AFRL-RY-WP-TR-2009-1042

MEMORY-BASED STRUCTURED APPLICATION SPECIFIC INTEGRATED CIRCUIT (ASIC) STUDY

Jay Brockman, Peter Kogge, Michael Niemier, and Larry Pileggi

University of Notre Dame

OCTOBER 2008
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

*//Signature//

KERRY L. HILL
Project Engineer
Advanced Sensor Components Branch
Aerospace Components & Subsystems
Technology Division

//Signature//

BRADLEY J. PAUL, Chief
Chief, Advanced Sensor Components Branch
Aerospace Components & Subsystems
Technology Division
Sensors Directorate

//Signature//

TODD A. KASTLE
Chief, Aerospace Components & Subsystems
Technology Division
Sensors Directorate

*Disseminated copies will show "//Signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YY)* October 2008	**2. REPORT TYPE** Final	**3. DATES COVERED *(From - To)*** 03 July 2007 – 30 September 2008

4. TITLE AND SUBTITLE MEMORY-BASED STRUCTURED APPLICATION SPECIFIC INTEGRATED CIRCUIT (ASIC) STUDY	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA8650-07-C-7734
	5c. PROGRAM ELEMENT NUMBER 69199F
6. AUTHOR(S) Jay Brockman, Peter Kogge, Michael Niemier (University of Notre Dame) Larry Pileggi (Carnegie Mellon University)	**5d. PROJECT NUMBER** ARPS
	5e. TASK NUMBER ND
	5f. WORK UNIT NUMBER ARPSNDBM
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Notre Dame, 300 Main Building, Notre Dame, IN 46556; Carnegie Mellon University, Hamerschlag Hall, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Sensors Directorate, Wright-Patterson Air Force Base, OH 45433-7320, Air Force Materiel Command, United States Air Force; Defense Advanced Research Projects Agency/ Information Processing Techniques Office (DARPA/IPTO), 3701 N. Fairfax Drive, Arlington, VA 22203-1714	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/RYDI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RY-WP-TR-2009-1042

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
PAO Case Number: DARPA 13536; Clearance Date: 13 May 2009. This report contains color.

14. ABSTRACT
According to the ITRS roadmap, 70 percent of the area of a typical ASIC today is memory and this increases to over 90 percent by the end of the roadmap. Yet despite the fact that ASICs are becoming more memory-intensive, commodity memory and ASIC design and manufacturing technologies are still on divergent paths. In this report, we examine methodologies for the design of Memory-Based Structured ASICs (SASICs) that include large amounts of dense, on-chip memory, as well as multiple processing cores, networks-on-chip, and I/O modules. We investigate regular fabrics as a means for designing logic circuitry compatible with the lithographic constraints imposed by the memory array for subwavelength geometries. We also develop midrange and high-level tools for exploring tradeoffs in power, area, and timing of complex, multicore SASICs. The report concludes with recommendations for follow-on work in the area of patterned ASICs.

15. SUBJECT TERMS
Structured ASICs, Smart Memory, Processor in Memory, Patterned ASICs

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Kerry L. Hill
Unclassified	Unclassified	Unclassified	SAR	62	**19b. TELEPHONE NUMBER** *(Include Area Code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Table of Contents

1. Summary ... 1
2. Introduction ... 2
  2.1 The Problem ... 2
  2.2 Approach ... 2
  2.3 Tasks ... 3
  2.4 Key Findings and Benefits ... 4
  2.5 Report Organization ... 6
3. Methods, Assumptions, and Procedures ... 7
  3.1 Evaluation of SASICs Using Regular Fabrics ... 7
  3.2 Integrated Power, Area, and Timing Modeling Framework for Multicore Designs ... 8
  3.3 High-Level SASIC Design Space and Performance Exploration ... 9
4. Results and Discussion ... 11
  4.1 An Evaluation of SASICs Using Regular Fabrics ... 11
    4.1.1 Overview of Regular Fabrics Technology and CAD Flow ... 11
    4.1.2 Example: ARM Processor Implementation ... 13
    4.1.3 PIM Lite Study ... 13
  4.2 Integrated Power, Area, and Timing Modeling Framework for Multicore Designs ... 20
    4.2.1 McPAT: Overview and Operation ... 20
    4.2.2 Validation ... 23
    4.2.3 Results ... 24
  4.3 High-Level SASIC Design Space and Performance Exploration ... 28
    4.3.1 Baseline System: Hierarchical CMP ... 28
    4.3.2 Use in a Structured ASIC Design Program ... 29
    4.3.3 Basic Process ... 31
    4.3.4 Sample Outputs ... 31
    4.3.5 Completed Functionality ... 32
    4.3.6 Functionality To Be Done ... 33
5. Conclusions ... 34
  5.1 Regular Fabrics ... 34
  5.2 Architectural Level Power, Area, and Timing Framework ... 34
  5.3 High-Level Design Space Exploration ... 35
6. Summary and Recommendations ... 36
  6.1 Summary ... 36
  6.2 RecommendationPatterned ASICs ... 36
7. References ... 41
8. Appendix: Integrated and Hierarchical Models for Power, Area and Timing ... 45
  8.1 Integrated Power, Area, and Timing Models ... 45
    8.1.1 Power Modeling ... 45
    8.1.2 Timing Modeling ... 46

8.1.3 Area Modeling 47
8.2 Hierarchical Modeling Framework 47
8.2.1 Multicore Architecture and Microarchitecture Level Modeling 48
8.2.2 Circuit Level Modeling 49
8.2.3 Technology-Level Modeling 50
8.3 Modeling Power-saving Techniques 51

List of Figures

Figure 1: Constraints along the axes of I/O links, processors, and memory interfaces define the region of interest 10
Figure 2: Comparison of lithography simulations for regular patterns (left) versus those for commercial standard cells (right) in the same 65nm CMOS technologies. Red layer is diffusion, green is polysilicon, blue is metal 1, and purple is metal 2. Simulation results courtesy of IBM and PDF Solutions. 12
Figure 3: Die shot of ARM926EJ implemented with logic bricks methodology 13
Figure 4: PIM Lite pipeline 14
Figure 5: PIM Lite custom VLSI implementation 15
Figure 6: Detail of PIM Lite layout, showing pitch matching between memory array and bit-sliced logic 15
Figure 7: PIM Lite standard cell implementation. Full chip (left) and cell detail (right).16
Figure 8: Double patterning lithography with gratings and trim customization........ 17
Figure 9: The unidirectional patterning is visible in these midrange views of M1 and poly. M1 fill metal omitted for clarity 18
Figure 10: Histogram of Fan-in for brick in PIM library........ 18
Figure 11: Example Layout of PIM Brick 19
Figure 12: Top level layout of the regularly patterned PIM 19
Figure 14: McPAT block diagram 21
Figure 15: McPAT Validation Results. For both validations there are miscellaneous components such as SOC logic that McPAT cannot model because the structures of those components are unknown. Therefore, 61.4W out of the total 63W are modeled for Niagara, and 77.9W out of the total 84W are modeled for Niagara 2. 23
Figure 17: Clustered many-core system architecture........ 25
Figure 18: Power consumption across configurations and technology nodes 27
Figure 20: Example component class 30
Figure 21: Sample outputs, showing the maximum number of processors feasible (Feasible Configs) for a given number of memory interfaces (MI) and link interfaces (LI). 32
Figure 22: Seedling study results........ 40
Figure A-23: Modeling methodology 47

List of Tables

Table 1: Comparison of advantages for pitch-matched, structured ASIC, and patterned ASIC designs........ 5
Table 2: Comparison of normalized delays for different PIM Lite implementations...... 20
Table 3: Validation Results of McPAT with regards to total power of Niagara and Niagara2 processors........ 24

List of Acronyms, Abbreviations, and Symbols

ASIC	application specific integrated circuit
attPSM	attenuated phase shift masks
CACTI	Cache Timing analysis tool
DRAM	dynamic random access memory
FPGA	field programmable gate array
G4FET	4-gate field effect transistor
ITRS	International Technology Roadmap for Semiconductors
MRAM	magnetic random access memory
OAI	off-axis annular illumination
PIM	processing in memory
SRAM	static random access memory

1 Summary

According to the International Technology Roadmap for Semiconductors, (ITRS) roadmap, 70 percent of the area of a typical application-specific integrated circuit (ASIC) today is memory and this increases to over 90 percent by the end of the roadmap. Yet despite the fact that ASICs are becoming more memory-intensive, commodity memory and ASIC design and manufacturing technologies are still on divergent paths. Commodity dynamic random access memory (DRAM) technology, for example, has 20X the density of embedded static random access memory (SRAM) and over 4X the density of embedded DRAM. Further, it is still unclear how new memory technologies, such as magnetic ram (MRAM) or 4-gate field effect transistors (G4FETs), can be effectively embedded into ASIC processes. This is particularly true for applications of critical DoD interest, where issues of rapid turnaround in heavily constrained power, weight, and complexity are system drivers. One might consider approaching the problem from the other direction, which is, embedding application-specific logic into a memory design and manufacturing methodology. Because ASICs are low-volume products, however, they are not compatible with either high-volume manufacturing flows or design methodologies preferred by a memory vendor.

In this study, we have explored the use of *regular fabrics* for lowering the barrier to using commodity memory technology for memory-intensive ASICs. A regular fabric is a system of circuits and design methodologies that best utilize the simple, regular patterns that *can be reliably printed for logic, memory, and analog circuits* with a single, compatible subwavelength lithography setup. In particular, results show that laying out logic circuitry on regular grids that are based on memory arrays spacing can substantially improve the manufacturability of memory-intensive ASIC designs. Regular fabrics are especially conducive to e-beam lithography, where electron beams transcribe patterns directly to the silicon wafer, without the use of photolithographic masks. Because of this, regular fabrics provide a path for cost-effective, low volume production by reducing mask cost. As an evaluation, we used the regular fabrics technology developed by Carnegie Mellon University and PDF Solutions to re-implement the PIM Lite processor-in-memory chip [49] and compared this with full-custom, conventional standard cell, and field programmable gate array (FPGA) implementations. Originally designed under DARPA's DIVA and Cascade projects, PIM Lite is a proof-of-concept chip that demonstrates both multithreading and wide-word instruction in a simple processor pipeline that is pitch-matched to the columns of an on-chip memory array. Results show that the performance of the regular fabrics implementation is at least comparable and likely better to that of the standard cell design, with superior manufacturability with regard to lithographic variation.

Given the base technology for implementing memory-intensive ASICs that regular fabrics provides, a further challenge is giving architects the tools to evaluate tradeoffs and explore the range of possible designs for complex, heterogeneous ASIC designs. Such designs may include many processor cores, interconnection networks, I/O components, and multiple varieties of memory, including 3D stacks or quilts. To address this challenge, we have developed both midrange and high-level tools for modeling power, area, and timing in multi-/many-core systems. The midrange tool, named McPAT (**M**ulti-**c**ore **P**ower, **A**rea and **T**iming), uses circuit level models coupled with statistics from an architectural simulation to obtain fairly accurate predictions: results show power estimates for Sun's Niagara multicore within 20% of actual. The high-level tools use simpler models, but permit rapid exploration and characterization of a broad design space.

2 Introduction

2.1The Problem

For air, space and ground-based systems, there is a clear need for high performance, lightweight, low-power, highly-reliable computing on data-intensive applications. A data-intensive application is one in which there is a very large volume of data, which is often accessed in irregular patterns. The growing number of graph-based, "connect-the-dots" problems fall into this class, as do many image and signal processing problems. In terms of computer systems architecture, data-intensive computing requires large amounts of memory, accessible at the highest possible bandwidth and lowest possible latency. Numerous research projects sponsored by DARPA and other agencies have explored a promising approach to data-intensive computing, variously called *Processing-in-Memory (PIM)*, *Intelligent RAM (IRAM)*, or *Smart Memory*, which seeks to embed processing logic into dense DRAM memory. The main problem addressed by this project is that there currently exists no cost-effective way of manufacturing such chips.

The main reason for this is the disparity between the design and manufacturing methodology of DRAM and logic-intensive application-specific integrated circuits (ASICs). Whereas the goal for DRAM manufacturing is to support a small set of high-volume products that can be fabricated with as few processing steps as possible, ASIC manufacturing requires the flexibility to support a relatively large number of designs, each running at a smaller volume, and with more processing steps than DRAMs nominally require. Furthermore, from the chip designer's viewpoint, in addition to the mask cost, each new product has a non-recurring engineering cost associated with the design effort. This design cost rises sharply with each successive technology generation because of the difficulty of producing reliable designs with good signal integrity and high manufacturing yield.

2.1 Approach

Our approach to addressing this problem is to investigate new methodologies for designing complex, multicore chips that inherently reconcile the design of the processing logic and on-chip memory. Specifically, we investigate the use of *regular fabrics* as a means for implementing such chips. A regular fabric is a gridded system of patterns for the mask layers of an integrated circuit (IC), designed to accommodate the distortions when electronic devices are printed using subwavelength lithography, which is one of the most significant design challenges for technologies at 45 nm and beyond. The patterns in a regular fabric are based on the memory arrays, and hence the logic circuitry will by designed to be compatible with memory. We will show in this report that the regular fabrics approach has virtually no adverse affect on either the area or performance of a chip implementation, as compared to a traditional standard cell implementation, while having superior manufacturability. Further, because regular fabrics are conducive to e-beam lithographic methods, they are especially suitable for low-volume production.

A second part of our approach is providing tools for system architects to explore the space of possible designs of memory-intensive, multicore, structured ASICs (SASICs) for area, speed, and power. These tools will allow designers to evaluate tradeoffs in the configuration of multicore chips, such as varying the number of processor cores and the type and amount of memory. We develop two frameworks for exploring these tradeoffs: a “midrange” tool that has embedded circuit models for high fidelity analyses, and a “high-level” tool with lower fidelity but that permits rapid evaluation of a broader range of designs.

2.2 Tasks

Our project was organized into four tasks, listed below:

- T1: Develop a representative synthesizable register-transfer level (RTL) PIM design for purposes of comparing implementations in FPGA, standard cell ASIC, and structured ASIC technologies. The reference design will consist of a hardware description language (HDL) model that is based on an existing PIM design, such as the PIM Lite chip developed at Notre Dame

- T2: Derive a set of structured ASIC “bricks” using the CMU/Fabbrix technology suited to the reference PIM design. These bricks are the low-level physical modules, with regular geometric patterns, that will be used in composing the implementation of the reference design from T1.

- T3: Synthesize and map the reference design to FPGA, standard cell ASIC, and structured ASIC technologies and compare layouts and performance. This will entail using the CMU/Fabbrix CAD flow to generate layouts for both a standard cell ASIC implementation and a structured ASIC using the bricks developed in T2, as well as using a commercial CAD tools licensed by Notre Dame for the FPGA implementation

- T4: Extend existing suite of PIM architectural tools to facilitate planning of heterogeneous tiled structured ASIC designs that include memory blocks, structured logic, on-chip networks and I/O. In particular, we will augment tools that were developed under the DARPA HPCS program to include technology-specific input into simulations.

2.3 Key Findings and Benefits

The key impacts of this work are:

- developing a design methodology that delivers performance comparable to ASIC logic that is compatible with the manufacturing process and cost structure of dense memory,

- reducing front-end design costs by facilitating high-level design space exploration

- reducing back-end design costs by exploiting design regularity and reuse.

During the execution of this study, our most significant finding was a shift in our thinking away from a conventional structured ASIC that has its roots in gate array technology, and towards a *patterned ASIC* that emphasizes regularity of printed geometries on the fabricated layers of an integrated circuit. The main reason for this shift was a recognition of the cost and performance drivers for ASICs at technologies below 32 nm, namely, the critical importance of designing for manufacturability with respect to the constraints of advanced lithographic techniques. Superficially, our initial pitch-matched, full-custom PIM Lite design from prior to this study, a gate array styled structured ASIC, and a regular patterned ASIC look similar, in that all three exhibit regular arrays of chip structures. The motivations behind these structures, as well as the savings in design and fabrication cost and time are very different, as summarized below:

- *pitch-matched design (original PIM Lite):* The bit-sliced datapath of the processor was custom designed to match the pitch or spacing of the columns of memory. The prime motivations of this approach were to maximize memory bandwidth while minimizing the wire length interconnecting the processing logic and memory. The regularity of the bit-sliced design also simplified debug and layout.

- *structured ASIC*: A structured ASIC consists of pre-patterned gates on the transistor layers of a chip such as diffusion and polysilicon, that are later personalized for the logic of a specific design through custom metal layers. Depending upon the implementation technology, this personalization typically would require no more than 4 metal masks and as little as single via mask. The underlying arrays of devices may be as simple as arrays of individual transistors, or as complex as a complete look-up table (LUT) as used in a field-programmable gate array (FPGA). There are two chief motivations behind this style of structured ASIC. The first is savings of both design cycle time and mask cost through the reuse of base silicon layers across multiple designs. For structured ASICs that use LUTs as primitives rather than individual transistors, as second motivation is complete compatibility with an FPGA design flow, such as Altera's hard copy technology.

- *patterned ASIC*: Unlike a gate-array based structured ASIC, a patterned ASIC does not preserve an identical array of transistors across multiple designs, that is, these layers, as well as the metallization, are personalized for each design. Instead, the emphasis is on a regular patterning grid that is common to all designs for all layers, developed to ensure manufacturability and to minimize lithographic variations, which is the most important implementation cost driver on nanometer-scale fabrication technologies. Even though polysilicon may be patterned differently for each design, there is still opportunity to reuse masks if e-beam technology is used to customize patterns from a common grid. Our analysis shows that this approach can be particularly cost effective for runs of up to several hundred wafers.

Table 1 summarizes the savings of each of these approaches, along with our high-level design tools, for the main steps of an IC design and manufacturing cycle:

Table 1: Comparison of advantages for pitch-matched, structured ASIC, and patterned ASIC designs

	Pitch-matched	structured ASIC	patterned ASIC
Design space exploration	high-level design space exploration tools reduce time and cost for each approach		
Register-transfer level (RTL) design and synthesis	requires user to specify bit-sliced design	same as conventional ASIC, may also be compatible with FPGA for LUT based technology	same as conventional ASIC
Place and route	requires either some manual layout or custom tools	largely compatible with conventional industry tools, with additional constraints	new tools required to discover regular pattern “bricks”, then compatible with industry standard tools
Mask development	no mask reuse	significant mask reuse	possible mask reuse when coupled with e-beam lithography
Initial Fabrication	complete process required, no advantage for low volume	uses pre-patterned wafers with only a few additional custom masks for personalization	uses predefined grid, less opportunity for reusing layers of pre-patterned wafers than structured ASIC
Respin	expensive	requires redesign of only personalized metal layers	may involve all layers, but use of e-beam may reduce costs for low volume
Volume fabrication	comparable to conventional ASIC	larger die size as a result of logic gate array increases cost	use of e-beam customization cost prohibitive runs beyond several hundred wafers

2.4 Report Organization

The remainder of the report is organized around each of the three major project studies: (1) evaluation of SASICs using regular fabrics, (2) development of the “midrange” framework for modeling the power, area, and timing of complex, multicore, memory-intensive chips, (3) development of the high-level modeling framework.

- Section 3, “Methods, Assumptions, and Procedures” provides an overview of the goals of each of these three studies

- Section 4, “Results and Discussion,” provides the details of the three studies

- Section 5, “Conclusions,” summarized conclusions of each of the studies

- Section 6, “Summary and Recommendations

- Summary

  The key findings of this study are:

- High level analysis tools can reduce design cycle time, cost, and risk by enabling design space exploration for area, speed, and power for multicore, memory-intensive systems. System-level tools that use simple heuristic and analytic models, as well as “midrange” tools that run in conjunction with a standard architectural simulator are both necessary. In this study, we have developed examples of each of these and have used them to characterize a range of designs.

- Using the PIM Lite chip as an example design, we have demonstrated that an implementation using the regular patterned “bricks” technology developed at Carnegie Mellon University and commercialized by Fabbrix/PDF Solution has performance comparable to a conventional ASIC, with a design flow little more complex than that for a conventional ASIC (addition of a single “brick discovery” step), that is much more compatible with advanced lithography.

It is difficult to estimate precise savings in design time and cost, as this is highly dependent on the specific application. Further study using a standard reference design more complex than PIM Lite could ascertain these differences in the future.

- RecommendationPatterned ASICs,” suggests a possible new program for follow-on work

- Section 8, “References” contains the bibliography for all citations

- Section 9, “Appendix: Integrated and Hierarchical Models for Power, Area and Timing,” provides additional details on the models used in the midrange framework.

3 3 Methods, Assumptions, and Procedures

3.1 Evaluation of SASICs Using Regular Fabrics

The fundamental premise of the regular fabrics approach is to develop circuits and design methodologies that best utilize the simple, regular patterns that *can be reliably printed for logic, memory, and analog circuits* with a single, compatible subwavelength lithography setup. This is synergistic with the commercial semiconductor interest in more *restricted design rules* for these technologies.

ASIC design methodologies generally call for the strict segmentation of logic and memory. Memory arrays are generally instantiated by an automated compiler. The arrays generated have limited configuration options and often adhere to special layout design rules that are incompatible with the rules for the logic blocks. Designs that incorporate highly granular logic and memory are not a possibility for today's ASIC designer because they require the handcrafting of such cells. Even if the ASIC engineer were able to meticulously assemble the array, lower density arrays would result because logic standard cells are not pattern-compatible with dense memory.

We developed methodologies that enable the use of dense tightly integrated memory and logic in future scaled technologies. We built on previous research on regular logic fabrics and bricks [24][20] to design pattern-compatible logic and memory that was developed by partner Larry Pileggi of Carnegie Mellon University (CMU). Furthermore, we demonstrate the efficacy of these methodologies for integrating specialized memory logic into application-specific architectures. Our primary focus is on the implementation of logic circuits based on critical patterns that are designed with *extreme regularity*; namely, simple *grating patterns* of fixed pitch and width polysilicon and metal structures. Specifically, we explore logic and memory circuits that are constructed by first forming dense gratings, followed by a second exposure for trimming the gratings [12][13]. Such approaches are being considered for 32nm and sub-32nm technologies using grating techniques such as maskless immersion interference lithography [12] for the critical dimension patterns, followed by conventional projection lithography for trimming.

Importantly, our methodology is not limited to embedded SRAM and DRAM, but can consider any embedded memory technology, including emerging technologies such as MRAM or G4FET. Building regular logic circuits often sacrifices a number of design freedoms to match the regular memory patterning, but with this regularity there is a significant architectural advantage that logic and memory will not have to be physically separated on the die for lithography or manufacturability reasons. This pattern compatibility between logic and memory can be an enabler for more granular integration of the two to provide application-specific computation capabilities and save significant power with more localized processing.

3.2 Integrated Power, Area, and Timing Modeling Framework for Multicore Designs

Power dissipation, and the resulting heat issues, have become possibly the most critical design constraint of modern and future processors. This concern only grows as the semiconductor industry continues to provide more transistors per chip in pace with Moore's Law. Industry has already shifted gears to deploy architectures with multiple cores, multiple threads [20][27] and large last-level caches so that the chip can be clocked at a lower frequency and burn less power, while still getting better overall performance.

Controlling power and temperature in future multi-core and many-core processors will require even more novel architectural approaches. However, our ability to propose, design, and evaluate new architectures for this purpose will ultimately be limited by the quality of tools used to measure the effects of these changes. To address this problem, we have developed a new power simulation framework called McPAT (**M**ulti**c**ore **P**ower, **A**rea, and **T**iming).

It has always been true in the architecture community that tools (or lack of) both limit and drive research directions. Wattch [7], first presented in 2000, has been such a tool, enabling a tremendous surge in power-related architecture research. However, several factors drive the need for new tools to address changes in architecture and technology. This includes the need to accurately model multicore and manycore architectures, the need to accurately model all sources of power dissipation, and the need to accurately scale circuit models into deep submicron technologies. The McPAT framework attempts to address these challenges.

McPAT advances the state of the art in several directions, compared to the Wattch tool, the current standard for power research. First, McPAT models more than just dynamic power, which is critical under deep-submicron technologies because the static power is skyrocketing and has become comparable to dynamic power [24][43]. McPAT models all three types of power dissipation - dynamic power, static power, and short-circuit power - to give a complete view of the power envelope of multicore processors.

Second, McPAT provides a complete, integrated solution for multi-core power. Some researchers have combined Wattch's core power model with a router power model (namely *Orion* [23][51]), but even that is an incomplete solution. Today's multi-cores are a complex system of cores, caches, interconnect, memory controllers, multiple-domain clocking, etc. McPAT models the power of most of the important components of multicore processors, including all of the components listed above.

Third, McPAT handles technologies that can no longer be modeled by the linear scaling assumptions used by Wattch. The simple linear scaling principles are no longer valid because device scaling has become highly non-linear in the deep-submicron era. The HotLeakage tool [54], sometimes used in conjunction with Wattch, presents a more complex model of static power that uses Ioff (subthreshold leakage current) values that are based on device data obtained through BSIM3 parameter extractions. McPAT provides an integrated solution that models all the power sources. Our power-modeling tool makes use of technology projections from ITRS and MASTAR [42] for dynamic, static and short-circuit power; as a result, this tool will naturally evolve with ITRS even beyond the end of the current road map. McPAT also supports modeling of emerging 3D circuit technology.

Fourth, because McPAT is actually an integrated power, area, and timing framework, it makes it difficult for the user to specify an architecture that cannot be built. McPAT actually specifies the low-level design parameters of regular components (interconnects, caches, other array-based structures, ...) based on high-level constraints (timing and/or area) given by the user, ensuring the user is always modeling a reasonable design. This approach enables the user, if they choose, to ignore many of the low-level details of the circuits being modeled.

3.3 High-Level SASIC Design Space and Performance Exploration

Now that frequency and ILP have flattened, *multi-core* chips, where multiple complete CPUs are placed on the same die with significant parts of their memory hierarchy, are an obvious approach to leveraging advancing silicon technology for more performance. These chips have become the heart of most modern *High Performance superComputing (HPC)* systems, and consequently an understanding of what their technological limits are, will give excellent insight into what future generations of HPCs that use evolutionary architectures will look like.

This section gives an overview of our approach to the problem of rapidly exploring the design space of such high performance multicore chips, at pre-design stages, before more long-term efforts are expended. We describe a tool which designers can use at these early stages which will take application characteristics, combined with models of various chip subcomponents, and produce both visual and numeric results to help guide further design. Whereas the framework described in Section 4.2 provides high fidelity estimates, it also requires detailed statistics from a cycle-accurate architectural simulator. The high-level tool described in this section trades this level of accuracy for the ability to explore a wide design space at low computational cost.

The part of the extrapolation we focus on here is that associated with the microprocessor chips. Matching growth in the amount of memory and inter-node bandwidth is assumed, but not considered further. What we do consider are the key technological factors that govern the properties of individual circuit blocks on future multi-core chips, all of which scale independently. We then consider how the need to maintain certain application-specific ratios might influence how much of what resources (core, cache, off-chip interfaces) would need to be placed on-chip for a balanced system. This is complicated by the fact that many key external memory performance parameters will not scale anywhere near that of the processor chip.

The general approach developed here to identify potential future configurations of such chips assumes sweeping through a multi-dimensional design space, and restricting the points selected for further analysis by several kinds of constraints.

Figure 1 illustrates the set of constraints defined by three dimensions related to the physical number of key subsystems placed on the die (processor cores, memory interfaces, and link interfaces).

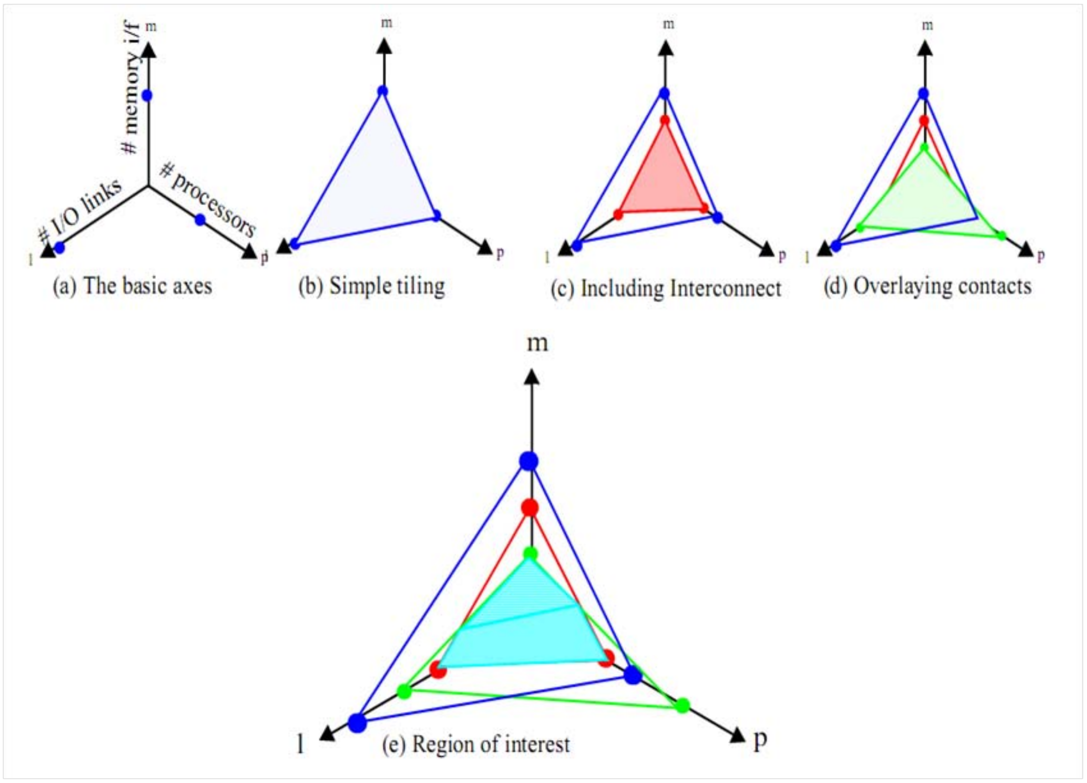


Figure 1: Constraints along the axes of I/O links, processors, and memory interfaces define the region of interest

Both hard and soft design constraints can be considered. Hard constraints are ones for which there is some physical limit (such as area) to some parameter (which of course may change with time). Die area and the number of off-chip contacts are two such. Softer constraints are ones that deal with the ``suitability'' of the configuration to meet the application needs. This includes relative bandwidth and latency (to memory), and overall chip power dissipation. Here there may be fuzzy areas where a chip may be feasible but has less than desirable characteristics, such as being bandwidth starved or exhibit overly high power dissipation.

Finally, after exploring the configuration space, and filtering down to feasible points, a second round of identifying optimal configurations and overall trends is performed. We may decide to identify different measures for optimality such as peak performance per chips, performance per watt, performance per unit die area, etc.

4 Results and Discussion

In this section, we take a detailed look at each of the three studies performed under this project: (1) evaluation of the CMU-developed regular fabrics for the implementation of memory-based structured ASICs; (2) development of an architectural level integrated power, area, and timing estimation framework for memory intensive, multicore systems; (3) development of high-level design space exploration tool for multicore systems, that support fast evaluation of a broad range of design, albeit with lower fidelity than the architectural level models.

4.1 An Evaluation of SASICs Using Regular Fabrics

4.1.1 Overview of Regular Fabrics Technology and CAD Flow

In the nanoscale era of CMOS scaling, the engineering abstractions that enabled the VLSI revolution are breaking down. The what-you-draw-is-what-you-get approach to layout and circuit design ceases to work as lithographic, etch and stress variations effect circuit parameters. Deep-sub-wavelength lithography has reduced the lithographic process window and destroyed the ability to reproduce arbitrary layout patterns, thereby making systematic variability a potential showstopper for continued scaling. Optical proximity correction (OPC) and resolution enhancement techniques (RET) have stepped in to preserve printability, but these techniques have increased mask data preparation costs and constrained the patterns that can be reproduced. Regular patterning design methodologies have garnered attention for their simplified application of OPC and their amenability to RETs such as attenuated phase shift masks (attPSM) and off-axis annular illumination (OAI). Regardless of which lithography and RET techniques are used, they all work better with extremely uniform fixed-pitch patterns, which has been the premise of the regular fabrics research.

The CMU regular fabrics [24][20][21] have not only been used to design manufacturable layouts under the constraints of subwavelength lithography, but have also optimized the logic-level design of cell libraries. This is important because the implementation of conventional standard cells in regular fabrics has drawbacks. The fixed-pitch unidirectional shapes make the most area efficient implementations more difficult. To overcome this, we take a different approached called *logic bricks*. A logic brick is like a standard cell but there are several significant differences: micro-regularity, macro-regularity and library composition.

Micro-regularity refers to the strict adherence to restricted design rules. Different fabrics have different restriction such as unidirectional fixed pitch patterning. Determining the exact fabric is an exercise in trading-off the capabilities of the lithography system with the needs of the circuit design. In the side-by-side lithography simulation of a 45nm micro-regular logic brick and a conventional standard cell, we observe significant variation in the critical dimensions (where red and green regions overlap) of the regular standard cell as compared to the regular fabric. In particular, line-end pull back, and diffusion corner rounding is highly visible in the less regular standard cell (Figure 2).

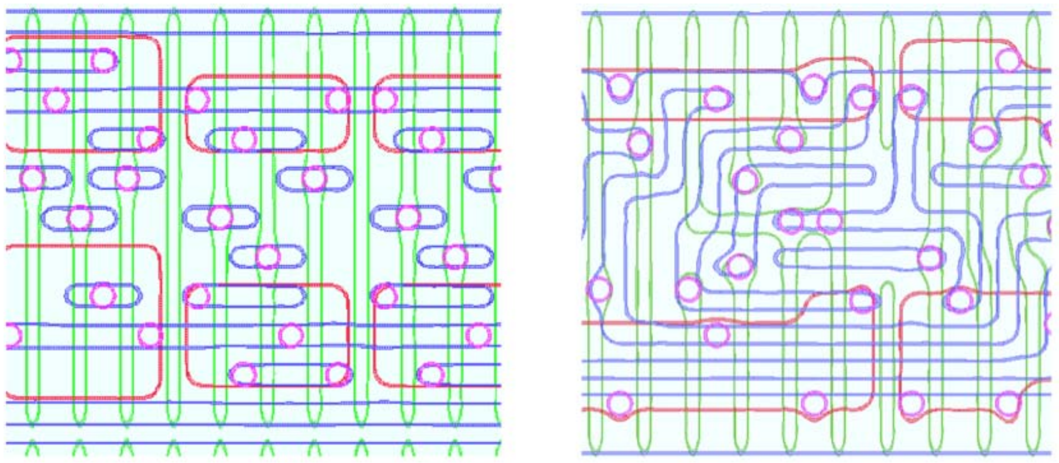


Figure 2: Comparison of lithography simulations for regular patterns (left) versus those for commercial standard cells (right) in the same 65nm CMOS technologies. Red layer is diffusion, green is polysilicon, blue is metal 1, and purple is metal 2. Simulation results courtesy of IBM and PDF Solutions.

Macro-regularity is the second characteristic of logic bricks. While micro-regularity refers to the individual shapes, macro-regularity refers to repeating groups of shapes. A canonical example of macro-regularity is the SRAM cell. Each SRAM array consists of a repeated tiling of SRAM cells. As a result, the boundaries of each cell is known and can be characterized for printability in advance. This enables much tighter control over systematic variability and is why macro-regularity is a desirable trait for a brick library.

The third unique attribute of a logic bricks is the composition of the library. Conventional standard cell libraries have well over a thousand different cells that can be used to implement any type of design. The cells mostly have the simple functionality of a basic logic gate. In contrast, brick libraries have on the order of twenty different unique bricks. The bricks implement a more complex logic function than their standard cell counterparts. This compacts the layout and recovers area penalties that may otherwise arise due to the micro-regularity constraints. Fewer logic cells also enhance macro-regularity by reducing the number of unique patterns to which each cell can abut. Because we are using fewer types of library elements, and those types are less generic, it is necessary to choose the logic functions of each brick based upon the application-domain of the RTL in order to guarantee that the bricks can efficiently implement the design. We have two methods to select the best logic function. The “packer” method works by merging logic gates that share physical proximity. In this method, the RTL is preliminarily synthesized to a set of logic primitives and then placed and routed. The design is then analyzed to identify repeating sets of adjacent logic primitives. These primitives are then merged into a larger single logic brick. The RTL is then re-synthesized to these bricks and then undergoes final placement and routing. The second method uses heuristics to analyze the netlist and selects a set of logic functions that are amenable to implementation in a regular fabric and efficiently cover the netlist with a minimum number of total library elements. This method is called “brick discovery” and was implemented by Fabbrix and PDF solutions.

4.1.2 Example: ARM Processor Implementation

The combinations of these techniques leads to ICs that are very manufacturable at scaled technology nodes, have reduced design times, and that don’t sacrifice performance as compared to their irregularly design counterparts. In 2006, the CMU team completed the backend implementation of an ARM926EJ microprocessor in 65nm technology using a logic bricks methodology (Figure 3). Simultaneously, industrial collaborators designed this same part using a conventional standard cell methodology. The bricks chip used only twenty logic bricks on a fabric of fixed pitch poly and M1. Despite this supposed limitation, the bricks chip fit into the same footprint as the standard cell approach, had equivalent performance, functional first silicon, and required only five person-weeks of design!

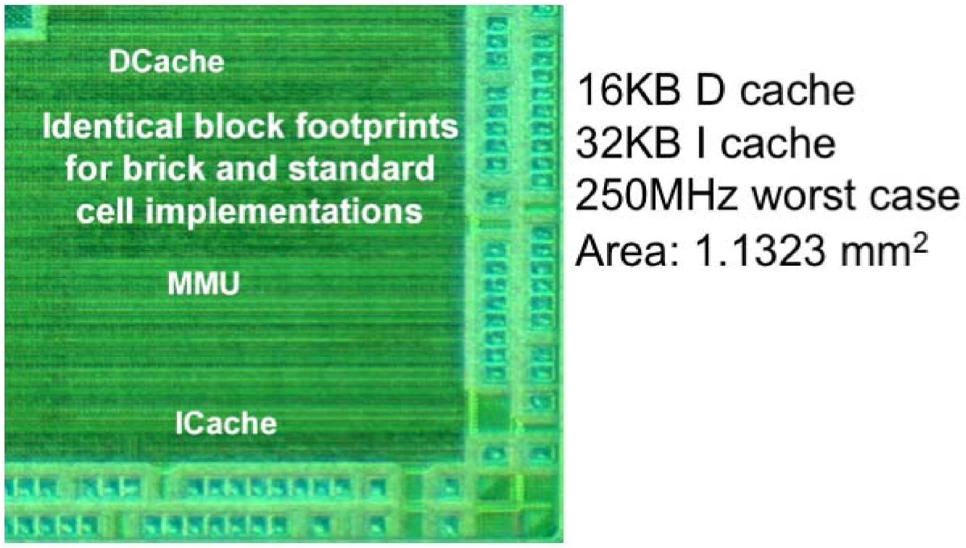


Figure 3: Die shot of ARM926EJ implemented with logic bricks methodology

4.1.3 PIM Lite Study

In order to evaluate the effectiveness of a regular fabrics implementation relative to other implementation styles, in this study we compared four implementations of the PIM Lite [49] processor-in-memory chip.

1. Original full-custom implementation
2. Conventional standard cell implementation
3. Regular fabrics implementation
4. FPGA implementation

PIM Lite is a proof-of-concept chip that demonstrates a PIM component suitable for a scalable distributed shared-memory system, particularly one nestled close to the kinds of dense memory blocks expected in many future ASICs. Figure 4 illustrates the PIM Lite pipeline. The instruction set is a 16-bit hybrid multithreaded/scalar/SIMD that makes use of thread frames in memory instead of processor-owned register files. Communication between PIM Lite nodes is through active messages called "parcels," that may be used for migrating threads to the site of a data object, as well as performing remote memory operations.

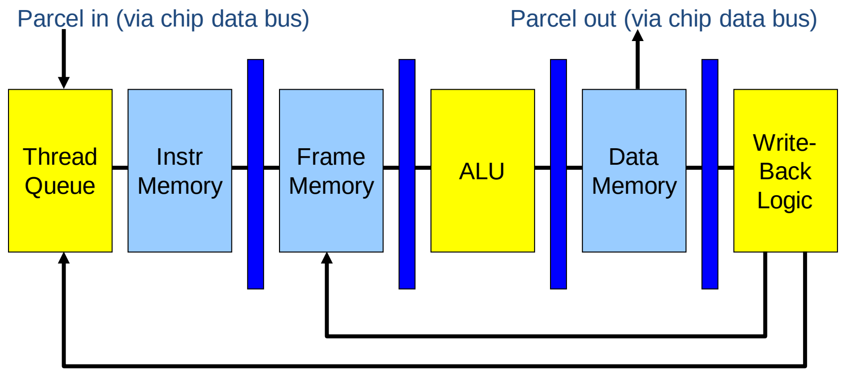


Figure 4: PIM Lite pipeline

4.1.3.1 Full-Custom Implementation

Figure 5 illustrates the original full-custom PIM Lite chip. Figure 6 shows details of the connection between the memory and the bit-sliced logic. The layout is highly regular, with datapath bit-slices pitch-matched and tightly coupled to columns of on-chip memory. A single-node version of the chip was fabricated on the TSMC 0.18 micron, 1.8 volt process through MOSIS, with over 800K transistors on a 13.4 sq. mm. die. The actual clock frequency of the fabricated chip was 40 MHz, but would have been approximately 100 MHz had an internal signal been properly buffered that caused a slow critical path.

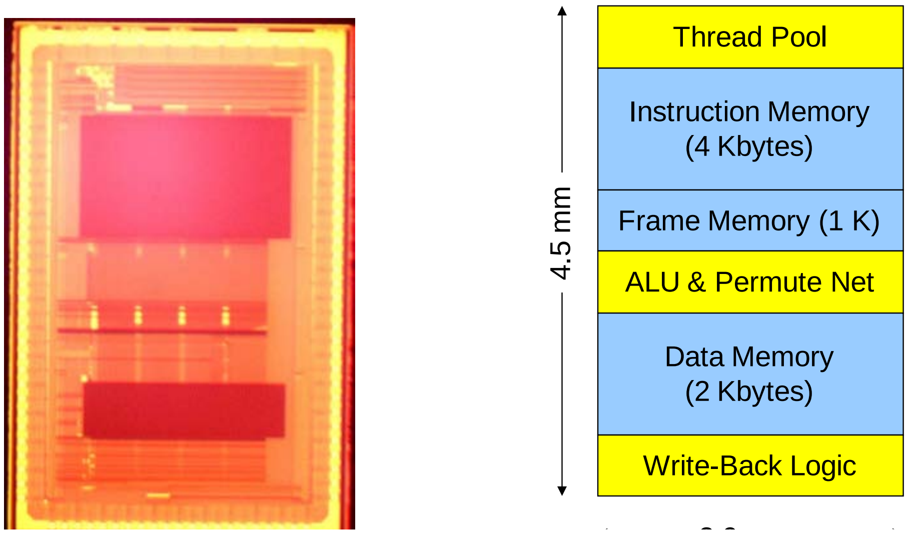


Figure 5: PIM Lite custom VLSI implementation

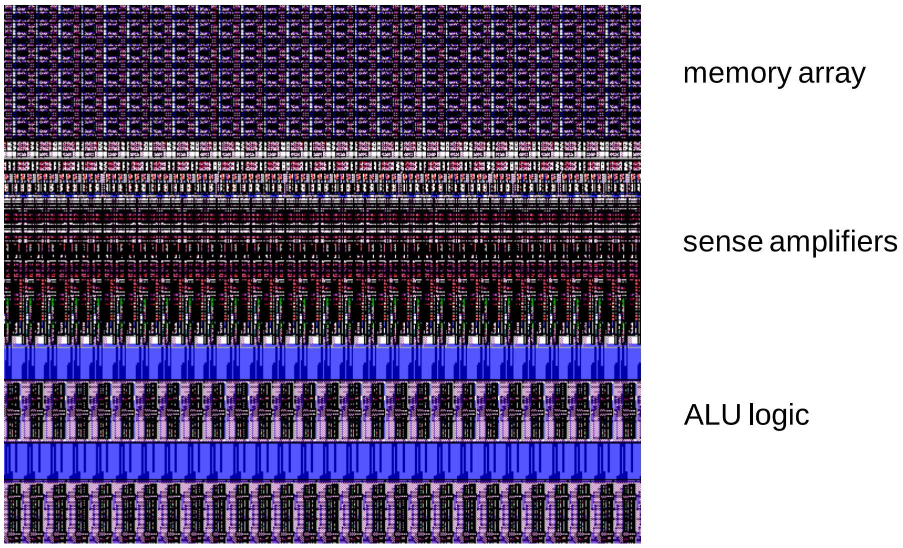


Figure 6: Detail of PIM Lite layout, showing pitch matching between memory array and bit-sliced logic

4.1.3.2 Standard Cell Implementation

For the standard cell implementation of PIM Lite, we used the AMI 0.5 micron, SCMOS Cadence cell library designed by Oklahoma State University, based on the technology files developed by North Carolina State University. Modification of the initial behavioral PIM Lite VHDL model for synthesis required approximately two months. Synthesis was performed with Synopsys Design Analyzer, and placement and routing was performed using Cadence Encounter. Default settings were used for both synthesis and placement and routing, with no additional optimization for area, speed, and power. The design synthesized with a total of 4829 cell instances, using 17 unique cells from the library. The critical path delay was 18.9 ns, for a clock rate of 53 MHz.

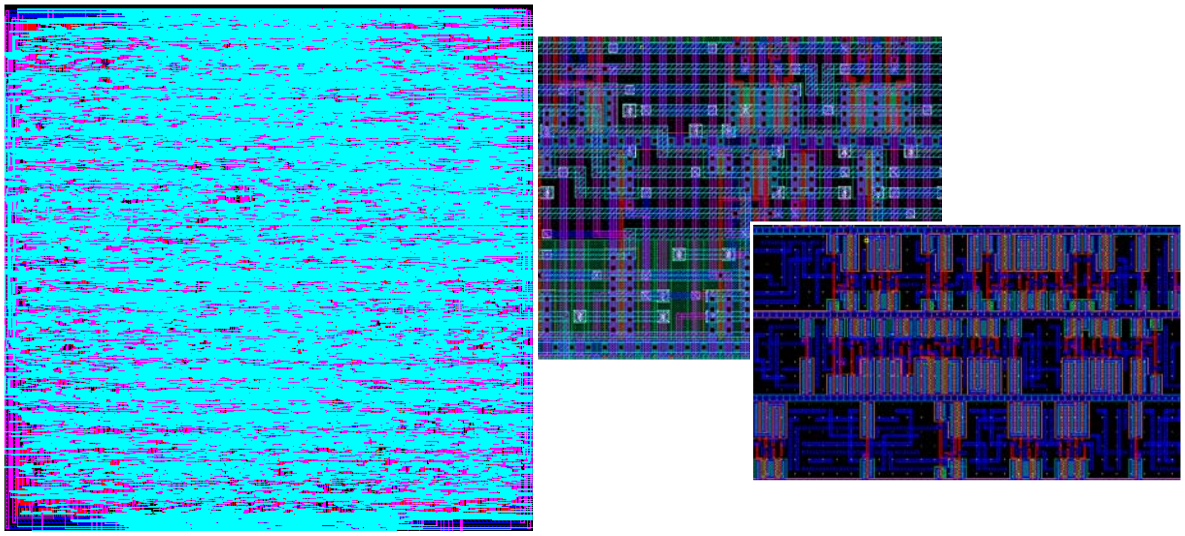


Figure 7: PIM Lite standard cell implementation. Full chip (left) and cell detail (right).

4.1.3.3 FPGA Implementation

To obtain a representative FPGA implementation of PIM Lite, we synthesized it for the Altera Stratix II. We used the same synthesizable VHDL code as was used in the standard cell experiment, with minor modifications to synthesize with the Altera tools. This particular FPGA is fabricated in a 90 nm, 1.2 V technology, and has 142,530 lookup table cells or ALUTs. The PIM Lite design required only 1201 ALUTs, or less than 1% of the total available logic. The design was synthesized with default settings and no additional optimization using the Altera Quartus II tool suite with a 35.6 ns worst-case propagation delay, or an operating frequency of 28 MHz.

4.1.3.4 Regular Fabric Implementation

The backend implementation of PIMLite was an extension of the methodology that was used very successfully on the ARM926EJ and several other pojects. The key steps in the design process were specifying the fabric, designing the brick library and the synthesis and physical implementation of the design.

A fabric consisting of purely unidirectional fixed pitch metal and polysilicon (poly) was selected for this design for its ability to use affordable low- to mid-volume lithography techniques at highly scaled technology nodes. 65nm design kits were used for this implementation but the fabric style and techniques are applicable at even more deeply scaled nodes. A potential fabrication scenario is shown in Figure 8. Highly regular grating patterning is printed onto the photoresist. The grating patterning is then trimmed with a lower tolerance optical lithography or e-beam. Since only the trimming is done with e-beam, the cycle time is reduced [12][13].

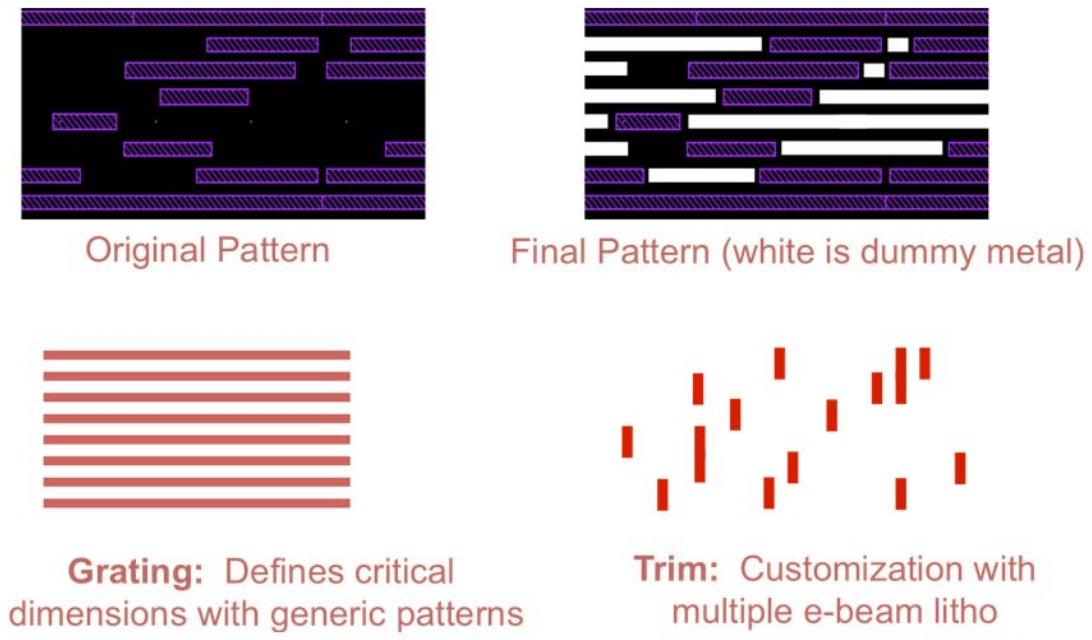

Original Pattern
Final Pattern (white is dummy metal)
Grating: Defines critical dimensions with generic patterns
Trim: Customization with multiple e-beam litho

Figure 8: Double patterning lithography with gratings and trim customization

A midrange view of the M1 and poly layers is visible in Figure 9. This clearly shows the high degree of regularity.

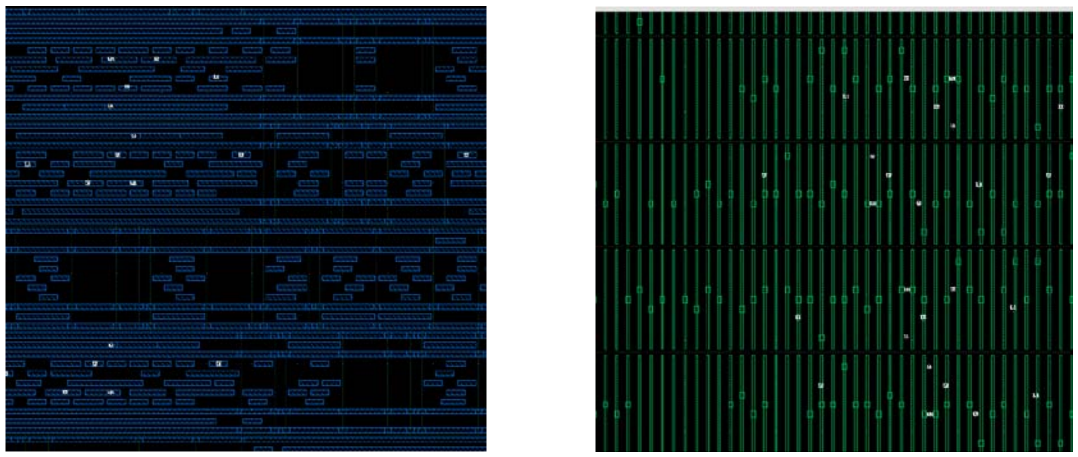


Figure 9: The unidirectional patterning is visible in these midrange views of M1 and poly. M1 fill metal omitted for clarity

By enforcing micro- and marco-regularity constraints, it is possible to specify more aggressive design rules. A standard cell needs to be able to print and etch in any legal pattern environment while a brick has a much more restricted set of pattern environments. It’s well known that by controlling the adjacent patterns, design rules can be aggressively tweaked. For example, the macro-regular SRAM cells have denser design rules than arbitrary logic cells. Additionally, highly regular structures like SRAM or FPGA’s often transition to the next process node before their less regular counter parts. For our 65nm PIM implementation, the design rules deviated just slightly from official fab supported rules. The rules that needed to be “pushed” involved line-end contact and via enclosure rules. The micro-regularity of the cells necessitated this adjustment so that each metal track could be used efficiently. The changes have been verified with lithography simulations but not with working silicon.

The Brick Discovery software package was used to analyze the PIMLite netlist to select the logic functions that would compose the brick library. Many of the selected functions are large. Twenty-five percent have five or more inputs. This result is shown in Figure 10.

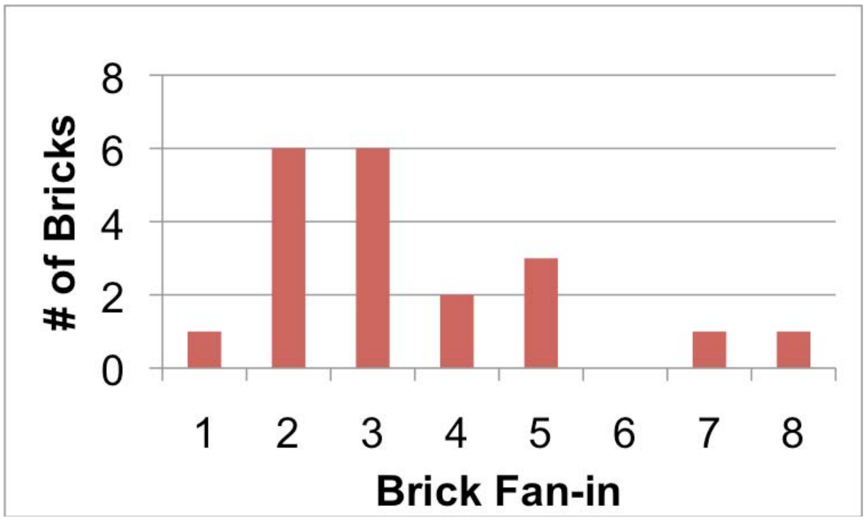
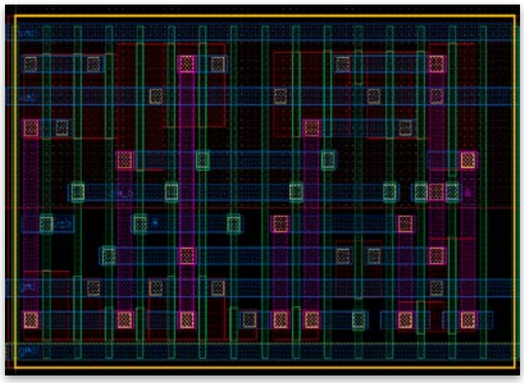
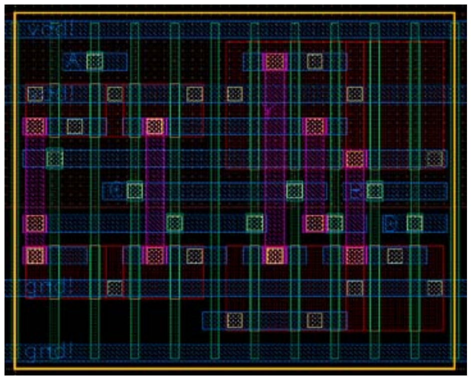


Figure 10: Histogram of Fan-in for brick in PIM library

Two of the highly regular bricks are shown in Figure 11. The brick on the left is a D flip- flop and the brick on the right implements a four input function.



D-Flip-Flop



F=A¬ B¬ C+¬ D

Figure 11: Example Layout of PIM Brick

With the brick library designed, the PIM design could be implemented using the Cadence Encounter software. A top level layout is shown in Figure 12. The total area is 350um x 300 um. There are 18,904 placed cells. The simulated cycle time is 1GHz.

The memories blackboxed as they were not designed as part of this exercise.

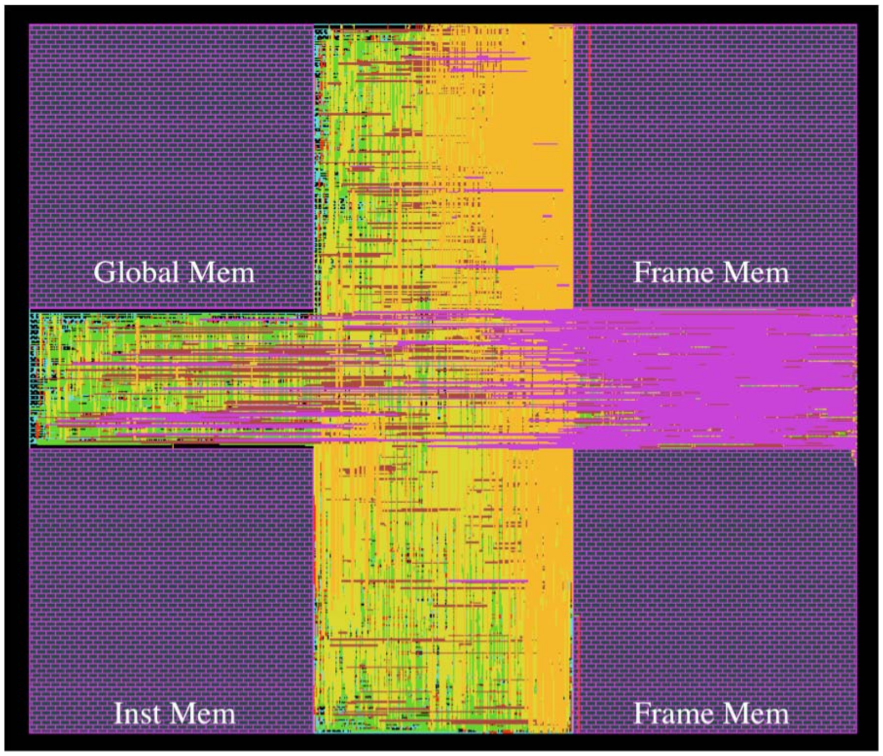


Figure 12: Top level layout of the regularly patterned PIM

4.1.3.5 Comparison

Table 2 provides a table of comparative delays and areas for the four PIM Lite implementations. We used linear scaling of delay to account for differences in geometry sizes, and also used linear scaling of delay with respect to voltage for the standard cell design, which was implemented in an 0.5 micron technology. We did not scale delay for voltage in the other designs, as the dependency is much less than linear for these geometries. Because of technology differences, as well as differences in the sophistication of the cell library designs (note that our standard cell library was a public-domain university library), it is difficult to make direct comparisons. Still, the regular fabric implementation compares very favorably with the other design styles. The normalized delay for the regular fabric was approximately 4 times as fast as that for the standard cell design, and more than 20 times as fast as the FPGA design.

The area for the regular fabric implementation, normalized for technology differences, was approximately 25 percent larger than our standard cell implementation (0.10 for the fabric as compared to 0.08 for the standard cell). It is difficult to draw a direct conclusion from this, however, because of the difference in sophistication of the libraries, especially with regard to issues such as buffering of signals. Both the standard cell and fabrics implementations had much lower normalized area than the custom design, where pitch-matching was a dominant constraint in the layout.

Table 2: Comparison of normalized delays for different PIM Lite implementations.

	technology (nm)	Vdd (V)	f (MHz)	delay (ns)	normalized delay	area (um^2)	normalized area
custom	180	1.8	110	9.09	1.00	3,497,400	1.00
standard cell	500	3.3	53	18.87	1.37	2,042,469	0.08
regular fabric	65	0.7	1000	1.00	0.30	47,400	0.10
fpga	90	1.2	28	35.71	7.86		

4.2 Integrated Power, Area, and Timing Modeling Framework for Multicore Designs

4.2.1 McPAT: Overview and Operation

The McPAT is an integrated power, area, and timing modeling framework designed to work with a variety of processor performance simulators (and thermal simulators, etc.) over a large range of technology generations. It is designed for both the user who wants to specify very low-level configuration details and the user who wants to ignore those details but still get them right.

Figure 13 is a block diagram of the McPAT framework. McPAT itself is a standalone program written in C++, that requires an external architectural simulator to generate statistics of switching activity within the processor. The actual execution time of McPAT is very fast—at most on the order of a few sections—although the execution time of the external architectural simulation may take from minutes to hours, depending upon the complexity of the benchmark programs being simulated. We note that in order to estimate peak power dissipation in a processor, it is not necessary to run an external architectural simulation to obtain an activity factor, but instead can assume that all components of the microarchitecture are active.

Rather than being hardwired to a particular simulator, McPAT uses a standard XML-based interface with any performance simulator. To date, we have successfully run McPAT in conjunction with the M5 simulator [32]. This interface allows the specification of both the static microarchitecure configuration parameters and the passing of dynamic activity statistics generated by the performance simulator. McPAT can also send runtime power dissipation back to the performance simulator through the XML-based interface, so that the performance simulator can react to power or (even temperature) data. This approach makes McPAT very flexible and easily ported to other performance simulators. Since McPAT provides complete hierarchical models from the architecture to technology level, the XML interface also contains circuit implementation style and technology parameters that are specific to a particular target processor. Examples are array types, crossbar types, and the CMOS technology generation with associated voltage and device types.

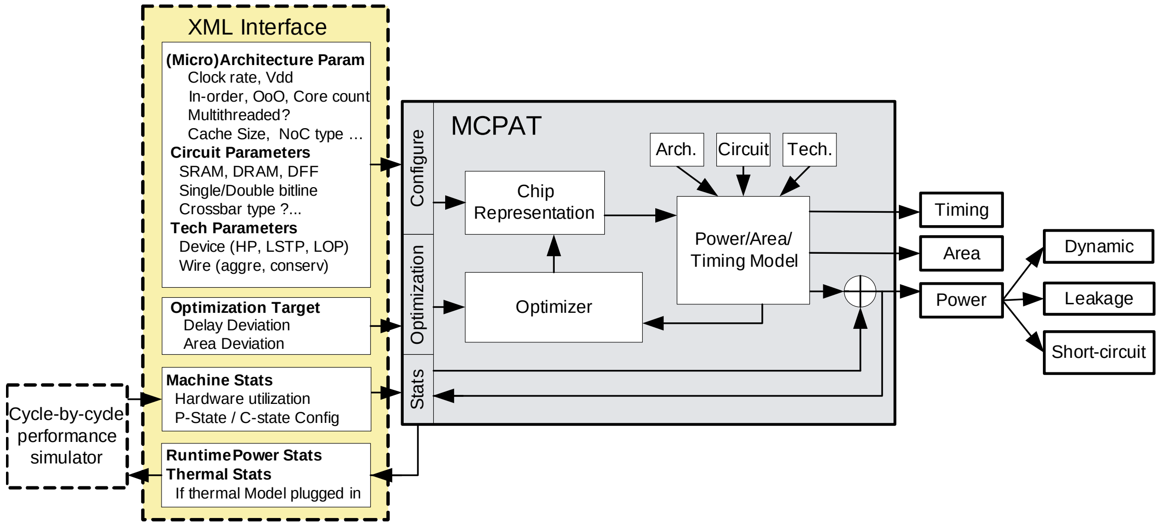


Figure 13: McPAT block diagram

The key components of McPAT are (1) the hierarchical power, area and timing models, (2) the optimizer for determining circuit level implementations, and (3) the internal chip representation that drives the analysis of power, area and timing. Most of the parameters in the internal chip representation, such as cache capacity and core issue width, are directly set by the input parameters.

McPAT's hierarchical structure allows it to model structures at a very low level, and yet still allow an architect to focus on only the high-level details. The optimizer determines missing parameters in the internal chip representation. McPAT's optimizer focuses on two major regular structures: interconnects and arrays. For example, the user can specify the bandwidth, frequency and bisection of the interconnect or the capacity, associativity and number of cycles to access a cache bank, while letting the tool determine the implementation details such as the choice of metal planes and effective signal wiring pitch for the interconnect or the length of wordlines and bitlines of the cache bank. These optimizations lessen the burden on the architect to figure out every detail, and significantly lowers the learning curve to use the tool. Users always have the flexibility to turn off these features and set the circuit level implementation parameters themselves.

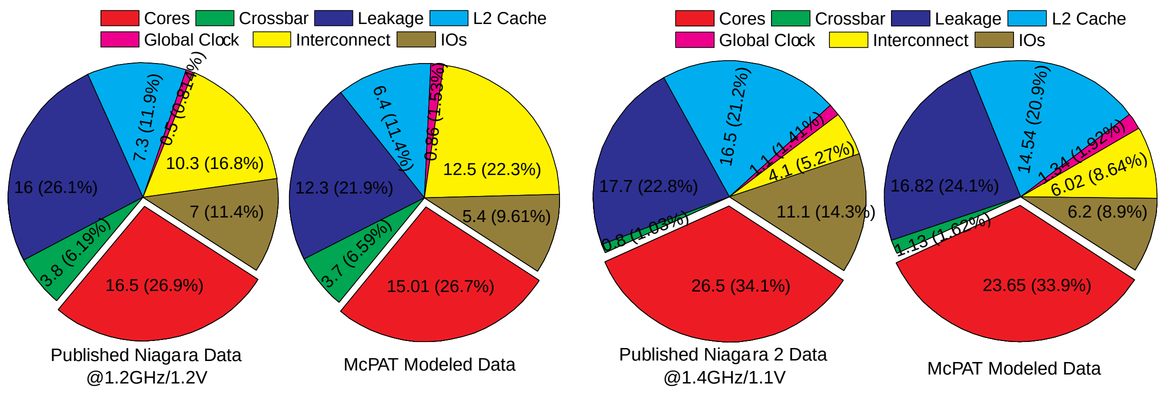
The main usage of McPAT is for designers to be able to obtain accurate power estimates, given constraints on chip are and clock rate.   Consider a multicore system to be analyzed for its runtime power dissipation. The architect specifies the target clock frequency, the target die area and other architectural/circuit/technology parameters. First, McPAT optimizes the interconnects to satisfy the timing constraints within the allowed deviation by trying out possible metal planes and signal wiring pitch. Second, McPAT identifies the subset of possible implementations with area that is within the allowed deviation. Finally, McPAT selects the interconnect implementation from the subset based on the optimization target which can be power, area or timing. Since an integrated circuit typically has many wiring layers on which signals may be routed, where each has different electrical characteristics, determining optimal routes for signals is a very complex problem. McPAT uses a heuristic approach for selecting the wiring plane and signal wiring pitch which is described in the Appendix. A similar optimization on array structures such as RAM-, CAM- and DFF-based arrays, is done using the built in CACTI (**Cac**he **Ti**ming) module with our enhancements. Thus, McPAT carries out multiple iterations and finally picks the configuration that is optimized for power, area, or timing and satisfies timing and area design constraints within the allowed deviation. The power, area and timing models together with the final chip representation generated by optimizer are used to compute area, timing, and peak power. The peak power of individual units and the machine utilization statistics (activity factor) are used to calculate the final runtime power dissipation. The statistics and activity factor may be determined by an architectural simulation tool such as SimpleScalar or M5 [32].

Another distinguishing feature of McPAT is its ability to model low-power states, such as the P- and C-state [18] power management of modern processors, which enable the processor to vary the clock rate and voltage to conserve power. After calling McPAT to finish initialization, a performance simulator can pass stats and invoke McPAT anytime during the simulation, and McPAT will calculate the corresponding power dissipation for the particular period and send it back to the performance simulator when required. This allows the simulator to react to simulated power or thermal sensors (assuming a temperature model attached to the backend), including entering these low-power states, changing voltage and frequency settings, or invoking one of multiple power-gating modes on various logic blocks. This allows the architect to use the framework to model the full range of power management alternatives.

4.2.2 Validation

The main focus of McPAT is accurate power modeling at the architectural level, given area and timing design constraints. Both *relative* accuracy and *absolute magnitude* accuracy are important for architectural level power modeling. Relative accuracy means that changes in modeled power as a result of architecture modifications should reflect on a relative scale the changes one would see in a real design. While relative accuracy is critical, absolute accuracy is also important if one want to compare results against thermal design power (TDP) limits, or to put power savings in the core in the context of the whole processor or whole system. In this section, we attempt to validate both absolute accuracy (by comparing against real processor data) and relative accuracy (by comparing against multiple generations of a similar architecture).

The biggest difficulty in validating a tool like this is getting reliable data on real processors. We compare the output of McPAT against published data of the 90nm Niagara processor [30] runing at 1.2GHz with a 1.2V power supply and the 65nm Niagara2 processor [34] running at 1.4GHz with a 1.1V power supply. Although the two processors have similar design styles they differ greatly in micro-architecture, which in turn makes their mapping to our hierarchical models very different. This comparison also tests our ability to cross technology generations and retain accuracy. The configurations for the validations are based on published Niagara and Niagara 2 data in[45][46][47][30][34], including target clock rate, die area, and architectural parameters. We assume the working temperature is 85 °C in both cases based on [30]. The target clock rate is used as the lower bound for the timing constraint. Therefore, the generated results must match the clock rate of the actual target processor.



(a) Validation against Niagara processor (b) Validation against Niagara 2 processor

Figure 14: McPAT Validation Results. For both validations there are miscellaneous components such as SOC logic that McPAT cannot model because the structures of those components are unknown. Therefore, 61.4W out of the total 63W are modeled for Niagara, and 77.9W out of the total 84W are modeled for Niagara 2.

Detailed validation exercises can be found in the McPAT tech report [5]; here we simply summarize the validation results. Figure 14 shows these results. Unfortunately, the best power numbers we have for these processors are for peak power rather than average power. Fortunately, McPAT can also output peak power numbers, based on maximum activity factors and maximum switching activity. Results show that modeled power numbers track the published numbers well. For the Niagara processor, the absolute power numbers for cores and crossbars, generated by McPAT, match very well with the published data. Over all seven components, the average difference in absolute power between the modeled power numbers and published Niagara data is just 1.47 W, for an average error per component of 23%. That number seems high, but the two big contributors are clock power (72% error, but a small contributor to total power) and leakage power (23% error). Both are significantly more accurate for the Niagara2. For the Niagara2, the average error is 1.87 W (26%), but by far the biggest contributor (4.9 W error) is the IO power. This arises because Niagara2 is a complicated SOC with different types of I/Os including memory, PCI-e and 10Gb Ethernet [34]; while McPAT only models on-chip memory controllers/channels as IOs. If we were measuring average power instead of peak power, this difference would shrink given the expected activity factors of those components.

Table 3: Validation Results of McPAT with regards to total power of Niagara and Niagara2 processors.

Processor	Published total Power Modeled	total power McPAT	error
Niagara	63W	56.17W	-10.84%
Niagara2	84W	69.70W	-17.02%

Table 3 shows the comparison of total power for validations against Niagara and Niagara2. Differences between the total peak power generated by McPAT and reported data are 10.84% and 17.02% for Niagara and Niagara2, respectively. Given the generic nature of McPAT, we consider these errors acceptable.

4.2.3 Results

4.2.3.1 Many-core Processor Implementation at different Technology nodes

We illustrate the utility of McPAT by applying it to the design and architectural evaluation of a scalable many-core architecture with optional stacked last level cache (LLC) die. We evaluate this architecture across five technologies-90, 65, 45, 32, and 22nm-which covers years 2004 to year 2016 of the ITRS roadmap.

4.2.3.2 System architecture

The many-core architecture we assume targets future high throughput computing. It is a highly parallel architecture with coherent L2 caches and an optional L3 cache die. We evaluate two aspects of this architecture: (1) benefits of organizing cores into clusters with local interconnect (3) some tradeoffs in including the optional stacked L3 cache.

At this point, we can only show very preliminary results, based on peak power modeling. It is worth noting that the weak link in this process is not the McPAT framework, but rather the performance simulation. That is, we lack a full system simulation engine that faithfully models many cores, coherent caches, a complete interconnection network, and memory controllers at the level of detail that matches what McPAT can model on the power side. However, full timing and average power dissipation results for a set of parallel applications will be available *very* shortly, based on a significantly enhanced version of the M5 simulator [32]. This exercise, even at this preliminary stage, serves to illustrate the breadth of devices that McPAT can model, the depth of detail at which those devices can be specified, and the ability of McPAT to configure the entire architecture to meet timing and area constraints, optimizing different pieces according to different design styles.

Figure 15 shows the clustered version of the many-core architecture. It consists of multiple clusters connected by a 2D mesh on-chip network. Inside a cluster, there are four multithreaded Niagara-like cores with one 4-way SIMD FPU per core. Each core has 32KB, 8-way set-associative private SRAM-based L1 instruction and data caches, with cache line size being 32B. Four cores in a cluster share a 1MB 4-bank 8-way set-associative SRAM-based L2 cache, with 64B cache lines. A crossbar is used for the local interconnect that provides fast access and uniform latency for intra-cluster communications. A hub handles packet encapsulation between cluster and router.

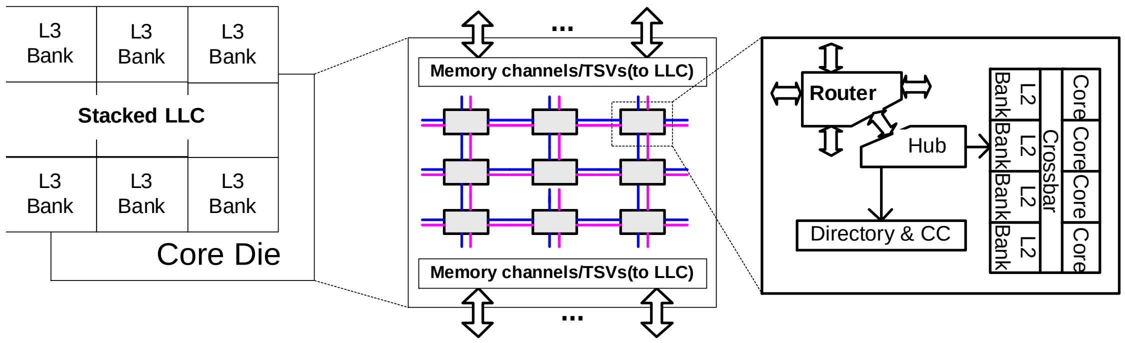


Figure 15: Clustered many-core system architecture

The mesh network has a data width of 256 bits. We use wormwhole routers with two virtual channels per physical port to avoid deadlock [10]. Each virtual channel has an 8-deep flit input buffer. Double-pumped crossbars [49] are used in all routers. According to McPAT's area modeling results, the double-pumped crossbars reduce the area of the 2D mesh router and the ring router by 27.3% and 35.7% respectively, compared to the routers with single-pumped crossbar. Routers in both the mesh and ring networks have a local port that connects the hub of a cluster as well as ports that connect the neighboring routers.

An optional shared last level cache die can be stacked on the core die to minimize the system penalty caused by finite-cache effects. The L3 cache die is designed to be truly optional in that the main die itself is a complete, coherent many-core processor with on-chip network, memory controllers, and the reserved high density through silicon vias (HDTSVs) for control/data/address buses of the optional L3 cache. In this way, processors can be easily produced in both configurations and sold into different markets at different price points with a common design.

The optional, fully-shared L3 cache is connected to the memory controllers and acts as a large shared cache of main memory using a write-back policy. Thoziyoor et. al show that a commodity DRAM based stacked L3 cache, when optimized for capacity, can provide both system performance and power reduction [4]. Therefore, we model a commodity DRAM-based L3 cache and optimize it for capacity. We also use similar operational considerations as described in [4] - we use a commodity DRAM based L3 cache using an SRAM-like interface with multisubbank interleaving and mapping multiple cache sets to DRAM pages. The total capacity of the L3 cache is decided by the area of the core die and grows linearly with the cluster count. To increase the number of memory controllers/channels linearly with the increased core count is difficult because of the limited pin count of the chip. In our many-core architecture, the memory channels are placed at the edge of the chip to minimize the routing overhead. Therefore, the number of memory controllers/channels grows proportional to the square root of the cluster/core count. A memory controller with dual/triple channels are connected to the routers at the edge and shared through the on-chip network as shown in Figure 15.

We evaluate the many-core processor fabricated using 90nm, 65nm, 45nm, 32nm and 22nm technology. We first sweep across the technology nodes and pick proper implementations at each node. We pick the best configuration based on performance, power, and cost.

The same design space exploration (minus full timing results unavailable at this time) is done at each technology node. At 90nm, only one cluster can fit in the die, and no 2D mesh global network is needed. At 65nm node, two clusters can be placed on the die. Dual-channel and a tri-channel memory controllers are used for the 90nm and 65nm implementation, respectively. At 45nm, 32nm, and 22 nm node, 4, 8 and 16 clusters can be placed on the die respectively, connected by on-chip 2D network. As shown in Figure 15, the L2 caches (including L2 directories that have shadowed L1 tags) of all the clusters are placed at the edge of cores along both horizontal and vertical directions so that the interconnects of the 2D mesh can route over L2 cache along both directions. Therefore, interconnects between routers can be routed over L2 caches with very small area overhead. (Our hierarchical wire model starts to assign wiring plane from semi-global metal plane for the interconnection between routers, semi-global wires can be routed over the cache arrays). We account for the fact that the sub-array in cache mats has to be moved to make space for the placement of wire buffers and latches.

We start from a conservative die size around 200 mm^2 and try to keep it constant across all the technologies by balancing area power, speed and area using our tool. The intrinsic speed of high-performance transistors increases by 17% per year according to ITRS. However, increasing CPU clock frequency at this pace will lead to unmanageable chip power density and total chip power dissipation. We increase the clock frequency conservatively by around 25% every 3-year generation. Our power tool optimizes the power of the many-core processor, while satisfying the target clock frequency and die area at each technology node. We allow 10% deviation from the target 200mm^2 die size during the optimizations. We also scale the main memory based on the availability of major DIMM products at each technology node.

4.2.3.3 Scaling many-core architecture across the technology nodes

In this section, we present the peak power characteristics of this architecture as we scale to future technologies. Dynamicerformance and average power simulations are underway and will be available by May 15, 2009, which we will publish in a separate technical report.

Not surprisingly, power grows, but less than linearly with the number of cores. The Figure 16 shows the breakdown of system power into core power, network-on-chip (NOC), I/O, L2 cache, global clock and L3 cache for four configurations: Clustered with L3, Clustered without L3, Non-clustered with L3 and Non-clustered without L3 across all technology nodes. The peak power for each individual block is broken down into its dynamic, standby leakage and short circuit power, which are calculated as described in the Appendix. In this breakdown the activity factor only affects dynamic power. Thus the leakage and short circuit power will remain the same in both average and peak power calculations.

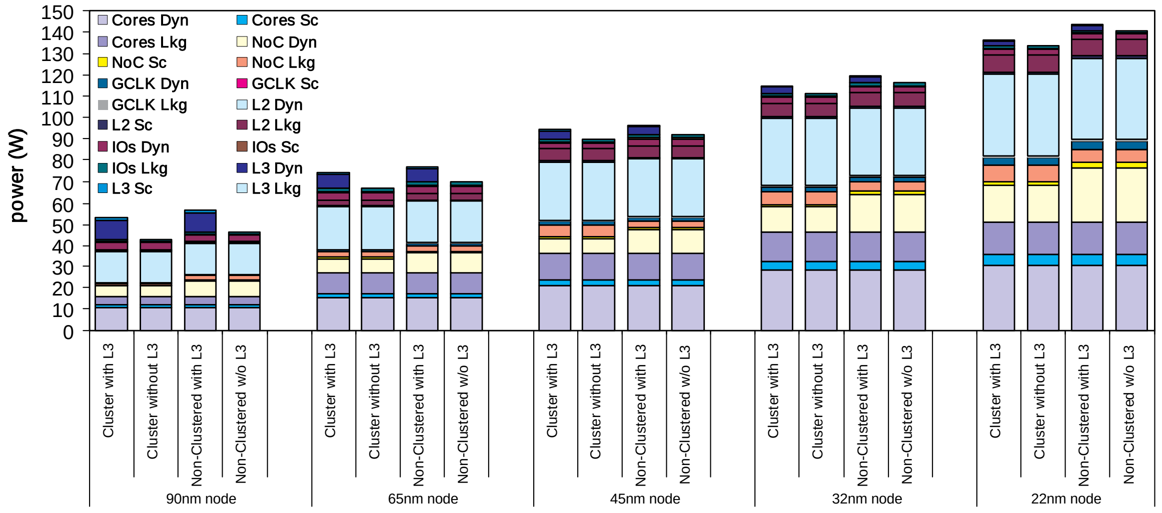


Figure 16: Power consumption across configurations and technology nodes

Also, we see that as technology shrinks, the incremental power of the L3 cache diminishes significantly, relative the rest of the processor. We also see that the leakage power, although a big factor in all configurations, remains a fairly stable contributor to total power.

4.3 High-Level SASIC Design Space and Performance Exploration

The hierarchical approach to power modeling used in McPAT supports system architects in managing the overall power budget of a design. For each component of the system, users can determine the consumption of each of the three types of power: leakage, short-circuit, and dynamic. With this information, designers can determine which blocks to customize in order to optimize the overall power consumption.

4.3.1 Baseline System: Hierarchical CMP

The class of systems into which we assume the SASIC chips discussed here are deployed, are highly scalable multi-node systems, where each node contains one or more multi-core microprocessor chips, some (large) amount of local DRAM packaged in some standard modules such as DDR2 or FB-DIMM, and high performance links to other such nodes (perhaps through routers or other network mechanisms). This design, shown in Figure 17, covers both large server farms and high end supercomputers such as Red Storm [39] and Blue Gene/L [12].

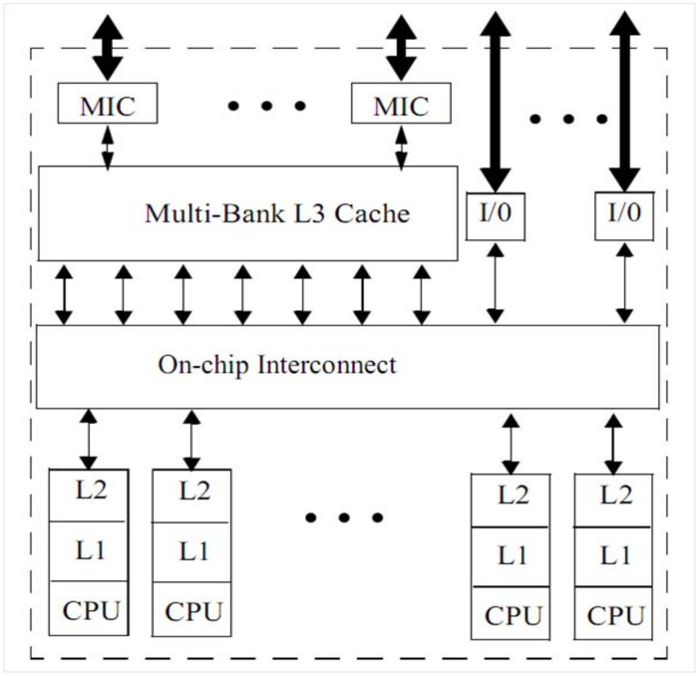


Figure 17: Multicore system block diagram

We also assume the general layout and system architecture of the microprocessor is as a hierarchical multi-core design [26]. The major blocks found on such chips include some number of: processor cores, off-chip memory interface controllers, off-chip communication links for node-to-node communication and/or I/O, shared cache between the cores and the memory controllers, and some sort of on-chip interconnect, such as a crossbar.

In addition, we also assume that applications that run on such chips and systems obey "weak scaling": i.e. they can be parallelized, where if sufficient memory and bandwidth is teamed with a unit of performance, then overall application execution time will scale with the amount of parallelism available.

4.3.2 Use in a Structured ASIC Design Program

A key feature of this method is that all optimization targets a specific application profile. However, as the design exploration changes due to changing availability of technologies to incorporate into a target chip, additional application profile data may be needed. Therefore, this approach is ideal for a situation in which hardware and systems designers have both access and familiarity with some set of applications, as would be the case in a SASIC design program.

4.3.2.1 Application Profiling

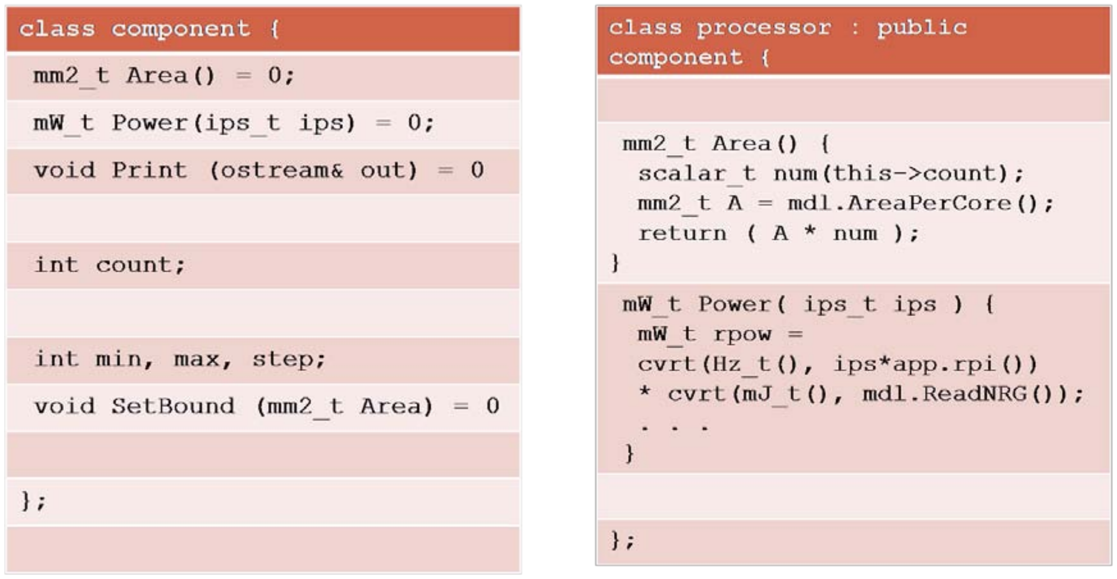
At the highest level, interest is in sustained performance over time, or application throughput. The types of parallel applications expected to use these large multicore systems measure performance in terms of hours or weeks of runtime, and so statistical information provides the required level of detail, allowing this type of tool to be useful. Even for the case of embedded system, although actual execution times on the hardware may be short, the *simulation* of execution may be very long, making a high level estimation tool very useful.

Application behavior is assumed mostly "blind" to chip structures. In other words, changing on-chip cache size does not significantly change the number of memory references. In cases where an application does not conform to this assumption, this approach remains useful. This is because changing behavior in this way almost always represents optimizations. Since our high level exploration looks at *performance limits*, rather than *performance improvements*, we can still bound non-optimal application behavior. This would just mean that the space boundaries are not as tight, and more detailed analysis would be required later.

Scaling is the major exception to static application behavior modeling. Since parallel applications are the target, we must model how does behavior change with the increased parallelism required to utilize increasing number of processing cores. This will typically be increased communication overheads, which then provide overall performance limits based on the chip resources and routing.

4.3.2.2 Component Modeling

This section uses an example scenario to explain how component modeling should proceed. Since our toolset is written in C++, examples shown here use that language specifically.



```
class component {
  mm2_t Area() = 0;
  mW_t Power(ips_t ips) = 0;
  void Print (ostream& out) = 0

  int count;

  int min, max, step;
  void SetBound (mm2_t Area) = 0

};
```

```
class processor : public
component {

 mm2_t Area() {
  scalar_t num(this->count);
  mm2_t A = mdl.AreaPerCore();
  return ( A * num );
}
 mW_t Power( ips_t ips ) {
  mW_t rpow =
  cvrt(Hz_t(), ips*app.rpi())
  * cvrt(mJ_t(), mdl.ReadNRG());
  . . .
 }

};
```

Figure 18: Example component class

Figure 18 shows a base component class. Several values allow a main loop to treat all components similarly. The count variable indicates a number of identical instances of some component. Since the main exploration will try all valid combinations of components, in this case the programmer can control the sweep values through min, max and step.

Empty virtual classes give the programmer a place to specialize each component with area and power calculation as well as IO functions for detailed text output. To help reduce debugging time and help multiple designers communicate component models through code, special types enforce consistency between objects.

An example component specialization, shown in Figure 18, implements the area and power functions. Inside the area calculation, the mdl object provides an AreaPerCore. Because of the return type of mm_t, each of the component base class, processor class, and mdl objects could be implemented at different times, by different designers. The units of measurement must be consistent, or the program will not compile.

Aside from coordinating multiple code sources, the unit type system ensures correct dimensional analysis at compile time. The power function in Figure 18, takes $\frac{Instruction}{Second}$ as input, which is then multiplied by app.rpi (reads per instruction), and converted into Hz, $\frac{1}{Second}$. Finally, a multiply by milli Joules yields $\frac{mJ}{S} = mW$. Again, the program will not compile if units do not match.

Other data types, with example method prototypes, used to model chip resources include:

- Level: organizational only, contains components
- Chip: contains arbitrary number of levels
- Topology : all wires on chip
- Wires: from 1 set identical components to another component

4.3.3 Basic Process

Using some set of components and application data, the basic algorithm is as follows:

1. Determine number of design points: Projected technology for a given year + sub-points for each year
2. For each design point, iterate through all area-feasible combinations of subcomponents
3. For each iteration, use component hierarchy and interconnect topology to determine latencies
4. Given a set latency, search for maximum performance within area/power constraints, where

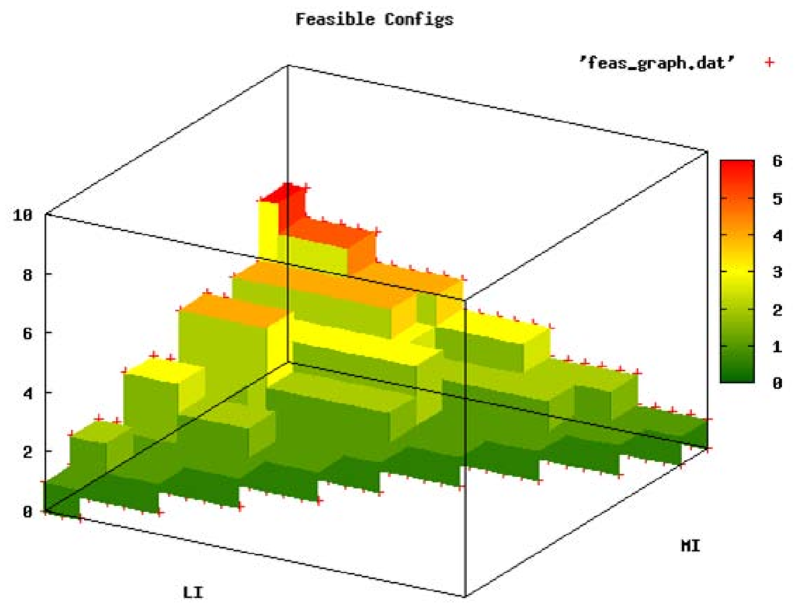
```
f1(Performance, Latency) = Required Bandwidth

f2(Bandwidth) = Required Interconnect & Component
Power

f3(Bandwidth, Clock Rate) = Interconnect Area
```

4.3.4 Sample Outputs

Using the baseline system from Figure 17, along with stub configuration values, the toolset produced sample outputs, shown in Figure 19. Each bar represents the maximum feasible number of processors given a number of link and memory interfaces. Figure 19 shows a two dimensional projection of those values. Note that these particular graphs do not show performance. Outputs could easily be changed to show max performance, max per-processor performance, parallel efficiency, energy delay, or any other metric of interest. The toolset includes graph data structures with simple update commands and output functions.



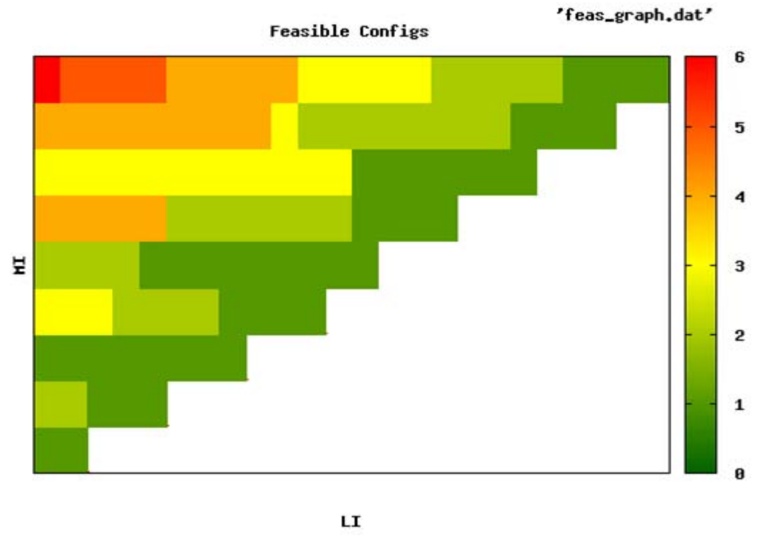


Figure 19: Sample outputs, showing the maximum number of processors feasible (Feasible Configs) for a given number of memory interfaces (MI) and link interfaces (LI).

4.3.5 Completed Functionality

- Graph data output structures, which produce text based outputs and scripts for use with the freely available gnuplot tool

- Special unit types / operations / dimensional analysis are implemented as a highly portable C++ template library

- Configuration file input format provides an integrated way to input values with special types so that unit validity extends out to this level

- Base class implementations are available -- Component, Profile, Topology, Wires, Chip

- Main looping framework uses these base classes to do exploration and output results

4.3.6 Functionality To Be Done

- To test the baseline system, more derived specific classes need be completed

- Though completed, the units functionality required more testing as the baseline system is tested

- Base class implementations for specific applications with scaling models must be added

- Currently, the search proceeds naively, because it is very fast. Larger chips and larger numbers of options are expected, so branch and bound search truncation will be added.

5 Conclusions

5.1 Regular Fabrics

Regular fabrics hold great promise as an implementation technology for low-volume, memory-intensive SASICs. Details on recommendations for continued work in this area are provided in Section 7.

5.2 Architectural Level Power, Area, and Timing Framework

As the number of transistors on an IC continues to increase in accordance with Moore's Law, power consumption has become a leading consideration in processor design. The growing concern over power has led to dramatic shifts in architecture, turning away from the increasing complexity of individual cores and aggressive clock rate advancement, and towards designs with more, less aggressive cores. While tools like Wattch [7] have served as invaluable tools for modeling dynamic power consumption for out-of-order, superscalar processors, new tools are needed to explore tradeoffs in power, area, and timing for multicore and manycore systems.

Building on the contributions of Wattch, its extension HotLeakage [54], the memory and cache simulator CACTI [4], and the NoC simulator Orion, McPAT is the first tool to integrate power, area, and timing models for a complete processor, including components needed to model multi-and many-core systems. Unlike prior tools which were tightly integrated with specific performance simulators, McPAT uses an XML interface to decouple the architectural simulator from the power, area, and timing analysis, so that it can be readily used with a variety of simulators.

McPAT's power models account for dynamic power, subthreshold leakage, and gate leakage, as well as short-circuit power, using calculated timing information to determine the interval when both the pull-up and pull-down devices are conducting. It is also the first processor power modeling environment to include both power gating and clock gating, as well as support for P-state and C-state power management schemes. Like CACTI and Orion, McPAT uses MASTAR [42] to calculate device models based on the ITRS roadmap [43], giving it the ability to model not only bulk CMOS transistors, but also SOI and double-gate devices that become increasingly important at the 32 nm technology node and beyond. Finally, as CACTI did for memory structures, McPAT includes the ability to determine power-optimal configurations for array structures and interconnect, given specified targets for area and timing values. Validation experiments show very good agreement between McPAT's power predictions and published results for the Sun Niagara and Niagara 2 processors.

In providing these capabilities, McPAT supports architects in exploring a broad design space for future multicore and manycore system. As an illustration, we show some very preliminary results for a large many-core architecture applied at several different technology points, and identify some trends as that architecture scales to smaller devices sizes and larger core counts.

5.3 High-Level Design Space Exploration

This is the type of tool for SASIC designers to use before moving onto more time consuming CAD tools. Ideally, the design process would eventually incorporate this tool in a feedback loop, where high level exploration would guide more detailed design and simulation which would then provide more insight into how to make better high level models.

As time goes by, technology specific changes can be quickly integrated into the toolset, without a major software engineering effort. For example, a more complex power model might require changes to the component classes so that power calculation methods would take additional parameters. Topology changes could add new methods to calculate communication overheads and change wire power and area methods.

In the end, these tools would produce outputs which show bottlenecks to performance and approximate chip parameters based on some large number of models and per year expected technology trends. The aim is to produce results much more complex than back-of-the-envelope calculations, without requiring much more in designer effort.

6 Summary and Recommendations

6.1 Summary

The key findings of this study are:

- High level analysis tools can reduce design cycle time, cost, and risk by enabling design space exploration for area, speed, and power for multicore, memory-intensive systems. System-level tools that use simple heuristic and analytic models, as well as "midrange" tools that run in conjunction with a standard architectural simulator are both necessary. In this study, we have developed examples of each of these and have used them to characterize a range of designs.

- Using the PIM Lite chip as an example design, we have demonstrated that an implementation using the regular patterned "bricks" technology developed at Carnegie Mellon University and commercialized by Fabbrix/PDF Solution has performance comparable to a conventional ASIC, with a design flow little more complex than that for a conventional ASIC (addition of a single "brick discovery" step), that is much more compatible with advanced lithography.

It is difficult to estimate precise savings in design time and cost, as this is highly dependent on the specific application. Further study using a standard reference design more complex than PIM Lite could ascertain these differences in the future.

6.2 RecommendationPatterned ASICs

The key observation behind the Fabbrix technology used in the PIM Lite portion of the study is that the challenges with lithographic patterning to create corners and bends as we reach smaller features sizes will become increasingly difficult. Consequently, in the demonstrated Fabbrix circuit block, each pattern level from diffusion through the topmost metal interconnect is unidirectional to the maximum extent possible, and differing unconnected line segments are grouped so as to share a center line. Thus any interconnection of patterning from one layer to the next tends to look like a set of parallel lines where each line may have multiple breaks in it. The Fabbrix technology has demonstrated that designs such as PIM Lite can be constructed with such regular patterning without loss of quality in terms of area, power, and performance.

In looking forward to approaches that can reduce the cost of designing new chips, particularly those for low to medium volume, a major consideration must be how to reduce the cost and complexity of the mask sets. Of particular interest isthe cost of such systems for applications that require a significant amount of on-chip memory, where variations in designs may simply reflect different compute engines (logic blocks) perhaps connected in different ways to different amounts of memory. This is in an environment where estimates are by 2010 that 90% of an ASIC die will be memory of some sort.

A classical solution to this has been gate arrays: create a sea of regular patterns done at the Front End of the Line (FEOL), with personalization left to application-specific BEOL metal patterns on the last few levels of interconnect where feature sizes are more relaxed and it is cheaper to develop specialized masks. This, however, becomes unviable when most of the die needs to be memory, since it would be very inefficient in terms of both area and power to configure SRAM cells from general regular patterns as opposed to custom designed memory cells.

Today's structured ASIC approaches try to avoid this latter problem in several ways:

- Predefine arrays of SRAM cells,
- Use higher levels of logic abstractions including configurable small cores,
- Use other methods of configuration: BEOL masks, laser trimming, non-volatile memory, via programming, etc.

Such approaches try to share at least subsets of masks from project to project, and possibly even share portions of pre-processed FEOL layers that can be made in a project-independent way. The trick is to develop all patterning to achieve the highest density of embedded memory, while grouping logic into pattern-compatible functions that most efficiently utilize the patterning. Current examples include:

- ChipX: top-level metal configurable logic X-Cell™, with configurable memory and I/O structures. http://www.chipx.com/asic-options.html
- eAsic: Nextreme: LUT-based with only changes to 1 via level. http://www.easic.com/index.php?p=nextreme2-overview
- Triad Semiconductor: Mixed signal ASICs using "Via Configurable Array" Preplaced resources underneath "global routing fabric" and a Single mask layer changes to specify vias between. http://www.triadsemi.com/vca-technology/
- ALTERA HardCopy:: architecturally equivalent to FPGAs: using same base arrays and customized using top two metal layers. http://www.altera.com/literature/lit-hrd.jsp

All of these approaches suffer from one or more common problems:

- They are logic-centric, not memory-centric, which means they optimize10% of the die.
- There is considerable discussion if it is even possible to build reliable SRAM from logic at 22nm and below.
- Preselected partitioning die area into memory and logic areas saves mask costs but often is the wrong mix for range of applications.
- And today's approaches are not using the densest memory macros.

Observations from our CMU partners led to an interesting alternative to any of the above:

Patterned ASICs. The goals of this approach include:

- Make low volume, scalable memory-intensive ASICs affordable for DoD and other advanced applications

- Create a new top-down design methodology to use alternative fabrication techniques
- Maximize use of current CAD flow
- Complement potential other advanced fab-oriented programs, and specifically DARPA advanced lithography programs investigate use of E-beam technology. The following techniques would reduce costs for low volume fab runs:
    - optics and resist materials to create just regular patterns,
    - patterning and cutting techniques, especially but not necessarily exclusively E-beam,
    - the use of multiple E-beams to do such cuttings,
    - the use of alternative techniques, such as E-beam, to do on-wafer inspection

A Patterned ASIC development would start with the regular patterns needed for a dense memory, with each pattern layer of interconnect starting with nothing more than a parallel set of lines covering the whole die. We would expect that the pitch between these, at least the lowest layers, would be selected to match what is needed for a surface full of memory cells. There is NO preplacement of either memory blocks or logic islands. Each layer of interconnect on a die starts with the creation of such sets of lines.

As an aside, this is exactly where much of the very deep submicron nanotechnology fabrication research approaches are going. It is far easier, for example, to build such patterns with simple diffraction than with any approach that requires any variability in the spacing.

The next step towards such a development approach is to constrain the family of logic circuits, including the support circuitry for the memory arrays, to use regions of such overlapping patterns with <u>*only cuts in the preexisting parallel lines.*</u> With this approach, a standard regular grid could be deposited on the chip, with sections removed using E-beam. This is already at the heart of the Fabbrix logic design approach, and would be extended to include both memory support and I/O blocks.

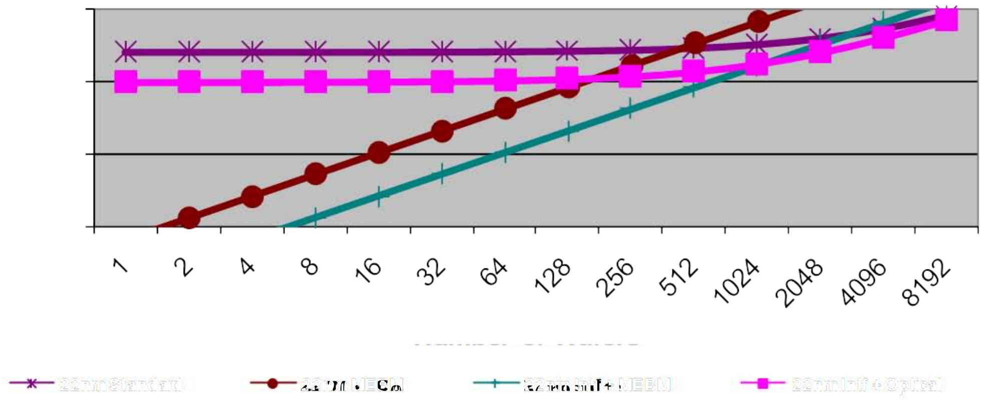
Now, laying out a custom design may start with positioning memory in any amount, anywhere on a die. Logic would be positioned in the gaps. Implementing this on the die requires making targeted cuts in the regular patterns. This could be done lithographically, or more interestingly, on a die-by-die basis using an E-beam. For low volume or prototype work such an approach means that the only masks needed are those for the regular patterns of lines – a HUGE decrease in mask costs both by dollars and in preparation time.

Multi-head E-beam devices are beginning to appear, and it is feasible to consider using such devices to make multiple such cuts concurrently. In general, the capital equipment cost for a single-head E-beam would be comparable to the capital costs for mask-based lithography equipment at around 32 nm technology. The most significant cost difference, however, results from throughput differences. At 32 nm, the processing time for patterning a wafer would be 2-3 orders of magnitude slower for E-beam than it would be for masks. Use of multi-head E-beam trades throughput for cost. In the case of low volume production, however, the high cost of masks makes them less cost effective than E-beam, as discussed below.

An initial exploration by CMU of the differences between this and a more conventional approach are astonishing. If we used an E-beam to "deposit" material in the lines, approximately 27% of a typical surface would need to be covered. If instead we used an E-beam to "cut" existing patterns, only 5% of the surface needs to be touched. If it takes equal time to do a unit length in either approach, then the cutting approach is OVER 5 TIMES faster.

As an example, the following chart (based on a study from a DARPA seedling project with Carnegie Mellon that was supported by Dr. Michael Fritze) summarizes the total wafer cost for a 22nm SRAM where an interference-based lithography grating pattern is used as a starting point, and where a multiple e-beam (MEBM) tool can achieve 30 wafers an hour for cut rather than deposition patterns. Four approaches are shown:

- The expected cost of a "stock conventional" 22 nm process to handle one wafer. This requires a complete mask set, and is north of $1M for even the first wafer, regardless of wafer volume.
- The cost of a 22nm process where diffraction patterns are used to create the lines, and then traditional optical masks used to make the cuts. Cost here is less than for the optical masks to create the patterns, but still in the $1M range for even wafer one.
- A deposition approach to create the patterned lines using a MEBM tool. Since this is mostly driven by time on machine, the cost per wafer is largely a linear function of the number of wafers.
- A full-scale Patterned Cut ASIC approach where the initial lines are drawn by a cheap diffraction pattern, and then cut by the MEBM. Again, the cost is linear in the number of wafers, and approximately 1/5 of the time of the prior approach because only 1/5 of the area needs to be touched.



Both of the latter approaches show huge savings over either of the first two conventional approaches, at least until volumes reach several hundred wafers. At a volume of 10 wafers, the Patterned Cut approach saves 160X over the conventional. As reference, a 300mm diameter wafer would support up to 176 400mm^2 die, so 10 wafers with a 50% yield would deliver just under 900 chips.A program to develop such a capability would include the following steps:

1. Develop a fab flow and cost model for starting with ONLY patterned mask sets and using E-beam to personalize via cuts
2. Develop pattern sets for different memory cell technologies
   - Silicon SRAM, eDRAM, DRAM
   - Non-CMOS silicon G4-FETs, Flash, PCRAM
   - Synergism with non-silicon: MRAM, mQCA, Nantero, CMOL, …
3. Use our Fabbrix experience as basis for “cut-based” logic families that implements memoryblock support circuits and local “at-the-memory” cores from cuts in memory patterns, allowing application-specific selection of width, capacity, and features.
4. Expand patterning to include core-core interconnect options
5. Size more complex processing cores in “cut-based” circuit families
6. Investigate memory based layout ideas such as “Whirlpool” PLAs, a technique of combining multiple small PLA arrays into a ring-like structure developed at UC Berkeley.
7. Develop new test and redundancy techniques that exploit patterns and cuts, such as fab time fault identification, analysis, and reconfiguration via E-beam scanning. This can improve yield as well as reduce test time and cost
8. Consider needs of scaled multi-chip systems, including advanced packaging requirements, I/O interface circuits, PLLs, DLLs, clock generation, …
9. Optimize CAD flow with potentially improved timing, area, power estimators, using high-level design space exploration to identify most profitable configurations early.

7 References

[1] K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka, "Power gating with multiple sleep modes," *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, pp. 5 pp.-, March 2006.

[2] H.-T. Ahn and D. Allstot, "A low-jitter 1.9-v cmos pll for ultrasparc microprocessor applications," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 3, pp. 450-454, Mar 2000.

[3] AMD, "Amd opteron processor benchmarking for clustered systems," *AMD WhitePaper*, 2003.

[4] S. Thoziyoor, J. Ahn, N. Jouppi, "CACTI-D Technical Report." Oct 2007.

[5] S. Li, N. Jouppi. "McPAT Technical Report. Details withheld to preserve author anonymity." Oct 2008.

[6] D. Brooks, P. Bose, V. Srinivasan, M. K. Gschwind, P. G. Emma, and M. G. Rosenfield, "New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors," *IBM J. Res. Dev.*, vol. 47, no. 5-6, pp. 653-670, 2003.

[7] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *ISCA*, 2000, pp. 83-94.

[8] R. Y. Chen, N. Vijaykrishnan, and M. J. Irwin, "Clock power issues in system-on-a-chip designs," in *WVLSI '99: Proceedings of the IEEE Computer Society Workshop on VLSI'99*. Washington, DC, USA: IEEE Computer Society, 1999, p. 48.

[9] B. Chi and B. Shi, "2/3 divider cell using phase switching technique," *Electronics Letters*, vol. 37, no. 14, pp. 875-877, Jul 2001.

[10] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[11] T. Fischer, F. Anderson, B. Patella, and S. Naffziger, "A 90 nm variable frequency clock system for a power-managed itanium architecture processor," in *International Solid-State Circuits Conference (ISSCC)*, IEEE. IEEE, Feb. 2005, pp. 294-295.

[12] M. Fritze, B. Tyrell, T. Fednyshyn and M. Rothschild, High-Throughput Hybrid Optical Maskless Lithography: All-Optical 32nm Node Imaging, Proceedings of SPIE, Vol. 5751, 2005.

[13] M. Fritze et al, Hybrid Optical Maskless Lithography: Scaling Beyond the 45nm Node, J. Vac. Sci. Technology B 23(6), Nov/Dec 2005.

[14] Gara, A. and Blumrich, MA and Chen, D. and Chiu, GLT and Coteus, P. and Giampapa, ME and Haring, RA and Heidelberger, P. and Hoenicke, D. and Kopcsay, GV and others. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49(2/3):195--212, 2005.

[15] M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power considerations in the design of the alpha 21264 microprocessor," in *DAC '98: Proceedings of the 35th annual conference on Design automation*. New York, NY, USA: ACM, 1998, pp. 726-731.

[16] M. Gries, "The impact of recent dram architectures on embedded systems performance," *Euromicro Conference, 2000. Proceedings of the 26th*, vol. 1, pp. 282-289 vol.1, 2000.

[17] R. Ho, "On-chip Wires: Scaling and Efficiency," Ph.D. dissertation, Stanford University, 2003.
[18] INTEL, "Power and thermal management in the intel? coretm duo processor," *Intel Technology Journal*, vol. 10, pp. 109-122, May 2006.
[19] --, "Introducing the 45nm next-generation intel core microarchitecture," INTEL, Tech. Rep., 2007.
[20] Tejas Jhaveri, Vyacheslav Rovner, Larry Pileggi, Andrzej J. Strojwas, et al., "Maximization of Layout Printability/Manufacturability by Extreme Layout Regularity", Journal of Micro/Nanolithography, MEMS, and MOEMS, Vol 6 (03), 2007.
[21] Tejas Jhaveri, Andrzej Strojwas, Larry Pileggi & Vyacheslav Rovner, OPC Simplification & Mask Cost Reduction using Regular Design Fabrics, SPIE Advanced Lithography Conference, February 2009.
[22] T. Johnson and U. Nawathe, "An 8-core, 64-thread, 64-bit power efficient sparc soc (niagara2)," in *ISPD '07: Proceedings of the 2007 international symposium on Physical design*. ACM, 2007, pp. 2-2.
[23] B. L. L.-S. P. Kambiz Samadi, Andrew Kahng, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," The University of California, San Diego DEPT. OF COMPUTER SCIENCE," Technical Report, 2008.
[24] V. Kheterpal, T. Hersan, V. Rovner, D. Motiani, Y. Takagawa, L. Pileggi and A. Strojwas, "Design Methodology for IC Manufacturability Based on Regular Logic-Bricks," *Design Automation Conference*, June 2005.
[25] N. S. Kim, *et al.*, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68-75, 2003.
[26] Kogge, P.M. An Exploration of the Technology Space for Multi-Core Memory/Logic Chips for Highly Scalable Parallel Systems. *Proceedings of the Innovative Architecture on Future Generation High-Performance Processors and Systems*, :55--64, 2005.
[27] P.M. Kogge. "EXECUBE-A New Architecture for Scaleable MPPs." *International Conference on Parallel Processing*, 1:77-84, Aug. 1994.
[28] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded sparc processor," *IEEE Micro*, vol. 25, no. 2, pp. 21-29, 2005.
[29] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling," in *ISCA '05: Proceedings of the 32nd annual international symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 408-419.
[30] A. S. Leon, K. W. Tam, J. L. Shin, D. Weisner, and F. Schumacher, "A power-efficient high-throughput 32-thread sparc processor," *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 1, pp. 7-16, Jan. 2007.
[31] X. Liang, K. Turgay, and D. Brooks, "Architectural Power Models for SRAM and CAM Structures Based on Hybrid Analytical/Empirical Techniques," in *ICCAD*, 2007.
[32] M5 Simulator System, http://www.m5sim.org.

[33] S. Mathew, M. Anders, B. Bloechel, T. Nguyen, R. Krishnamurthy, and S. Borkar, "A 4-ghz 300-mw 64-bit integer execution alu with dual supply voltages in 90-nm cmos," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 44-51, Jan. 2005.

[34] U. Nawathe, M. Hassan, K. Yen, A. Kumar, A. Ramachandran, and D. Greenhill, "Implementation of an 8-core, 64-thread, power-efficient sparc server on a chip," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 6-20, Jan. 2008.

[35] K. Nose and T. Sakurai, "Analysis and future trend of short-circuit power," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 19, no. 9, pp. 1023-1030, 2000.

[36] K. Nowka, *et al.*, "A 0.9 V to 1.95 V dynamic voltage-scalable and frequency-scalable 32 b powerPC processor," in *International Solid-State Circuits Conference (ISSCC)*, IEEE. San Francisco, CA: IEEE, Feb. 2002, pp. 340-341.

[37] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-effective superscalar processors," *Computer Architecture, International Symposium on*, vol. 0, p. 206, 1997.

[38] B. D. S. N. Patrick Mahoney, Eric Fetzer, "Clock distribution on a dual-core multi-threaded itanium family processor," in *International Solid-State Circuits Conference (ISSCC)*, IEEE. IEEE, Feb. 2005, pp. 292-293.

[39] -. Red Storm. http://www.cs.sandia.gov/platforms/RedStorm.html, 2007

[40] A. F. Rodrigues, "Parametric sizing for processors," Sandia National Laboratories," Technical Report, 2007.

[41] Semiconductor Industries Association, "International Technology Roadmap for Semiconductors," Tech. Rep., 2001.

[42] --, "Model for Assessment of CMOS Technologies and Roadmaps (MASTAR)," 2005, http://www.itrs.net/models.html.

[43] --, "International Technology Roadmap for Semiconductors. http://www.itrs.net/," 2007.

[44] D. B. Shashank Gupta, Stephen Keckler, "Technology independent area anddelay estimates for microprocessor building blocks," The University of Texasat Austin DEPT. OF COMPUTER SCIENCE, Technical Report TECH. REPORT TR2000-05, 2000.

[45] I. Sun Microsystems, "Opensparc T1 microarchitecture specification," Sun Microsystems, Inc., Tech. Rep., July 2005.

[46] --, "Opensparc T2 core microarchitecture specification," Sun Microsystems, Inc., Tech. Rep., July 2007.

[47] --, "Opensparc T2 system-on-chip (soc) microarchitecture specification," Sun Microsystems, Inc., Tech. Rep., July 2007.

[48] T. N. Theis, "The future of interconnection technology," *IBM Journal of Research and Development*, vol. 44, no. 3, pp. 379-390, 2000.

[49] S. Thoziyoor, J. Brockman, and D. Rinzler. “PIM Lite: A Multithreaded Processor-In-Memory Prototype.” In *Proceedings of Great Lakes Symposium on VLSI*, Chicago, IL, April 17, 2005.

[50] S. Vangal, N. Borkar, and A. Alvandpour, "A six-port 57gb/s double-pumped nonblocking router core," *VLSI Circuits, 2005. Digest of Technical Papers. 2005 Symposium on*, pp. 268-269, June 2005.

[51] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," *Microarchitecture, IEEE/ACM International Symposium on*, vol. 0, p. 294, 2002.
[52] H. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," in *MICRO*, Nov 2002.
[53] K. L. Wong, *et al.*, "Cascaded pll design for a 90nm cmos high performance microprocessor," in *International Solid-State Circuits Conference (ISSCC)*, IEEE. IEEE, Feb. 2003, pp. 422-424.
[54] Yan Zhang, Dharmesh Parikh, "Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects," UNIV. OF VIRGINIA DEPT. OF COMPUTER SCIENCE, Technical Report TECH. REPORT CS-2003-05, 2003.
[55] V. Zaccaria, D. Sciuto, and C. Silvano, *Power Estimation and Optimization Methodologies for VLIW-Based Embedded Systems*. Norwell, MA, USA: Kluwer Academic Publishers, 2003.

Appendix A: Integrated and Hierarchical Models for Power, Area and Timing

In order to model the power, area, and timing of a complex multicore processor, McPAT takes an *integrated* and *hierarchical* approach. The approach is integrated in that it models power, area, and timing *simultaneously*. Because of this McPAT is able to ensure that the results are mutually consistent from an electrical standpoint. McPAT's approach is hierarchical in that it decomposes the models into three levels: architectural/microarchitectural, circuit, and technology. This provides users with the flexibility to model a broad range of possible multicore configurations across a range of implementation technologies. Taken together, this integrated and hierarchical approach enables the user to paint a comprehensive picture of a design space, exploring tradeoffs between design and technology choices in terms of power, area, and timing. In the remainder of this section, we first provide details of the integrated models and then describe the model hierarchy.

8.1 Integrated Power, Area, and Timing Models

8.1.2 Power Modeling

As shown in Equation (A-1), power dissipation of CMOS circuits has three main components: dynamic power, short circuit power and leakage power. All three contribute significantly to the total power dissipation of multicore processors fabricated using deep-submicron technology.

$$P_{total} = \underbrace{\alpha C V_{dd} \Delta V f_{clk}}_{Dynamic} + \underbrace{V_{dd} I_{short\_circuit}}_{Short\_circuit} + \underbrace{V_{dd} I_{leakage}}_{Leakage} \qquad \text{(A-1)}$$

The first term is the *dynamic power* that is spent in charging and discharging the capacitive loads when the circuit switches state, where C is the total load capacitance, V_{dd} is the supply voltage, ΔV is the voltage swing during switching, and f_{clk} is the clock frequency. The total capacitance C depends upon the circuit design and layout of each of the components of the IC; we calculate this using analytic models for regular structures as was done in CACTI [4], Wattch [7], and Orion [52], along with empirical models for random logic structures. The activity factor α indicates what fraction of the total circuit capacitance actually switches state during a typical clock cycle. We calculate α using access statistics from architectural simulation together with properties of the circuit, in the manner of Wattch [7] and PowerTimer [6].

The second term in the power model is the *short circuit power* that is consumed when both the pull-up and pull-down devices in a CMOS circuit are partially on for a small, but finite amount of time. Since the V_{dd} to V_{th} ratio , where V_{th} is the threshold voltage, shrinks as the technology scales, short circuit power is expected to become more significant in the future. It has been reported in [35][55] that the short-circuit power can be approximately 25% of the dynamic power in some cases. The equation for calculating $I_{short\_circuit}$ is complex and not reproduced here, but it is based on [35]. We use RC delays and rise/fall times from the timing model to determine the fraction of a clock cycle when both devices are on. Note that this is only possible because we have integrated both power and timing analysis into a common framework.

The third term in the power model is the *static* power consumption. Static power consumption is the result of *leakage* currents through the transistors, which in reality function as "imperfect" switches. There are two distinct leakage mechanisms, and for each type, the magnitude of the corresponding leakage current is proportional to the width of the transistor and depends on the logical state of the device-that is, whether the device is in the on or off state. The first type of leakage, *subthreshold leakage*, occurs when a transistor that is supposedly in the off state actually allows a small current to pass between the source and drain. We let I_{sub} denote the subthreshold through a unit-sized (off) transistor. The second type of leakage current, *gate leakage*, is current that leaks through the gate terminal. Gate leakage current when the device is on (I_{gon}) differs greatly from the gate leakage current when the device is off (I_{goff}). We determine the unit leakage currents from the ITRS [43] using MASTAR [42], and the approach described by Intel in [19].

In order to model the leakage current for a circuit block with many transistors, we need to consider which logical state each transistor is in, and accordingly sum up the leakage current components for each. If a circuit block is in some logical state s, we can express the *effective width* of all (off) transistors exhibiting subthreshold leakage in that state as $W_{sub}(s)$; similarly, we can express the effective widths for gate leakage for on and off transistors as $W_{gon}(s)$, and $W_{goff}(s)$. If Pr(s) is the probability that a circuit is in state s, we can express the total leakage current for a given block as the average leakage current over all possible states S, as shown in Equation (2):

$$I_{leakage} = \sum_{s=1}^{S} Pr(s)[W_{sub}(s)I_{sub}+W_{gon}(s)I_{gon}+ W_{goff}(s)I_{goff}] \qquad \text{(A-2)}$$

For simple logic blocks or for memory structures such as caches and register files, we estimate the average effective widths of a circuit from the results of a cycle-accurate simulation. For more complex circuit blocks where it is not practical to keep track of the internal logical state, we estimate the effective widths using representative values from existing designs. By adding up the average static power of all circuit blocks, we obtain the total static power for the whole chip.

8.1.3 Timing Modeling

Like the power model, McPAT's timing model breaks the system down into components and stages. While the power model requires only the capacitance to compute dynamic power, the timing model uses both resistance and capacitance to compute RC delays, using an approach similar to CACTI [4] and [37]. McPAT uses the delay information from each component to calculate short-circuit power consumption as described above, and determines the clock frequency from the delays of components along the critical path.

8.1.4 Area Modeling

McPAT's area model is fully integrated with the timing and power models as part of the hierarchical methodology, ensuring that calculations for all three are electrically consistent. Area models for basic logic gates and array structures is based on CACTI [4], with our enhancements. Buffered wire models are based on work from Ho [17] and Kumar, et al. [29]. Models for NoC links and routers use an approach similar to [29], [48], and Orion 2 . While an algorithmic approach to area modeling is accurate for regular structures such as those listed above, it does not work for complex structures that have custom layouts, such as ALUs. For structures such as these, McPAT uses an empirical approach similar to [40][44], using curve fitting to build a numerical model for area from published information on existing processor designs, and then scaling the area for different target technologies.

8.2 Hierarchical Modeling Framework

McPAT's integrated power, area, and timing models are organized in a three-level hierarchy, as illustrated by Figure 21: Modeling methodology. The highest level is the *architectural/microarchitectural level*, where a multicore processor is decomposed into major architectural components such as cores, NoCs, caches, memory controllers and clocking. The second level is the *circuit level*, which consists of set of four basic circuit structures-hierarchical wires, arrays, complex logic, and clocking networks-into which the architectural building blocks are mapped. The lowest level is the *technology level*, where the physical parameters of devices and metal wires-such as unit resistance, capacitance, and current densities-are calculated using technology data from the ITRS roadmap [43].

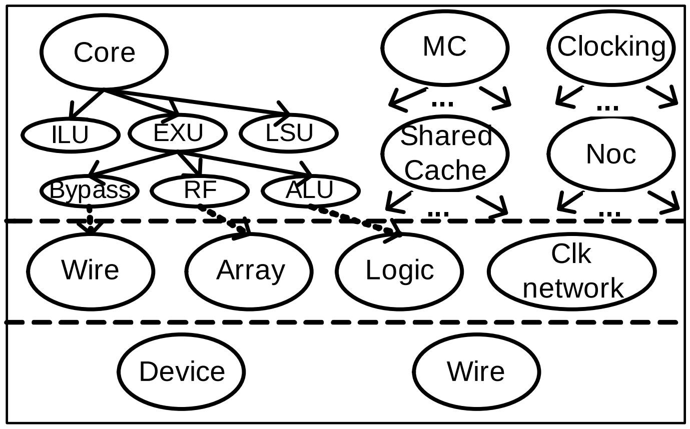


Figure A-21: Modeling methodology

8.2.1 Multicore Architecture and Microarchitecture Level Modeling

The top level of the model hierarchy is the architectural/microarchitectural level, which represents the basic building blocks of a multicore processor system. Below, we provide an overview of the models for these high-level blocks and how they are mapped to the circuit level. Further details may be found in a technical report [5].

Core: A core can be divided into several main units: an instruction fetch unit (IFU), an execution unit (EXU), and a load and store unit (LSU). Each of these units can be further divided into hardware structures. For example, the EXU may contain an ALU, FPU, bypass logic, and register files. In our hierarchical framework, the ALU and FPU are mapped to the complex logic model at the circuit level. Bypass logic can be mapped to the combination of wire and logic models, while register files can be mapped to the array model.

NoC: A NoC has two main components: signal links and routers. Routers are modeled in a manner similar to Orion 2.0, with signal links mapped to hierarchical wires at the circuit level. We also provide a model for a double-pumped crossbar [49].

On-chip caches: There will typically be a hierarchy of caches on chip. McPAT supports both private and shared caches. Shared caches that back up private caches may also have coherence support. We model coherent caches by also modeling the directory storage associated with each bank of the cache. Depending on the architecture, a directory can be mapped to CAM structures at the circuit level as in Niagara and Niagara2 processors [20][27] or to memory structures.

Memory controller: In order to provide sufficient bandwidth, memory controllers are now commonly integrated onto a processor die, and McPAT provides models for these. McPAT's on-chip memory controller model is based on the design for the AMD Opteron [3], as well as [16]. A memory controller contains four main hardware structures: input/output data buffers for temporarily storing the data to/from the main memory, request queues for storing the memory requests from the CPU to exploit the pipelined memory interface, arbiter/schedulers for rescheduling the memory requests according to some schedule policy, and memory channels for communicating off-chip to memory. It is especially important to model the large output drivers of the memory channels, which consume significant power. McPAT sizes these drivers according to DDR2 and DDR3 electrical requirements.

Clocking: Clocking circuitry has two main parts: the phase-locked loop (PLL) to generate the clock signal and the clock distribution network to route the clock. Multicore processors may be viewed as systems-on-a-chip with multiple clock domains. Different clock frequencies for different clock domains can be generated using a single PLL with fractional dividers as in Niagara 2 [34] and Intel Itanium processors [36]. McPAT uses an empirical model for the power of a PLL and fractional divider, based on scaling published results from Sun and Intel [2][9][53]. The clock distribution network can be directly mapped to the clock network model at the circuit level.

8.2.2 Circuit Level Modeling

Hierarchical Repeated Wires and 3D TSVs: Hierarchical repeated wires are used to model both local and global metal wires on-chip as well as the wire and buffer arrays in memory channels. Performance of wires is governed by two important parameters: resistance and capacitance. We model plain wires using a one-section π-RC model as described in [17]. Wires actually scale more slowly than gates with respect to RC delays, and unbuffered plain wires cannot keep up with the improved transistor delay. Our buffered wire model is based on Ho [17], and we optimize sizing parameters for energy-delay product rather than only for speed. High density through silicon vias (HDTSVs) are also modeled so that the 3D circuit modeling can be supported. HDTSVs are modeled in a manner similar to wires, but with different technology parameters, according to the 2007 ITRS roadmap [41].

Our hierarchical wire model assumes 10 metal planes, with 4 layers for local interconnect, and 2 each for intermediate, semi-global, and global interconnect. Wires on different planes have different pitches and aspect ratios. As a result, the assignment of signals to wiring planes plays a key role in determining power, area, and timing characteristics. McPAT's optimizer automatically assigns wires to planes to achieve specified objectives. First, our model optimizes the buffers for each wiring plane. Then, the delays of buffered wires are calculated for the first appropriate layer in the wiring plane hierarchy. If the current wring layer cannot satisfy the delay constraints, the wires are moved to a higher wiring plane for lower latency, at the expense of a higher power and area penalty. The effective wire pitches on semi-global and global planes are also increased from the minimum width to satisfy the target latency as described in [29]. Latches are also inserted and modeled when necessary to satisfy the target clock rate.

Arrays: McPAT includes models for three basic array models at the circuit level: RAM, CAM, and DFF (D flip flop)-based arrays. The models for these structures are based on CACTI 5.3, with several important extensions. McPAT revises the CAM model from CACTI to more accurately reflect its bank structure and also adds a write operation, which is needed to model replacing entries in a TLB, for example. Second McPAT adds a DFF array model similar to that used in Orion 2.0, but with the added capability of modeling timing. Finally, McPAT adds gate leakage and improved timing models for arrays.

Logic: McPAT employs three different schemes for modeling logic blocks, depending on the complexity of the block. For highly regular blocks with predictable structures, such as memories or networks, McPAT uses an algorithmic approach similar to CACTI [4]. For structures that are less regular but that can still be parameterized, such as thread selection or decoding logic, McPAT uses analytic models similar to those in [37] that are modeled after the Niagara processors [45]. Finally, for highly customized blocks such as functional units, McPAT uses empirical models based on published data for existing designs scaled to different technologies, in a manner similar to Wattch [7]. For example, the ALU and FPU models are based on actual designs by Intel [33] and Sun [30].

Clock distribution network: A clock distribution network is responsible for routing clock signals of different frequencies to clock domains, with drops to individual circuit blocks. It is essentially a special case of a hierarchical repeated wire network, but because they have strict timing requirements, with large fanout loads spanning the entire chip, and consume a significant fraction of total chip power [15], we represent a clock distribution network using a separate circuit level model.

We use a clock distribution network model similar to that in [8] that uses three distinct levels: global, domain, and local. The global-level network routes clock signals to different clock domains across the whole chip, including cores, NoCs, caches and I/O channels. The domain-level network routes clock signals within the clock domains to different hardware blocks, while the local clock network routes clock signals within the building blocks. Both the global- and domain-level clock networks can be routed on the same metal plane. We assume an H-tree topology for global-level and domain-level networks and a grid topology for the local network, in the manner of the clock distribution schemes in both the Sun Niagara processors [27][34] and the Intel multithreaded Itanium processor family [11][38]. Following Ho's approach [17], we insert tri-state buffers at all final clock heads to enable clock gating.

8.2.3 Technology-Level Modeling

The lowest level models in McPAT are technology models for devices and wire interconnect. McPAT takes an approach similar to CACTI [4], using the MASTAR software tool (Model for Assessment of CMOS Technologies and Roadmaps) [42] to derive device parameters from the ITRS 2007 roadmap [43], with additional parameters to support Ho's wire projections [17]. The current implementation of McPAT includes data for the 90 nm, 65 nm, 45 nm, 32 nm, and 22 nm technology nodes, which covers the ITRS roadmap through 2016.

Devices: McPAT includes models for each of the three device types defined by the ITRS-High Performance (HP), Low Standby Power (LSTP), and Low Operating Power (LOP)-for each technology node. The HP transistors have short gate lengths, thin gate oxides, low Vth, relatively high VDD, and high on-currents, making them fast but also very leaky. The LSTP transistors have longer gate lengths, thicker gate oxides, and higher Vth and VDD, thus trading lower on-currents and hence slower speeds for reduced leakage. The LOP transistors have gate lengths between the HP and LSTP devices and use the lowest VDD, and thus are somewhat slower than the HP transistors but have lower dynamic power consumption. The ITRS assumes that planar bulk CMOS device will reach practical scaling limits at 36 nm, at which point the technology will switch to silicon-on-insulator (SOI). Below 25 nm, the ITRS predicts that SOI will reach its limits and that double-gate DG devices will be the only option. McPAT captures each of these options, making it scalable and evolvable with the ITRS through the end of the roadmap.

Wires: As is done in CACTI [4], McPAT uses a combination of parameters from the ITRS [43], together with parameters from Ho's projection models [17]. Ho considers both aggressive (optimistic) and conservative (pessimistic) assumptions regarding interconnect technology, with respect to the use of low-k dielectrics, effects of resistance degradations from dishing and scattering, and tall wire aspect ratios.

8.3 Modeling Power-saving Techniques

McPAT models the two major power saving techniques: clock gating to reduce dynamic power and power gating to reduce static power. This enables the modeling of C-states in modern multicore processors. Clock gating effectively separates inactive components from the clock distribution network; McPAT models the actual gating buffer circuitry at the distribution network heads, rather than using a heuristic approach as Wattch [7] did. McPAT is the first architectural modeling tool to include power gating, which is used in many modern multicore processors such as the Intel Core 2 Duo [18]. Our framework models power-gating with multiple sleep modes as described in [1]. The circuit uses footer devices and adjustable gate bias of the footer device to achieve four operating modes: Active, Sleep, Dream and Snore. Having different sleep modes allows tradeoffs between the power-savings and the wakeup overhead with respect to both wakeup power and wakeup delay. For example, dream mode can save 50% more static power than sleep mode, but at the expense of twice the wake delay and three times the wakeup energy. The user can specify power-gating modes for various components.