# Floorplanning Optimization with Trajectory Piecewise-Linear Model for Pipelined Interconnects[*]

Changbo Long Lucanus J. Simonson Weiping Liao Lei He
EE Department, University of California at Los Angeles
Los Angeles, CA, 90095

## ABSTRACT

Interconnect pipelining has a great impact on system performance, but has not been considered by automatic floorplanning. Considering interconnect pipelining, we study the floorplanning optimization problem to minimize system CPI (cycles per instruction) and in turn maximize system performance. We develop an efficient table-based model called trajectory piece-wise linear (TPWL) model to estimate CPI with interconnect pipelining. Experiments show that the TPWL model differs from cycle-accurate simulations by less than 3.0%. We integrate this model with a simulated-annealing based floorplan optimization to obtain CPI-aware floorplanning. Compared to the conventional floorplanning to minimize area and wire length, our CPI-aware floorplanning can reduce CPI by up to 28.6% with a small area overhead of 5.69% under 100nm technology and obtain better results under 70nm technology. To the best of our knowledge, this paper is the first in-depth study on floorplanning optimization with consideration of interconnect pipelining.

## Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

## General Terms

Algorithms, Design.

## Keywords

Floorplanning, pipeline, interconnect, piecewise-linear, performance

## 1. INTRODUCTION

Traditional high performance design separates the architectural optimization minimizing the average CPI (cycles per instruction) and the physical optimization maximizing clock frequency. This separation is based on the assumption that inter-block communication can be finished within one clock cycle. This assumption is expected to become invalid due to technology scaling in the future. In fact, the delay of a wire crossing a leading edge chip in ITRS 0.07$\mu m$ technology can be up to eight clock cycles [1]. Given this, it is highly possible that flip-flops are needed for inter-block communication. The number of flip-flops used for interconnect pipelining has been estimated at the full-chip level in [2, 3] and the number is expected to increase exponentially between generations of microprocessors [2].

We measure system performance by CPI (cycles per instruction). The lower the CPI, the higher the system performance. Pipelined interconnects impact the system performance significantly. For instance, increasing the latency between the fetch logic and L1 instruction cache from one cycle to two can increase the average CPI by 12.6% for the microprocessor similar to Alpha 21264 in Table 1. However, this impact has not yet been considered in physical design. Because floorplanning decides the length of interconnects between blocks and therefore decides interconnect pipelining, we study in this paper the floorplanning optimization problem to maximize system performance with consideration of interconnect pipelining.

The first contribution of this work is to develop an efficient yet accurate model for CPI estimation of out-of-order modern SuperScalar processors. Our model is based on the trajectory piecewise-linear (TPWL) approach originally developed for nonlinear circuits and micro-machined devices [4]. Existing work on CPI estimation relies upon techniques such as statistically generated synthetic benches [5] and statistical sampling of cycle accurate simulation [6]. These methods are too slow to be used in iteration based floorplanning optimization. Our model is table-based with one time use of a limited number of cycle accurate simulation, and has an average estimation error of less than 3.0% compared to cycle accurate simulation.

Our second contribution is to develop an efficient and effective floorplanner using the proposed TPWL model. Our experiment results for a microprocessor implemented in 100nm technology show that our floorplanning with consideration of pipelined interconnects achieves 28.6% CPI reduction compared to conventional floorplanning minimizing area and total wire length. Such CPI reduction is equivalent to a 40.0% increase in throughput for a fixed clock rate.

The rest of the paper is organized as follows. In Section 2 we present background information on architecture and floorplanning. In Section 3 and Section 4, we propose the trajectory piecewise-linear model for pipelined interconnect and CPI-aware floorplanning, respectively. We report the experiment results in Section 5 and conclude the paper in Section 6. More details of this study is

[*]This paper is partially supported by NSF CAREER award CCR-0093273, SRC grant HJ-1008, a UC MICRO grant sponsored by Analog Devices, Fujitsu Laboratories of America, Intel and LSI Logic, and a Faculty Partner Award by IBM. Address comments to lhe@ee.ucla.edu.

*DAC 2004,* June 7–11, 2004, San Diego, California, USA.
Copyright 2004 ACM 1-58113-828-8/04/0006 ...$5.00.

in a technical report [**?**].

# 2. BACKGROUND AND PROBLEM STATEMENT

## 2.1 Architecture and Partitioning

We demonstrate the effectiveness of our CPI-aware floorplanning on two out-of-order SuperScalar implementations of the MIPS instruction set. One is similar to the Alpha 21264 implemented in 100nm technology with an issue width of four and the other is implemented in 70nm technology with an issue width of eight. A summary of the microprocessor configurations is presented in Table 1. We group these modules into blocks that are each treated as an independent unit during floorplanning. We assume that interconnects between modules within the same block will not be pipelined. Blocks that are composed of multiple modules are the RUU block, including Register Update Unit and Load Store Queue, Decode block, including Fetch Queue and the Decoder, Branch block, including Fetch Unit and Branch Predictor, DL1 block, including the Level 1 Data Cache and the DTLB, and the IL1 block, including the Level 1 Instruction Cache and the ITLB. The L2 unified cache and all functional units are treated as independent blocks. We summarize the block area in Table 2.

| Technology | 100nm | 70nm |
|---|---|---|
| ISA | MIPS | MIPS |
| SuperScalar | Width 4 | Width 8 |
| Functional Units | 3 Integer ALU<br>1 Integer Mult.<br>1 FP Adder<br>1 FP Mult. | 6 Integer ALU<br>2 Integer Mult.<br>2 FP Adder<br>2 FP Mult. |
| Register Update Unit | 64 Instructions | 128 Instructions |
| Load Store Queue | 32 Instructions | 64 Instructions |
| Fetch Queue | 8 Instructions | 16 Instructions |
| Clock Frequency | 3 GHz | 6 GHz |
| FF Insertion Length | 2000$\mu m$ | 707$\mu m$ |

**Table 1: Processor used in experiments. The four-way SuperScalar is similar to Alpha 21264.**

| Block | Area ($mm^2$) | Block | Area ($mm^2$) |
|---|---|---|---|
| IALU | 1.00 | IMULT | 1.00 |
| F_ADD | 1.94 | F_MULT | 2.07 |
| RUU | 3.04 | Decode | 1.44 |
| Branch | 2.27 | L2 | 75.6 |
| IL1 | 8.99 | DL1 | 10.03 |

**Table 2: Area of logical blocks in 100nm technology. The area for 70nm technology is scaled down by a factor of $(10/7)^2$.**

## 2.2 Bus Latency Vectors

The latencies of the interconnects between two blocks in Table 2 are computed according to the Manhattan distance between the centers of two blocks in the floorplan. We treat the latency of each such interconnect as an independent variable. Changing the latency of one of these interconnects is effectively a change in the microarchitecture and will impact performance. In Table 3 we specify these interconnects with respect to their terminal blocks.

We define a bus latency vector ($\vec{B}$) for characterizing a floorplan as a vector containing the latency of each interconnect in Table 3. If, for a given floorplan, Bus 1 has a latency of 3, Bus 2 has a latency of 4, Bus 3 has a latency of 7 etc. The $\vec{B}$ for that floorplan would be $\vec{B} = \{3, 4, 7, ...\}$. These latencies can be determined from the floorplan by dividing the length of each interconnect by the flip-flop(FF) insertion length as computed by the simultaneous buffer and FF insertion algorithm presented in [3].

| Bus id | Terminal blocks | Bus id | Terminal blocks |
|---|---|---|---|
| 1 | IALU, RUU | 6 | IL1, L2 |
| 2 | IMULT, RUU | 7 | DL1, L2 |
| 3 | FPAdd, RUU | 8 | Branch, IL1 |
| 4 | FPMul, RUU | 9 | Decode, Branch |
| 5 | LSQ, DL1 | 10 | Decode, RUU |

**Table 3: Buses that potentially affect IPC.**

## 2.3 Cycle Accurate Simulation

For a given $\vec{B}$ we use out-of-order issue, cycle accurate simulation in the SimpleScalar 2.0 [7] framework to measure CPI for a subset of the SPEC2000 benchmark suite. The floating point benchmarks we use are $equake$ and $mesa$ while the integer benchmarks are $gzip$, $vortex$ and $mcf$. These benchmarks were chosen to be representative of a varied workload.

In order to summarize the performance of a floorplan as described by $\vec{B}$ we take the arithmetic mean of the CPI for above benchmarks. To obtain results more quickly we simulate truncated runs of twenty million instructions for each benchmark. During the initialization period of a program the cache behavior and instruction mix is not representative of its typical workload. For this reason the CPI of a benchmark is disregarded for the first ten million instructions of simulation. Implementation details to simulate interconnect latency will be included in a technical report as cycle accurate simulation is not the focus of this paper.

## 2.4 Floorplanning Formulation and Algorithm

Floorplanning determines the positions and shapes of blocks on a chip subject to the minimization of a cost function. In traditional floorplanning, the cost function is often based upon area and/or total wire length. In this paper, we additionally consider CPI, i.e., our objective is to minimize

$$W_{area} \cdot Area + W_{wire} \cdot wire\_length + W_{CPI} \cdot CPI,$$

where $Area$ and $wire\_length$ are the area and total wire length of the floorplan, respectively. We denote the objective of area, total wire length and CPI as $ALC$. We also denote the objective of area and total wire length as $AL$, and the objective of area and CPI as $AC$.

A category of floorplanning tools are based on simulated annealing (SA) [8, 9, 10]. SA starts with an initial floorplan and $moves$ to a new one by changing the positions or shapes of blocks. In each iteration the cost of the new floorplan is evaluated and the $move$ is accepted if the cost of the new floorplan is smaller than the old one. The $move$ may also be accepted regardless of cost with a probability dictated by the simulated “temperature” of the annealing. A $move$ that increases the cost is more likely to be accepted at a higher temperature. The temperature is decreased throughout the simulation based upon a schedule so that by the end only $moves$ that reduce the cost are likely to be accepted.

# 3. TRAJECTORY PIECEWISE-LINEAR CPI MODEL

In this paper, we propose the TPWL model for pipelined interconnects to estimate CPI for the following reasons. 1) The accuracy of the CPI estimation during the SA optimization process has a tremendous impact to the quality of the final floorplan, and the TPWL model is accurate with error less than 3.0%. 2) The TPWL model is analytical. Once the model is built using limited cycle accurate simulation the cost of an estimation is negligible.

## 3.1 Construction of the TPWL model

The construction of the TPWL model consists of the following phases. 1) $Sampling$: Assume the total number of buses is $n$, each bus latency vector $\vec{B} \in \mathbf{R}^n$ represents a floorplan and a point in the $n-$dimension space ($n = 12$ in this paper). The $\vec{B}$ points traversed by a SA process during floorplanning defines a trajectory within the solution space. We sample these points in a particular SA process to represent the trajectory with reduced complexity. Specifically, a SA process of floorplanning for $AL$ is conducted first. We construct the trajectory based on the moves in this SA optimization process.

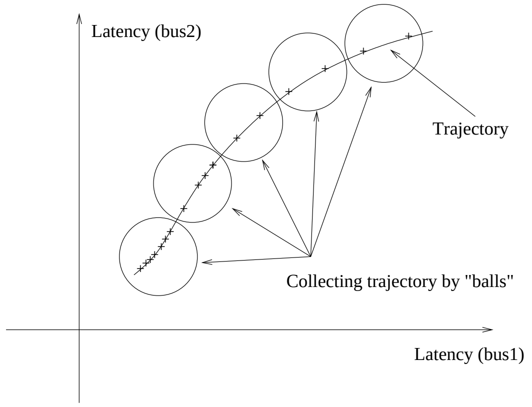


**Figure 1: Illustration of collecting trajectory in $2$-dimension space.**

2) $Collecting$: In the $collecting$ phase, we collect the points of the trajectory in as few "balls" as possible. Shown in Fig. 1 is an illustration of using "balls" to collect points in the 2-dimension space, where the x-axis represents the latency of $bus1$ and y-axis represents the latency of $bus2$.

We formulate the problem of minimizing the number of "balls" required to collect the points in a given trajectory as follows.

FORMULATION 1. **Trajectory points collecting (TPC)**: *Given a set of points $\mathcal{P} \subset \mathbf{R}^n$ and radius $r \in \mathbf{R}$, find $\mathcal{C} \subset \mathbf{R}^n$ with minimum $|\mathcal{C}|$ while satisfying*

$$\min_{c_j \in \mathcal{C}} \parallel p_i - c_j \parallel \leq r, \quad \forall p_i \in \mathcal{P}. \tag{1}$$

With respect to Fig. 1, $c_j$ is the center of a ball and minimum $|C|$ leads to the smallest number of balls. One can see that TPC is NP-hard via analogy to the $set\text{-}cover$ problem [11]. In this paper, we solve TPC by the greedy algorithm employed to solve the $set\text{-}cover$ problem [12, 13]. The idea is to consecutively find a ball $c$ which covers as many points in $\mathcal{P}$ as possible. We define a termination criteria $T$ such that when the number of remaining points is smaller than $T$, the algorithm terminates.

3) $Simulating$: By solving the TPC problem via the greedy algorithm, we obtain a set of points $\mathcal{C} \subset \mathbf{R}^n$, which covers most points in the trajectory with a given radius $r$. In fact, each point $c_j \in \mathcal{C}$ represents a bus latency vector $\vec{B}$, which specifies the latency for all buses in Table 3. We obtain CPI by cycle accurate simulation for each $\vec{B}$ (See Section 2.3) and build a CPI table indexed by the bus latency vector $\vec{B}$.

## 3.2 CPI estimation under TPWL model

Assume the size of $\vec{B}$ is $n$ and there are $m$ entries in the CPI table. The $i$th table entry is represented as $(\vec{B}_i, CPI_i)$. To compute $CPI$ for a particular $\vec{B}$, we first obtain the distance $d_i$ between $\vec{B}$ to each $\vec{B}_i$ as

$$d_i = \parallel \vec{B} - \vec{B}_i \parallel, \quad i = 1, \cdots, m. \tag{2}$$

Then, we compute the weight of each entry by $d_i$:

$$\hat{w}_i = e^{-\beta d_i / \overline{d}}, \quad i = 1, \cdots, m, \tag{3}$$

where

$$\overline{d} = \min d_i, \quad i = 1, \cdots, m. \tag{4}$$

Note that $\beta$ is a positive constant and is set to 25 in this paper[1]. Afterward, we compute the normalized weights as

$$w_i = \hat{w}_i / \sum_{i=1}^{|\mathcal{C}|} \hat{w}_i, \quad i = 1, \cdots, m. \tag{5}$$

Actually, from each entry in the CPI table, we can estimate the $CPI$ value for the target $\vec{B}$ by the first order expansion,

$$CPI^i_{\vec{B}} = CPI_i + \overrightarrow{\nabla CPI_i} \cdot (\vec{B} - \vec{B}_i), \tag{6}$$

where

$$\overrightarrow{\nabla CPI_i} = \begin{bmatrix} \frac{\partial CPI}{\partial \vec{B}(1)} \\ \vdots \\ \frac{\partial CPI}{\partial \vec{B}(n)} \end{bmatrix}_{|\vec{B}=\vec{B}_i}, \tag{7}$$

and $\frac{\partial CPI}{\partial \vec{B}(j)}$ is computed as follows

$$\frac{\partial CPI}{\partial \vec{B}(j)} = \frac{CPI(\vec{B}_i + \Delta) - CPI(\vec{B})}{\Delta}. \tag{8}$$

Note that $CPI(\vec{B}_i + \Delta)$ should be obtained from cycle accurate simulation. However, it is time-consuming to compute $\overrightarrow{\nabla CPI}$ for each entry in the CPI table. In this paper, we compute the average $\overrightarrow{\nabla CPI}$ and use it for all entries in the CPI table. The average $\overrightarrow{\nabla CPI}$ is obtained as follows:

$$\begin{aligned} \Delta_{max}(j) &= CPI(\vec{B}_{max-min}(j)) - CPI(\vec{B}_{max}), \\ \Delta_{min}(j) &= CPI(\vec{B}_{min}) - CPI(\vec{B}_{min-max}(j))), \\ \overrightarrow{\nabla CPI}(j) &= \frac{\Delta_{max}(j) + \Delta_{min}(j)}{2 \cdot D}, \quad j = 1, \cdots, n, \end{aligned} \tag{9}$$

where $\vec{B}_{max}$ and $\vec{B}_{min}$ is the bus latency vector with maximum and minimum latency on all buses, respectively. In this paper, we assume the maximum and minimum latency of a bus is 10 and 0, respectively, and denote the difference between them as $D$, i.e. $D = 10$. Also, $\vec{B}_{max-min}(j)$ is the same as $\vec{B}_{max}$ except the latency of the $j$th bus is the minimum. Similarly, $\vec{B}_{min-max}(j)$ is the same as $\vec{B}_{min}$ except the latency of the $j$th bus is the maximum.

The estimated $CPI$ value for $\vec{B}$ is computed as the weighted sum of $CPI^i_{\vec{B}}$, i.e,

$$CPI_{\vec{B}} = \sum_{i=1}^{m} w_i \cdot CPI^i_{\vec{B}}. \tag{10}$$

[1] This setting is proposed in [4].

For convenience, we summarize the computations to estimate CPI for a bus vector $\vec{B}$ as in Fig. 2.

**Compute CPI for $\vec{B}$**

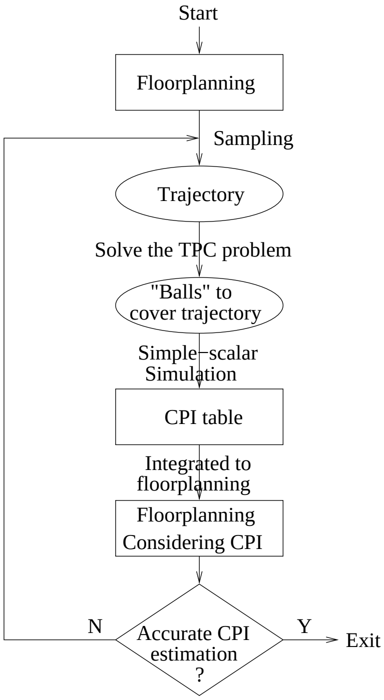
$d_i = \| \vec{B} - \vec{B}_i \|, \quad i = 1, \cdots, m.$
$\overline{d} = \min d_i, \quad i = 1, \cdots, m.$
$\hat{w}_i = e^{-\beta d_i / \overline{d}}, \quad i = 1, \cdots, m., \beta = 10.$
$CPI^i_{\vec{B}} = CPI_i + \overrightarrow{\nabla CPI} \cdot (\vec{B} - \vec{B}_i)$
$CPI(\vec{B}) = CPI_{\vec{B}} = \sum_{i=1}^{m} w_i \cdot CPI^i_{\vec{B}}.$

**Figure 2: CPI estimation under the TPWL model.**

# 4. CPI-AWARE FLOORPLANNING

We implement a prototype CPI-aware floorplanning tool based on the $Parquet$ package [10]. $Parquet$ employs SA to minimize the given floorplanning objective. In this section, we first introduce the overview of CPI-aware floorplanning and then focus on how to integrate the TPWL model with the SA optimization process of floorplanning. To improve the accuracy of the TPWL model, we also introduce an iterative TPWL (iTPWL) model.

## 4.1 Overview



Start
Floorplanning
Sampling
Trajectory
Solve the TPC problem
"Balls" to cover trajectory
Simple-scalar Simulation
CPI table
Integrated to floorplanning
Floorplanning Considering CPI
N
Accurate CPI estimation ?
Y
Exit

**Figure 3: Overview of CPI-aware floorplanning.**

Shown in Fig. 3 is the overview of the CPI-aware floorplanning process. It starts with a traditional floorplanning with objective of $AL$, and generates a trajectory. Then, the set of balls $\mathcal{C}$ is found by solving the TPC problem on the trajectory. Accurate CPI value for all balls in $\mathcal{C}$ are found by cycle accurate simulation to generate the CPI table.

As shown in [10], the objective of $AL$ is a linear combination of area and total wire length. In [10], after each move in SA, $\Delta$ of the objective is computed as the weighted sum of the changes of area and total wire length. In this paper, we optimize $ALC$ in the floorplanning by changing the objective function. Similar to [10], in our floorplanning, we compute $\Delta$ of the objective as the weighted sum of the changes in area, total wire length and CPI after each move. I.e.,

$$\Delta_{area} = \frac{area_{new} - area_{old}}{\sum area_{block}} \tag{11}$$

$$\Delta_{wire} = \frac{wire_{new} - wire_{old}}{wire_{old}} \tag{12}$$

$$\Delta_{CPI} = \frac{CPI_{new} - CPI_{old}}{CPI_{old}} \tag{13}$$

$$\Delta = w_{area} \cdot \Delta_{area} + w_{wire} \cdot \Delta_{wire} + w_{CPI} \cdot \Delta_{CPI}, \tag{14}$$

where $\sum area_{block}$ is the area of all blocks in the floorplan, and $w_{area}$, $w_{wire}$ and $w_{CPI}$ are the weights for area, total wire length and CPI, respectively. Note that in SA, a move is accepted if and only if

$$\begin{cases} \Delta < 0 & \\ R < exp(\frac{-\Delta \cdot t_i}{t_c}) & \Delta \geq 0 \end{cases} \tag{15}$$

where $R$ is a random value between 0 and 1, $t_i$ and $t_c$ are initial temperature and current temperature, respectively.

One may notice in Fig. 3 that when the CPI estimation is not accurate enough, we return to the sampling phase. Because the TPWL model is built upon the trajectory that is sampled from a SA optimization process with objective of $AL$, this trajectory might differ from the trajectory of the SA with objective of $ALC$. Under this circumstance, the CPI estimation by TPWL model may be inaccurate. Therefore, we use iteration to improve the accuracy of the TPWL model, and denote this as iterative TPWL (iTPWL) model.

The iTPWL model is based on the TPWL model and constructed by expanding the CPI table in each iteration. After building the CPI table for the TPWL model (we call it the first iteration), we employ this CPI table to estimate CPI and conduct SA with objective of $ALC$. A new trajectory is sampled from the SA optimization process and the CPI table is expanded by adding more entries obtained from the new trajectory[2]. Our experiment results will show that two extra iterations can improve the accuracy of the model and produce a considerably better final floorplan.

## 4.2 Justification of TPWL

The TPWL model was originally proposed to model nonlinear dynamic systems[4]. In [4], the trajectory is generated by performing a single simulation of the system for a fixed "training" input. At first glance, one may think the TPWL model can be accurate only around the trajectory. It is shown in [4], however, that in practice TPWL can easily outperform other recently developed techniques based on quadratic reduction.

TPWL model is suitable for our floorplanning optimization. The objective of traditional floorplanning is $AL$, and SA starts with an initial solution and reaches the high quality solution by following a trajectory of decreasing $AL$. With objective of $ALC$, this trajectory changes but remains close to the original because $AL$ is still part of the objective. Therefore, by tracing the original trajectory of SA which optimizes $AL$, we obtain a prediction of the trajectory that SA follows with consideration of CPI. When these two trajectories are close the TPWL model is very accurate. When they are not close we employ the iTPWL model. The iTPWL model iteratively employs an established CPI model and traces the trajectory of SA with consideration of CPI. The convergence of the trajectories in consequent iterations indicates a highly accurate model. Note

[2] A new entry with a small enough distance to an entry in the CPI table in fact is not calculated.

that we have improved the original TPWL model in the following aspects: 1) We introduce the TPC problem and solve it by applying the greedy $set\text{-}cover$ algorithm to reduce the number of "balls". 2) We expand the TPWL to iTPWL with considerable improvement in accuracy.

# 5. EXPERIMENT RESULTS

In this section, we compare the accuracy of the proposed TPWL and iTPWL models and the quality of the floorplans obtained using TPWL and iTPWL models. Finally, we report the running time to build the models. We use clock frequency and FF insertion length from Table 1.

## 5.1 CPI Models

The purpose of the CPI model developed in this paper is to estimate CPI accurately during a SA optimization process with consideration of CPI minimization. To verify the accuracy of the proposed model, we compare it with the CPI obtained by cycle accurate simulation. We present both the TPWL and iTPWL models to justify the cost of extra iterations in iTPWL. Note that each model generates a CPI table to estimate CPI, but the CPI table of the iTPWL model expands upon that of the TPWL model with each iteration.

When build up the TPWL and iTPWL models, the radius $r$ of "balls" is empirically decided. Small $r$ leads to large $|C|$ and large number of cycle-accurate simulations to construct the CPI table. In general, a bigger CPI table has better accuracy. Therefore, we choose $r$ as small as possible while the number of cycle-accurate simulations is acceptable in our experiment.

We conduct SA to minimize $AL$ and build up the CPI table of the TPWL model by sampling the SA process and running cycle accurate simulations. We repeat the SA process two more times with objective of $ALC$ to build the iTPWL model.

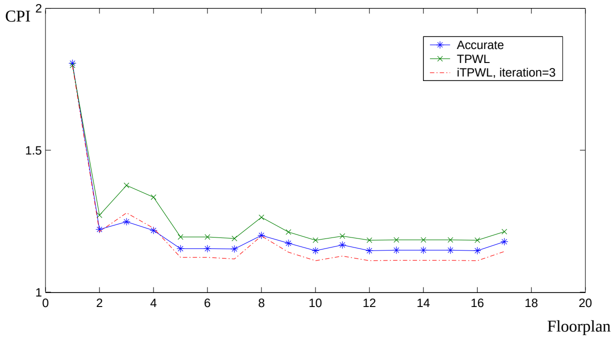


**Figure 4: TPWL model estimation results. (The error of the TPWL model and iTPWL model are 2.62% and 1.66%, respectively)**

We compare TPWL and iTPWL models with cycle-accurate simulations for floorplanning solutions selected from SA-based floorplanning. We select one solution from every 500 accepted moves, then calculate bus latency vector $\vec{B}$ and obtain CPI via TPWL, iTPWL and cycle accurate simulations. As shown in Fig. 4, for the processor in 100nm technology, the error of TPWL and iTPWL model are 2.62% and 1.66%, respectively. At the first glance, both models are accurate enough. However, we will show in Section 5.3 that the more accurate iTPWL model eventually leads to a considerably better floorplan.

## 5.2 Comparison of floorplanning with different objectives

We compare the floorplans obtained by SA subject to the objectives of $AL$, $AC$ and $ALC$. Note that the $AL$ objective is used by the traditional floorplanning and the area for each block in the floorplan is shown in Table 2. We summarize the results of floorplanning using these objectives in Table 4. For each objective we ran ten cases of floorplanning as the floorplanning algorithm in [10] is not deterministic. We choose the best solution for comparison.

We first show the white space rate of these floorplans in the first row. The low rate indicates the good quality of these floorplans. We show the CPI of floorplans in the second row. We observe that the $ALC$ and $AC$ solutions reduce CPI over $AL$ by 28.6% and 24.3%, respectively under 100nm technology and 56.8% and 60.9%, respectively under 70nm technology. It is observed that the total wire length of the $AL$ solution is significantly smaller than that of $AC$ in both cases. Together with the significantly larger CPI of $AL$ compared to $AC$, we conclude that minimizing total wire length does not necessarily maximize performance. Instead, latency of pipelined interconnects in CPI-critical paths should be considered to maximize performance. Similarly, the results in the fourth row show that minimizing total wire length does not necessarily reduce the total number of flip-flops and the maximum number of flip-flops in a single bus. We show the area in the fifth row. The $ALC$ results in the lowest CPI with only a small area overhead of 5.69% over $AL$ under 100nm technology and no area overhead but 18.3% wirelength increase under 70nm technology.

## 5.3 Comparison of floorplans under different CPI models

It has been shown in Fig. 4 that both TPWL model and iTPWL model are accurate. However, if these accurate CPI models are not available, heuristics such as minimizing the weighted sum of bus latencies, where the weight is the access ratio, can be employed. Note that access ratio is the fraction of clock cycles in which that bus is accessed during program execution. In this section, we compare the floorplan under TPWL, iTPWL and this heuristic based on bus access ratio.

Table 5 compares the three floorplans for the processor in 100nm technology. It can be seen that iTPWL model produces the best floorplan with a few percentage better result than the other two. But the floorplans obtained by both TPWL model and access ratio heuristics are acceptable.

| Metrics | Access ratio | TPWL | iTPWL |
|---|---|---|---|
| White Space (%) | 10.98 | 16.16 | 12.20 |
| CPI | 1.06 | 1.09 (+2.83%) | 1.00 (-5.66%) |
| TWL ($10^2$ $mm$) | 5.37 | 5.08 (-5.40%) | 5.08 (-5.40%) |
| Total/Max. #FF | 18/2 | 39/11 | 24/4 |
| Area ($10^2$ $mm^2$) | 1.28 | 1.36 (+6.25%) | 1.30 (+1.56%) |
| Error (%) | – | 2.62 | 1.66 |

**Table 5: Comparison between the floorplans obtained by access ratio heuristic, TPWL and iTPWL model with objective of minimizing area, total wire length and CPI simultaneously.**

## 5.4 Running time

Because the running time of SA for the floorplanning is negligible compared to that of cycle accurate simulation with over 20 million instructions we only present the number of simulations. Shown in Table 6 is the number of simulations required to build the TPWL and iTPWL models for each benchmark. Because iTPWL

| | | Objective | | | | | |
|---|---|---|---|---|---|---|---|
| | | 100nm | | | 70nm | | |
| Row | Metrics | $AL$ | $AC$ | $ALC$ | $AL$ | $AC$ | $ALC$ |
| 1 | White Space(%) | 7.47 | 5.45 | 12.2 | 16.7 | 15.5 | 16.7 |
| 2 | CPI | 1.40 | 1.06 (-24.3%) | 1.00 (-28.6%) | 2.71 | 1.17 (-56.8%) | 1.06 (-60.9%) |
| 3 | TWL($10^2$ $mm$) | 5.56 | 10.1 (+81.7%) | 5.08 (-8.63%) | 7.04 | 17.7 (+151%) | 8.33 (+18.3%) |
| 4 | Total/Max. #FF | 41/6 | 28/5 | 24/4 | 354/46 | 109/33 | 95/27 |
| 5 | Area($10^2$ $mm^2$) | 1.23 | 1.21 (-1.63%) | 1.30 ( +5.69%) | 7.66 | 7.56 (-1.31%) | 7.66 (-0.00%) |

**Table 4: Comparison between floorplans obtained by different objectives (CPI is estimated by iTPWL model).**

model is based on TPWL model, We show the simulation times in an accumulative fashion. For example, under 100nm technology we carried out 53 simulations to build the TPWL model. To build iTPWL model with one extra iteration, we need 20 more simulations. So, the simulation time is 73. For the third iteration, we need another 20 simulations, which increases the total number of simulations to 93. The number of simulations under 70nm is larger due to increased number of buses and larger solution space. It is much smaller than brute-force enumeration of ten latency cases for each of the twelve buses[3].

| | # Simple-scalar simulations | |
|---|---|---|
| | 100nm | 70nm |
| TPWL | 53 | 138 |
| iTPWL, i=2 | 73 | 186 |
| iTPWL, i=3 | 93 | 238 |

**Table 6: Simple-scaler simulation times to build up the TPWL and iTPWL model.**

## 6. CONCLUSIONS AND DISCUSSIONS

To consider the performance impact of pipelined interconnects, we have developed an efficient yet accurate iterative trajectory piecewise-linear (iTPWL) model with error less than 3.0% compared to cycle accruate simulation. We have also developed a floorplanning optimization minimizing CPI (cycles per instruction). Compared to the conventional floorplanning minimizing area and wire length, the new floorplanning formulation obtains a floorplan with 28.6% CPI reduction and a small area overhead of 5.69% under 100nm technology and even better result for 70nm technology. For a fixed clock rate, 28.6% CPI reduction is equivalent to 40.0% system throughput increase. To the best of our knowledge, this paper is the first in-depth study on floorplanning optimization with consideration of interconnect pipelining.

CPI optimization during floorplanning is achieved by shortening the lengths of CPI-critical buses. At first glance, the set of CPI-critical buses is a subset of all global interconnects and the traditional floorplanning objective of minimizing total wire length should lead to a floorplan with optimized system performance. However, we have shown in the experiment that minimizing total wire length does not necessarily lead to minimization of CPI. Furthermore, the accuracy of iTPWL model leads to floorplanning solutions with high quality and enables us to develop good heuristics minimizing CPI without explicit CPI calculation. We show that minimizing weighted wire length with bus access ratio as the weight is such a good heuristic. Since heavier interconnect pipelining is expected for future generation circuits and systems [2], we expect that floorplanning considering interconnect pipelining for CPI reduction will gain a growing importance.

A related work has studied micro-architecture and floorplanning co-optimization without considering interconnect pipelining [14]. A table-based CPI model in a brute-force fashion (i.e., cycle accurate simulation for each micro-architecture change) was employed. We intend to apply our iterative TPWL model to co-optimization of micro-architecture and floorplanning with interconnect pipelining. We expect that the effectiveness and efficiency of the iterative TPWL model will help to explore a bigger solution space and obtain better micro-architecture and floorplanning designs.

## 7. REFERENCES

[1] D. Matzke, "Will physical scalability sabotage performance gains?," *Computer*, vol. 30, pp. 37–39, 1997.

[2] P. Cocchini, "Concurrent flip-flop and repeater insertion for high performance integrated circuits," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 268–273, Nov 2002.

[3] W. Liao and L. He, "Full-chip interconnect power estimation and simulation considering concurrent repearter and flip-flop insertion," in *ICCAD*, pp. 574–580, 2003.

[4] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 155–170, 2003.

[5] S. Nussbaum and J. Smith, "Modeling superscalar processors via statistical simulation," in *International Conference on Parallel Architectures and Compilation Techniques*, pp. 15–24, 2001.

[6] R. Wunderlich, T. Wenisch, B. Falsafi , and J. Hoe, "Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling," in *International Symposium on Computer Architecture*, pp. 84–95, 2003.

[7] D. Burger and T. Austin, *The simplescalar tool set version 2.0*. University of Wisconsin-Madison, 1997.

[8] N. Sherwani, *Algorithms For VLSI Design Automation*. Kluwer, 3rd ed., 1999.

[9] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence pair," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1518–1524, 1996.

[10] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning through better local search," in *Proc. IEEE Int. Conf. on Computer Design*, pp. 328–334, 2001.

[11] S. Rajagopalan and V. Vazirani, "Primal-dual rnc approximation algorithms for (multi)-set (multi)-cover and covering integer programs," in *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pp. 322–331, 1993.

[12] D. S. Johnson, "Approximation algorithms for combinatorial problems," *J. Comput. Sys. Sci.*, vol. 9, pp. 256–278.

[13] L. Lovasz, "On the ratio of optimal integral and fractional covers," *Discrete Math.*, vol. 13, pp. 383–390.

[14] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis, "Microarchitecture evaluation with physical planning," in *Proc. Design Automation Conf*, pp. 32–35, 2003.

[3] $10^{12}$ times of cycle accurate simulations are needed for enumeration.