Microarchitecture Configurations and Floorplanning Co-Optimization

Changbo Long, Lucanus J. Simonson, Weiping Liao and Lei He *Member, IEEE,*

***Abstract*—Microarchitecture configurations and floorplanning are keys to boost throughput, and they are strongly related. In this paper we propose a new method to optimize them simultaneously. We first concentrate on floorplanning under given microarchitecture configurations. In addition to the objectives of conventional floorplanning methods, we minimize the throughput degradation caused by pipelined global interconnects based on efficient yet accurate models for microarchitecture throughput over pipeline stages of global interconnects. Our results show that an accurate trajectory piecewise-linear (TPWL) model incurs more offline setup time to obtain 13% better throughput than a rough access ratio based model, and both models lead to much better throughput (up to 64% higher) compared to conventional floorplanning methods. We then build a unified throughput model parameterized for pipelined global interconnects and microarchitecture configurations based on the TPWL method, and apply this model to efficiently explore over one million microarchitecture configurations and corresponding floorplan variations. We obtain microarchitecture configurations and floorplans with throughput 26.9% better than manually chosen microarchitecture followed by automatic floorplanning in a very recent paper[1].**

***Index Terms*—Floorplanning, pipeline, interconnect, piecewise-linear, performance**

I. Introduction

The throughput of a common computer system is the product of average instruction-per-cycle (IPC) and clock rate. To boost throughput, both microarchitecture configuration and floorplanning that are strongly related need to be optimized. The microarchitecture configuration, including choosing issue width, branch prediction method, cache/TLB sizes and the number of functional units directly determines IPC. For example, as shown in [2], resizing critical components improves throughput over 20% for Power5 processors. Microarchitecture configuration also decides the area of components in physical layout and thus affect the floorplan of the microarchitecture. The floorplan of a microarchitecture, on the other hand, not only determines the clock rate of the microarchitecture but also has a significant impact to IPC due to pipelining of global interconnects[1], [3], [4]

The traditional design flow, however, separates microarchitecture tuning and floorplan optimization. IPC was improved by the means of microarchitecture configuration alone without considering floorplanning, and floorplanning was employed only to determine the clock rate. Consequently, this separation may lead to inferior designs. Because it is imperative that multiple stages for global interconnects will be adopted in the future[5], [6], [7], [8], it is necessary to optimize microarchitecture configuration and floorplanning simultaneously in order to avoid the throughput degradation caused by the separation of the design flow.

In this paper we develop a method to optimize microarchitecture configuration and floorplanning simultaneously to maximize throughput for SuperScalar-like [9], [10] microarchitecture. Firstly, we concentrate on floorplanning under given microarchitecture configurations. In addition to the objectives of conventional floorplanning methods, we minimize the throughput degradation caused by pipelined global interconnects as well. The key is to develop efficient yet accurate models for microarchitecture throughput over pipeline stages of global interconnects during floorplanning. Note that the accurate evaluation of throughput for a microarchitecture requires cycle-accurate simulations over a set of benchmarks and one simulation lasts for hours. A fast throughput model is essential during floorplanning. For this purpose we propose the trajectory piecewise-linear (TPWL) model for CPI (cycle-per-instruction)[1] over pipeline stages of global interconnects. Our results show that the TPWL model needs more offline setup time but obtains 13% higher throughput than a rough access ratio based model. The floorplanning approach based on the two models can improve the throughput of microarchitecture by up to 64.7% (for the TPWL model) than conventional floorplanning methods without considering the influence of pipelining global interconnects.

Secondly, we build a unified throughput model parameterized for pipelined global interconnects and microarchitecture configurations based on the TPWL method, and then applied this model to efficiently explore over one million microarchitecture configurations and corresponding floorplan variations. This is in sharp contrast with existing works [11], [1], which enumerated a limited number of microarchitecture configurations. Our experiments show that the average error of the TPWL model is about 3.0% compared to cycle-accurate estimations. We obtain microarchitecture configurations and corresponding floorplans less than 20.0% from the ideal IPC. Also, our solutions are 26.9% better than the manually chosen configurations in [1].

The idea of the TPWL model, as originally proposed in [12] to model nonlinear dynamic systems, is to build a piecewise-

Changbo Long is with Synopsys inc. Mountain View, CA 94043 (e-mail: longchb@synopsys.com). Lucanus J. Simonson is with Intel inc. Weiping Liao is with Advance Micro Device. Lei He is with electrical engineering department, University of California, Los Angeles, CA, 90095 (e-mail: lhe@ee.ucla.edu)

Manuscript received September 8, 2005, revised February 8 and September 10, 2006. This paper was supported in part by the National Science Foundation under CAREER Award CCR-0401682 and SRC Grant 1116.

[1]By practical consideration we assume clock rate to be concrete and explore them one by one. Under each clock rate, CPI fully represents throughput. On average, we assume $CPI = 1/IPC$.

linear model along the trajectory of a typical system response excited by a "training input". This model is particularly accurate to model system responses that have a trajectory close the one of the "training input". In Section III we will show that the floorplanning optimization based on simulated annealing (SA) can be mathematically described by a form similar to nonlinear systems described as state space approaches in [12]. We will also show that multiple SA runs starting with different initial floorplans have close trajectories which enables a highly accurate TPWL model. Moreover, we have improved the original TPWL model from [12] for higher accuracy.

Related to our work, the floorplanning for interconnect optimization has been studied for ASIC and System-On-Chip (SOC) designs. Early work on interconnect-driven floorplanning [13], [14], [15], [16] for ASIC chips focus on buffer block planning without considering interconnect pipelining. More recently, [17] minimizes the degradation of throughput caused by pipelined interconnects by adding throughput as one of the objectives during floorplanning in SOC designs. In [17], the throughput of SOCs are normalized values rather than specified throughput values in this paper.

The floorplanning optimization for microarchitecture has also been studied. [11] developed a primitive co-optimization method for microarchitecture configuration and floorplanning without considering interconnect pipelining. Specifically, the IPC values of 32 microarchitecture configurations are obtained by cycle-accurate simulations and then stored in a look-up table. The best configuration is obtained by comparing throughput of these configurations and corresponding floorplans. Microarchitecture floorplanning with interconnect pipelining was studied by [1] and the earlier version [3] of this paper. [1] profiled module-to-module communication and solved an interconnect-pipelining aware floorplanning using mixed integer non-linear programming (MILP). Iterations between profiling and MILP are needed to guarantee the convergence of the overall design flow. Again, the microarchitecture configurations were limited as only four candidates were considered in the paper. [18] evaluated a bus-driven floorplanning method to optimize the routability and timing of buses in a given microarchitecture configuration. To solve the microarchitecture floorplanning problem, [3] has proposed a TPWL model for interconnect pipelining, which has been summarized in this paper. Also, in this paper we propose a unified TPWL model parameterized with respect to interconnect pipelining and microarchitecture configuration, which enables us to optimize microarchitecture configuration and floorplanning simultaneously. More recent related work after the development of the method proposed in this paper will be summarized in conclusion.

In the rest of this paper, we introduce background knowledge in Section II, and present the TPWL model in Section III. We develop methods for microarchitecture floorplanning considering pipelined interconnects and present experiment results in Section IV. We present the co-optimization of microarchitecture configuration and floorplanning with experiment results in Section V, and conclude the paper in Section VI.

II. Background

A. Bus Latency Vectors

We assume an out-of-order SuperScalar implementation of the MIPS instruction set. For microarchitecture floorplanning under a given configuration, we summarize the configuration in Table I, which is similar to the Alpha 21264 with an issue width of four. We group the modules in this implementation 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.

Generation	100nm
ISA	MIPS
SuperScalar	Width 4
Functional Units	3 Integer ALU 1 Integer Mult. 1 FP Adder 1 FP Mult.
Register Update Unit	64 Instructions
Load Store Queue	32 Instructions
Fetch Queue	8 Instructions
Clock Frequency	3 GHz
FF Insertion Length	$2000\mu m$

TABLE I
THE CONFIGURATION SIMILAR TO ALPHA 21264 IS USED FOR MICROARCHITECTURE FLOORPLANNING.

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 II, which is obtained by scaling the area of corresponding components in Alpha 21264 to the ITRS[8] 100nm generation.

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 II
AREA OF LOGICAL BLOCKS IN THE 100NM GENERATION.

The lengths of interconnects between two blocks in Table II are computed according to the Manhattan distance between the centers of two blocks in the floorplan[2]. 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 the performance. In Table III we specify these interconnects with respect to their terminal blocks[3].

[2]Note that our method can be applied to the exact bus length and forked bus if such design information is provided.

[3]L2 cache is composed by three banks in our experiment and we consider the worst case latency.

Bus id	Terminal blocks	Access ratio
1	IALU, RUU	0.953
2	IMULT, RUU	0.002
3	FPAdd, RUU	0.137
4	FPMul, RUU	0.078
5	LSQ, DL1	0.409
6	IL1, L2	0.001
7	DL1, L2	0.046
8	Branch, IL1	0.391
9	Decode, Branch	0.390
10	Decode, RUU	0.390

TABLE III
BUSES THAT AFFECT CPI.

We summarize all the interconnects that affect CPI in Table III and form them into a vector $\vec{B}$, called as bus latency vector, which is used to characterize a floorplan. For example, in a floorplan if 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 the floorplan would be $\vec{B} = \{3, 4, 7, ...\}$. The latency of each interconnect is obtained by dividing the total wire length of the interconnect, measured from the floorplan, by a constant value called flip-flop (FF) insertion length, which is computed based on the simultaneous buffer and FF insertion algorithm proposed in [7].

B. Cycle accurate simulations and CPI metrics

To measure the impact of pipelining stages of global interconnects to CPI we use out-of-order issue, cycle-accurate simulations in the SimpleScalar 3.0 [19] framework. The latencies (stages) of various global interconnects, which are obtained from the floorplan and recorded in the bus latency vector $\vec{B}$, are first specified in SimpleScalar. In some cases these latencies can be specified by simply modifying the configuration file of SimpleScalar. The interconnect between L1 and L2 data cache is a good example. Note that because the interconnect lengths are different for L2 instruction cache and L2 data cache the miss penalties for them are different. In more complicated cases we insert queues between modules to buffer data to realize these latencies. These cases include the buses for instruction L1 cache, fetch to dispatch, and dispatch to issue. Specifically, the L1 instruction cache interconnect latency is modeled by a FIFO queue placed between the fetch unit and the cache. A branch cannot be identified by the fetch unit until it moves through the queue, therefore prefetching proceeds speculatively to consecutive memory locations. When a branch is taken the prefetched contents of the queue are flushed and fetch proceeds from the target location. The latencies between the fetch and dispatch units and between the dispatch and issue units are modeled by a FIFO queue between the fetch and dispatch units with length equal to the sum of the latencies of the two buses. This is because the dispatch stage is completely self contained and the effect of latency that comes immediately before dispatch is identical to that of latency that comes immediately after dispatch.

After all interconnect latencies in a bus latency vector $\vec{B}$ are specified in SimpleScalar, the CPI value of this vector is computed by the arithmetic mean of the CPI of ten benchmarks, including *equake, mesa, gzip, art, bzip2, parser, vpr, gcc, go* and *mcf*, representing both integer and floating-point workloads. The CPI value of each benchmark is obtained by cycle-accurate simulations, where the first 200 million instructions are fast-forwarded[4] and the next 100 million instructions are actually simulated. Fast-forwarding the first 200 million instructions is to skip the initial false setup stage and warm up the architecture structure such as caches and branch predictors, which improves the accuracy of CPI measurement. Simulating the next 100 million instructions only is to improve the efficiency of the measurement because it is long enough to obtain a steady-state estimation.

C. Floorplanning

The method for microarchitecture floorplanning used in this paper is based on traditional floorplanning approaches. The objective of traditional floorplanning is to determine the positions and shapes of blocks in a chip subject to the minimization of a cost function, which is usually a combination of area and total wire length. This is usually presented in the form

$$\alpha \cdot \frac{area}{area_{norm}} + \beta \cdot \frac{wire\_length}{wire\_length_{norm}}, \quad (1)$$

where $area$ and $wire\_length$ are the area and total wire length of the floorplan, respectively, α and β are user-defined weights. Because the metrics of $area$ and $wire$ are in different magnitude, they are normalized by typical values ($area_{norm}$ and $wire\_length_{norm}$) in the objective function.

A widely used floorplanning approach is based on simulated annealing (SA)[20], [21], [22]. 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 unconditionally accepted if the cost of the new floorplan is smaller than the old one. The $move$ may also be accepted if the cost increases but 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 annealing based upon a schedule so that by the end only $moves$ that reduce the cost are likely to be accepted.

III. TRAJECTORY PIECEWISE-LINEAR MODEL

A. Overview of the TPWL model

The TPWL model was originally proposed to model nonlinear dynamic systems [12] such as the following one described by a state-space approach

$$\begin{cases} \frac{dg(x(t))}{dt} = f(x(t)) + B(x(t))u(t) \\ y(t) = C^T x(t) \end{cases} \quad (2)$$

where $x(t) \in R^N$ is a vector of states at time t, $f : R^N \to R^N$ and $g : R^N \to R^N$ are nonlinear vector-valued functions. B is a state-dependent $N \times M$ input matrix, $u : R \to R^M$ is an input signal, C is an $N \times K$ output matrix and $y : R \to R^K$ is

[4]Fast-forwarding is an option provided by SimpleScalar which skips a specified number of instructions by using functional simulation before starting cycle-accurate simulations to reduce runtime.

the output signal. In essence, the TPWL model is a weighted combination of linear models at different linearization points in the state space, say x_0, ..., x_{s-1}, where the key is how to find these linearization points.

By Taylor's expansion, it is clear that the weighted combination of linear models at linearization points of x_0, ..., x_{s-1} would be accurate for a given state x, which needs to be evaluated, if it is close to any of x_0, ..., x_{s-1}. However, it is difficult to obtain linearization points x_0, ..., x_{s-1} to guarantee that there is a "close" point for any given state x. Therefore, [12] proposed to perform a single simulation of the nonlinear system for a fixed "training input", $u(t)$, and initial state x_0, and find the linearization points x_0, ... x_{s-1} along the trajectory of this simulation. The reasoning is that system responses may have similar trajectories and the trajectory of a "training input" can guide us to find linearization points x_0, ..., x_{s-1} that are close to the points on the trajectories of other inputs. Experiment results show that this TPWL model is highly accurate, especially for trajectories close to the trajectory of the "training input".

In this paper we adopt the TPWL approach to model CPI over pipelining stages of global interconnects during floorplanning. Similar to the state-space approach, the SA optimization process could be described as

$$\begin{cases} \vec{B}(n+1) = \vec{B}(n) + \Delta\vec{B}(n) + u(n) \\ CPI(n) = g(\vec{B}(n)) \end{cases} \tag{3}$$

where the moves in SA are labeled by numbers of 1, 2, ..., and, $\vec{B}(n)$ is the bus latency vector of the floorplan after n moves, $\Delta\vec{B}(n)$ is the change to $\vec{B}(n)$ in the $n+1$ move which is caused by the change to the floorplan, $u(n)$ is the initial floorplan where $u(0)$ is the bus latency vector of the initial floorplan before SA starts and $u(n) = 0, \forall n > 0$. The CPI value of the floorplan after n moves is a function of $\vec{B}(n)$, which is represented by $g(\vec{B}(n))$. Similar to [12], we perform a single-start SA run for a fixed initial floorplan, which is the "training input", to obtain linearization points $\vec{B}_0, ..., \vec{B}_{s-1}$ along the trajectory by treating each move in SA as a state in the nonlinear system. The weighted combination of linear models at these linearization points $\vec{B}_0, ..., \vec{B}_{s-1}$ is the TPWL model for floorplanning. We can see that this TPWL model would be accurate if the trajectories of SA starting from other initial floorplans are close to the trajectory of the "training input".

Fortunately, the trajectories starting from different initial floorplans are close to each other in floorplanning. We illustrate the trajectory of SA for microarchitecture floorplanning in Fig. 1 (a). We are particularly interested in the trajectory of latencies for buses because these latencies impact the throughput. We represent these latencies by the distance between the corresponding bus latency vector (as $\vec{B}$ defined in Section II-A) and the original point [5] in the Y axis. Although this distance metric cannot fully represent bus latency vectors[6], it serves as a good example to illustrate the trend of bus latency vector changes in the SA procedure. The X axis shows the process of SA procedure, where 0% and 100% represents the beginning and end of the SA procedure, respectively. Fig. 1 (a) shows that the trajectory becomes greatly concentrated at the end of SA. This is because the temperature of SA is low at the end and there are smaller number of moves accepted to change the floorplan compared to a high temperature at the beginning stage of the SA procedure. This point has been further demonstrated in Fig. 1 (b), which shows the distribution of latencies in the SA trajectory. The latencies are heavily concentrated in a relatively small range. As shown in the figure, 86% of them fall between the value of 8 and 16. Because the SA procedure at the lower temperature explores a relatively small and concentrated solution space as shown above, employing the TPWL model to build a model just in the vicinity of the solution space explored by SA is likely more effective than models targeting at the whole solution space such as design of experiments in [4][7].

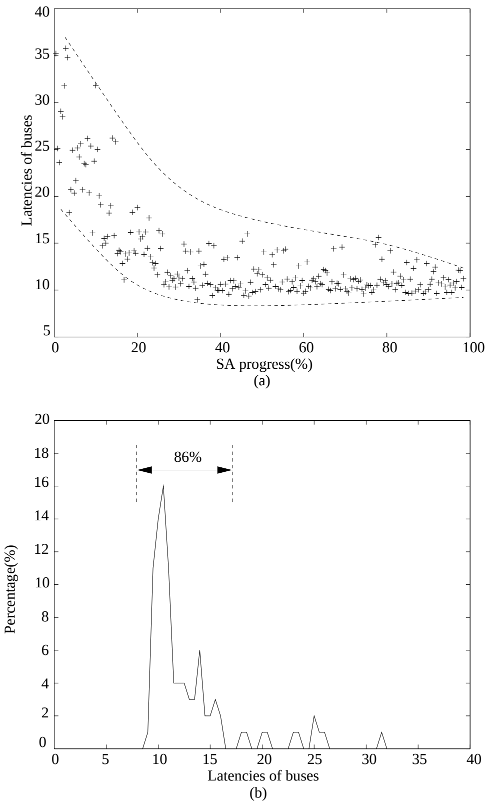


Fig. 1. (a) The trajectory of SA. (b) Distribution of latencies of buses. 86% of them have total bus latencies between 8 and 16.

In this work we improve the original TPWL model by introducing a TPC problem and iterations. The TPC problem reduces the cost of building the piecewise-linear model by sampling the trajectory, then collecting key points among those sampled points as discussed in the following section. The strategy to build the TPWL model based on several trajectories from multiple SA starts of floorplanning optimization, called

[5]The original point means that all bus latencies are zero.

[6]Two different bus latency vectors may have the same distance to the original point.

[7]Note that the TPWL and design of experiments are orthogonal to each other and can be combined.

as iteration, is used because the trajectory of floorplanning is subject to small changes when the CPI metric is added into the objective function of floorplanning. This improves the accuracy of the CPI model is improved (Please refer to Section IV-B for the details of iterations).

B. Construction of the TPWL model

The TPWL model is built in three phases.

1) Sampling: Each move in SA explores a new floorplan by a small change to the current one, and these explored floorplans define a trajectory in the solution space. We capture the trajectory by sampling one floorplan in every n $(n \geq 1)$ consecutive moves in SA. By sampling it is meant that the bus latency vector $\vec{B}$ is extracted from the floorplan and will be used in the next phase. Note that a large n reduces the costs but may lose the details of the trajectory. In order to obtain a good trade-off between cost and performance we assume $n = 2$ or 3 depending on the size of the floorplan.

2) Collecting: To capture the trend of the trajectory with as few as possible sampling points we perform the *collecting* phase. Intuitively we describe this phase as using as few as possible "balls", which are defined as spherical areas in the solution space where any point inside these areas has a distance to the center smaller than a given radius, to cover all the bus latency vectors obtained in the *sampling* phase. The center points of these balls are used to represent all others in the same ball and will be used in the simulation phase while all others are discarded. We formulate the problem as follows.

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

$$\min_{c_j \in \mathcal{C}} \| p_i - c_j \| \leq r, \quad \forall p_i \in \mathcal{P}. \tag{4}$$

Note that c_j is the center of a ball and the minimum $|\mathcal{C}|$ leads to the smallest number of balls. The TPC problem can be rephrased as follows.

Formulation 2: Given a set of points $\mathcal{P} \subset \mathbf{I}^n$ and radius $r \in \mathbf{I}$, $\mathcal{P}_i$ for each $p_i \in \mathcal{P}$ contains all points p_j satisfying

$$\| p_j - p_i \| \leq r, \quad \forall p_j \in \mathcal{P}, \tag{5}$$

find the smallest number of sets $\mathcal{P}_i$ to cover all points in $\mathcal{P}$.

One can see that the re-phrased TPC problem falls into the category of *set-cover* problem. We adopt the greedy algorithm proposed in [23], [24] to solve the TPC problem. The idea is to iteratively find a ball c which covers as many points in $\mathcal{P}$ as possible. The implementation details are similar to those in [23], [24].

The TPC problem was not explicitly presented in [12]. As one of the contributions of this paper to improve the TPWL approach, the introduction of the TPC problem can significantly improve the efficiency of the model.

3) Simulation: From the collecting phase, we obtain a set of bus latency vectors $\mathcal{C} \subset \mathbf{I}^n$, which are the center points of the "balls" representing the trend of the trajectory. Each point $c_j \in \mathcal{C}$ is a bus latency vector $\vec{B}$ and the corresponding CPI can be estimated by cycle-accurate simulations (See Section II-B). We call the phase to simulate all these bus latency vectors in $\mathcal{C}$ to obtain corresponding CPIs the *simulating* phase. These bus latency vectors and corresponding CPI values form a table, called CPI table, which is used to evaluate the CPI value for any given bus latency vector $\vec{B}$ described in the following sub-section.

C. CPI estimation under the TPWL model

By Taylor's expansion, the CPI value of any given bus latency vector $\vec{B}$ could be estimated from each entry in the CPI table

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

where $\vec{B}_i$ stands for the i^{th} entry in the CPI table and

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

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

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

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 that the latency of the j^{th} bus is the minimum. Similarly, $\vec{B}_{min-max}(j)$ is the same as $\vec{B}_{min}$ except that the latency of the jth bus is the maximum. It can be seen from above that $\overrightarrow{\nabla CPI}(j)$ actually represents the sensitivity of the bus.

The final estimation is computed as the weighted sum of $CPI^i_{\vec{B}}$,

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

To determine the weight w_i for each entry in the CPI table we follow the method adopted in [12]. We first compute the distance between each entry and $\vec{B}$, and then employ an exponential function of the distance as a weight to compute the average estimation. The distance d_i between $\vec{B}$ to each $\vec{B}_i$ is computed as as

$$d_i = \| \vec{B} - \vec{B}_i \|_2, \quad i = 1, \cdots, m, \tag{11}$$

where m is the size of the CPI table. Then, we compute the weight of each entry by d_i:

$$\hat{w}_i = e^{-\beta d_i/\overline{d}}, \quad i = 1, \cdots, m, \tag{12}$$

where

$$\overline{d} = \min d_i, \quad i = 1, \cdots, m. \tag{13}$$

Note that β is a positive constant and is set to 25 [12]. Afterward, we compute the normalized weights as

$$w_i = \hat{w}_i / \sum_{k=1}^{|\mathcal{C}|} \hat{w}_k, \quad k = 1, \cdots, m. \tag{14}$$

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

Computing CPI for $\vec{B}$

```
$d_i = \| \vec{B} - \vec{B}_i \|, \quad i = 1, \cdots, m.$
$\overline{d} = \min d_i, \quad i = 1, \cdots, m.$
$\hat{w}_i = e^{-\beta d_i/\overline{d}}, \quad i = 1, \cdots, m., \beta = 25.$
$CPI^i_{\vec{B}} = CPI_i + \overrightarrow{\nabla CPI} \cdot (\vec{B} - \vec{B}_i)$
$CPI(\vec{B}) = CPI_{\vec{B}} = \sum_{i=1}^{m} w_i \cdot CPI^i_{\vec{B}}.$
```

Fig. 2. CPI estimation under the TPWL model.

As stated in [12], computing the weights based on an exponential function of distance is a simple heuristic. However, it is suitable for CPI which is a strong nonlinear function of bus latencies. The estimation based on the Taylor's expansion from a CPI table entry which is not close to the target bus latency vector can be error-prone because of this nonlinearity. Therefore, the exponential weight function is more accurate than others such as linear functions, because table entries that are close to the target bus latency vector contribute to the estimation.

IV. Microarchitecture floorplanning considering pipelined interconnects

In this section, we study microarchitecture floorplanning to minimize the impact of interconnect pipelining to CPI for given microarchitecture configurations. In order to model the impact of interconnect pipelining on CPI we employ the TPWL and access ratio based model to be presented in this section and then compare them.

A. Microarchitecture floorplanning

As shown in Section II-C, the objective of traditional floorplanning is the weighted sum of area and total wire length. To consider the microarchitecture performance during floorplanning, we add CPI to the objective function and obtain

$$\alpha \cdot \frac{area}{area_{norm}} + \beta \cdot \frac{wire\_length}{wire\_length_{norm}} + \gamma \cdot \frac{CPI}{CPI_{norm}}, \tag{15}$$

where $area$ and $wire\_length$ are the area and total wire length of the floorplan, respectively, α and β are user-defined weights, CPI, CPI_{norm}, and γ are the CPI value of the floorplan, the normalization value of CPI, and user-defined weights for CPI, respectively. For simplicity, we denote the objective combining area and total wire length as AL, and combining all area, total wire length and CPI as ALC.

The CPI value in (15) is computed by the TPWL model as shown in Section III and via an access ratio based approach. *Access ratio* of an interconnect (bus) is defined as the number of bus access over the total clock cycles for a benchmark. Intuitively, the latency of a bus has more impact on performance if it is accessed more frequently. Therefore, the access ratio metric is an indicator of the impact of a bus. The data of bus access numbers are collected from cycle-accurate simulations and we present the arithmetic mean of the data over all benchmarks in Table III. In the access ratio based approach to compute the "CPI" term in (15) we use the access ratio weighted sum of the pipelining stages for all interconnects in Table III. Note that a similar idea has been used in an independent study [1].

B. Microarchitecture floorplanning with the TPWL model

We develop the microarchitecture floorplanning with the TPWL model based on the $Parquet$ package [22]. Fig. 3 shows the overview of the flow. It starts with an initial floorplanning optimization with an objective function of weighted sum of area and total wire length (as in traditional floorplannings) to obtain an SA trajectory. This trajectory is sampled, collected and simulated to build an initial CPI table (refer to Section III). This initial CPI table constructs a TPWL model which makes CPI estimation possible. Thereafter, a few iterations of above processes may be needed to expand the CPI table and improve the accuracy of the model. Note that the initial SA trajectory is obtained without CPI included in the objective function. But CPI is added into the objective function after the first round. The iterations stop when the change on the estimated CPI is smaller than a given threshold, which is corresponding to the condition for 'accurate CPI estimation' at the bottom of Figure 3.

Typically the iterations converge in two to three iterations and our experiment results show that these extra iterations can improve accuracy significantly. Note that the idea of using iterations to improve the accuracy of the TPWL model is proposed in this paper for the first time.

Regarding implementation details, we describe the way to add CPI into the objective function in [22] as follows. As shown in [22], the objective of AL is a linear combination of area and total wire length. In [22], after each move in SA, Δ 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 [22], in our floorplanning, we compute Δ of the objective as the weighted sum of the changes in area, total wire length and CPI after each move, i.e.,

$$\Delta_{area} = \frac{area_{new} - area_{old}}{\sum area_{block}} \tag{16}$$

$$\Delta_{wire} = \frac{wire_{new} - wire_{old}}{wire_{old}} \tag{17}$$

$$\Delta_{CPI} = \frac{CPI_{new} - CPI_{old}}{CPI_{old}} \tag{18}$$

$$\Delta = w_{area} \cdot \Delta_{area} + w_{wire} \cdot \Delta_{wire} + w_{CPI} \cdot \Delta_{CPI}, \tag{19}$$

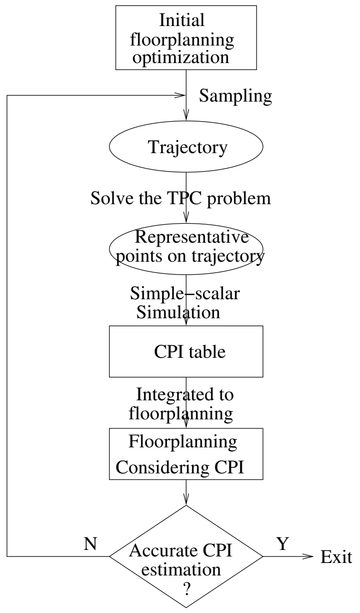

Initial floorplanning optimization
Sampling
Trajectory
Solve the TPC problem
Representative points on trajectory
Simple−scalar Simulation
CPI table
Integrated to floorplanning
Floorplanning Considering CPI
N
Accurate CPI estimation ?
Y
Exit

Fig. 3. Overview of the microarchitecture floorplanning.

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

$$\begin{cases} \Delta < 0 \\ R < exp(\frac{-\Delta \cdot t_i}{t_c}) \quad \Delta \geq 0 \end{cases} \tag{20}$$

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

C. Experiment results

We have implemented the proposed methods for microarchitecture floorplanning based on the *Parquet* package [22]. According to [7], below we assume that the interconnect distance between two adjacent flip-flops is $2000\mu m$ under $3GHz$ clock and 1200 μm under $5GHz$ in the ITRS $100nm$ generation.

1) Validation of the TPWL model: The procedure to build a TPWL model follows the procedure outlined in Section III-B. Specifically, We conduct SA to minimize an objective function of combining area and total wire length (AL) and build up an initial CPI table for the TPWL model by sampling this SA procedure and running cycle-accurate simulations. Then, SA is repeated for a couple more times with CPI added to the objective function (ALC), where the CPI value is evaluated by the TPWL model that has been already established. Each time, we add more entries to the CPI table to improve the accuracy.

The total number of cycle-accurate simulations for building a TPWL model is controlled by two factors. The first one is the radius r of the "balls" in the sampling phase. A small value of r leads to a large number of cycle-accurate simulations in the SA procedure. The second factor is the total number of SA procedures performed. In our experiment, we choose r to be five and perform three times of SA. This setting ends up with 90 simulations. Fig. 4 shows the accuracy of the TPWL model against the total number of simulations. As we can see from Fig. 4, the maximum error decrease from 15% to 4% and average error decrease from 4% to 1% as the total number of cycle-accurate simulations increase from 15 to 90.

The total number of simulations to build a TPWL model is scalable as shown in Fig. 4. However, the accuracy of the TPWL model indeed affects the quality of the floorplans obtained by SA. For example, using a CPI model with 15% error, SA could obtain a floorplan anywhere between 0% and 30%[8] from the optimum even if the SA procedure is optimal. Considering that the SA procedure is sub-optimal and cycle-accurate simulations are time-consuming, we believe that using 90 cycle-accurate simulations to build a TPWL model with 4% maximum error is a good balance between cost and quality. Note that each cycle-accurate simulation includes 10 benchmarks and takes 3-4 hours in a 2.8GHz Xeon machine. Comparing to cycle-accurate simulations, the cost of floorplanning optimization is negligible (2-3 minutes for each run). Applying the TPWL to evaluate CPIs is highly efficient and almost has no impact on the speed of floorplanning optimization once the TPWL tables are built because the evaluation is based on formulae.

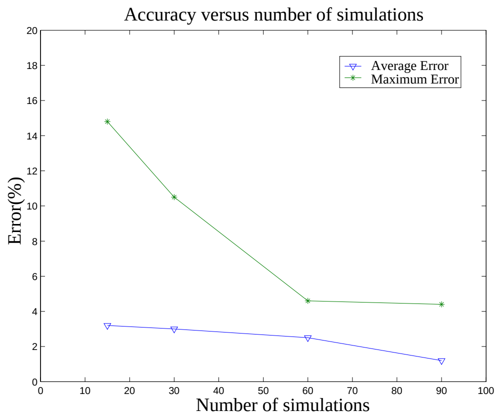


Fig. 4. Accuracy of the TPWL model versus total number of cycle-accurate simulations

2) Comparison of floorplanning with different objectives: We compare the floorplans obtained by SA subject to the objective functions of AL (area, total wire length) and ALC (area, total wire length and CPI), respectively. Note that the AL objective is used by the traditional floorplanning with throughput degraded by pipelined global interconnects. The area of each block in the floorplan could be found in Table II. We summarize the results of floorplanning using these objectives in Table IV. For each objective in Table IV we run floorplanning optimizations ten times as the floorplanning algorithm in [22] is not deterministic. We show both best and average results from these ten runs for comparison in the table.

We first present the white space rate of these floorplans in

[8]For the model that has 15% error, a solution 30% worse than the optimal solution could be 15% overestimated according to the model. If the optimal solution is 15% underestimated by the model, the 30% worse solution could be choosen by SA to become the "optimal" solution.

Metrics	μp in 3GHz				μp in 5GHz			
	best		avg		best		avg	
	AL	ALC	AL	ALC	AL	ALC	AL	ALC
White Space(%)	3.62	7.12	9.87	13.2	16.59	16.59	10.10	12.98
CPI	1.79	0.78 (-56.4%)	1.73	0.81 (-46.8%)	1.90	0.82 (-56.8%)	2.66	0.94 (-64.7%)
TWL(10^2 mm)	4.61	5.35 (+16.1%)	4.84	5.19 (+7.2%)	4.00	4.36 (+9.0%)	5.08	5.30 (+4.33%)
Total/Max. #FF	53/6	42/6	54/8	40/7	84/17	42/13	88/16	54/11
Area(10^2 mm^2)	1.18	1.23 (+4.2%)	1.27	1.32 (+3.9%)	1.37	1.37 (-0.0%)	1.27	1.31 (+3.1%)

TABLE IV
COMPARISON BETWEEN FLOORPLANS OBTAINED BY DIFFERENT OBJECTIVES.

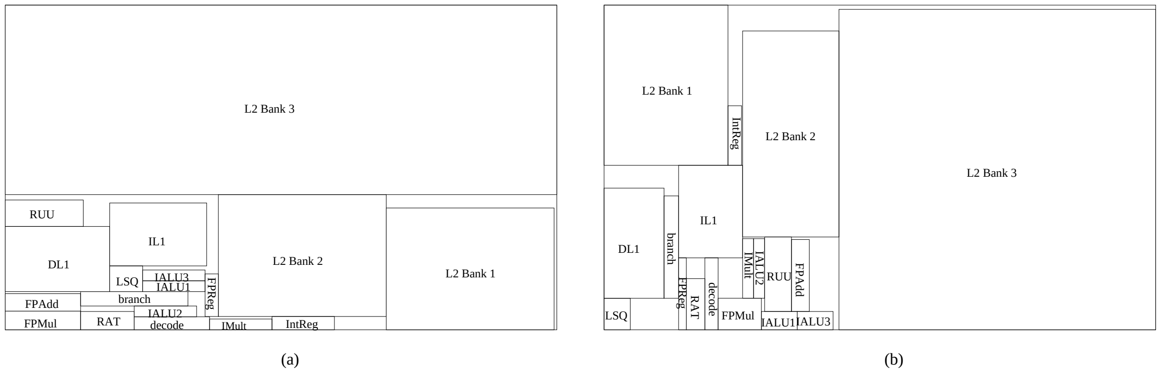


Fig. 5. Comparison between floorplanning without (a) and with (b) CPI minimization.

the first row. The low rate indicates the good quality of these floorplans. We then present the CPI of floorplans in the second row. The ALC and AC solutions can reduce CPI over AL by up to 64.7%, which is corresponding to a 183.0% increase in throughput. The results on total wire length show that there is no strong correlation between CPI and total wire length. This indicates that the objective of minimizing total wire length in traditional floorplanning does not 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 present the area in the fifth row. The ALC results in the lowest CPI with only a small area overhead of less than 5.0% over AL.

To demonstrate how CPI minimization affects the final floorplan, we compare the floorplan obtained without CPI minimization and with CPI minimization in Fig. 5. In our experiment, the aspect ratio of the die is fixed but those of blocks are flexible. As shown in Fig. 5 (b), with consideration of CPI minimization during floorplanning, the length of critical buses that have high access ratios and intuitively significant effect on CPI is generally shorter than that in (a). For example, in (b) the "RUU" module is placed close to the modules of "IALU", "FPAdd" and "FPMul". Note that the buses between these modules are critical buses based on the access ratio in Table III.

3) Comparison of floorplans under different CPI models: In terms of computational time to setup, the access ratio model is more efficient than the TPWL model. As discussed above, it takes 90 simulations to build an accurate TPWL model while access ratio data can be collected in one simulation. However, because both models use formulae to evaluate CPI, they are equally efficient when used in floorplanning once the TPWL tables are built.

To compare the two models, we choose the best and average results among ten optimization results and show them in Table V. We compare the floorplans with an ALC objective function under both 3GHz and 5GHz. Metrics of white space ratio, CPI, total wire length (TWL), total/max number of flip-flops and area are shown in the table for comparison. As shown in the table, in terms of objective function, the TPWL model is 7.8% and 5.5% better than the access ratio model under 3GHz and 5GHz, respectively, on average. In terms of CPI, the TPWL model is 10.0% and 13.0% better. We have also shown the real throughput in the Table in BIPS, too. From the table we can see that although IPC is degraded when the clock rate is increased, throughput is still significantly improved.

This result can be explained as follows. As we discussed in footnote 8, in an ideal SA procedure a CPI model with X% error leads to a floorplan anywhere between 0% and 2X% away from the optimum. Because the access ratio model can be treated as a rough CPI model and the TPWL model is more accurate, the floorplan obtained by the TPWL model should be better on average.

V. CO-OPTIMIZATION OF MICROARCHITECTURE CONFIGURATION AND FLOORPLANNING

To develop the co-optimization method for microarchitecture configuration and floorplanning we build a unified throughput model to pipelined interconnects and microarchitecture configurations based on the TPWL approach. This model is then integrated in the SA engine to efficiently explore over 1 million microarchitecture configurations and corresponding floorplan variations for each of them. Below we first introduce the TPWL model for pipelined interconnects

	μp in 3GHz				μp in 5GHz			
	best		avg		best		avg	
Metrics	AR	$TPWL$	AR	$TPWL$	AR	$TPWL$	AR	$TPWL$
White Space(%)	15.13	7.12	15.21	13.2	5.84	16.59	14.00	12.98
CPI	0.78	0.78 (-0.0%)	0.90	0.81 (-10.0%)	0.96	0.82 (-14.6%)	1.08	0.94 (-13.0%)
Throughput (BIPS)	3.85	3.85 (-0.0%)	3.33	3.70 (+11.0%)	5.21	6.10 (+17.1%)	4.63	5.32 (+14.9%)
TWL(10^2 mm)	4.95	5.35 (+8.0%)	5.83	5.19 (-11.0%)	5.84	4.36 (-25.3%)	5.30	5.30 (-0.0%)
Total/Max. #FF	25/5	42/6	54/8	26/6	33/8	42/13	41/11	54/11
Area(10^2 mm^2)	1.35	1.23 (-8.9%)	1.35	1.32 (-2.2%)	1.21	1.36 (+12.4%)	1.34	1.32 (-1.5%)
Objective	3.00	2.88 (-4.0%)	3.33	3.07(-7.8%)	3.18	2.95 (-7.2%)	3.45	3.26 (-5.5%)

TABLE V
COMPARISON BETWEEN TPWL AND ACCESS RATIO BASED APPROACH.

and microarchitecture configurations and then discuss the co-optimization method based on the SA engine. Finally we present our experiment results.

A. TPWL model for pipelined interconnects and microarchitecture configurations

To build a TPWL model for pipelined interconnects, the interconnects that have a large impact on throughput have been first identified and then recorded into the Bus latency vector, as shown in Section II-A. To build a unified TPWL model on both pipelined interconnects and microarchitecture configurations, not only these interconnects but also the components in configurations that affect throughput need to be identified. These components are described as follows:

1) Issue width: Issue width, also called machine width, is one of the most important parameters determining IPC. It is defined as the number of instructions that can be issued to the execution stage of the pipeline in a single cycle. As shown in [25], CPI can be approximated as

$$CPI = CPI_{ss} + CPI_{brmiss} + CPI_{imiss} + CPI_{dmiss}, \quad (21)$$

where CPI_{ss} is the maximum performance sustainable in the ideal case, and CPI_{brmiss}, CPI_{imiss}, and CPI_{dmiss} are the additional CPI caused by branch misprediction, instruction miss and data cache miss, respectively. Note that (21) shows that the upper bound of IPC is $1/CPI_{ss}$.

It is observed in [26] that the total number of instructions that can issue per cycle is roughly the square root of the number of instructions in the issue window. Therefore, issue width determines the size of RUU (Register Update Unit [27]) and LSQ. In this paper, we explore three issue width values of 2, 4 and 8. This range covers most current architectures.

2) Branch prediction: Equation (21) shows that branch miss-events increase CPI. A branch misprediction event causes fetching of useless instructions into pipeline with five to ten cycles misfetch delay penalty in a pipeline and with five to nine front-end depth, as shown in [25].

The missing rate of branch prediction depends on the prediction method. We assume a combination of bimodal and 2-level method [28] in this study. [28] shows that this combinational method has a good performance in practice. The BTB size of the bimodal and 2-level methods are parameters of our model. In general, a larger size of BTB leads to a more accurate prediction.

3) Cache and TLB size: The cache-missing event is another major factor that increases CPI, as shown in (21). Cache-missing events cause stalling of the pipeline to bring data from next level caches and introduce delay penalties. For example, the L1 instruction missing penalty is about eight cycles in a pipeline with five to nine front-end depth[25]. The occasion of cache-missing events depends on the size and organization of caches. In this study, we assume a smaller associativity for small cache/TLB sizes and a larger one for large cache sizes, and treat the cache/TLB size as model parameters. The range of cache/TLB size is from 4K to 2048K in this study.

The physical size of cache/TLB in the floorplan is calculated by Cacti[29] under the ITRS 100nm generation. Cacti takes the size and organization of a cache/TLB as input and estimates the physical size of the cache/TLB with a high accuracy.

4) Number of functional units: Insufficient number of functional units can significantly affect performance. Our study shows that one less or more integer ALU can cause over 20% difference in IPC. However, redundant functional units waste area and power. To a certain degree, the purpose of exploring the number of functional units is to find the appropriate number of functional units that just satisfies the requirement for computation resources based on other parameters such as issue width, cache/TLB size and so on. Because the number of functional units cannot exceed issue width, in this study it is limited between one and eight. Also, we consider four types of functional units: integer ALU (IALU), integer multiplier (IMult), floating point ALU (FPALU), and floating point multiplier (FPMult).

Together with pipeline interconnects in Table III, these configuration components are recorded in a vector, which is similar to a bus latency vector. To build a TPWL model for throughput with respect to this vector, we use the same procedure described in Section III-B. Specifically, we sample an SA trajectory, collect the sample points in as few as possible "balls", and then simulate the center points of these "balls" to build a CPI table. Note that here SA optimizes both floorplans and microarchitecture configurations. Moves in SA may modify the floorplanning as well as the microarchitecture configurations. The details of the SA engine will be discussed in the following Section V-B.

B. SA based co-optimization method

In floorplanning the solution space is explored by moves in SA that change the shape, position, or orientation of a module in the floorplan. To additionally explore the solution space of microarchitecture configurations, we use new moves including resizing branch predictor and cache/TLB, and changing the number of functional units. We assign around 15% of the total number of moves in SA to moves for micro-architecture configurations and these moves are evenly distributed into each type mentioned above.

At first glance, changing issue width can also be a new move. However, to achieve high performance, most parameters should stay in a suitable range with respect to issue width. As observed in our experiment, in most cases a change of issue width in SA brings in inconsistency among parameters and degrades the quality of the solution. To explore the solution space more efficiently, we propose to employ multiple starts for issue width. In each start, issue width is fixed and the best case is selected among all starts.

The explored microarchitecture configurations are summarized in Table VI. Note that each row in the table represents an independent variable and the total number of microarchitecture configurations in this table is over one million, which is in sharp contrast with previous work of [1], [11] to enumerate no more than 32 candidates.

Table VI is built by allowing each parameter to change within a range around the baseline configuration, which is in bold font and close to those appeared in the literature [1]. We find that the microarchitecture throughput is sensitive to changes in these ranges. Specifically, we start with the baseline configuration and then scan one parameter each time to determine the range for the parameter. Note that as suggested by [25], the RUU and LSQ size are deterministic for the given issue width.

C. Experiment results

1) IPC versus area: We have integrated the TPWL model and the methodology to explore micro-architecture configurations into *Parquet*. Because the metric of total wire length has no strong relation with throughput (as shown in Section IV-C) we use SA minimizing the objective function AC, i.e., area and CPI.

We show the curve of IPC versus area in Figure 6 for the purpose of validating the TPWL model. The data points are obtained by executing multiple SA runs with different weights assigned to area and IPC in the objective function. We also choose the best IPC value for solutions with close area values.

We first validate the TPWL model by comparing the IPC value obtained by the model and by cycle-accurate SimpleScalar simulation in Figure 6. The data points in the figure indicate that the average error of TPWL model is around 3.3%. We then show the trend of IPC with respect to increasing area. The dotted lines in the figure represent the ideal IPC value with no performance loss caused by insufficiency of instruction parallelism, missing events, lack of functional units and interconnect pipelining. We approximate the ideal IPC value for each issue width by providing large enough branch predictors, caches, TLBs and enough number of functional units in the micro-architecture configurations and conduct SimpleScalar simulations. As shown in the figure, the

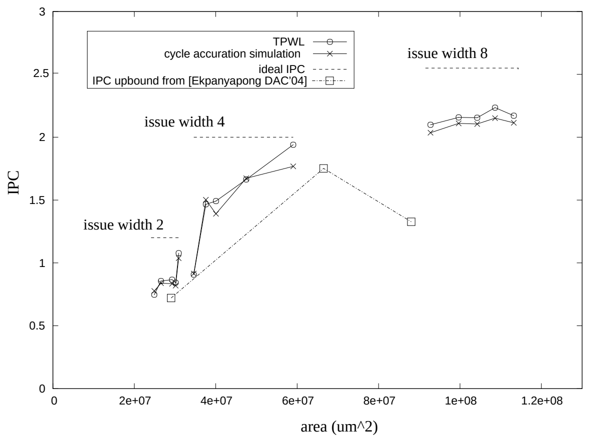


Fig. 6. IPC versus area (under TPWL model and cycle-accurate simulation).

optimization results are less than 20% away from ideal IPC. We also compare our IPC value with the IPC value of the first three configurations enumerated in [1], which is also shown in Table VII as the last value. These three configurations are for issue width 2, 4 and 8, respectively. We show the IPC value of these three configurations but without considering interconnect latency. Therefore, shown in the figure is ideal IPC for [1]. Also, we obtain the area of each configuration by mapping configuration parameters to corresponding physical size used in this paper. As we can see from the figure, the co-optimization methodology proposed in this paper leads to much better designs (26.9% higher IPC) than configurations manually chosen in [1]. Note that the case of issue width 8 in [1] experiences a major performance degradation because its cache size is too small.

As shown in Figure 6, increasing chip area helps to increase IPC mainly because of the increase in cache size and the number of functional units. However, there is a diminishing for the same issue width, as it is difficult to improve performance by increasing the chip area when IPC is close to the ideal. This trend holds for different issue widths, too. In addition, compared to issue width 4, issue width 8 is less effective to improve IPC by the same amount of area increase. The primary reason is that there is not enough instruction level parallelism (ILP) when issue width increases. Note that the increase in global interconnect latency with the increase in area also contributes to this diminishing return. On the other hand, a too small area may lead to significant degradation in IPC. The leftmost data point from the issue width 4 in the TPWL model is a good example. It has the smallest area among all data points in the issue width 4, but the IPC value is much smaller than that from the issue width 2 with a similar area (the rightmost data point from the issue width 2).

2) Configuration details: To reveal the micro-architecture configuration details, we show all micro-architecture configuration parameters for average and best cases of multiple SA runs in Table VII. All the IPC values in the table are obtained from cycle-accurate simulations with global interconnect latencies extracted from corresponding floorplans. As shown in the Table, average values are fairly close to the best case value, indicating a good convergence of SA optimization.

μ-arch modules	issue width 2	issue width 4	issue width 8
RUU size	**32**	**64**	**256**
LSQ size	**16**	**32**	**128**
Bpred (KB)	1, 2,**4**, 8, 16	1, 2, **4**, 8, 16	1, 2, **4**, 8, 16
L1 Icache (KB)	4, 8, **16**, 32, 64	8, 16, **32**, 64, 128	32, 64, **128**, 256, 512
L1 Dcache (KB)	4, 8, **16**, 32, 64	8, 16, **32**, 64, 128	32, 64, **128**, 256, 512
L2 Ucache (KB)	16, 32, **64** , 128, 256	32, 64, **128**, 256, 512	64, 128, **256**, 512, 1024
ITLB (entry)	16, 32, **64**, 128, 256	16, 32, **64**, 128, 256	32, 64, **128**, 256, 512
DTLB (entry)	16, 32, **64**, 128, 256	16, 32, **64**, 128, 256	32, 64, **128**, 256, 512
IALU	1, **2**,3,4	1,**2**,3,4	4, 5, **6**, 7, 8
FPALU	**1**, 2, 3	1, **2**,3,4	1, **2** 3, 4
IMult	**1**, 2, 3	**1**, 2, 3	1, **2** 3, 4
FPMult	**1**, 2, 3	**1**, 2, 3	1, **2** 3, 4

TABLE VI
THE RANGES OF CONFIGURATION VARIABLES.

μ-arch modules	issue width 2	issue width 4	issue width 8
Bpred	4.1KB/8KB/128	4.2KB/8KB/512	3.2K/8K/512
L1 Icache (KB)	12.8/16/8	27.2/64/64	129.9/256/8
L1 Dcache (KB)	12.8/4/8	31.5/8/64	97.0/32/8
L2 Ucache (KB)	60.4/32/64	60.8/64/512	209.5/128/128
ITLB (Entry)	32.0/32/32	46.9/32/128	62.1/32/128
DTLB (Entry)	26.3/32/32	42.1/128/128	68.9/32/128
IALU	1.7/2/1	1.9/4/3	5.8/7/3
FPALU	1.5/1/1	1.7/2/1	1.9/4/1
IMult	1.5/1/1	1.8/1/1	2.3/4/1
FPMult	1.5/2/1	1.7/2/1	2.2/2/1
Area(mm^2)	29.8/32.9/29.4	51.2/59.0/71.2	100.7/108.6/91.5
IPC	0.95/1.05/0.72	1.50/1.77/1.75	2.11/2.15/1.25

TABLE VII
CONFIGURATION DETAILS (AVG/BEST/[1]).

Also, the obtained configuration by the proposed method has a significant performance advantage over the upper bound of designated configurations used in [1] without considering the impact of global interconnects.

VI. CONCLUSIONS AND DISCUSSIONS

Considering the impact of interconnect pipelining on throughput, we have developed methods for microarchitecture floorplanning to minimize CPI (cycles-per-instruction) for a given microarchitecture configuration. Compared to the conventional floorplanning minimizing area and wire length techniques, the new floorplanning formulation obtains a floorplan with CPI reduced by 64.7% with a small area overhead that is less than 5.0%.

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 that minimizing total wire length does not necessarily lead to minimization of CPI. Yet, using the bus access ratio as weight and minimizing the weighted interconnect length is shown to be a good heuristic for microarchitecture floorplanning.

We have developed a TPWL model for CPI to consider micro-architecture configurations and interconnect pipelining, and further developed co-optimization of microarchitecture configuration and floorplanning based on it. We explore over one million configurations candidates and find microarchitecture configurations and corresponding floorplans with an IPC (instructions per cycle) less than 20.0% away from the ideal IPC. Our solutions are 26.9% better than [1] that was able to consider only a limited number of micro-architecture configurations.

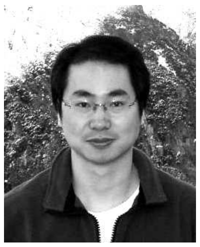
There is a trade-off between quality and simulation runtime to build the TPWL model. Tracing SA reduces the simulation runtime without sacrificing the accuracy around the SA trajectory. In addition, cycle-accurate simulation may be replaced by traced based simulation [25], or incremental simulation to reduce model building time. Note that once models are built, the runtime is virtually the same for TPWL model and access ratio based approach. The TPWL model is a general model and can be expanded to consider more design freedoms and objectives, such as power, and still maintain high quality.

REFERENCES

[1] M. Ekpanyapong, J. R. Minz, T. Watewai, H.-H. S. Lee, and S. K. Lim, "Profile-guided microarchitectural floorplanning for deep submicron processor design," in *Proc. Design Automation Conf*, 2004.

[2] J. Clabes, J. Friedrich, M. Sweet, J. Dilullo, S. Chu, D. Plass, J. Dawson, P. Muench, L. Powell, M. Floyd, B. Sinharoy, M. Lee, M. Goulet, J. Wagoner, N. Schwartz, S. Runyon, G. Gorman, P. Restle, R. Kalla, J. McGill, and S. Dodson, "Design and implementation of the power5 microprocessor," in *Proc. IEEE Int. Solid-State Circuits Conf.*, pp. 56 – 57, 2004.

[3] C. Long, L. Simonson, W. Liao, and L. He, "Floorplanning optimization with trajectory piecewise-linear model for pipelined interconnects," in *Proc. Design Automation Conf*, pp. 640–645, 2004.
[4] V. Nookala, Y. Chen, D. J. Lilja, and S. S. Sapatnekar, "Microarchitecture-aware floorplanning using a statistical design of experiments approach," in *Proc. Design Automation Conf*, pp. 579–584, 2005.
[5] D. Matzke, "Will physical scalability sabotage performance gains?," *Computer*, vol. 30, pp. 37–39, 1997.
[6] 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.
[7] 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.
[8] International Technology Roadmap for Semiconductors (ITRS). 2003.
[9] J. Smith and G. Sohi, "The microarchitecture of superscalar processors," *Proceedings of the IEEE*, vol. 83, pp. 1609 – 1624, Dec. 1995.
[10] B. A. Gieseke and et al, "A 600mhz superscalar risc microprocessor with out-of-order execution," in *Proc. IEEE Int. Solid-State Circuits Conf.*, pp. 176–177, 1997.
[11] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis, "Microarchitecture evaluation with physical planning," in *Proc. Design Automation Conf*, pp. 32–35, 2003.
[12] 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.
[13] J. Cong, T. Kong, and D. Pan, "Buffer block planning for interconnect-driven floorplanning," in *Proc. Int. Conf. on Computer Aided Design*, pp. 358–363, Nov. 1999.
[14] I.-R. Jiang, Y.-W. Chang, J.-Y. Jou, and K.-Y. Chao, "Simultaneous floor plan and buffer-block optimization," pp. 694–703, May 2004.
[15] K. Wong and E. Young, "Fast buffer planning and congestion optimization in interconnect-driven floorplanning," in *Proc. Design Automation Conf*, pp. 411–416, 2003.
[16] C. wing Sham, E. Young, and H. Zhou, "Interconnect-driven floorplanning by searching alternative packings," in *Proc. Asia South Pacifi c Design Automation Conf.*, pp. 417–422, 2003.
[17] M. R. Casu and L. Macchiarulo, "Floorplanning for throughput," in *Proc. Int. Symp. on Physical Design*, pp. 62–69, 2004.
[18] F. Rafiq, M. Chrzanowska-Jeske, H. Yang, M. Jeske, and N. Sherwani, "Integrated floorplanning with buffer/channel insertion for bus-based designs," pp. 730–741, June 2003.
[19] D. Burger and T. Austin, *The simplescalar tool set version 2.0*. University of Wisconsin-Madison, 1997.
[20] N. Sherwani, *Algorithms For VLSI Design Automation*. Kluwer, 3rd ed., 1999.
[21] 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.
[22] 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.
[23] D. S. Johnson, "Approximation algorithms for combinatorial problems," *J. Comput. Sys. Sci.*, vol. 9, pp. 256–278.
[24] L. Lovasz, "On the ratio of optimal integral and fractional covers," *Discrete Math.*, vol. 13, pp. 383–390.
[25] T. Karkhanis and J. Smith, "A first-order superscalar processor model," in *omputer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pp. 338 – 349, June 2004.
[26] E. Riseman and C. Foster, "The inhibition of potential parallism by conditional jumps," *IEEE Trans. on Computers*, vol. C-21, pp. 1405–1411, 1972.
[27] G. Sohi, "Instruction issue logic for high-performance, interruptible, multiple functional unit, pipelined computers," *IEEE Trans. on Computers*, vol. C-39, pp. 349–359, Mar. 1990.
[28] McFarling, "Combining branch predictors," Tech. Rep. TN-36, DEC WRL, June 1993.
[29] S. Wilton and N. Jouppi, "Cacti: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 677 – 688, 1996.



Changbo Long Changbo Long received the BS and MS degree in electrical engineering from Tsinghua University in 1999 and 2001, respectively, the MS degree in computer engineering from University of Wisconsin, Madison in 2003, and the Ph.D degree in electrical engineering from UCLA in 2006.

He joined Synopsys Inc. in June, 2006. His research interests include computer-aided design of VLSI circuits and systems, power-efficient circuits and designs.

PLACE PHOTO HERE

Lucanus J. Simonson

PLACE PHOTO HERE

Weiping Liao Weiping Liao received B.S. degree and M.S. degrees, both in physics, from the University of Science and Technology of China, Hefei, China, in 1996 and 1999, respectively, the M.S. degree in computer engineering from the University of Wisconsin, Madison in 2002, and the Ph.D. degree in electrical engineering from the University of California at Los Angeles in 2005. He is currently a senior architecture engineer at NVIDIA, Santa Clara, CA, working on next-generation graphics processor design.

Lei He Dr. Lei He is an associate professor at electrical engineering department, UCLA, and was a faculty member at University of Wisconsin, Madison between 1999 and 2001. He also held visiting or consulting positions with Intel, Hewlett-Package, Cadence, Synopsys, Rio Design Automation, and Apache Design Solutions.

His research interests include VLSI circuits and systems, and electronic design automation. He has published over 140 technical papers and has been a technical program committee member for a number of conferences including Design Automation Conference, International Conference on Computer-Aided Design, International Symposium on Low Power Electronics and Design, and International Symposium on Field Programmable Gate Array.

Dr. He obtained Ph.D. degree in computer science from UCLA in 1999. He was granted National Science Foundation CAREER award in 2000, UCLA Chancellor's faculty career development award (highest class) in 2003, IBM Faculty Award in 2003, Northrop Grumman Excellence in Teaching award in 2005, and Best Paper Award in the 2006 International Symposium on Physical Design.