

Measuring the Complexity-effectiveness of Future-generation Silicon Architectures Using FPGAs: A Status Report

Andrew Schwerin

Steven Swanson

Mark Oskin

University of Washington
Computer Science and Engineering
{schwerin, swanson, oskin}@cs.washington.edu

Abstract

Historically, there have been two methods for assessing microarchitectural ideas. Most groups use cycle-by-cycle simulation in software, but a few prefer to build hardware prototypes. The inexorable increase in scale and complexity of new architectural features is making both methods more difficult. Simulations are increasingly difficult to verify, and prototypes are growing costly and more time-consuming to build. This paper discusses our preliminary ideas on a middle ground between these two methodologies: using FPGA-based infrastructure for architecture research. The goal of such an approach is to reduce the conceptual distance between technological constraints and software simulation, but without the costs of a full-custom prototype.

Our research group is designing and building the Bathysphere, a large-scale future-generation silicon emulation system from FPGA components. In this preliminary status report, we consider some of the aspects of traditional software simulation that might be complemented by emulation in FPGAs, as well as some of the modeling differences that remain when FPGAs are used instead of ASICs or full-custom ICs. We also present some preliminary data that highlights one of the many challenges in extracting meaningful data from an FPGA-based implementation: properly emulating multi-ported memory structures.

1 Introduction

The increase in microprocessor design complexity that has accompanied the exponential growth in transistor density has led to a commensurate increase in

the cost of developing new microprocessors. This manifests itself in the research community as an increasing difficulty in accurately assessing the benefits of new architectural ideas.

Cycle-accurate software simulation is a popular method for performing these assessments. Constructing a simulator forces researchers to explore some of the complexity of the interactions among, and to a lesser extent the circuit complexity within, major architectural features. For the most part, researchers use the simulator to compare the clock cycle time independent performance (the ubiquitous “IPC”) of the architecture with and without a new feature. Since they are built for this purpose, they often do a poor job of assessing the real complexity-effectiveness of the microarchitecture.

Unfortunately, software simulation has several serious drawbacks. It is slow, limiting the amount of data that can be collected and the kinds of comparisons that can be made. Second, simulators often implicitly mask significant global communication, making “guesses” of an architecture’s cycle time dubious. Finally, its accuracy can be questionable, due to a variety of factors from simplifying assumptions to undiscovered bugs.

To make assessment of new designs more accurate, some processor architects have built prototypes. While building a complete ASIC / full-custom prototype provides valuable insights, it also requires a significant (and often prohibitive within an academic setting) time and monetary investment. Field programmable gate arrays (FPGAs) are seen by many as a potential vehicle to avoid the costs of “fabbing a chip.” Despite the fact that this has been a popular notion discussed at many gatherings of architects for a very long time, few published research studies have used such a methodology.

Emulation atop an FPGA substrate would allow the execution of complex experiments that require more computation than may be reasonably simulated by a cycle-by-cycle simulator. Furthermore, constructing a model for the behavior of an ASIC implementation in terms of the behavior of an equivalent FPGA implementation would allow performance and efficiency properties of new designs to be more reliably evaluated. It would also allow something that is simply not possible with a full-custom IC prototype: The evaluation of many different design alternatives in hardware. Researchers rarely, if ever, build several versions of a chip. On an FPGA, however, tuning a parameter or implementing a new idea is as simple as recompiling and reloading your design.

Despite these advantages, one cannot blindly use FPGAs and expect any more accurate results than cycle-level simulation. Real concerns, such as limited configurable logic blocks (CLBs), limited off-chip pins, and limited flexibility of on-chip memory structures make using FPGAs problematic. Nevertheless, if a reliable scientific methodology can be developed that uses FPGAs then there is great potential to solve some long standing problems of conventional architecture simulation. Namely, a degree of realism will come to designs, since practical considerations such as spatial distribution of microarchitecture structures, memory and port requirements, delays, and long-wire communications will be accounted for. In addition we can begin to study issues that are otherwise difficult to explore, such as operating system effects. In the rest of this paper, we highlight our early thinking about and initial results from developing such a methodology.

In the next section we discuss some well-known drawbacks to conventional cycle-by-cycle simulation. Section 3 introduces the Bathysphere, the FPGA-based deep-submicron architecture exploration device we are currently building. Next in Section 4 we present our early results toward developing a systematic methodology with which to use FPGAs for architecture research. In Section 5 we describe some prior art in this area and finally in Section 6 we conclude.

2 Drawbacks of Simulation

Cycle-by-cycle simulation is, by far, the most widely used method to evaluate architectural ideas, but it suffers from some well known problems. We discuss three of them here: First, software simulators are slow, and this limits our ability to study long-running applications and workload/operating system interactions. Second, simulators are notoriously difficult to verify, frequently resulting in inaccurate results. Finally, the conceptual gap between writing a cycle-level simulator and real hardware makes understanding subtle issues of microarchitecture, such as the actual critical path and wire-distance, challenging. This potentially leads to false conclusions about improvements in instructions-per-cycle always equating improvements in real wall-clock time.

Simulation speed: Simulators make it difficult to perform in depth analysis of the effects of design decisions on long-running computations, because they can only simulate a very small period of processor time. They also fail to facilitate study of interaction between decisions made in the hardware architecture and those in the OS design. The slowness of simulation has led to a number of recent research endeavors in choosing a proper sampling of an application [11, 17]. Particularly shocking are the results reported in [12] which demonstrate just how poor a scientific practice it is to choose an arbitrary subset of an application and assume it is representative of the entire benchmark.

Inaccuracies: Writing a model of an architecture in C is quite different than designing that hardware for real. It is extremely difficult to verify that such models accurately reflect real (or even plausible) hardware structures. Indeed, researchers in [8] attempted to model the Alpha 21264 in detail, and found many seemingly small implementation details complicated their efforts. Also in [8] they attempted to show how such real details lead to inaccurate results in prior work [6], but they later had to publish a retraction [9] due to a modeling error! This simply highlights the difficulty in building accurate simulators and conveying architectural ideas in short conference-paper form. Minor implementation de-

tails are not described due to space limitations, leading to an inability to properly re-implement ideas.

Conceptual-distance: There is no inherently reliable mechanism in cycle-accurate simulation methodologies for estimating the effect of architectural changes on the critical path through pipeline stages. Analytic models (such as CACTI [16]) are sometimes used to estimate the cost of changing the size of memory structures, and in turn to estimate the cost of changing the complexity of components that use memories (caches, branch predictors, etc.). Using these models, architects can compute the number of cycles of delay associated with structures that access large, associative memory structures. For data communications, complex control circuits and computation logic, however, architects are left to making reasonable estimates. In either case, the conceptual distance between a software simulation written in C, and an actual hardware design is quite large. This distance has limited the ability of architects to properly gauge pipeline stage depth, wire delay and architectural complexity.

In an effort to overcome the limitations of existing simulation methodologies, we propose the emulation of architectural designs for future, billion-plus-transistor processors using arrays of FPGAs. Emulation in FPGAs promises to address many of the problems posed by traditional simulations, and to complement, but not replace, simulation.

3 Enter the Bathysphere

There will be two early challenges to the viability of an architectural analysis methodology based upon prototyping designs with FPGAs. The first will be to design an infrastructure capable of emulating the functionality of devices consisting of billions of transistors. The second, and the more thought-provoking, will be to devise a set of guidelines for designing prototypes that are neither artificially restricted by limitations unique to FPGAs, nor over-represent the capabilities of future technology.

To address the first challenge, we are building the Bathysphere – a deep-sub-micron exploration vehicle. The Bathysphere is a two-dimensional array of Xilinx Virtex 1000 FPGAs. Attached to each FPGA

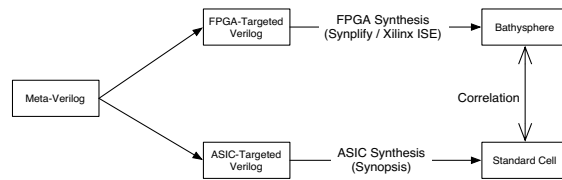


Figure 2: The tool-chain employed for our correlation study.

is 512K words of 32-bit (2 megabytes total) SRAM. These FPGA / memory units are arranged into a grid network in which the cost of long-distance communication (between FPGAs) is exposed to the architecture and is far higher than that of local communication (within an FPGA). Each FPGA has limited, nearest-neighbor connections to other FPGAs in each of the four cardinal directions, but communication over longer distances must be routed through the intervening FPGAs.

Physical design of the board is ongoing, however, our current plan is to have a custom PCB board with 16 FPGA / memory units by this Fall. Between each adjacent FPGA is 76 dedicated programmable I/O pins, with those FPGAs on the edge of the board having a similar I/O interface (Figure 1). A distributed power regulation scheme is used with segmented power planes within a multi-layer PCB board. Each of the FPGAs can be programmed individually or they can all be programmed simultaneously by an external device. We intend to fabricate four of these boards for a total 64M-gate (approximately 256M transistor) logic emulation system with a total of 128 megabytes (6B transistor) memory. After testing this board we our planning on making the entire design source files available in the public domain to further facilitate architecture research.

Despite the flexibility of this logic emulation system for modeling future generation ASICs, there are serious challenges to using FPGAs for architecture research. Our ultimate goal is to develop a systematic methodology for using the Bathysphere such that one can target the FPGA directly, and, through a set of design techniques, pre-packaged parameterized Verilog models, and analytical processes, “un-factor” the effects of the FPGA targeting to arrive at reasonable estimates of the equivalent area and delay within the ASIC process (Figure 2).

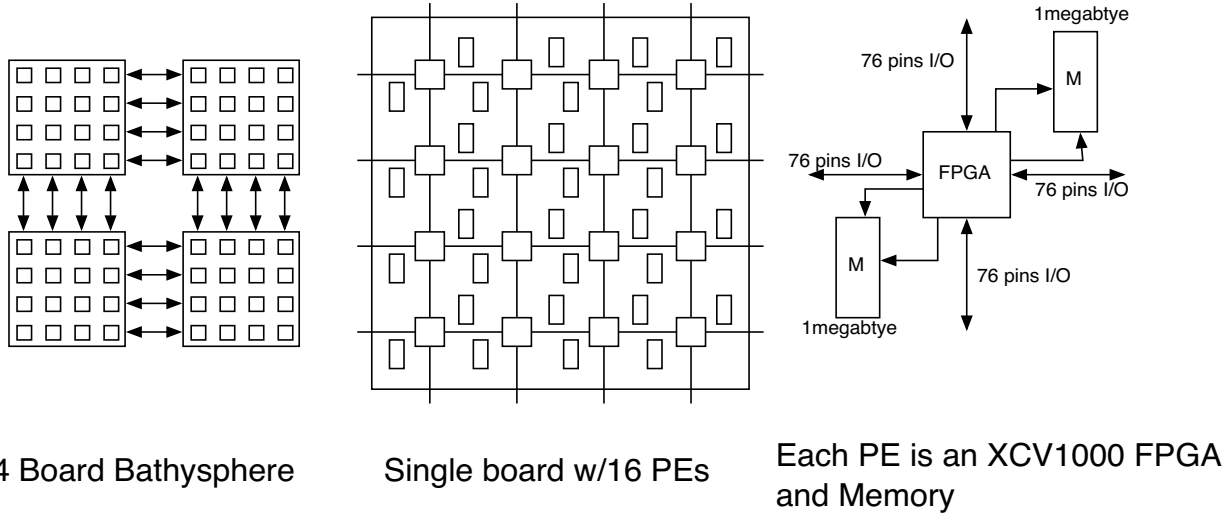


Figure 1: The Bathysphere emulation system.

This goal may not be possible. In this case our secondary goal is to develop and use a careful Verilog coding style that allows easy retargeting to the Bathysphere for emulation and to a standard-cell ASIC process for area and delay measurements. Our work is ongoing in this area, but we describe some initial results in exploring memory structures in the next section.

4 Emulating Multi-ported Structures

Most of the details of using FPGAs for architectural research will need to be explored in future work. In this section we present our initial investigation into only one aspect, memory structures. It is one area where the resource costs between an FPGA and a full-custom / ASIC is substantially dissimilar.

Using the numerical models from CACTI [16, 10, 13], one can see that the access time and die area of a small, non-associative memory with one read port and one write port grow roughly linearly with the size of the memory (Figures 3 and 4). Synthesizing a similar memory using the configurable logic blocks of an FPGA for storage (FPGA LUT RAM in the graphs), one sees a very similar trend for area, measured as the fraction of CLB resources consumed. To a lesser extent, there is a similar trend for latency. However, when using the dedicated RAM resources

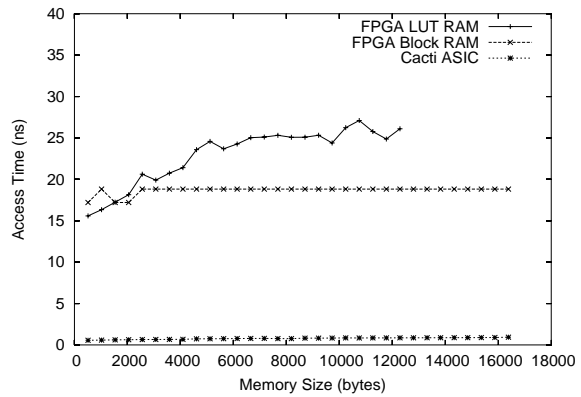


Figure 3: Memory Size vs. Access Time

found on many FPGAs today (denoted FPGA Block RAM in the graphs), the access time holds relatively constant, while the total resources required grows linearly with memory size. Regardless of synthesis technique, though, the amount of memory that may be synthesized on a single FPGA is quite limited – only about sixteen kilobytes on a Virtex 1000 FPGA. For this reason the Bathysphere includes 2 megabytes of dedicated SRAM for attached to each FPGA..

Worse than the limitations on quantity of memory available in an FPGA are the limitations on the number of ports that memory structures in the FPGA can reasonably support. A 16-entry, 64-bit, single-cycle-latency memory with two write ports and a single read port consumes over a quarter of the CLB

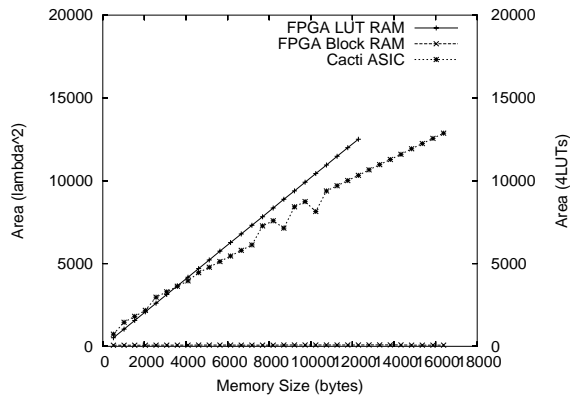


Figure 4: Memory Size vs. Area

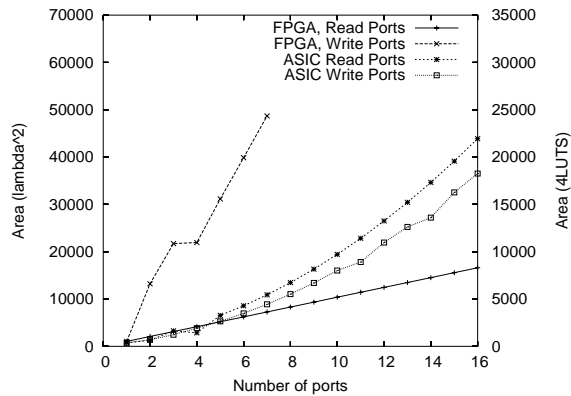


Figure 6: Memory Ports vs. Area

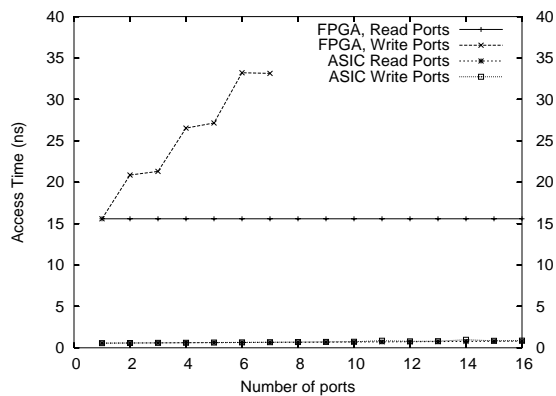


Figure 5: Memory Ports vs. Access Time

resources on a Virtex 1000 (which has over 12K CLBs). A similar memory with a single write port consumes a mere two percent, and one with three write ports overwhelms our synthesis tools.

Figures 5 and 6 show how the number of read and write ports on a memory structure affect its latency and physical resource consumption. Access time grows very slowly, but linearly, with the number of access ports, except in the case of write ports in the FPGA implementation. Here, the access time grows quickly, though still linearly, as the synthesis tools produce time-consuming consistency logic that allows all writes to be visible in the subsequent cycle.

Forcing prototypes built atop the Bathysphere to use single-ported memory structures would be unreasonable, because they are available in direct silicon implementations. Finding an alternative solution is imperative. We are currently considering solutions based upon an emulation clock running at a

fraction of the speed of the physical clock. This allows single-ported memories to be time-multiplexed, within a single emulation clock cycle, giving the appearance of multi-ported structures. For physical resource consumption studies, an analytical model relating portedness to die area could be used.

Though this solution lifts the false barrier restricting the number of memory ports, it introduces the possibility of inadvertently implementing prototypes in a manner that masks the resource costs of other logic. For example, in a time-multiplexed, multi-ported cache, it would be easy to accidentally re-use the tag matching logic in every multiplexing cycle. This effectively time-multiplexes that logic, requiring that its resource consumption, too, be evaluated analytically. Our current focus is on the development of abstractions that will separate the details of emulating multiple ports from the designs that require multiported structures. We hope that such abstractions may also be generalized to the emulation of wide communication channels, similar to the Virtual Wires of [3].

5 Related Work

The authors of [2] built a prototype of an extensible logic emulator and simulation accelerator. They use a time-multiplexing technique [3] to overcome the limited pin count of FPGAs. Their emphasis is primarily on fast functional emulation. Our work aims to extend this by using the prototyping substrate to bring the design complexity and communication costs to the fore and formally assess them.

There has been a wealth of work in simulator design [4, 15], and simulation languages such as SystemC [1]. While useful, they often suffer from the conceptual gaps and limitations in measuring complexity-effectiveness previously described in Section 2. In the future it would be interesting to merge simulation design in toolkits such as Liberty [15] with hardware design and prototyping with the Bathysphere.

In [5] and [7], groups at Berkeley and Stanford assessed the costs of using ASIC processes as opposed to full custom IC processes in terms of area and delay. The authors of [7] found a large penalty in area and a lesser, but significant penalty in delay for using fully automated place-and-route of standard cells to implement the microprocessor register fetch stage, as compared to a full custom implementation. They also found that the penalties could be mitigated by introducing more custom design techniques into the standard cell ASIC flow.

In [5], the authors explore the causes for the performance gap between ASIC processes and custom processes. They conclude that all of the performance penalty of ASIC processes, except for a factor of 2-3 \times , stems from poor pipelining options in standard cell libraries and high tolerance for fabrication process variation leading to conservative design libraries.

These efforts are relevant to the Bathysphere, because they have already established ASIC to full-custom correlation, and we hope to establish a similar correlation between FPGAs and ASIC designs. There two consequences for our work: First, some of the lessons of the work on ASICs apply to FPGAs as well. For instance, hand floor planning can help close the gap between ASICs and full-custom designs, and the same should hold for FPGA designs. Second, our goal is to quantify the differences between FPGA-based and full-custom implementations of the same circuit, and one approach is to quantify the difference between FPGA and ASIC implementations and use this related work to extend the analysis to full-custom designs.

6 Conclusion

Work on the Bathysphere is ongoing. In this paper we have described our preliminary efforts and early results towards correlating this hardware platform based on FPGAs to an ASIC design flow. In the future we intend to continue our work in developing techniques allowing us to use FPGAs for architectural analysis and apply to building a prototype WaveScalar [14] processor. Ultimately, our goal is to develop and release a complete methodological framework that allows researchers to combine measurements of complexity-effectiveness, wire-delay, and real area requirements with traditional instructions-per-cycle measurements.

References

- [1] Guido Arnout. C for system level design, September 1999.
- [2] J. Babb, R. Tessier, M. Dahl, S. Hanono, D. Hoki, and A. Agarwal. Logic emulation with virtual wires. *IEEE Transactions on Computer Aided Design*, June 1997.
- [3] Jonathan Babb, Russell Tessier, and Anant Agarwal. Virtual wires: Overcoming pin limitations in FPGA-based logic emulators. In Duncan A. Buell and Kenneth L. Pocek, editors, *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 142–151, Los Alamitos, CA, 1993. IEEE Computer Society Press.
- [4] Doug Burger, Todd M. Austin, and Steve Bennett. Evaluating future microprocessors: The simplescalar tool set. Technical Report CS-TR-1996-1308, University of Wisconsin - Madison, 1996.
- [5] D. G. Chinnery and K. Keutzer. Closing the gap between asic and custom: An asic perspective. In *Proceedings of the Design Automation Conference*, 2000.
- [6] José-Lorenzo Cruz, Antonio González, Mateo Valero, and Nigel P. Topham. Multiple-banked register file architectures. In *Proceedings of the*

- 27th annual international symposium on Computer architecture*, pages 316–325. ACM Press, 2000.
- [7] William J. Dally and Andrew Chang. The role of custom design in ASIC chips. In *Proceedings of the Design Automation Conference*, 2000.
- [8] Rajagopalan Desikan, Doug Burger, and Stephen W. Keckler. Measuring experimental error in microprocessor simulation. In *28th International Symposium on Computer Architecture*, pages 266–277, June 2001.
- [9] Rajagopalan Desikan, Doug Burger, Stephen W. Keckler, Llorenc Cruz, Fernando Latorre, Antonio Gonzalez, and Mateo Valero. Errata on "measuring experimental error in microprocessor simulation". *ACM SIGARCH Computer Architecture News*, 30(1):2–4, 2002.
- [10] Glen Reinman and Norman P. Jouppi. CACTI 2.0: An integrated cache timing and power model. Technical report, Compaq Western Research Laboratory, 2000.
- [11] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2002.
- [12] T. Sherwood, Erez Perelman, and Brad Calder. Block distribution analysis to find periodic behavior and simulation points in applications. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, September 2001.
- [13] Premkishore Shivakumar and Norman P. Jouppi. CACTI 3.0: An integrated cache timing, power and area model. Technical report, Compaq Western Research Laboratory, 2001.
- [14] Steve Swanson, Ken Michelson, Andrew Schwerin, and Mark Oskin. Dataflow: The road less complex. In *Proceedings of the 2003 Workshop on Complexity Effective Design*, 2003.
- [15] Manish Vachharajani, Neil Vachharajani, David A. Penry, Jason A. Blome, and David I. August. Microarchitectural exploration with liberty. In *Proceedings of the 35th International Symposium on Microarchitecture*, November 2002.
- [16] Steven J.E. Wilton and Norman P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical report, Digital Western Research Laboratory, 1994.
- [17] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *30th International Symposium on Computer Architecture*, June 2003.