

Microarchitecture Evaluation With Physical Planning

Jason Cong, Ashok Jagannathan, Glenn Reinman, Michail Romesis
Computer Science Department
University of California, Los Angeles
{cong,ashokj,reinman,michail}@cs.ucla.edu

ABSTRACT

Conventionally, microarchitecture designs are mainly guided by the maximum throughput (measured as IPC) and fail to evaluate the impact of architectural decisions on the physical design, and in particular, the impact on the interconnects. In this paper, we propose MEVA, a system to consider both IPC and cycle time in the design space search for a given microarchitectural design. MEVA can consider a variety of user-specified architectural alternatives that trade IPC and cycle time in the design, and performs accurate floorplanning and simulation to fully evaluate each alternative. The resulting solution will maximize the benefit from both IPC and cycle time to provide a better solution than a design space exploration based simply on IPC or cycle time alone. For a sample architectural design, we are able to search a space of 32 architectural configurations with physical planning in less than 2 hours to find a processor configuration that, in terms of BIPS, outperforms the configuration with the best IPC performance by 14%, and the configuration with the fastest clock by 27%. This initial exploration only considers the boundary cases of a much larger design space, but still features substantial IPC and cycle time variation.

Categories and Subject Descriptors

B.8.2 [Hardware]: PERFORMANCE AND RELIABILITY

General Terms

Algorithms, Design, Performance

Keywords

Microarchitecture evaluation, physical planning

1. INTRODUCTION

There are a number of hardware challenges that future architects will need to face in the design of the next generation of microprocessors. As feature sizes continue to shrink,

it has become evident that interconnect delay is not scaling at the same rate as transistor delays. Architects have been adding a variety of components to the chip to improve processor performance. These components have caused the chip area to increase with successive technology generations [1], complicating layout and routing. Also, as the clock speed of the processor continues to increase correspondingly dropping the cycle time of the processor, the interconnect scaling problem becomes even more severe and the processor may be unable to communicate across the chip in a single cycle [1, 2]. Agarwal et al. [1] predict that current processor designs will improve by at best, 12.5% per year in terms of performance over the next fourteen years due to hardware scaling concerns.

Prior work [1, 3, 4] has demonstrated the need to consider both cycle time and throughput (IPC) when measuring overall processor performance. However, architects often have little or incomplete physical design information about the architectural space they are considering. Accurate area and delay information for a given logic block can be difficult to derive without an actual implementation of the architecture. Moreover, without accurate physical planning, designers cannot measure the considerable impact interconnect can have on a given architecture.

In general, the execution time for a given program is defined as

$$T_{exec} = num_instructions * CPI * cycle_time$$

where *num_instructions* is the dynamic instruction count of the program, *CPI* or cycles/instruction is the average number of cycles required to execute a given instruction, and *cycle_time* is the inverse of the processor frequency (measured in seconds). Much research has been dedicated to exploring the interaction between compilers and architectures to better improve both the CPI and dynamic instruction count of a given application space. However, it is equally important to consider the interaction between architectural design and physical design (the latter two variables in the execution time equation). This requires designers to carefully explore both spaces together, making architectural decisions that maximize benefits to both throughput and cycle time.

The design space explored by architects during processor design can grow quite large when considering different algorithms (i.e. different branch predictors), component sizes and characteristics (i.e. cache size or associativity), or pipeline depth (i.e. cache access latency). To better manage and explore this space in the face of future hardware scaling challenges, we propose a Microarchitec-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2003, June 2–6, 2003, Anaheim, California, USA.
Copyright 2003 ACM 1-58113-688-9/03/0006 ...\$5.00.

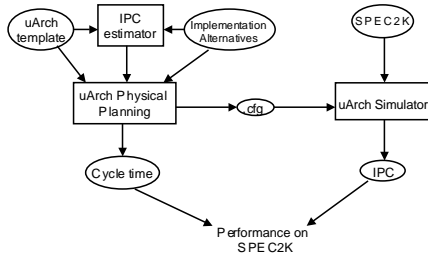


Figure 1: Overview of the MEVA system

tural EVALuation (MEVA) system to provide a set of flexible and customizable tools to explore an architectural design space with both accurate physical planning information and cycle-accurate architectural simulation. Through this joint exploration of physical and architectural design, architects can better study what designs are able to tolerate poor wire scaling and still achieve high performance.

The rest of the paper is organized as follows: In Section 2, we present an overview of the MEVA system. Section 3 discusses the physical planning engine. Experimental results are presented in Section 4 and the paper is concluded with some suggestions for future work in Section 5.

2. OVERVIEW OF MEVA

Figure 1 shows the overall picture of the MEVA system. In the following section, we briefly discuss the inputs, outputs and the various components of this system.

Inputs to MEVA

The MEVA system takes two inputs – (a) an *architectural template*, which is essentially a block-level netlist that captures connectivity of the major functional blocks and (b) a *library* of architectural alternatives for the different blocks in the template. Together, the architectural template and the library of alternative block implementations capture a class of microarchitectures where the underlying connectivity is fixed, while the individual block properties such as area, timing, latency can vary significantly. Such alternative architectures for the blocks affect the IPC through varying latency properties, and also affect the cycle time as they have different area/timing characteristics. It is important to note that it is impossible to represent all classes of microarchitectures using a single template. Thus, one can come up with a variety of templates and corresponding library files to model any architectural space, and use the MEVA system to evaluate each class of microarchitectures with physical planning.

Since the choice of any architectural alternative on the physical design space is measured by its impact on the achievable cycle-time, the area and timing properties of these alternatives should be modeled in sufficient detail. Each alternative is characterized with the following information:

- (i) **Area** of the block.
- (ii) **Longest delay** D of any combinational path inside the block.
- (iii) **Input-to-clock time** (T_{su}): For each input pin on the block, this value specifies the maximum delay from this pin to any flip-flop inside this block.
- (iv) **Clock-to-output time** (T_{co}): For each output pin on the block, this value specifies the maximum delay to this pin from the output of any flip-flop inside this block.

The T_{su} and T_{co} values are very important in deciding the

freedom available for the interconnects that are connecting this block to the rest of the design. We believe that the above mentioned information is good enough for careful interconnect planning.

In MEVA, we use structural Verilog to represent a given architectural template and a Synopsys .lib like format for our library of architecture alternatives.

Components of MEVA

The MEVA system consists of three main components which are (a) physical planning engine, (b) IPC estimator and (c) cycle-accurate microarchitecture simulator.

The *physical planning engine* takes as input the architectural template and the library of architecture alternatives and performs a floorplanning of the design with interconnect planning to optimize a given cost function, which can include a combination of objectives from the architectural and physical design spaces such as IPC, cycle time, power etc. During this process, the planning engine also chooses different alternative implementations for each block from the library to achieve the best set of block implementations for the given objective. The planning engine also considers various layout related issues such as pin-assignment for the blocks, routability of the floorplan etc., when it attempts to find the best microarchitecture in terms of the given constraints. This planning engine is presented in Section 3.

The goal of the *IPC estimator* is to provide the planning engine with a quick and accurate IPC estimation for any configuration of block alternatives chosen by the planning engine at any point of time. In this work, we use a cycle-accurate simulator to generate IPC values for any given architectural configuration as discussed in [5].

Once the planning engine determines the best configuration for each block based on the IPC estimates and the results of the floorplan, this result can be fed to a cycle-accurate microarchitecture simulator. The *simulator* performs a detailed simulation of the underlying architectural model for a given set of benchmark programs, and provides accurate IPC information characterizing the microarchitecture.

Outputs from MEVA

The output from MEVA includes the selected architectural alternative for each block in the template along with the best possible cycle time information that it can derive using its physical planning engine, subject to the cost function specified to the planning engine. The latency of the architectural alternative for each block can then be fed to a cycle-accurate architecture simulator for that template to generate an accurate IPC value. The cycle time information from the physical planning engine and the accurate IPC value from the simulator can then be used to get a good estimate of the performance of the microarchitecture on the given set of benchmarks.

3. PHYSICAL PLANNING ENGINE

The physical planning engine in MEVA has been developed to address the following goals: (i) Optimize the performance of the system measured as the number of instructions executed per second – i.e., IPC/cycle_time, subject to other physical design constraints such as area. (ii) Consider different architectural alternatives for the blocks when searching

for the architecture with the best performance. (iii) Consider interconnect planning during the floorplanning stage.

The inputs to the engine are (a) an architectural template, (b) a library file that contains area and timing information for different architecture alternatives of each block and (c) a list of allowed architectural combinations along with the IPC for that configuration. The output of the engine is a layout of the blocks with a selection of an architecture combination such that the performance of the system is maximized under given area constraints. Below, we explain how we address each of the three main goals:

Objective function: The objective function for our planning engine is as follows:

$$\frac{\sum_{i=1}^n w(i)wl(i)}{IPC(c)}$$

where $w(i)$ is the weight of a net i , $wl(i)$ is the wirelength of i , and $IPC(c)$ is the IPC of the current configuration c . The weights for the nets are computed according to the slacks of their pins. We use a traditional simulated annealing engine and at every temperature we perform static timing analysis. The delays for every pin-to-pin connection are computed by the IPEM estimator [6] for the $0.10\mu\text{m}$ technology which considers optimal buffer insertion, sizing and wire sizing. Using static timing analysis, we can estimate the cycle time for the current layout and the slack time of each pin. For a net n , suppose that the slack time of its source pin is s picoseconds. If the total cycle time is c picoseconds, the weight of the net n is computed as: $(1 - s/c)^a$. The exponential factor a is initialized to 1, and is gradually increased as we move to lower temperatures.

Alternative Block Selection: We introduce a new type of move in the simulated annealing engine, called the configuration selection. When a new configuration is selected, the dimensions and the timing characteristics of some blocks change, as well as the IPC of the floorplan. This is usually a significant change to the floorplan, therefore, in order to evaluate it we first perform a small number of low-temperature additional moves on the new configuration, and we then decide to accept or reject the new configuration.

Interconnect Planning: As mentioned earlier, we use the interconnect performance estimator IPEM [6] for estimating the wire delays under optimal buffer insertion, sizing and wire sizing. Currently, we perform a simple pin assignment algorithm. The algorithm is initially computing a location of the pins while ignoring the spacing constraints. The pins then are spread out until they satisfy the width and spacing requirements. For the immediate future, we plan to integrate pin assignment with global routing according to the algorithm described in [7]. For the global router, a good candidate is the L-Z router presented in [8], as it is very fast and can provide congestion information.

We have implemented a version of our physical planning engine extending the floorplanner PARQUET [9] to support timing optimization, alternative block selection and interconnect planning. Table 1 shows the quality of our physical planning engine compared to existing state-of-the-art floorplanners [10][11] on a set of MCNC benchmarks.

4. EXPERIMENTAL RESULTS

In this section, we report results to support the viability of the proposed approach. For the purpose of this study,

Circuit	Sim-Tempering		TCG		MEVA	
	Area	WL	Area	WL	Area	WL
ami33	1.29	48.6	1.24	50.3	1.28	47.7
ami49	39.24	715.8	38.20	663.1	38.47	646.8
hp	9.58	114.9	9.49	151.8	9.96	101.2
xerox	20.50	417.4	20.42	385.0	21.02	384.3
apte	48.50	223.8	48.48	378.0	49.58	276.6
Average	0.99	1.03	0.97	1.19	1	1

Table 1: Comparison of MEVA with existing floorplanners on MCNC benchmarks.

we developed an architectural template [5] which represents a 4-way out-of-order superscalar processor. This template has been carefully chosen to allow us to study many of the interesting architecture alternatives available to a microprocessor designer. The architecture simulator used is a cycle-accurate simulator built on the SimpleScalar [12] tool set. We perform simulations on 20 randomly selected SPEC2000 benchmarks and use the arithmetic mean to obtain the IPC for any architectural configuration. We have chosen to vary the size and latency of the branch predictor, instruction window and the instruction and data caches primarily due to the fact that variation in these block characteristics will result in significant change in the IPC of the architecture, leading to interesting design tradeoffs. All our area/delay estimates are generated using CACTI [13] and are based on the 100nm technology parameters built within the tool. Based on the delay values, we derive a minimum and maximum latency value for each of the blocks based on the cycle-time of existing state-of-the-art microprocessors. We assume that these blocks can be pipelined using a given number of flip-flops, such that the delay of each stage inside the block is the same. It is important to note that the minimum and maximum latency options will imply different longest combinational delays and T_{su}/T_{co} requirements, which will significantly affect the freedom during physical planning. We generate 32 architectural combinations, 16 for the minimum latency values and 16 for the maximum latency values, as presented in [5].

Figure 2 presents three data points for each of the 32 configurations derived – (a) the average IPC number for each architectural configuration obtained using our cycle-accurate simulator, (b) the cycle time of the best floorplan obtained using our physical planning engine and (c) the corresponding BIPS rating. It can be seen from Figure 2 that configuration #16 has the best IPC and configuration #18 has the best cycle time. However, configuration #24 has the best performance in terms of BIPS, underlining the fact that it is important to look at both the architectural and physical design spaces together to draw conclusions on the overall performance of any microarchitecture. Also, despite the relatively lower IPC of the maximum latency configurations (17-32), we see a dramatic increase in BIPS for these configurations due to the cycle time we are able to achieve with the deeper pipelining. The extra data cache latencies and larger branch misprediction depth are particularly hard to tolerate, as shown in [3], and have a large impact on IPC for these configurations. This again, demonstrates the importance of examining both IPC and cycle time when exploring a design space. Ultimately, these results emphasize the importance of taking interconnect effects into account when exploring an architectural design space. The latencies

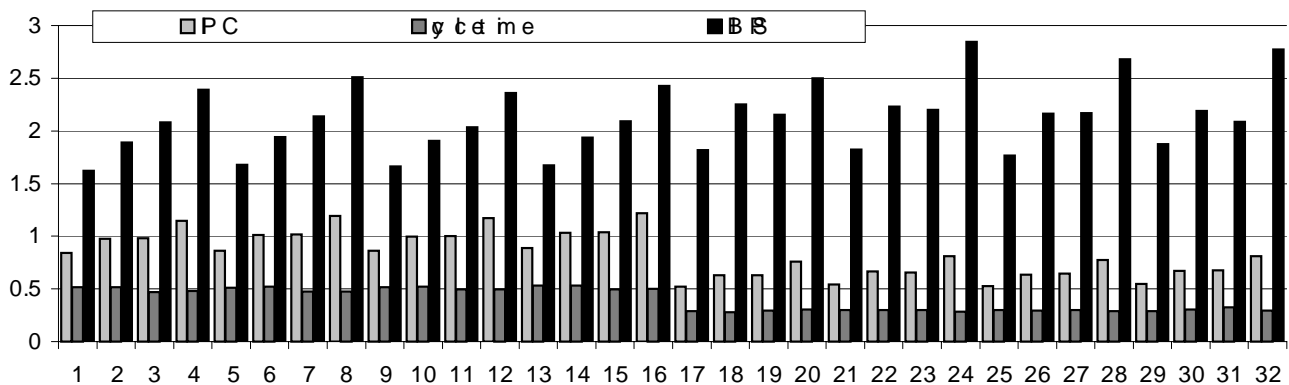


Figure 2: IPC, cycle time and BIPS results for the 32 configurations.

in the architectural alternatives impact the IPC of the configuration, and can be tuned for a desired cycle time, but the true cycle time is only known once the floorplanning is complete. This can have surprising results, as in the case of configurations #2 and #3. Despite a comparable IPC, the wirelength impact of the larger branch prediction logic has a significant impact on BIPS (nearly a 10% reduction in BIPS from configuration #3 to #2).

It should be noted that these results only represent the performance of the boundary cases (corresponding to the minimum and maximum latencies of the blocks) in our driver architecture design space. A configuration with latency between the boundary configurations should provide the most beneficial tradeoff between IPC and cycle time, as the minimum latency alternatives do not provide an aggressive enough clock speed and the maximum latency alternatives are too heavily pipelined (impacting IPC) relative to the impact of wirelength.

Finally, to validate the alternative architecture selection part of our physical planning engine, we tried to search the combined solution space using the method explained in Section 3. Since the tool is searching a very large solution space, we measure the effectiveness of the tool by the quality/runtime tradeoff – i.e., what is the quality of the best configuration identified by the tool, given a fixed amount of computing time? Over several runs, we found that the tool finds the best configuration (#24 in our case) in about 1/4th of the total running time required to generate the best possible floorplan for each of the 32 configurations individually.

5. CONCLUSION AND FUTURE WORK

We have presented a microarchitectural evaluation system named MEVA that attempts to combine the architectural design space with the physical design space. A representative architectural template with corresponding block alternatives was presented to demonstrate the validity of the approach. Both IPC values and cycle times for these different architecture configurations were shown, along with the BIPS value, to emphasize the need for a combined consideration of physical planning during microarchitecture evaluation. Experimental results show the viability of this approach and the need for better tools to evaluate microarchitectures with careful physical planning. Future work will also involve a more efficient search of a much larger design space, beyond the boundary alternatives and more efficient ways to quickly estimate the IPC of microarchitectures.

Acknowledgements

This work is supported in part by Semiconductor Research Corporation under Contract 2001-TJ-910, the MARCO/DARPA Gigascale Silicon Research Center (GSRC), the National Science Foundation under Grant CCR-0096383, and a grant from Intel Corporation under the California MICRO program.

6. REFERENCES

- [1] V. Agarwal, M. Hrishikesh, S. Keckler, and D. Burger, “Clock rate versus ipc: The end of the road for conventional microarchitectures,” in *27th Annual International Symposium on Computer Architecture*, 2000.
- [2] J. Cong, “An interconnect-centric design flow for nanometer technologies,” in *Proceedings of IEEE*, pp. 505–527, April 2001.
- [3] E. Sprangle and D. Carmean, “Increasing processor performance by implementing deeper pipelines,” in *29th Annual International Symposium on Computer Architecture*, 2002.
- [4] G. Reinman, T. Austin, and B. Calder, “A scalable front-end architecture for fast instruction delivery,” in *19th Annual International Symposium on Computer Architecture*, May 1999.
- [5] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis, “Microarchitecture evaluation with physical planning,” Technical Report CSD-030022, University of California, Los Angeles, Mar. 2003. <http://cadlab.cs.ucla.edu/~michail/arch/030022.pdf>.
- [6] J. Cong, and D. Pan, “Interconnect estimation and planning for deep submicron designs,” in *Proc. Design Automation Conference*, pp. 507–510, 1999.
- [7] J. Cong, “Pin assignment with global routing for general cell design,” *IEEE Trans. on Computer Aided Design*, vol. 10, pp. 1401–1412, 1991.
- [8] C. Chang, J. Cong, D. Pan, and X. Yuan, “Physical hierarchy generation with routing congestion control,” in *Inter. Symposium on Physical Design*, pp. 36–41, 2002.
- [9] S. Adya, and I. Markov, “Fixed-outline floorplanning through better local search,” in *Proc. International Conference on Computer Design*, pp. 328–334, 2001.
- [10] J. Cong, T. Kong, D. Xu, F. Liang, J. Liu, and W. Wong, “Relaxed simulated tempering for vlsi floorplan design,” in *Proc. Asia and South Pacific Design Automation Conference*, pp. 13–16, 1999.
- [11] J. Lin, and Y. Chang, “Tcg: A transitive closure graph-based representation for non-slicing floorplans,” in *Proc. Design Automation Conference*, pp. 764–769, 2001.
- [12] 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.
- [13] S. Wilton and N. Jouppi, “Cacti: An enhanced cache access and cycle time model,” in *IEEE Journal of Solid-State Circuits*, May 1996.