Timing Driven Force Directed Placement with Physical Net Constraints

Karthik Rajagopal, Intel Corporation
Tal Shaked, Intel Corporation & University of Washington
Yegna Parasuram, Intel Corporation
Tung Cao, Intel Corporation
Amit Chowdhary, Intel Corporation
Bill Halpin, Intel Corporation & Syracuse University

ABSTRACT
This paper presents a new timing driven force directed placement algorithm that meets physical net length constraints as well as constraints on specific pin sets. It is the first force directed placement algorithm that meets precise half perimeter bounding box constraints on critical nets. It builds on the work of Eisenmann et al. [12], adding a new net model that changes the contribution of constrained nets in the quadratic programming problem, during solving for each force generation step. We propose several methods for selecting and constraining critical nets to achieve improved timing. Our work suggests that the force directed method with net constraints is a powerful tool for placement and timing convergence, achieving an average worst negative slack optimization exploitation of 64% and average total negative slack optimization exploitation of 48% results on 16 industry circuits from a 1.5GHz microprocessor.

Categories and Subject Descriptors
B.7.2 [**Hardware, Integrated Circuits, Design Aids**]: Placement & routing.

General Terms: Algorithms, Design.

Keywords: Timing Driven Placement, Force Directed Placement, Net Constraints.

1. Introduction
Placement has long been an important step for timing optimization during the IC design flow. Placement is significant because it determines the length of interconnects. Semiconductor process advances increase the impact of interconnect delay in determining circuit performance since the wire delays do not reduce as rapidly as gate delay [4] [17]. In high speed interconnect dominated designs, placement has a significant impact on design functions such as buffering, gate sizing, logic synthesis and logic design. Process scaling also allows larger designs and the importance of the placement step grows with design size [25].

The placement field has been the subject of much research [1][2][7-12][14-16][19-25]. There are 3 main goals in the automated placement problem: minimizing chip area, achieving routable designs, and maximizing circuit performance. Maximizing circuit performance, the topic of this paper, has been the focus of continued attention. In the placement of very high performance circuits there are only a few combinational cells between the timed elements in a path. Given a circuit, the best possible delay for a path is the sum of the gate delays assuming no interconnect. This delay is referred to as the unloaded delay of the path. Assuming all of the unloaded delays meet the desired circuit performance requirement, the timing critical paths will be caused by interconnects whose delay pushes the unloaded delays beyond the timing specification.

Timing driven placement techniques can be classified as either path-based [1][9][7] or net-based [2][12][14][15][24]. Path based modeling is the most natural as it reflects the true nature of the timing problem, but for large circuits it is not possible to enumerate all of the paths. For this reason much of the recent timing driven work has been net-based [12][15][24]. In the net-based approach, critical nets are identified and either given higher weights or net length constraints. Since net weights do not have any direct analogy in the timing space, the net weighting approaches [12] suffer from the problem of identifying the proper net weights. Net length constraints are a more natural choice since there is a direct and computable correspondence between a net length and the delay of a net. In [24], a net constraint approach is used with recursive bi-section placement. But this placement technique is limited in its ability to do iterative refinement.

2. Motivation
Traditionally, timing optimization of a design is achieved by repeated application of various steps – timing driven placement, buffer insertion, cell sizing and circuit re-synthesis. In the present interconnect-dominated deep sub-micron era, a simple sequential execution of these optimization steps is not sufficient to converge critical designs. Hence, a powerful timing convergence flow should tightly integrate the optimization steps of buffer insertion, cell sizing and circuit re-synthesis with the timing-driven placement engine. It is important that the placement engine should be able to smoothly integrate the changes made by these operations.

Force directed placement techniques like [12] lend themselves easily to introduction of optimizations like sizing and re-synthesis due to their very incremental nature. This technique works well in tandem with sizing, re-synthesis and buffering. Though there have been a number of works in literature on timing driven placement, not too many of them are incremental and handle netlist changes as smoothly as this method. Another important attribute of this method is that it is ideal for eco placement.

In Eisenmann's work, timing driven placement was implemented as weights on timing critical nets in the network. Recent work [24] suggests that using net length constraints derived from timing is a better way of converging timing. This is because the delay through a cell and the net driven by it depends on the wire capacitance of the net, which depends on the wire length of the net. Thus a net constraint bounds the delay across the driving cell as well as the net. Since the importance of all receivers on a net is not uniform, it is desirable to be able to constrain specific net segments between the driver and specific receiver(s) on the net. This work proposes a new formulation of the force directed placement technique with general net length and net segment constraints and proposes several methods of setting the constraints to achieve improved timing results.

3. Overview of Force Directed Placement

We first give an overview of the force-directed placement algorithm by Eisenmann et al. [12]. The circuit is modeled as a graph with cells as vertices and nets as sets of edges. A net connecting k cells is modeled as a star with an additional movable vertex and an edge connecting each of the k cells to this vertex (star model) [10]. In the quadratic placement problem, the cost of a net is the sum of the cost of all its edges, where the cost of an edge is modeled as the squared distance between the two vertices (cells) of the edge. The overall objective function is to minimize the sum of the cost of all nets. In matrix notation, the objective function is given below in terms of a 2n-dimensional placement vector p, where n is the number of vertices:

$$Objective = \frac{1}{2}\vec{p}^{\,T}.C.\vec{p} + \vec{d}^{\,T}.\vec{p} + const$$
$$where, \vec{p} = \left(x_1, \ldots, x_n, y_1, \ldots, y_n\right)^T$$

Here, the *x* and *y* locations of vertex *i* are denoted by x_i and y_i, respectively. The cost of an edge between two movable vertices *i* and *j* is given by $(x_i-x_j)^2 = x_i^2 - 2.x_i.x_j + x_j^2$. The first and the last terms represent the *i*th and *j*th diagonal entry in *C* matrix, while the middle term represents the negative entries at *(i, j)* and *(j, i)* positions in *C*. We can assign a weight of *w* to an edge by multiplying the corresponding entries in *C* by *w*. In case of a fixed cell *f*, the squared distance is given by $(x_i-x_f)^2 = x_i^2 - 2.x_i.x_f + x_f^2$, where the first term represents the *i*th diagonal entry in *C*, the second term represents the negative entry in row *i* of *d* vector and the third term contributes to the constant. The quadratic objective function is optimized by solving the following system of linear equations:

$$C.\vec{p} + \vec{d} = 0$$

The force-directed placement algorithm in [12] modifies the above formulation by including an additional force vector $\vec{e}$ that is derived from the cell density distribution in the placement area. The force vector $\vec{e}$ is used to remove cell overlaps by moving cells from areas of high cell density to areas of low cell density.

$$C.\vec{p} + \vec{d} + \vec{e} = 0$$

The force-directed placement approach iteratively solves the above system of linear equation. At every iteration, the force vector is first derived from the density distribution of cells based on the current placement and then used to determine the new location of cells by solving the above system of equations.

4. Problem Statement

The problem of timing-driven placement can be specified as follows. Given a circuit and a fixed placement area, find an overlap-free placement of all cells in the placement area such that the total wirelength and the timing of the circuit are minimized. The total wirelength is a good measure of the ease of routing the design. Here we focus on extending the force-directed placement approach to optimize the timing of the circuit.

The timing of a circuit is measured in terms of two metrics: worst negative slack (WNS) and total negative slack (TNS). The slack at a timing point (an input/output pin of a cell or an input/output pad of the circuit) is the difference between the required and arrival times of the signal at that timing point. A negative slack implies that the signal is arriving later than required time. WNS is defined as the worst slack among all timing endpoints (input pin of latches/flip-flops or output pads of the circuit), while TNS is the total sum of negative slacks at the timing endpoints. Given a placement of all cells of the circuit, a timing analysis engine gives the WNS and TNS for the current placement.

5. Proposed Approach

We propose a timing-driven placement approach where a set of net constraints on critical nets is derived from the timing information of the circuit. A net constraint is an upper bound on the half perimeter of the smallest bounding box enclosing all connection points to a net. A connection point is the location on a cell where a net connects to the cell. Setting net constraints on a small number of nets ensures that the effect of net constraints on the total wirelength of the circuit is marginal; we want to minimize wirelength while meeting the net constraints. The timing-driven formulation of our approach can be described in two steps.

- The timing constraints of the circuit are translated into net constraints on a small number of nets.
- The net constraints are modeled in the force-directed problem formulation such that these net constraints can be met with only a marginal impact on the other nets and cells in the circuit.

We outline our approach in Figure 1.

5.1 Timing Constraints to Net Constraints

Choosing the right nets to set net constraints is crucial to minimize WNS and TNS. Net constraints can be either specified by the user or derived from the timing report using heuristics. The timing report generated by a timing analysis engine lists the critical paths in the circuit, given the current placement of the cells. The timing report also specifies the delay and transition time values for cells and nets on the critical paths. After a fixed number of iterations of force-directed placement, we run the timing analysis engine to get new timing report for the current placement of the cells. We use heuristics to identify the critical cells from the timing report, and then set new net constraints or tighten the existing constraints on the critical nets.

```
procedure timing-driven force-directed placement
begin
  create a graph vertex for every movable cell
  for all nets {
     k = number of cells connected to the net
     define an additional vertex for the center of the net
     create k edges, one from each cell to the center vertex
     assign a weight of 1 to all edges
  }

  while (placement is not spread out) {
    for  (fixed number of iterations) {
      for (all net constraints) {
        if (net not meeting constraint) {
           distribute additional weight to bounding cells on this net
        }
      }
      calculate spreading forces to form e vector
      construct the C matrix and the d vector
      solve for the new placement of cells
    }
   run the timing analysis engine on the current placement
   use heuristics to constrain critical nets
  }
end
```

Figure 1. Pseudo-code for proposed timing-driven placement approach

Our heuristics for selecting critical nets are based on the following factors for every net.

- Drive strength of the cell driving the net.
- Current wire capacitance of the net.
- Pin capacitance of pins and fixed pads connected to the net (this is the lower bound on the load capacitance on the driving cell, when the net length is 0).
- Transition time change across the driving cell and the net.

5.2 Modeling net constraints

We now discuss in detail the modeling of net constraints in the force-directed placement approach. An approach to model a net constraint is to increase the weight on the cost of all edges of the net by the same amount, if the half perimeter of the net is more than the constraint on the net. We have made several key enhancements to such an approach of modeling net constraints, which have resulted in significant improvements in the final timing of the design. Next we present the various aspects of our modeling of net constraints.

5.2.1 Additional weight for nets not meeting constraints

During each force spreading iteration, we assign additional weights for newly constrained nets and for nets whose constraints are not met currently. The exact value of the additional weight is determined iteratively and is adjusted before each spreading iteration. If a net is not meeting its net constraint c, then we add an additional weight proportional to the amount of the net's half perimeter in excess of c, see Figure 2. The additional weight assigned to a net is then distributed among all edges of the net. This is described in section 5.2.3.

Once a net is identified as a critical net and a net constraint is determined for that net, then the net constraint is only tightened in subsequent iterations. If the net is no longer critical, we continue to impose the constraint determined when it was previously critical. If the constraint and additional weights on a non-critical net is removed, in some cases, the net's length will increase and eventually the net becomes critical, once again. This does not lead to a converging solution for reducing overall timing slack.

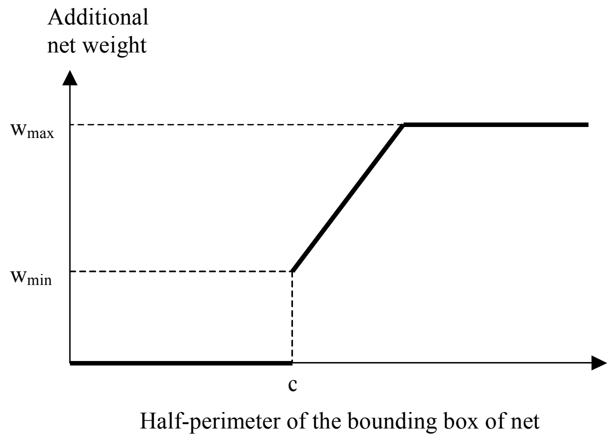

Figure 2. Deriving additional net weights from net constraints: additional net weight *w* depends on half-perimeter of the net exceeding the net constraint *c*. Increasing net weight beyond an upper bound does not help in meeting the net constraint.

5.2.2 Duplicating nets with severe violations on net constraints

We found that nets that are in severe violation of net constraints do not improve (i.e. meet net constraint) by further increasing the weights on their edges. Instead we use an upper bound on the weight of the net. For all nets with weight at the upper bound, we propose the duplication of every edge of the net in the formulation. We have shown experimentally that duplication of all edges of nets with severe constraint violation helps in meeting net constraints for such nets.

5.2.3 Distributing additional weight of a net among its edges

Let us consider the star net model, where a net connecting k cells has an additional vertex for the net center and k edges connecting the k cells to this vertex. For star model, assigning a weight to an edge can be thought about as assigning a weight to its corresponding cell. The additional weight has two components – horizontal weight wx and vertical weight wy. We distribute horizontal weight wx and vertical weight wy of a net among k cells depending on the placement of these cells in the bounding box of the net. Consider the example of a net with six cells, v_1, … , v_6, shown in Figure 3. We add a vertex v_c for the star model of the net; this vertex will get assigned to the mean center of the k cells in the solution of the force-directed placement formulation. If a cell lies strictly inside the bounding box of the net then we assign a weight of 0 to the cell, since moving this cell closer to v_c will not help in meeting the net constraint. If the cell lies on the boundary of the bounding box then we assign it a fraction of the weight that is proportional to the distance of the cell from the center v_c. An example of how the additional horizontal and vertical weights are distributed, for the net shown in Figure 3 is given below:

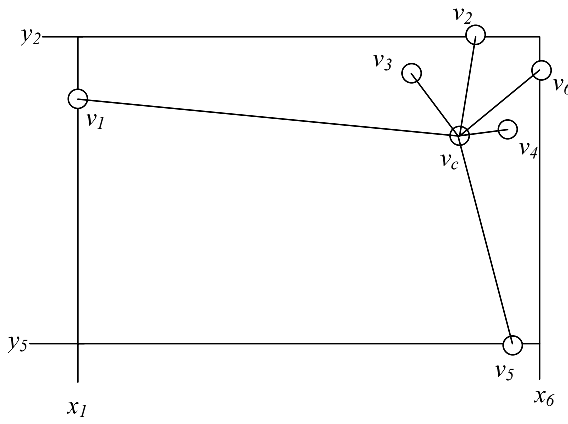

Figure 3. Example of a net with six cells used to illustrate distribution of weights.

$$wx_1 = wx.\frac{(x_c - x_1)}{(x_6 - x_1)}$$

$$wx_2 = wx_3 = wx_4 = wx_5 = 0$$

$$wx_6 = wx.\frac{(x_6 - x_c)}{(x_6 - x_1)}$$

$$wy_1 = wy_3 = wy_4 = wy_6 = 0$$

$$wy_2 = wy.\frac{(y_2 - y_c)}{(y_2 - y_5)}$$

$$wy_5 = wy.\frac{(y_c - y_5)}{(y_2 - y_5)}$$

The additional weight assigned to the center vertex v_c is the sum of weights on all cells, since v_c shares an edge with every cell on the net. The weights are distributed in this manner so as to move the cells on the boundary of bounding box towards the mean center of the cells. In the example above, two of the six vertices on the net do not need to be moved at all for meeting the net constraint. Thus, our weight distribution approach will reduce the impact of net constraint on total wirelength.

Once weights are assigned to a cell for meeting a net constraint, the weights are maintained even after the cell is no longer an outlier for that constraint. If this is not done, the cell may be pulled away from the net-center due to several reasons including connectivity to other nets and spreading forces, and may once again become an outlier. If a cell was not an outlier previously and is now an outlier, we start pulling it in, by assigning additional weight. Our experimental results show that the distance-based weight distribution helps tremendously in meeting net constraints with only a small increase in total wirelength. The iterative approach prevents us from adding weights on cells unless it has an impact on reducing the net's bounding box. We find that this approach results in lesser total wirelength as compared to an approach that adds weights to all the cells on a net being constrained.

5.2.4 *Specifying constraints between driver and critical receiver(s) of a net*

Of the *k* cells on the net, only the driving cell and a few receiving cells are on the critical timing paths of the circuit. We propose net segment constraints that are constraints defined only for the segment of the net connecting the driver and a critical receiver. We meet net segment constraints by using the idea of weight distribution described earlier. Here, the weight of the net segment is distributed between the driver and the critical receivers, which has the effect of bringing the critical cells close to each other. This approach is a refinement to that described in section 5.2.3. A net-segment can be thought of as a virtual net that connects the driver with the critical receivers. The same approach for meeting a net constraint is applied to this virtual net.

6. Experimental Results

We have implemented the net constraint driven placement algorithm in C++ on LINUX. Instead of using MCNC benchmarks [26], we used circuits from a recent microprocessor, since the effect of net constraints on meeting timing is more accurately studied by using data from a recent manufacturing process and standard cell library, and by using state of the art RC estimation and timing analysis engines. For our experiments, we used a set of 16 circuits from a 1.5 GHz microprocessor designed on 0.18 micron process. The circuits range from 548 to 6223 cells, as listed in Table 1.

Table 1. Number of cell instances and number of nets.

Design	Number of standard cells	Number of nets
i1	6039	7081
i2	3441	4228
i3	2096	2569
i4	559	810
i5	1242	1662
i6	1967	2589
i7	1308	1651
i8	3374	4122
i9	686	989
i10	2039	2450
i11	1268	1353
i12	2891	3283
i13	1042	1279
i14	6223	7296
i15	548	708
i16	3399	4150

We used a static timing analysis engine that accurately estimates the delay and transition times across cells and nets of the circuit. Parameters from a 0.18 micron process were used for characterizing interconnect resistance and capacitance and the timing response for standard cells.

For every circuit, we set up two experiments: wirelength driven and timing driven. In both runs, cells were placed using the force directed algorithm. The difference between the runs was that, in the timing driven run, the circuit was timed several times during the course of placement to setup net constraint as discussed above in section 5.1.

We found that apart from the calls to the timing engine, there was no measurable runtime degradation between the wirelength driven and timing driven placement runs. Introducing the net length constraints did not increase the runtime of the core placement algorithm.

Table 2. WNS for global and legalized placements.

Design	**WNS**					**Optimization Exploitation**	
	Unloaded	**Global**		**Legal**		**Global**	**Legal**
		A	**B**	**A**	**B**	**%**	**%**
i1	-0.021	-0.401	-0.168	-0.409	-0.184	61.3%	58.0%
i2	-1.283	-1.394	-1.298	-1.402	-1.297	86.5%	88.2%
i3	-0.083	-0.163	-0.085	-0.17	-0.098	97.5%	82.8%
i4	-0.495	-0.542	-0.496	-0.542	-0.496	97.9%	97.9%
i5	-0.09	-0.12	-0.092	-0.123	-0.109	93.3%	42.4%
i6	-0.071	-0.219	-0.144	-0.227	-0.15	50.7%	49.4%
i7	-0.058	-0.26	-0.182	-0.29	-0.182	38.6%	46.6%
i8	0	-0.561	-0.102	-0.614	-0.12	81.8%	80.5%
i9	-0.061	-0.112	-0.09	-0.135	-0.112	43.1%	31.1%
i10	-0.266	-0.345	-0.267	-0.35	-0.273	98.7%	91.7%
i11	-0.026	-0.372	-0.13	-0.439	-0.139	69.9%	72.6%
i12	-0.294	-0.32	-0.311	-0.321	-0.314	34.6%	25.9%
i13	-0.069	-0.284	-0.182	-0.294	-0.191	47.4%	45.8%
i14	-0.134	-0.298	-0.16	-0.295	-0.183	84.1%	69.6%
i15	-0.003	-0.711	-0.163	-1.077	-0.387	77.4%	64.2%
i16	-0.07	-0.561	-0.201	-0.58	-0.217	73.3%	71.2%
Average						71.0%	63.6%
Column "A" are results with out net constraints							
Column "B" are results with net constraints.							

Table 2 compares the slack of the most critical path (WNS) for the global placements as well as legalized placements. The timing for unloaded placement is obtained by assuming that interconnect has zero capacitance and resistance, i.e. ignoring the delay and the loading due to wires. The improvement in WNS is computed according to the optimization exploitation. The optimization potential is a measure of how much a density result (placement optimized only for minimum wirelength), can be improved relative to the unloaded path delay. We calculate the exploitation as a percent of the optimization potential that was achieved by the timing driven method over its density result.

Across all 16 circuits the average WNS exploitation improvement is 71.0% before legalization and is 63.6% after legalization. The amount of improvement is different for different circuits depending on size, number of nets and circuit topology.

Table 3. TNS for global and legalized placements

Design	**TNS**					**Optimization Exploitation**	
	Unloaded	**Global**		**Legal**		**Global**	**Legal**
		A	**B**	**A**	**B**	**%**	**%**
i1	-0.021	-34.929	-7.601	-45.392	-14.09	78.3%	69.0%
i2	-15.782	-29.26	-21.049	-34.402	-24.52	60.9%	53.1%
i3	-0.291	-1.759	-0.799	-3.581	-2.559	65.4%	31.1%
i4	-0.495	-0.736	-0.512	-0.817	-0.556	92.9%	81.1%
i5	-0.331	-2.018	-0.926	-2.407	-1.334	64.7%	51.7%
i6	-0.895	-6.577	-4.243	-8.143	-5.916	41.1%	30.7%
i7	-0.145	-4.944	-2.281	-5.968	-3.578	55.5%	41.0%
i8	0	-48.252	-6.424	-68.183	-9.669	86.7%	85.8%
i9	-0.215	-1.105	-1.005	-2.877	-1.642	11.2%	46.4%
i10	-4.583	-25.153	-20.194	-33.063	-26.232	24.1%	24.0%
i11	-0.051	-7.736	-3.862	-10.089	-6.892	50.4%	31.8%
i12	-2.593	-8.587	-6.499	-10.641	-8.066	34.8%	32.0%
i13	-0.318	-10.502	-5.479	-12.062	-7.481	49.3%	39.0%
i14	-2.39	-43.97	-21.382	-50.332	-29.28	54.3%	43.9%
i15	-0.003	-14.782	-7.782	-25.245	-12.493	47.4%	50.5%
i16	-0.498	-32.745	-13.378	-38.66	-19.188	60.1%	51.0%
Average						54.8%	47.6%
Column "A" are results with out net constraints							
Column "B" are results with net constraints.							

The net constraint driven algorithm improved TNS as well. In our experiments, the timing engine reported two worst paths for every timing endpoint. One corresponds to the rising signal at that timing endpoint and the other corresponds to the falling signal. TNS is the sum of all negative slack of these paths. From Table 3, the average TNS exploitation is 54.8% before legalization and 47.6%after.

We were able to achieve improvements in timing without much degradation in the wirelength metric. Table 4 compares the wire length between the wirelength driven and timing driven runs. The average increase in wirelength due to running the net length constraint algorithm is 3.3%

Table 4. Wirelength comparison before and after legalization.

Design	Wirelength					
	Global			Legal		
	A	B		A	B	
i1	76,508,072	8.39E+07	9.7%	79,415,344	8.63E+07	8.7%
i2	30,623,224	32,407,498	5.8%	32,869,660	33,605,680	2.2%
i3	16,350,303	17,113,024	4.7%	17,326,240	17,990,284	3.8%
i4	3,047,238	3,227,253	5.9%	3,277,813	3,426,342	4.5%
i5	9,107,331	9,414,814	3.4%	9,796,280	9,891,563	1.0%
i6	18,120,192	18,325,508	1.1%	18,893,960	19,097,088	1.1%
i7	11,031,834	11,378,824	3.1%	11,606,858	11,927,216	2.8%
i8	47,061,444	51,837,004	10.1%	48,546,960	52,830,900	8.8%
i9	6,417,038	7,546,817	17.6%	8,131,880	7,857,532	-3.4%
i10	18,653,544	19,825,388	6.3%	20,024,322	20,821,298	4.0%
i11	6,991,472	7,819,749	11.8%	7,346,770	8,176,857	11.3%
i12	19,675,146	19,797,084	0.6%	20,326,330	20,445,920	0.6%
i13	9,945,988	10,158,826	2.1%	10,308,774	10,566,140	2.5%
i14	68,856,776	79,793,504	15.9%	71,779,504	82,462,656	14.9%
i15	10,705,754	8,610,920	-19.6%	12,177,296	9,814,097	-19.4%
i16	36,970,936	40,938,436	10.7%	38,596,564	42,367,024	9.8%
Average			5.6%			3.3%
Column "A" are results with out net constraints.						
Column "B" are results with net constraints.						

Table 5. Timing comparison with a net-constraint driven recursive-bisection placement algorithm [24]

Design	WNS			TNS		
	A	B	%	A	B	%
i1	-0.203	-0.184	9.4%	-25.52	-14.09	44.8%
i3	-0.192	-0.098	49.0%	-6.67	-2.56	61.6%
i4	-0.498	-0.496	0.4%	-0.7	-0.56	20.0%
i6	-0.213	-0.15	29.6%	-5.71	-5.92	-3.7%
i7	-0.186	-0.182	2.2%	-2.64	-3.58	-35.6%
i9	-0.079	-0.112	-41.8%	-1.58	-1.64	-3.8%
i10	-0.286	-0.273	4.5%	-39.42	-26.23	33.5%
i15	-0.211	-0.387	-83.4%	-11.09	-12.49	-12.6%
Average			-3.8%			13.0%
Column "A" is recursive-bisection placement with net constraints.						
Column "B" is force directed placement with net constraints.						

Table 6. Wirelength comparison with a net-constraint driven recursive-bisection placement algorithm [24]

Design	Wirelength		
	A	B	%
i1	93,549,792	8.63E+07	7.7%
i3	20,919,034	17,990,284	14.0%
i4	3,537,436	3,426,342	3.1%
i6	20,258,164	19,097,088	5.7%
i7	13,111,048	11,927,216	9.0%
i9	7,763,368	7,857,532	-1.2%
i10	23,401,844	20,821,298	11.0%
i15	10,796,181	9,814,097	9.1%
Average			7.3%
Column "A" is recursive-bisection placement with net constraints.			
Column "B" is force directed placement with net constraints			

We also compared our algorithm with the timing driven recursive-bisection placement algorithm based on net-constraints [24]. The algorithm in [24] shows much better timing results for the MCNC bench-marks [26]. Table 5 compares the timing metrics (WNS and TNS) for the two algorithms. On average, TNS from our placement algorithm is 13% better than TNS from the recursive-bisection placement algorithm [24]; WNS from the two algorithms are about the same. The wirelength from our placement algorithm is on an average 7.3% better than the wirelength from the recursive-bisection placement algorithm. The results shown are for legalized placements. A possible explanation for better timing results from the net-constraint driven force-directed placement algorithm is that the force-directed placement approach allows more flexibility of cell movement in later stages of placement that helps in meeting the net constraints on critical nets.

7. Conclusions

We have presented an effective way to introduce net length constraints into a force directed placement algorithm. This technique enables the placer to gently meet these constraints without significantly impacting the total wirelength. We present results using this model in a timing driven placement flow on a number of designs from a high performance microprocessor. In our current work we build on the iterative nature of this algorithm and use it in conjunction with buffering, sizing and re-synthesis to rapidly converge timing critical designs.

8. REFERENCES

[1] Michael A. B. Jackson, Arvind Srinivasan and E. S. Kuh, "A Fast Algorithm for Performance-Driven Placement", Digest of Technical Papers, ICCAD, pp. 328-331, Nov. 1990.

[2] Bernhard M. Riess and Gisela G. Ellelt, "SPEED: Fast and Efficient Timing Driven Placement", pp. 377-380, 1995.

[3] J. Rubenstein , J. Paul Penfield and M. A. Horowitz, "Signal Delay in RC Tree Networks", IEEE Transactions on Computer-Aided Design, vol. 39, no. 11, pp. 825-840, 1992.

[4] H. B. Bakoglu, Circuits, Interconnects and Packaging for VLSI. Addison Wesley, 1990.

[5] W. C. Elmore, "The transient response of Damped Linear network with particular regard to wideband amplifier", Journal of Applied Physics, 1948, pp.55-63.

[6] Lawrence T. Pillage and Ronald A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis", IEEE Transactions on Computer-Aided Design, pp. 352-366, 1990.

[7] William Swartz and Carl Sechen, "Timing Driven Placement for Large Standard Cell Circuits", DAC, pp. 211-215, 1995.

[8] Wern-Jieh and Carl Sechen, "Efficient and Effective Placement for Very Large Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 349-359, 1995.

[9] A. Srinivasan, A. K. Chaudhary, E. S. Kuh, "RITUAL: Performance Driven Placement Algorithm for Small Cell ICs", ICCAD, pp. 48-51, Nov. 1991.

[10] Jurgen M. Kleinhans, Georg Sigl, Frank M. Johannes, and Kurt Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Transactions on Computer Aided Design, Volume 10, No. 3 pp. 356-365, 1991.

[11] K. Doll, F. M. Johannes, and K.J. Antreich, "Iterative placement improvement by network flow methods", IEEE Transactions on CAD, vol. 13 pp.1190-1200, Oct. 1994.

[12] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning", ACM/IEEE DAC, 1998.

[13] J. Cong, "Timing models for Interconnects and Devices", DAC, 1997.

[14] R.S. Tsay, "Timing-Driven Placement", DAC, 1997.

[15] Shih-Lian Ou and Massoud Pedram, "Timing-driven Placement Based on Partitioning with Dynamic Cut-Net Control", DAC, 2000.

[16] Habib Youssef and Eugene Shragowitz, "Timing Constraints for Correct Performance", ICCAD, 1990

[17] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron", pp. 203-211, ICCAD 1998.

[18] Jorge Nocedal and Stephen J. Wright, "Numerical Optimization", Springer-Verlag, 1999.

[19] Bill Halpin, C.Y. Roger Chen, and Naresh Sehgal, "A Sensitivity Based Placer for Standard Cells", GLS-VLSI, 1999.

[20] Ren-Song Tsay and Juergen Koehl, "An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement", DAC, pp. 620-624, 1991.

[21] X. Yang, B-K. Choi, and M. Sarrafzadeh, "A Standard-Cell Placement Tool for Designs with High Row Utilization". International Conference on Computer Design, pages. Sept. 2002.

[22] S. Hur and J. Lillis. ``Mongrel: Hybrid Techniques for Standard Cell Placement", in International Conference on Computer-Aided Design, pp. 165-170, IEEE, 2000.

[23] A. B. Kahng, S. Mantik and I. L. Markov, "Min-max Placement For Large-scale Timing Optimization", in Proc. ACM/IEEE Intl. Symp. on Physical Design, pp. 143-148, Apr. 2002.

[24] Bill Halpin, C.Y. Roger Chen, and Naresh Sehgal, "Timing Driven Placement using Physical Net Constraints" Design Automation Conference, pp. 780-783. IEEE/ACM, June 2001.

[25] Yih-Chih Chou, Youn-Long Lin, "A performance-driven standard-cell placer based on a modified force-directed algorithm.", pp. 24-29, ISPD 2001.

[26] www.cbl.ncsu.edu/benchmarks/layoutsynth92/.

9. Author Contact information

Address for Intel Corporation authors (except Tal Shaked):
Intel Corporation MS SC12-606
2200 Mission College Blvd.
Santa Clara, CA 95052
USA

Karthik Rajagopal
Email: karthik.rajagopal@intel.com
Phone: 408 765 3222

Yegna Parasuram
Email: yegnashankar.parasuram@intel.com
Phone: 408 765 7864

Tung Cao
Email: tung.d.cao@intel.com
Phone: 408 653 4942

Amit Chowdhary
Email: amit.chowdhary@intel.com
Phone: 408 765 0721

Bill Halpin
Email: william.halpin@intel.com
Phone: 408 765 9867

Tal Shaked
Email: tshaked@cs.washington.edu
Department of Computer Science & Engineering
University of Washington
Box 352350
Seattle, WA 98195-2350