Interconnect-Driven Floorplanning with Fast Global Wiring Planning and Optimization *

Chin-Chih Chang, Jason Cong, David Zhigang Pan, and Xin Yuan
Department of Computer Science
University of California, Los Angeles, CA 90095
Email: {cchang, cong, pan, yuanxin}@cs.ucla.edu

Abstract

This paper presents an interconnect-driven floorplanning (IDFP) flow and algorithm integrated with multi-layer global wiring planning (GWP). It considers a number of interconnect performance optimizations during floorplanning, including interconnect topology optimization, layer assignment, buffer insertion, wire sizing and spacing. It also includes fast routability estimation and performance-driven routing for congestion control. Our experiments on the SUN picoJava-II^{TM} *core test circuit show that over 74% delay reduction can be achieved using our interconnect-driven floorplanner, compared to a conventional floorplanner without consideration of interconnect performance optimization/planning. We expect that IDFP with GWP will play a central role in designing interconnect-limiting, high-performance integrated circuits.*

1 Introduction

Global interconnect is commonly recognized as a key factor for designing high-performance integrated circuits, as VLSI process technology migrates into deep submicron (DSM) dimensions and operates in giga-hertz clock frequencies. By using a wide range of interconnect synthesis and optimization techniques, such as topology optimization, buffer insertion, layer assignment, wire sizing, and wire spacing, the performance of a global interconnect could be improved by a factor of 5 or more. In the current design flow, however, these interconnect optimizations are mainly used in the post-layout stages (e.g., global and/or detailed routing). As the global interconnects are largely determined by floorplanning, it becomes critical for floorplanning engines to be able to handle efficient interconnect planning and optimizations, so that the overall timing and design convergence can be achieved.

So far, very limited work has been done toward a better understanding of interconnect-driven floorplanning. The work in [1] presented a unified channeled-BSG (bounded-sliceline-grid) structure to combine floorplanning with routing. But it can only handle channel routing instead of the multi-layer, over-the-cell routing. In [2], a fast grid-based geometry routing is used in a simulated annealing-based floorplanning engine. But neither performance metric nor interconnect performance optimization was considered. In [3], buffer block planning (used for interconnect optimizations) problem was studied. The concept of *feasible region* was introduced and shown to be effective for planning buffer locations for a given floorplan. Yet the routability/congestion was not taken into account in [3]. The work of [3] was later extended by [4] to consider *independent feasible region* with routability consideration. However, none of these works considered multi-layer interconnect planning and optimization (e.g., layer assignment, wire sizing and spacing optimization).

In this paper we present an *interconnect-driven floorplanning* (IDFP) flow and algorithm integrated with multi-layer *global wiring planning* (GWP). The integrated framework emphasizes interconnect performance optimizations throughout the entire flow. We demonstrate our flow and algorithm using a simulated annealing-based floorplanning engine, tightly integrated with efficient GWP that considers a set of effective interconnect optimization techniques, including interconnect topology optimization, layer assignment, buffer insertion, wire sizing and spacing. We show that our IDFP significantly outperforms the conventional floorplanning by over 74% delay reduction, on a SUN *picoJava-*II^{TM} *core* microprocessor test circuit.

The remainder of this paper is organized as follows. Section 2 formulates the problem. Section 3 presents our integrated floorplanning and wire planning/optimization flow and algorithm. Experimental results are shown in Section 4, followed by conclusions in Section 5.

2 Problem Formulation

The purpose of this study is to investigate a unified flow of *interconnect-driven floorplanning* with consideration of *global wiring planning and optimization.* The problem is formulated as follows. Given a set of functional blocks (or modules), the number of available routing layers and their process parameters specified by a given technology, we want to compute, as shown in Figure 1: (i) the best locations and shapes of each functional block, and (ii) the global routing solution (including layer assignment, buffer location, wire sizing and spacing) for each net, and generation of buffer blocks (optional), such that a certain set of design objectives are optimized. The design objectives include not only those considered by traditional floorplanners, i.e., total wire length and chip area, but also the overall circuit performance (largely determined by the performance of global interconnects defined by a floorplanning solution) and routability/congestion optimization.

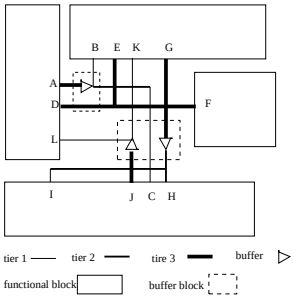

Figure 1: Output of IDFP is a floorplan with multi-layer global routing with buffers (or buffer blocks) generated. There are four global nets $\{A, B, C\}$, $\{D, E, F\}$, $\{G, H, I\}$, and $\{J, K, L\}$.

Because all existing benchmarks in the public domain are lack of path delay constraints, in this paper we assume a simple interconnect performance model. That is, the maximum delay for all nets

* This work is partially supported by Semiconductor Research Corporation under Contract 98-DJ-605.

is used to determine the overall clock frequency. This performance model is overly simplified and unrealistic, yet it still requires us to carefully address the interconnect planning problem during floorplanning and can be used as a "test vehicle" to drive our research on interconnect-driven floorplanning. Our primary goal is to minimize the maximum delay for *all* global nets defined by the floorplanning solution. Our secondary goal is to minimize the overall chip area and total wire length, while maintaining good routability.

We adopt the widely used Elmore delay model for interconnect and the switch-level RC model for buffers. Similar to [5], we define a *tier* to be a pair of adjacent metal layers with similar cross-sectional dimension. For the two layers in the same tier, one is mainly devoted for horizontal routing, and the other for vertical routing. Note that for a metal layer (or tier), one may choose to use wire width and space wider than the minimum for interconnect performance optimization.

3 Overall Flow and Algorithm

3.1 Overall Framework for Floorplanning and Interconnect Planning

Figure 2 shows the overall framework for our floorplanning and interconnect planning methodology. The key engine in this framework is an interconnect-driven floorplanner (IDFP) that fully incorporates the impacts from interconnect performance optimizations throughout the entire flow, using fast global wire planning for performance and/or routability optimization. Due to the inherent complexity of the IDFP problem, we use multi-stage, adaptive cost functions within IDFP to gradually consider more interconnect optimization, planning, and/or global routing features. Using the adaptive cost functions gives us flexibility to tune for different design objectives, and to tradeoff between performance and run time. After the "best" floorplan and its global routing (with interconnect optimizations built in) are obtained, some optional post-processing and refinement steps may follow to further improve the overall circuit performance and routability.

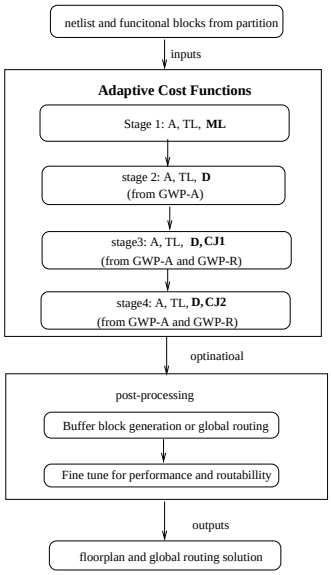

Figure 2: The overall flow of interconnect-driven floorplanning and global wiring planning and optimization.

Our floorplan engine is based on the simulated annealing algorithm and uses the Polish expression to represent the slicing floorplan structure [6]. Besides considering traditional measurements (such as total wire length and chip area) for a good floorplanning candidate, we focus on interconnect performance optimization and estimation with consideration of layer assignment, buffer insertion and wire sizing/spacing. We propose two phases of global wire planning to handle the performance evaluation: *global wire planning under area constraint* (GWP-A) and *global wire planning under routability constraint* (GWP-R). GWP-A will estimate the maximum net delay when considering layer assignment, buffer insertion and wire sizing/spacing, while GWP-R will estimate the routability for a given floorplan.

At high temperature, simulated annealing behaves more like a random walk and is less sensitive to the cost function. So we use some rough but fast estimations to let the simulated annealing engine quickly explore more solution space and find a good initial solution. When the temperature becomese lower, we need more accurate cost function to guide the search so that the solution are optimized toward our objectives. Therefore we divide the simulated annealing temperature region into four different stages from high to low temperatures. From Stage 1 to Stage 4, we use progressively more accurate interconnect performance and routability measurements as the temperature goes down.

- During Stage 1, we use a weighted cost function ψ of area (A), total wire length (TL), and the maximum length (ML) of a net, i.e., $\psi = A + \beta TL + \gamma ML$. We use half-perimeter metric to estimate the net length. The maximum wire length is used as a rough measure of the maximum interconnect delay, since for a long wire with optimal buffer insertion and wire sizing, its delay is a linear function of wire length [7].

- During Stage 2, we replace the maximum wire length by the maximum net delay D with consideration of layer assignment, buffer insertion, wire sizing and spacing optimizations for a given floorplan candidate. Then $\psi = A + \beta TL + \gamma D$. We have developed a fast GWP-A algorithm to compute D, and it works in linear time complexity in terms of total number of nets (to be explained in Section 3.2 in detail).

- During Stage 3, we further add the routability/congestion metric into the cost function by performing GWP-R. That is, $\psi = A + \beta TL + \gamma D + \delta CJ_1$, where CJ_1 denotes the congestion estimation using the Z-shape estimation approach (to be explained in Section 3.3). Note that during computation of D, GWP-A already takes care of the overall routing resource allocation, but it does not have "localized" congestion information. Our routability/congestion estimation is obtained by using a probabilistic model, similar to that developed in [2].

- During Stage 4, we perform a fast performance-driven, multi-layer global routing (to be explained in Section 3.3) to evaluate the routability of the final stage floorplanning solutions and obtain a more accurate total wire length based on the global routing solution. Then, $\psi = A + \beta TL + \gamma D + \delta CJ_2$, where CJ_2 denotes the average tile boundary congestion obtained by performing a global routing.

The parameters β, γ, δ are constants which control the relative importance of the terms in the cost function.

3.2 Global Wire Planning Under Area Constraint (GWP-A)

At Stage 2 of our interconnect-driven floorplanning engine, we perform a fast *global wire planning* for delay minimization under given area constraint (GWP-A). Given a floorplan candidate, we quickly determine from GWP-A (i) the overall performance estimation, (ii)

the wire width and spacing optimization for each layer, and (iii) the layer assignment (with consideration of optimal buffer insertion) for each net.

For simplicity of routing and via structure, we assume that each net is assigned to one routing tier during GWP-A, similar to [8, 9]. Also, we use single width/spacing for each tier, since compared to many-width wire sizing or tapering, it provides a much simplified routing architecture, yet obtains reasonable performance compared to that obtained by using many discrete wire widths, especially when buffer insertion is performed [8, 10]. In this study, we use the maximum net delay as a simple performance measure.

The overall algorithm for GWP-A combines a binary search (for the best delay target under given area constraint) and a greedy wire packing (under the given delay target for each net). The initial range for the binary search can be easily obtained using some minimum width/spacing setting. Then given a delay target, we will check the feasibility of every net meeting its delay constraint or not (under the given routing area constraints for each tier). This can be done by a layer assignment in a greedy bottom-up manner [9] by assigning short wires in lower metal layers and longer wires in upper layers. For each tier, we will find the best width/spacing pair (w_i^*, s_i^*) such that it can accommodate as many nets as possible while meeting both delay and routing area constraints. Note that there is a delicate balance in choosing the optimal (w_i^*, s_i^*). If (w_i^*, s_i^*) is small (e.g., minimum wire width and spacing), then the resulting delay (even after optimal buffer insertion) may be large, and not many interconnects may meet the delay constraint. In this case, the number of nets that can be assigned to a current layer is mainly constrained by delay. However, if (w_i^*, s_i^*) is too large, although the number of nets that could meet the delay constraint may increase, each wire now uses a larger area. In this case, the resulting number of nets that can be accommodated in the tier is mainly constrained by area.

For given wire width and spacing for each tier, we use an accurate capacitance look-up table to obtain unit length capacitance (including area, fringing and coupling components). From unit length resistance and capacitance (r, c), we use the closed form expression in [11] to compute the best delay by optimal buffer insertion. It first computes the optimal number of buffers.

$$k_{opt} = \left\lfloor -\frac{1}{2} + \sqrt{1 + \frac{2[rcl + r(C_l - C_b) + c(R_d - R_b)]^2}{rc(R_b C_b + T_b)}} \right\rfloor \quad (1)$$

and then plug $k = k_{opt}$ into the following optimal delay formula with k buffers inserted: [1]

$$\begin{aligned} T_k &= \frac{1}{k+1}[rl(kC_b + C_l) + cl(R_d + kR_b) \\ &\quad + (kC_b + C_l)(kR_b + R_d) + kT_b + \frac{rcl^2}{2} \\ &\quad - \frac{kr(C_b - C_l)^2}{2c} - \frac{kc(R_b - R_d)^2}{2r}] \end{aligned} \quad (2)$$

3.3 Global Wire Planning Under the Routability Constraint (GWP-R)

Given the outputs of the GWP-A algorithm, that is, the layer assignment results of all the nets with the maximum net delay and the wire width and spacing for each layer in a floorplan, the objective of GWP-R is to quickly evaluate the routability of this floorplan. We use the multi-layer routing model and assume that only one kind of wire can be used in each layer. We partition the floorplan into n by n tiles and evaluate the routability by estimating the routing congestion over the boundary of the tiles, i.e., CJ_1 and CJ_2 in the cost function. Since we already have a net's preferred routing tier provided by the GWP-A algorithm, we assume that all nets will be routed in the preferred routing tier as much as possible, and we allow a net to cross no more than two tiers without worsening the delay of the floorplan.

We adopt two kinds of approaches to estimate the routing congestion. The first approach is based on a stochastic approach similar to that developed in [2]. We assume all the nets will be routed using 2-bend wires (Z-shape) and can easily obtain the routing usage probability for each boundary of a tile and use the greatest tile boundary routing probability to estimate the routability. For multiple nets, we decompose them into 2-pin nets. We call this kind of estimation *Z-shape estimation.*

The second approach is to do a fast global routing using the graph-based A-tree algorithm[12] (GA-tree algorithm) to route nets one by one. The GA-tree algorithm works on a routing graph that represents the 3-dimension tile-layer structure. In a routing graph, a node represents a tile in a layer. If a wire can be routed through two adjacent tiles, then an edge is added to the corresponding nodes in the routing graph. So a routing edge actually corresponds to a boundary of a tile or a via, and the tile boundary congestion can be handled by assigning weight to the edges. Since we know the optimal wire width and spacing, we can derive the boundary routing capacity. Highly congested/overflowed routing edges can be assigned a large weight. The GA-tree algorithm will find a min-cost A-tree on the routing graph. By using a slope-like cost function for edge weight to penalize overflowed/highly congested edges and updating the weight after routing each net, we can get a global routing solution with congestion control without do having any iterations. For an edge in a routing tree, the routing congestion is defined as the number of nets crossing the corresponding tile boundary. We use the average routing congestion per routing edges in all net routing trees to estimate the overall routability. We call such approach *GA-tree estimation.*

For runtime purposes, we use the Z-shape estimation approach in Stage 3 to provide the CJ_1 in the cost function of floorplanning, and the GA-tree estimation approach in Stage 4 to provide the CJ_2 and more accurate total wire length TL. It is obvious that there is a tradeoff between runtime and accuracy of routability estimation. Z-shape estimation is faster but less accurate compared to GA-tree estimation.

4 Experimental Results

We have implemented the IDFP algorithm in C++ language and tested it on a 440 Mhz Sun Ultra 10 workstation with 256M memory. Our test circuit is generated from the Verilog source codes of the *picoJava-II*^{TM} *core* microprocessor from Sun Microsystems. We use the VIS system from UC Berkeley to convert the Verilog description to a gate-level descriptions. We break down the top level units of the picoJava-II to obtain a test case of 33 blocks and 3153 nets, of which 77% of the nets are 2-pin-nets.[2]

We estimate the area of a circuit by the following method. Since the number of two inputs gates needed to implement a combinatorial circuit is roughly as same as the number of literals in its disjunctive normal form representation, we will use the literal count as the gate count in our estimation. We use the area of a 2NAND gate ($1458\ \lambda^2$) and the area of a LATCH ($6561\lambda^2$) from a $0.25\mu m$ SCMOS cell library in MOSIS (*http://www.mosis.org*) to estimate the area for the two-inputs gates and latches in the design. Because we use the $0.10\mu m$ technology in ITRS'99 [13] for our experimental setting, the λ is set to $0.05\mu m$. We assume the total area occupied by a circuit is 3 times the area occupied by the gates and latches considering the routing area. The total estimated area is $31.6mm^2$.

[1] Note there is some typo in Eqn. (6) of [11].

[2] We ignore some simple gates in the floorplanner. We also exclude all the clock nets as they should be planned differently.

module	#IOs	area (μm^2)	module	#IOs	area (μm^2)
dcram_top	154	4.49e+4	smu_ctl	81	4.24e+4
icu_dpath	548	4.10e+6	icram_misc	46	4.41e+3
pipe	409	2.68e+5	dtag_misc	70	7.15e+3
itag_top	56	1.12e+4	dcram_misc	84	7.96e+3
dcu_dpath	577	5.30e+5	dcctl	157	1.25e+5
prils	159	2.62e+5	mantissa	470	1.24e+6
multmod	152	1.24e+6	code_seq	142	2.88e+6
exponent	94	7.38e+5	incmod	218	1.00e+6
nxsign	15	1.71e+4	rsadd	215	1.07e+5
ff_sr_2	5	426	ff_s	2	92
ff_s_5	10	465	pcsu_state	9	3.87e+3
smu_dpath	377	2.15e+5	icctl	112	2.92e+5
icram_top	115	1.69e+4	trap	87	4.04e+4
ex	1266	4.42e+6	ifu	371	7.03e+6
rcu	616	2.71e+6	hold_logic	41	4.13e+3
ucode	560	4.16e+6	dtag_top	101	2.75e+4
itag_misc	31	3.21e+3			

Table 1: Estimated areas of second-level modules in picoJava.

	FP in [6]	IDFP	Improvement
delay (ns)	0.476	0.123	74%
area(mm^2)	33.3	34.4	-3%
average congestion	125	84	33%
total wirelength(m)	8.10	3.58	56%
runtime(min)	43	194	—-

Table 2: Comparison between the conventional floorplanning and our IDFP on the test circuit picoJava-II.

Table 1 shows the number of IOs and the estimated areas of all the second- level modules of the picoJava II core.

The parameters used in our experiments are based on the $0.10\mu m$ technology in ITRS'99 [13]. The minimum widths (and spacings) for the local, semi-global, and global metal layers are 0.13, 0.17, and 0.28 μm, respectively. The sheet resistance for local, semi-global, and global tiers are 0.098, 0.046, 0.029, respectively. For the buffer, the intrinsic delay is 50ps, input gate capacitance is $7.2fF$, and output resistance is 234Ω. We set wire width to be equal to wire spacing for the same metal layer. Our wire width choices for each tier are from 1 to 6 x minimum width.

We compare our IDFP to the conventional floorplanning flow [6]. For the conventional floorplanning, the cost function consists of only area and total wire length, and neither interconnect performance metric nor interconnect performance/routability optimizations is considered. Then for both flows, we will use GWP-A and GWP-R followed by buffer insertion for each net to obtain the actual maximum net delay and routing congestion information.

Table 2 compares the maximum delay for all nets, chip area, average congestion, and total wire length of these two approaches. Compared to the conventional floorplanner, our interconnect-driven floorplanner considerably reduces the delay (for global interconnects) by 74%. It also reduces the congestion metric and total wire length by 33% and 56%. Our IDFP uses 3% more area than the conventional FP since the latter is more area-minimization driven.

5 Conclusions

In this paper, we have presented and demonstrated the effectiveness of a unified flow for interconnect-driven floorplanning (IDFP) with multi-stage, multi-layer global wiring planning (GWP). It efficiently incorporates into IDFP a number of interconnect performance optimizations, such as interconnect topology optimization, layer assignment, buffer insertion, wire sizing and spacing optimizations. It also includes fast routability estimation and performance-driven routing for routability and congestion control. Our experimental results show that over 74% delay reduction can be achieved by using this unified interconnect-driven floorplanner, compared to a conventional floorplanner without consideration of interconnect performance optimization/planning. We expect that IDFP with GWP will play a central role in designing interconnect-limiting, high-performance integrated circuits.

It shall be noted that the net-based delay model we adopt here is not practical and we use it mainly to demonstrate the necessity to consider inteconnect-driven floorplanning. We plan to study on the path-based delay model and obtain large circuits with path delay constraints from industry to further test our interconnect-driven floorplan flow.

References

[1] S. Nakatake, K. Sakanushi, Y. Kajitani, and M. Kawakita, "The channeled-BSG: A universal floorplan for simultaneous place/route with IC application," in *Proc. Int Conf. on Computer-Aided Design*, pp. 418–425, 1998.

[2] H.-M. Chen, H. Zhou, F. Young, D. Wong, H. H. Yang, and N. Sherwani, "Integrated floorplanning and interconnect planning," in *Proc. Int. Conf. on Computer Aided Design*, pp. 354–357, 1999.

[3] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning," in *Proc. Int. Conf. on Computer Aided Design*, pp. 358–363, 1999.

[4] P. Sarkar, V. Sundararaman, and C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," in *Proc. Int. Symp. on Physical Design*, 2000.

[5] J. Davis and J. Meindl, "Is interconnect the weak link?," *IEEE Circuits and Devices Magazine*, vol. 14, no. 2, pp. 30–36, 1998.

[6] D. Wong and C. L. Liu, "Floorplan design of VLSI circuits," in *Algorithmica*, pp. 263–291, 1989.

[7] J. Cong and D. Z. Pan, "Interconnect delay estimation models for synthesis and design planning," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 97–100, Jan 1999.

[8] J. Cong and D. Z. Pan, "Interconnect estimation and planning for deep submicron designs," in *Proc. Design Automation Conf.*, pp. 507–510, June 1999.

[9] P. Chong and R. K. Brayton, "Estimating and optimizing routing utilization in DSM design," in *Int. Workshop on System Level Interconnect Planning (SLIP)*, April 1999.

[10] C. J. Alpert, A. Devgan, and S. Quay, "Is wire tapering worthwhile?," in *Proc. Int. Conf. on Computer Aided Design*, pp. 430–5, 1999.

[11] C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proc. Design Automation Conf.*, 1997.

[12] J. Cong, A. Kahng, and K. Leung, "Efficient algorithm for the minimum shortest path steiner arborescence problem with application to VLSI physical design," *IEEE Trans. on Computer-Aided Design*, vol. 17, pp. 24–38, 1998.

[13] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 1999.