	3.40	4.06	75.84	52.62	1.03	7	76.76	55.27	77.96	56.45	77.36	55.25
apex4	1262	4.72	4.98	71.17	67.61	0.89	15	81.14	77.56	72.43	71.60	72.45	70.05
ex5p	1064	4.56	4.56	63.38	63.38	0.83	16	76.30	76.30	66.48	66.48	66.12	66.12
Average		108.7%		84.0%		0.89				87.2%		80.4%	

TABLE I
EXPERIMENTAL RESULTS

run-time. One thing to note is that as more time is given, our tool will almost always produce better solutions.

The following columns are the critical path delays for routed circuits. Empty space means the circuit is not routeable. The column of **CW** shows the minimum routeable tracks (Channel Width) needed for either VPR placement or ours. The next a few columns are the critical path delays for actual routed circuits with the minimum CW, *Min*, and with additional tracks, e.g., $Min + 4$. More than half of our placements are routeable at the minimum CW given by VPR placement. Some need more tracks but no more 3 tracks are needed. And the last column shows the delay values for routing the circuits with infinite number of available tracks (100 used here, but actual used is *much smaller* than this value for both versions of placements). Note that sometimes when increasing the routing resource, VPR gives worse instead of better solution. All circuits are routed by the VPR tool with default parameter settings.

IV. CONCLUSIONS AND DISCUSSION

In this paper, a method for timing driven placement based on LP-relaxation is presented. Prototype has been implemented and experiments have shown promising results. At the placement level, the estimated critical path delay has reduced by average of 16.0%. This is consistent with actual routed results, where average reduction of 19.6% is achieved. Ongoing work includes application to pre-buffering placement in the standard-cell domain, improved legalization procedures, improved convergence and initial placement strategy.

REFERENCES

[1] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proceedings of the 35th annual conference on Design automation*, pages 269–274, 1998.

[2] M. Hrkic;, J. Lillis, and G. Beraudo. An approach to placement-coupled logic replication. In *Proceedings of the 41st annual conference on Design automation*, pages 711–716, 2004.

[3] S.-W. Hur and J. Lillis. Relaxation and clustering in a local search framework: application to linear placement. In *Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 360–366, 1999.

[4] S. W. Hur and J. Lillis. Mongrel: hybrid techniques for standard cell placement. In *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 165–170, 2000.

[5] M. A. B. Jackson and E. S. Kuh. Performance-driven placement of cell based ic's. In *Proceedings of the 26th ACM/IEEE conference on Design automation*, pages 370–375, 1989.

[6] 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*, pages 203–213, 2000.

[7] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary, and B. Halpin. Timing driven force directed placement with physical net constraints. In *Proceedings of the 2003 international symposium on Physical design*, pages 60–66, 2003.

[8] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proceedings of the 32nd ACM/IEEE conference on Design automation*, pages 211–215, 1995.

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

[10] http://www.ilog.com/products/cplex/.