

Microarchitecture Evaluation With Floorplanning And Interconnect Pipelining

Ashok Jagannathan¹, Hannah Honghua Yang², Kris Konigsfeld², Dan Milliron², Mosur Mohan², Michail Romesis¹, Glenn Reinman¹, Jason Cong¹

¹Computer Science Department, Univ. of California, Los Angeles, CA 90095 USA

²Intel Corporation, Hillsboro, OR 97124 USA

Email: {ashokj@cs.ucla.edu, honghua.yang@intel.com}

Abstract—As microprocessor technology continues to scale into the nanometer regime, recent studies show that interconnect delay will be a limiting factor for performance, and multiple cycles will be necessary to communicate global signals across the chip. Thus, longer interconnects need to be pipelined, and the impact of the extra latency along wires needs to be considered during early micro-architecture design exploration. In this paper, we address this problem and make the following contributions: (1) a floorplan-driven micro-architecture evaluation methodology considering interconnect pipelining at a given target frequency by selectively optimizing architecture level critical paths. (2) use of micro-architecture performance sensitivity models to weight micro-architectural critical paths during floorplanning and optimize them for higher performance. (3) a methodology to study the impact of frequency scaling on micro-architecture performance with consideration of interconnect pipelining.

For a sample micro-architecture design space, we show that considering interconnect pipelining can increase the estimated performance against a no-wire-pipelining approach between 25% to 45%. We also demonstrate the value of the methodology in exploring the target frequency of the processor.

I. INTRODUCTION AND MOTIVATION

The throughput of a microprocessor in BIPS or *billion instructions per second* is given by

$$\text{Throughput}_{\text{BIPS}} = \frac{\text{IPC}}{\text{cycle_time}}$$

where IPC is the throughput in *instructions per cycle*. It can be seen that maximizing the BIPS requires optimizing both the IPC and the cycle time. Traditionally, this is done by evaluating the micro-architecture for a set of chosen target frequencies. At each frequency, architects estimate the latency of the blocks in the architecture and use this information to experimentally study the IPC through architecture simulations. The micro-architecture and the target frequency are chosen based on the best BIPS estimated through simulations. The circuit designers then work on the physical design of the processor to meet the target frequency goals.

With the advent of deep sub-micron technology, it has become a well-known fact that wire delay [1], [2], [3] contributes significantly to overall system performance. As the actual interconnect delay can only be derived from a physical layout of the processor, early micro-architecture planning and optimization without considering interconnect delay is no longer a valid approach to optimizing the processor performance.

For example, there are multiple instances where wires are pipelined in the Pentium 4 processor [4], [5] and lack of such interconnect design information could lead to sub-optimal design decisions during the early design space exploration process. As frequencies continue to scale [3], more wires will need to be pipelined and hence it becomes necessary to perform early micro-architecture evaluation with interconnect planning.

A. Prior Work

In [6], Cong et al. first proposed an early micro-architecture evaluation methodology with consideration of physical information. Assuming a target frequency, the authors model the latency of each block and the timing slack available at the input and output pins and perform timing-driven floorplanning to optimize for the performance of the micro-architecture configuration in terms of BIPS. This helps in identifying interconnect bottlenecks created by layout constraints. The work successfully demonstrated that such a combined approach is necessary to identify the quality of micro-architecture configurations.

A severe limitation of this approach is that the cycle time determined by the layout process for maximum BIPS and the cycle time assumed for pipelining the block may not match. This is due to the fact that interconnect pipelining is not considered and long wires result in a cycle time larger than the target. Thus, it is necessary to consider pipelining for interconnects and not just the raw wire delay during early micro-architecture performance evaluation. As pipelining different wires can have varied impact on the IPC of the microprocessor, it is important to understand the criticality of pipelining different wires based on their impact on performance.

Recent works [7][8][9] have looked at the problem of floorplanning with interconnect pipelining and proposed different interesting solutions in this regard. In [7], the authors propose a technique to optimize a floorplan for throughput by wire pipelining instead of optimizing the cycle time. Ekpanyapong et al. [8] formulated the micro-architecture floorplanning problem as a mixed-integer linear program where the wire weights were extracted from detailed micro-architecture simulations. They demonstrated that using weights derived from simulations provided better floorplan solutions in terms of system

performance against pure wirelength objectives. The work by Long et al. [9] is closer to the one presented here as it employs a CPI estimation method using the trajectory piecewise linear model [10] during floorplanning. For each bus in the design, the authors study the impact of extra latency along the bus when it is pipelined, and use a table lookup method to extract the CPI when different buses are pipelined during floorplanning. They showed that this approach can improve the estimated CPI by around 28% compared to the CPI obtained using wirelength optimization alone.

We differ from prior works in that we look at system level critical paths instead of two-pin connections. Micro-architecture performance is dependent upon a set of critical loops [11], and our work attempts to minimize the latency along these loops during floorplanning. Also, optimizing complete paths gives a global view of the design and reduces the time spent in estimating the impact of individual wires on the performance.

B. Contributions and Paper Outline

In this work, we propose a methodology to evaluate a micro-architecture with interconnect pipelining estimation and explore different target frequencies. Specifically, our contributions include:

- Early floorplanning methodology for evaluating micro-architectures with consideration of interconnect pipelining at a given target frequency by selectively optimizing latency-sensitive high-level critical paths [11] in the architecture.
- Use the idea of performance sensitivity models [5], [12] to estimate the changes in the IPC due to extra latency from the wires along the paths during floorplanning. As these sensitivity models are independent of the target frequency and only use the abstract notion of latency, they can be used to optimize the floorplan for any target frequency.
- Application of the proposed technique to evaluate the impact of interconnects on the performance at different target frequencies.

The remainder of the paper is organized as follows: Section III discusses the overview of the proposed approach. Section III-B presents the use of IPC sensitivity models to understand the impact of latency on critical paths on the performance of the architecture. Section III-C describes the proposed floorplanning methodology which considers interconnect pipelining and its impact on performance using the IPC sensitivity models. Experimental results are presented in Section IV. Section V summarizes the paper and discusses future research directions.

II. ASSUMPTIONS

As the work presented here is targeted toward providing feedback to architects during very early design stages (even before the pipeline stages and HDL are generated), we assume that a given microarchitectural block can be pipelined to any depth [5], [12] as required by the target cycle time. This is not unreasonable as modern processors such as Intel

Pentium 4 [4] has more than 20 pipeline stages, while Intel Prescott [13] has more than 30 pipeline stages. Such an abstraction also helps in understanding the impact of frequency scaling and interconnect delay along critical microarchitectural paths during early design planning. Finally, we assume that the overall *functionality* of the microarchitecture after wire pipelining can be fixed either manually or using some formal techniques [14].

III. METHODOLOGY

As mentioned earlier, the overall performance of the processor measured in instructions per second (BIPS) is maximized by optimizing both the IPC and the cycle time. Since simultaneous optimization of IPC and cycle time considering layout impact is a difficult problem, we split this into a two-step process:

- 1) For a given target frequency, optimize the floorplan and maximize the IPC¹ with consideration of interconnect pipelining estimation.
- 2) Apply Step 1 for different target frequencies and identify the best BIPS solution for the micro-architecture.

At a fixed target frequency, the IPC of the processor for a given workload is a function of: (a) the properties of each micro-architectural block, and (b) latency in number of cycles through critical micro-architectural paths. The block properties include instruction and data cache sizes and associativity, number of entries in the register file, the issue width of the processor etc. Critical micro-architectural paths include branch misprediction resolution latency, dependent instruction wakeup latency, etc. Typically, there are very few critical paths at a high-level (around 15 to 20). Since the path latencies come from both blocks and wires, the early floorplanning process can provide estimates of extra latency contributed by wires which can be used to estimate the performance in terms of BIPS.

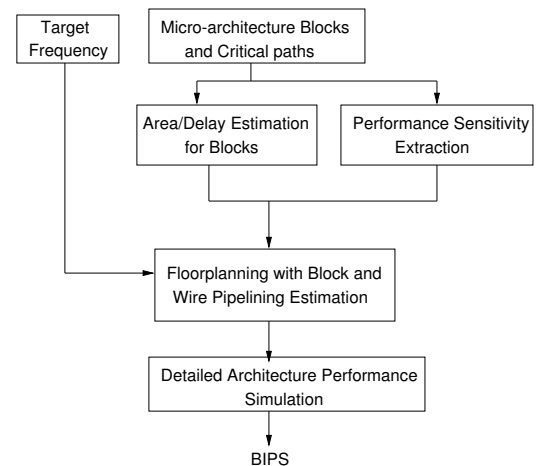


Fig. 1. Overall flow of the proposed approach.

Figure 1 outlines the proposed approach to optimize for IPC at a *fixed* target frequency. The input set consists of: (a) a set of architecture level blocks along with their area and delay,

¹At a fixed frequency, maximum IPC corresponds to maximum BIPS.

(b) a set of critical micro-architecture level paths which affect the performance of the processor, (c) IPC sensitivity models corresponding to the critical paths and blocks, and (d) a target frequency under which the performance is to be optimized.

The delay of each block is the maximum delay between any input to any output of the block without any pipelining. This delay is used to estimate the number of pipeline stages required to cover a path at the given target frequency based on the floorplan constraints. The IPC sensitivity models provide the relative changes in the IPC of the processor when the latency of critical paths changes based on the layout. These models are generated to be independent of the target frequency (and only parameterized using latency), and hence can be used for different target frequencies.

Since the target frequency is fixed, the maximum IPC corresponds to the maximum BIPS and hence the floorplanner optimizes for the IPC of the processor directly. The floorplanner uses the IPC sensitivity models as a guide to optimize for the latency through critical paths subject to the layout constraints. The output of the floorplanner is an estimation of the block and wire latencies for each of the critical paths that produced the best IPC estimate. This information is then fed to the detailed micro-architecture simulator in order to get accurate IPC estimation. The IPC from the simulator and the input target frequency provide an accurate measure of the throughput in BIPS for the given micro-architecture.

By changing the target frequency, we can evaluate the BIPS for the micro-architecture and accurately identify the operating frequency that provides the best performance. In what follows, we explain the notion of IPC sensitivity models and the details of the floorplanner.

A. Area and Delay Estimation

For the purpose of modeling area and delay, the micro-processor components can be categorized as below based on their implementation style:

- Array Structures: Data and instruction caches, cache tag arrays, all register files, register alias table, branch predictors and large portions of the instruction window and load/store queue.
- Fully Associative Content-Addressable Memories: Instruction window/reorder buffer wake-up logic, load and store order checks, TLBs etc.
- Combinational Logic and Wires: Functional units, instruction window selection logic, dependence check logic, and result buses.

The area and delay of each logic block depends very much on the implementation, particularly the transistor sizes and internal capacitances that makes up the circuit. We model the area and delay of the blocks using assumptions similar to Wilton and Jouppi [15] and Palacharla et. al. [16] in which the authors analyzed the delay of many of the microprocessor logic blocks. In both these works, the delay was analyzed by breaking the circuit into stages and calculating the RC for each stage. The delay of the block was derived by adding the delays along the critical path of the block.

B. IPC sensitivity Model

Since estimating IPC is a crucial part of our micro-architecture evaluation framework, we discuss our approach here. Traditionally, the IPC of a micro-architecture configuration is obtained by detailed cycle-accurate simulation using performance simulators such as SimpleScalar [17]. As this is a time-consuming process, such simulation methods cannot be used in the inner loops of physical planning tools. In order to have accurate IPC values, the work in [6] used an IPC table generated by simulating all the architecture configurations to be explored and obtaining the IPC for each configuration at a presumed target frequency. Though this method provides accurate IPC numbers, we identify the following problems with such an approach:

- Micro-architecture simulations to generate IPC for a large set of configurations and target frequencies is very time-consuming, and hence this approach is not scalable when exploring several configurations during the early design process, in spite of its accuracy.
- Such tables also limit the search space – i.e., it is necessary that each architecture configuration at every explored target frequency is simulated *a priori* and its IPC value obtained.

Several techniques [18], [19] have been proposed by architecture researchers to quickly estimate the IPC of the processor based on statistical and/or analytical methods. Since these techniques try to predict the actual IPC value, they are not accurate enough to be used as a guide during the floorplanning process.

We observe that it is not necessary to know the exact IPC obtained through simulations for a configuration during floorplanning. As long as the relative IPC degradation due to extra latency along critical paths can be captured, it can be used to guide any physical planning process. Though this may not provide *exact* IPC values, this method allows us to quickly estimate the changes in the IPC due to any variation in the latency of the blocks and paths as determined by the floorplan. Furthermore, this method has also been successfully used in micro-architecture research [5] for early design space evaluation.

To clarify the idea of a sensitivity model, Figure 2 shows a simple architectural critical path that corresponds to the latency of the availability of data accessed from the data cache. The results shown here were collected for a baseline architecture and methodology presented in Section IV by simply varying the latency of this path and keeping all other path latencies fixed. We can see from Figure 2 that the IPC drops by nearly a constant factor (around 2.4% on average) with each extra cycle latency along this path. This trend also appears to be consistent across different cache sizes. Thus, if the latency of this path increased by 2 cycles during floorplanning, we approximate the new IPC as $(1 - 0.024) \times (1 - 0.024)$ relative to the old IPC.

The total latency of a given micro-architectural path is made up of the latency through the blocks and the wires along the path, as shown in Figure 3. Since the models should be applicable under different target frequencies, the minimum

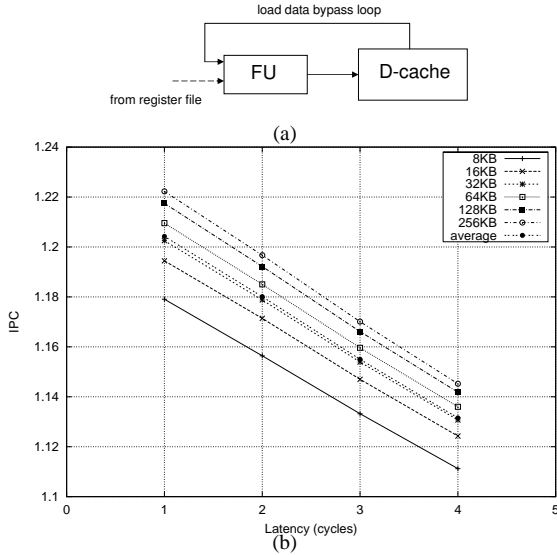


Fig. 2. Illustration of the sensitivity model for the load use latency path.

latency of any path P in the micro-architecture is set to the number of blocks along the path, assuming each block will correspond to one cycle latency. We denote this using $baseLatency(P)$, and use IPC_{base} to represent the IPC of the micro-architecture when the latency of every path is equal to its minimum latency.

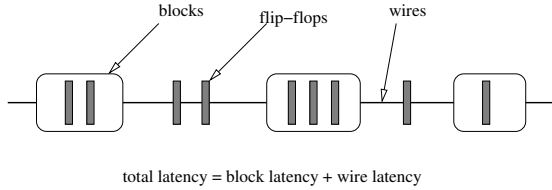


Fig. 3. Illustration of total latency through the micro-architecture path.

For a path P , the actual latency may be higher depending on the target frequency and the impact of interconnects after floorplanning. The IPC sensitivity model for P gives the relative IPC due to increase in the path latency, and therefore is a value in the range $[0, 1)$. The IPC sensitivity to the latency of a path is studied by varying its latency alone, keeping all other path latencies constant. The sensitivity is approximated using well known mathematical models (constant, linear, piece-wise linear, etc.) if the response is predictable, or as a lookup function, if the response cannot be captured using a simple mathematical model. For example, the response for the cache latency path in Figure 2 is modeled as a constant factor degradation per extra cycle latency. Thus, if c is the percentage degradation per extra cycle, and δ is the number of extra latency cycles over the minimum path latency, then the new IPC is computed using the sensitivity model $f(\delta) = (1 - c)^\delta$ relative to IPC_{base} .

Such sensitivity models can be generated for individual blocks and paths assuming all other parameters are fixed, or can be used to study the sensitivity to simultaneous changes along multiple paths or blocks, depending on the accuracy required. The work presented in [20] shows how to develop higher-order models to consider interactions among multiple

architectural parameters. However, in this work, we assume that the effect of each path or block latency on the IPC is *independent*, and we apply the individual models separately. Thus, if P_1, P_2, \dots, P_n are the critical paths considered, $f_{P_1}, f_{P_2}, \dots, f_{P_n}$ correspond to the IPC sensitivity models (as described earlier) for these paths respectively, and $\delta_{P_1}, \delta_{P_2}, \dots, \delta_{P_n}$ be the increase in the latency of these paths, then the new IPC is computed as

$$IPC_{new} = IPC_{base} \times f_{P_1}(\delta_{P_1}) \times f_{P_2}(\delta_{P_2}) \times \dots \times f_{P_n}(\delta_{P_n}).$$

where $0 < f_{P_i}(\delta_{P_i}) \leq 1$, for $i = 1, 2, \dots, n$.

We would like to clarify that generating such sensitivity models requires performing a number of architectural simulations which are time-consuming. However, once the models are generated, it is fairly straightforward to apply them, and they are scalable when exploring larger solution spaces.

C. Floorplanning

The floorplanning problem that we investigate here considers only the die area and the performance of the system. However, it is not hard to see that the floorplanner can be easily extended to trade off performance for other constraints which stem from power and thermal considerations. Formally, we define the performance-aware area-constrained floorplanning problem as follows:

Given:

- target cycle time T_{cycle}
- clocking overhead $T_{overhead}$
- list of blocks in the micro-architecture with their area, dimensions and total logic delay
- set of critical micro-architectural paths with performance sensitivity models for the paths
- required width W_r and height H_r of the floorplan

Objective: Generate a floorplan which obeys die size constraints and has the best performance in terms of BIPS.

The floorplanner used in this work is based on a simulated annealing [21] framework using sequence pair [22] and is derived from the open-source tool Parquet [23]. Since the tool already considered fixed die size constraints, we extended the floorplanner to consider the performance of the design instead of a wirelength optimization objective. We use a weighted combination of area and performance as follows:

$$cost = w * \frac{1}{IPC} + (1 - w) * area_cost$$

The $area_cost$ tracks the violation of the die area constraints and is calculated as in Parquet [23] while the computation of the IPC of the floorplan configuration is explained below.

As mentioned earlier, the IPC of a floorplan configuration is a function of the latencies of critical blocks and micro-architecture paths. Let T_{cycle} be the target cycle time. Typically, a portion $T_{overhead}$ of the cycle time is used to account for the skew, jitter, latching and other clocking overheads for a given technology [12], [5]. The remaining time $T_{useful} = T_{cycle} - T_{overhead}$ is essentially the computation time available in each cycle for useful work, and we use this time to calculate the number of pipeline stages for a block or a path.

Given a floorplan configuration, we now discuss how to compute the latency of P_k . Let $block(i)$ represent the i -th block along P_k and $e(i-1, i)$ represent the edge between blocks $i-1$ and i along P_k . If P_k has n blocks and $n-1$ edges, the total delay through the entire path is given by

$$delay(P_k) = \sum_{i=2}^{n-1} delay(block(i)) + \sum_{i=1}^{n-1} delay(e(i, i+1))$$

i.e., $delay(P_k)$ is the sum of the block and wire delays from the output of the first block on P_k up to the input of the last block on P_k .

To calculate the delay for an edge $e(i, j)$, we fix the connecting pins on the blocks i and j at the positions obtained from the intersection of the center-to-center line between the blocks and the corresponding block boundaries. We then compute the length of the edge $e(i, j)$ as the manhattan distance between the two pin locations and use this information to compute the delay along that edge. Once the propagation delay through P_k is known, the number of pipeline stages required to cover that delay at the given target frequency can be calculated as

$$latency(P_k) = \lceil \frac{delay(P_k)}{T_{useful}} \rceil$$

The increase in the latency of P_k compared to its base latency is given by

$$\delta_{P_k} = latency(P_k) - baseLatency(P_k)$$

For each critical path, the latency of the path is determined as mentioned above. We process the paths according to their criticality (and the impact on IPC) from the highest to the lowest so that blocks which are on multiple paths get pipelined according to the most critical path that passes through the block. Once the latencies have been computed, we use the IPC sensitivity models to estimate the new IPC based on the IPC estimation method discussed in Section III-B.

IV. EXPERIMENTAL VALIDATION

To experimentally study the methodology presented in this work, we chose a baseline out-of-order superscalar processor micro-architecture with a pipeline as shown in Figure 4. The baseline processor parameters are shown in Table I.

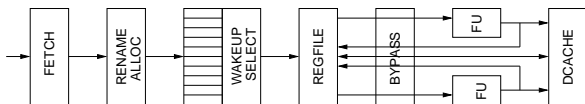


Fig. 4. Superscalar pipeline explored in this work.

The simulator used in this study was derived from the SimpleScalar 3.0 tool set [17], a suite of functional and timing simulation tools for the Alpha AXP ISA. The timing simulator executes only user-level instructions, performing a detailed timing simulation of a dynamically scheduled microprocessor. Simulation is execution-driven, including execution down any speculative path until the detection of a fault, TLB miss, or branch misprediction. We modified the simulator to model the pipeline in Figure 4 and made it configurable so that the impact of different critical path latencies can be studied.

Instruction Cache	32KB, 32B/block, 2-way
Decode Width	4
ROB Size	128 entries
Issue Queue	32 entries
Issue Width	4
Register File	128 entries
Load/Store Queue	32 entries
L1 Data Cache	16KB, 32B/block, 4-way, 3RW ports
Unified L2 Cache	1MB, 64B/block, 8-way

TABLE I
BASELINE PROCESSOR PARAMETERS.

Block	Area (mm^2)	Delay(ps)	AR (H/W)
Branch Predictor	0.275	530	1.20
I-cache	0.656	556	1.20
D-cache	0.701	881	0.37
Mapper	0.356	681	2.00
Register File	1.298	415	1.30
Issue Queue	0.279	671	2.30
IntAlu	0.237	468	1.00
IntMult	0.330	3172	1.75
FPAU	0.350	1820	1.00
FPMult	0.591	1820	1.75
Load/Store Unit	0.400	400	1.25
L2 cache	15.200	1890	1.49

TABLE II
LIST OF BLOCKS IN THE CHOSEN MICRO-ARCHITECTURE AND THEIR PROPERTIES.

To perform our evaluation, results were collected for 23 SPEC2000 [24] benchmarks. The programs were compiled on a DEC Alpha AXP-21164 processor using the DEC C and C++ compilers under an OSF/1 V4.0 operating system using full compiler optimization (`-O4 -if0`). Each program has been simulated using the *ref* set for that application for 100 million instructions after fast-forwarding past the initialization portion of the benchmark (as described in [25]).

In order to consider the impact of pipelining based on interconnect delays, we use the following critical blocks and paths in this study:

- ALU : latency through the ALU and the bypass logic.
- DCACHE : latency through the L1 data cache.
- L2CACHE : latency for access to L2 cache.
- MPLAT : latency through the branch resolution path, which is the time between when a branch is predicted and when the actual outcome of the branch is known.

Table II lists the blocks in our micro-architecture along with their respective area and delay numbers. The delay shown in the table is the total logic delay through the block, assuming no pipelining. The area and delay of the blocks were derived using the method in Section III-A for a futuristic $70nm$ process technology. Based on [12], we assume that the clock cycle overhead $T_{overhead}$ is $46ps$, which corresponds to roughly 1.8FO4 for $70nm$ technology parameters. Thus, for a 4Ghz target cycle time, we set $T_{useful} = 250ps - 46ps = 204ps$ and use this to calculate the number of pipeline stages required to cover a given path delay. As we study the impact of semi-global interconnects in the micro-architecture, we assume that all the signals are routed in the routing layers meant for semi-global wires. The delay of interconnects is derived using the IPEM models [26] which consider several optimizations such as wire sizing, buffering and sizing, etc.

A. Sensitivity Models

In order to study the impact of extra latencies through these critical paths, we performed a number of architecture simulations and summarize the results in Figure 5. Since the actual latency of the paths depends on the target frequency, these results are obtained without any assumption on the target frequency and mainly by using the baseline processor parameters listed in Table I. We can see from Figure 5 that

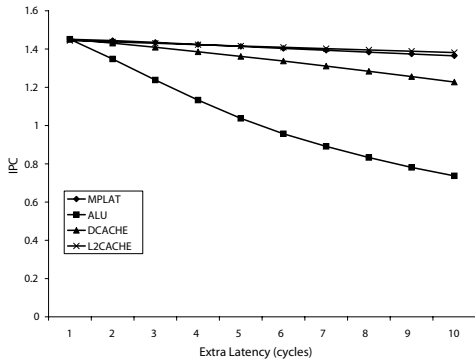


Fig. 5. Sensitivity of the micro-architecture to critical components in the pipeline.

the latency through the ALU (along with the bypass logic latency) impacts the performance most. This is because any delay through the ALU affects the latency between dependent instruction execution. The L1 data cache latency degrades the performance to a lesser extent as only memory operations are dependent upon this latency. Finally, the performance seems to be quite tolerant of the latency through the L2 cache as well as the branch misprediction resolution path. This is due to the small number of L1 data cache misses as well as high branch prediction accuracy due to advanced algorithms. Consequent to the sensitivities shown in Figure 5, we use a *constant* factor based sensitivity model for the critical paths ALU, DCACHE and L2CACHE, and a *piece-wise linear* model for the path MPLAT.

To understand the accuracy of using these models to estimate the changes to IPC, we generated *eight* micro-architecture configurations by varying the properties of some of the blocks as listed in Table III. We chose five dif-

No.	Width	D-cache	ROB/IQ/LSQ/RegFile	Area
1	4/4/4	16K, 3 RW	128/32/32/128	0%
2	4/4/4	16K, 3 RW	256/64/64/256	7%
3	8/8/8	16K, 3 RW	128/32/32/128	19%
4	8/8/8	16K, 3 RW	256/64/64/256	41%
5	4/4/4	32K, 2 RW	128/32/32/128	-1%
6	4/4/4	32K, 2 RW	256/64/64/256	6%
7	8/8/8	32K, 2 RW	128/32/32/128	19%
8	8/8/8	32K, 2 RW	256/64/64/256	41%

TABLE III

ARCHITECTURE CONFIGURATIONS USED IN THE STUDY. WIDTH DENOTES THE DECODE/ISSUE/COMMIT WIDTH OF THE PROCESSOR IN THAT ORDER. AREA IS THE RELATIVE CHANGE FROM THE BASELINE PARAMETERS, WHICH ALSO CORRESPONDS TO CONFIGURATION 1.

ferent target frequencies from 4Ghz to 8Ghz to study the accuracy of the models when the frequency is changed. For this study, the latency of each block was obtained as

$[block_delay/cycle_time]$. We ignored the impact of wire delays, and the latency of each path was the sum of the latencies of the blocks along that path. Based on these latency numbers, we obtained the actual IPC through simulations and compared the change in the IPC to the values predicted using the sensitivity models. The error between these two values for all the configurations is shown in Figure 6. We can see that the

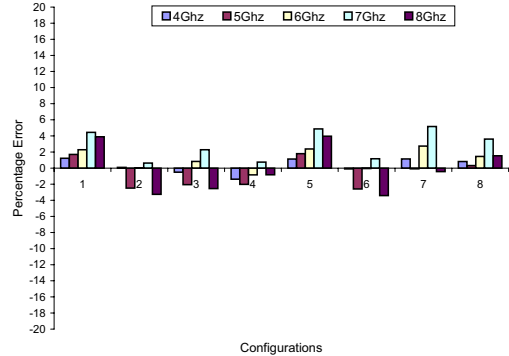


Fig. 6. Error in tracking IPC changes using the sensitivity models.

sensitivity models are within 5% error in tracking the changes in IPC for the chosen design space.

As mentioned earlier, for a larger solution space, some of these paths could be interacting and it may become necessary to study their interaction using advanced techniques as in [20]. However, as the above results show that these models are sufficiently independent for our sample design space, we present results here which assume no interaction among these paths.

B. Effect of Wire Pipelining

The work in [6] did not consider any wire pipelining and produces cycle time estimates based on the longest delay of critical interconnections. In order to understand the difference between this approach and the proposed approach with wire pipelining, we used the MEVA tool [6] and generated performance results at different target frequencies ranging from 4Ghz to 8Ghz for each of the eight configurations in our design space. The BIPS results from this experiment are shown in Figure 7. Also, for these configurations, we generated performance data using the proposed approach where long interconnects were pipelined based on the floorplanner in Section III-C. These results are shown in Figure 8.

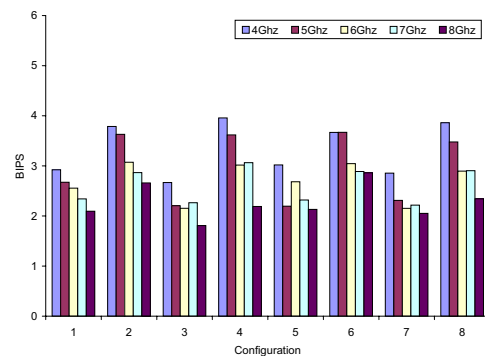


Fig. 7. BIPS for the architecture configurations based on the approach from previous work where wires are NOT pipelined.

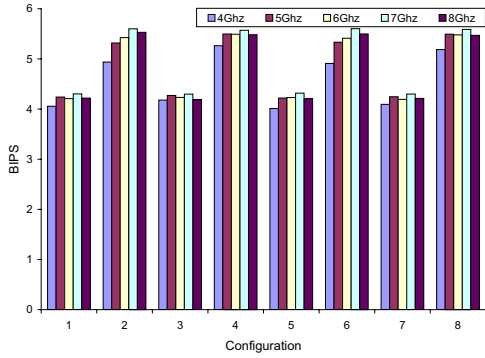


Fig. 8. BIPS for the architecture configurations based on the proposed approach with interconnect pipelining.

Figures 7 and 8 show a few interesting results. First, the previous approach (Figure 7) always estimates a lower performance for all the configurations. This is due to the phenomenon explained through our example in Figure ?? where long wires determine a cycle time larger than the targeted cycle time at each frequency. Analyzing the floorplan output, we found that typically such long delay wires were connected to the execution units from either the branch predictor or the data cache. Since these wires are not pipelined, this results in very low estimates of performance for the configurations. However, based on the results from the proposed approach (Figure 8), we found that these interconnects with large delays were pipelined and the impact of the extra latency due to this wire pipelining did not degrade the IPC significantly. This, coupled with the fact that we were able to achieve the target frequency, results in a much higher estimate of the BIPS compared to the method used in [6]. We can see that the improvement in the estimated performance is anywhere between 25% to 45%.

Secondly, we find that identifying the interconnect bottlenecks (as in the previous approach) is alone not sufficient in identifying the quality of individual configurations. For example, Figure 7 shows that configuration 5 performs better than configuration 3 at 5GHz, while we can see that this is not the case in Figure 8. Also, Figure 7 shows a relatively insignificant performance difference between configurations 4, 5 and 7 at 8GHz. However, based on the proposed approach, we see a remarkable difference in BIPS between 4, 5 and 7, as seen in Figure 8.

Finally, the results in Figure 7 make us to conclude that performance drops significantly with increasing frequency. However, we see from Figure 8 that in most cases, performance increases with frequency, even if the increase is not significant.

C. Impact of Interconnect Latency on Frequency Exploration

To demonstrate the need for the proposed approach in identifying good target frequencies, we present results for two cases in Figure 9.

- **layout:** Without considering any layout information, we use $\lceil \frac{\text{delay}}{\text{cycle_time}} \rceil$ to calculate the latency of the paths and blocks. For example, a block with 500ps delay will be pipelined to 3 cycles at 200ps cycle time and 4 cycles at 150ps cycle time. Notice that this is a lower bound on the

latency through any path or block, and the wire latency will only increase the latency through these components. Based on these latency numbers, we use the architecture simulator to obtain the IPC and calculate the BIPS at each frequency.

- **layout:** Based on the proposed approach, we obtain the block and wire pipelining solution from the floorplan at each of the target frequencies and generate the BIPS based on these latency numbers for the blocks and the paths.

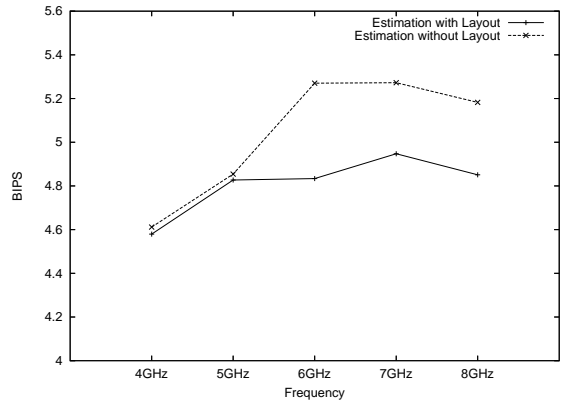


Fig. 9. Overestimation of the BIPS for the architecture configurations when impact of wire latency is ignored. Data also shows importance of wire latency in target frequency exploration.

The results in Figure 9 show the average over *all* eight configurations. Our first observation is that the BIPS estimated without considering any layout information in Figure 9 is overly optimistic and considerably higher than the BIPS estimated with layout information, especially at higher frequencies. This is expected because wire latency from the layout can only degrade performance, and this effect is completely ignored when layout is not considered. The difference between the two approaches is very small for lower frequencies (4GHz and 5GHz) because the wires that were pipelined at this frequency were on the less critical branch misprediction resolution path (MPLAT), which has the least impact on performance. However, at higher frequencies (6GHz and above), wires on some of the most critical paths such as the ALU bypass and the data cache access are pipelined, and this significantly degrades performance as shown in the Figure 9. As these paths are highly sensitive to extra latency, we see a degradation of 8%-10% when considering the impact of wire latency on the performance.

Secondarily, Figure 9 also shows the importance of exploring the target frequencies with consideration of wire latency impact. Without wire delay, the BIPS obtained at 6GHz and above appears to be around 8% better than the BIPS at lower frequencies. However, a layout-driven approach shows that the BIPS does not really vary significantly after 5GHz target frequency, and the benefit of increased frequency is offset by the impact of interconnect latency. As the dynamic power consumption is a function of the processor frequency, such insights into performance variation with increasing frequency helps when choosing the right power/performance tradeoff. These results together underline the importance of early micro-

architecture planning with consideration of wire latency.

V. CONCLUSIONS

In this work, we presented a micro-architecture evaluation methodology with floorplanning and interconnect pipelining. Specifically, we coupled architecture performance sensitivity models with traditional floorplanning to optimize for micro-architectural critical path latencies. Experimental results demonstrated that interconnect pipelining based approach improved the estimated performance between 25% to 45%. Also, results showed the value of simultaneous optimization of architecture and physical design in providing insights into the behavior of the processor at different operating frequencies. As future work, we would like to study the use of higher order IPC sensitivity models and consider power/thermal effects during physical evaluation of microprocessors.

VI. ACKNOWLEDGEMENTS

This research was partially supported by the Semiconductor Research Corporation under contract 2001-TJ-910, National Science Foundation under Grant CCR-0096383, and Intel Corporation under the California MICRO program.

REFERENCES

- [1] M. Bohr, "Interconnect Scaling - The Real Limiter to High-Performance ULSI," in *Tech. Dig. of the International Electron Devices Meeting*, pp. 241–244, Dec. 1995.
- [2] R. Ho, K. W. Mai, and M. A. Horowitz, "The Future of Wires," in *IEEE*, vol. 89, pp. 490–504, April 2001.
- [3] J. Cong, "An Interconnect-Centric Design Flow for Nanometer Technologies," in *Proceedings of IEEE*, pp. 505–527, April 2001.
- [4] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel, "The Microarchitecture of the Pentium 4 Processor," *Intel Technology Journal Q1*, 2001.
- [5] E. Sprangle and D. Carmean, "Increasing Processor Performance by Implementing Deeper Pipelines," in *29th Annual International Symposium on Computer Architecture*, 2002.
- [6] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis, "Microarchitecture Evaluation With Physical Planning," in *Design Automation Conference*, pp. 32–35, June 2003.
- [7] M. R. Casu and L. Macchiarulo, "Floorplanning For Throughput," in *Proceedings of the 2004 International Symposium on Physical design*, pp. 62–69, 2004.
- [8] M. Ekpanyayong, J. R. Minz, T. Watwai, H.-H. S. Lee, and S. K. Lim, "Profile-Guided Microarchitecture Floorplanning For Deep Submicron Processor Design," in *Design Automation Conference*, June 2004.
- [9] C. Long, L. J. Simonson, W. Liao, and L. He, "Floorplanning Optimization with Trajectory Piecewise-Linear Model for Pipelined Interconnects," in *Design Automation Conference*, June 2004.
- [10] M. Rewienski and J. White, "A Trajectory Piecewise Linear Approach to Model Order Reduction and Fast Simulation of Nonlinear Circuits and Micromachined Devices," in *IEEE Trans. on Computer Aided Design*, pp. 155 – 170, 2003.
- [11] E. Borch, E. Tune, S. Manne, and J. Emer, "Loose Loops Sink Chips," in *Proc. of 8th International Symposium on High-Performance Computer Architecture*, Jan 2002.
- [12] M. Hrishikesh, K. Farkas, N. Jouppi, D. Burger, S. Keckler, and P. Sivakumar, "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays," in *Proc. of 29th International Symposium on Computer Architecture*, May 2002.
- [13] D. Carmean Intel Corporation, Personal Communication.
- [14] V. Nookala and S. S. Sapatnekar, "A Method for Correcting the Functionality of a Wire-pipelined Circuit," in *Proceedings of the 41st annual conference on Design automation*, pp. 570–575, 2004.
- [15] S. Wilton and N. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model," in *IEEE Journal of Solid-State Circuits*, May 1996.
- [16] S. Palacharla, N. Jouppi, and J. E. Smith, "Complexity Effective Superscalar Processors," in *Proc. International Symposium on Computer Architecture*, pp. 206–218, June 1997.
- [17] D. C. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0," Technical Report CS-TR-97-1342, University of Wisconsin, Madison, June 1997.
- [18] S. Nussbaum and J. Smith, "Modeling Superscalar Processor via Statistical Simulation," in *International Conference on Parallel Architectures and Compilation Techniques*, pp. 15–24, Sept. 2001.
- [19] L. Eeckhout and K. D. Bosschere, "Hybrid Analytical-Statistical Modeling For Efficiently Exploring Architecture and Workload Design Spaces," in *International Conference on Parallel Architectures and Compilation Techniques*, pp. 25–34, Sept. 2001.
- [20] B. Fields, R. Bodik, M. Hill, and C. J. Newburn, "Using Interaction Cost for Microarchitectural Bottleneck Analysis," in *Proc. of 36th International Symposium on Microarchitecture*, Dec 2003.
- [21] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Number 4598, 13 May 1983, vol. 220, 4598, pp. 671–680, 1983.
- [22] H. Murata, F. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement on Rectangle Packing by Sequence-Pair," *IEEE Transactions on Computer-Aided Design*, vol. 15, pp. 1518–1524, Dec 1996.
- [23] S. Adya and I. Markov, "Fixed-outline Floorplanning Through Better Local Search," in *Proc. International Conference on Computer Design*, pp. 328–334, 2001.
- [24] "The Standard Performance Evaluation Corporation," 2000. <http://www.spec.org>.
- [25] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.
- [26] J. Cong and D. Pan, "Interconnect Estimation and Planning for Deep Submicron Designs," in *Proc. Design Automation Conference*, pp. 507–510, 1999.