



Invited Talk

Application Specific Architectures: A Recipe for Fast, Flexible and Power Efficient Designs

Chris Weaver, Rajeev Krishna, Lisa Wu, Todd Austin
Advanced Computer Architecture Lab
University of Michigan
Ann Arbor, Michigan
{chriswea, rkrishna, wul, austin}@eecs.umich.edu

Abstract

The general purpose processor has long been the focus of intense optimization efforts that have resulted in an impressive doubling of performance every 18 months. However, recent evidence suggests that these efforts may be faltering as pipelining and ILP processing techniques experience diminishing returns. Application specific architectures hold great potential as an alternative means to continue scaling application performance. The approach works by specializing a design to a small domain of important applications, and it benefits from improved performance, greater power efficiency, and reduced area costs. This technique is well matched to embedded targets, where application domains are typically narrow.

In this paper we present a case study of an application specific processor design. Our design, called the CryptoManiac processor, is an architecture specialized to efficiently execute cryptographic ciphers. We carefully highlight the domain specific application characteristics we identified and their accompanying optimizations in the CryptoManiac design. Detailed analyses of the design makes a strong case for application specific optimization. The CryptoManiac processor runs popular ciphers at twice the speed of a high-end general purpose processor, and the design renders nearly two orders-of-magnitude reduction in power consumption and area costs. Finally, we identify two key challenges that stand as barriers to wide spread adoption of application specific architectures.

1 Introduction

For more than three decades, architects have lavished attention on the design and optimization of general purpose processors. During this time, significant advances have been made in their organization and implementa-

tion. Today, advanced techniques such as superpipelining, superscalar execution, dynamic scheduling, multi-level memory caching, and aggressive speculation have become commonplace in designs. These optimizations combined with fabrication technology improvements have resulted in a steady doubling of processor performance every 18 months, a trend referred to as Moore's Law.

There is a growing body of evidence suggesting that general purpose processor optimizations are diminishing in value. A recent study by Agarwal *et. al.* examined the scalability of future general purpose processor designs [1]. The study observes that general purpose processor optimizations come in two flavors: increased clock speed and higher instruction throughput. The combined strength of these optimizations have lead to the impressive performance gains experienced by the industry. Agarwal points out that clock rate improvements from pipelining must soon diminish because current designs have little logic within pipe stages. As such, latch delay and clock skew will soon dominate the clock period. Current designs have approximately 10 fanout-of-four gate delays, leaving little logic that can be bisected to produce higher clocked rates. While it is conceivable that increased instruction throughput could make up for clocking deficiencies, additional evidence shows diminishing return on optimizations to increase instruction throughput. For example, the recently released Pentium 4 microprocessor attains only about 80% the instruction throughput of a Pentium III processor (measured in SPECInt/Mhz for the same technology) [2]. If architectural optimizations remain anemic, a primary source of value in the computer industry, namely performance, could be threatened.

As illustrated in Figure 1, it is possible to improve both the performance and efficiency of a design through specialization. It is well established that producing a dedicated H/W implementation of a specific algorithm can render large improvements in performance, increased power efficiency, and reduced area costs. The approach works by specializing an implementation to the specific needs of an algorithm, and by eliminating all other aspects of a design that would otherwise serve more general forms of computation. The drawbacks with a dedicated H/W design is an increased marginal design cost due to smaller production volumes and reduced

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CASES'01, November 16-17, 2001, Atlanta, Georgia, USA.
Copyright 2001 ACM 1-58113-399-5/01/0011...\$5.00.

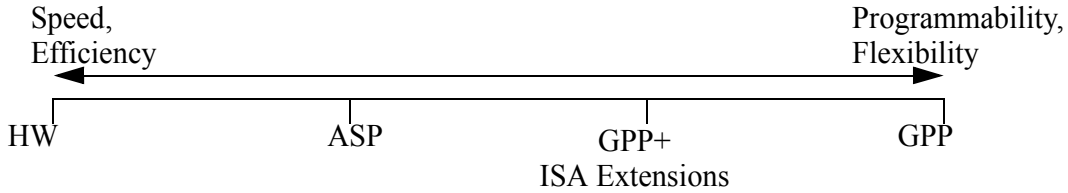


Figure 1 Architectural Design Space

design flexibility because the H/W implementation cannot be changed after it has been manufactured.

Two important design points lie between the extremes. The general purpose processor with instruction set extensions (GPP+ISA) incorporates instruction-level enhancements to service a particular application domain. An example of this type of optimization is Intel’s MMX extensions [3] or Sun’s VIS extensions [4], both crafted for multimedia applications. The application specific processor (ASP) is a more aggressively optimized design incorporating hardware optimizations into the microarchitecture, in addition to ISA extensions. Both approaches share the flexibility of a programmable core that permits upgrades to be installed in the field, and both designs experience performance and efficiency by specializing portions of their design to a specific application domain. The key difference between these designs is in their dedication to serving general purpose workloads. Instruction set enhancements provide some benefits, but not to the degree that is possible by tuning a microarchitecture to an application. Application specific processors have better overall performance for a domain, but at the expense of general purpose program performance. Embedded targets are a particular good match to this approach. The restricted function of typical embedded devices allows for a good deal of refinement in both the ISA and the microarchitecture.

Application specific processors are by no means a new idea, with several companies already providing tailor made designs to meet their customers specific demands. For example, Tensilica’s Xtensa architecture provides easily customizable cores for embedded solutions. On EEMBC’s ConsumerMark benchmark a customized 200 MHz Xtensa outperformed a 550 MHz AMD K6 by 5.8 times [5]. HP is developing a similar technology called PICO, Program In Chip Out. Their technology takes a specific program and produces an application specific VLIW processor [6].

In this paper, we present a case study of an aggressively optimized application specific processor design. Our design, called the CryptoManiac, is a simple architecture optimized to efficiently execute cryptographic cipher kernels. We carefully detail the cryptographic workload characteristics that we identified, and the optimizations implemented in the CryptoManiac that take advantage of these characteristics. The application specific optimizations implemented in CryptoManiac have

rendered significant improvements in design performance, power efficiency and area cost while retaining a flexible programmable design with field upgradability. Finally, we list the challenges that must be addressed before application specific architectures can play a significant role in scaling application performance.

2 A Case Study: CryptoManiac

The CryptoManiac processor is an application specific architecture focused solely on efficient execution of cryptographic cipher kernels [7,8]. In this section, we give an overview of the CryptoManiac architecture and then detail the application specific optimizations that it leverages.

2.1 CryptoManiac System Architecture

Figure 2 details the high level architecture of the CryptoManiac (CM) processor. The host processor interfaces to the CM through the input (InQ) and output (OutQ) request queues. Cryptographic processing requests are inserted into the InQ by a host processor over a connecting bus. The request scheduler distributes host processor requests, in the order received, to CM processing elements. The CM processing elements service requests from the InQ, placing any results produced in the OutQ for the host processor. In a typical application, the input and output queues would be accessible by the host processor via a standard bus interface, such as the PCI bus.

The CM processing elements block waiting on a request from the host processor. When a request is dispatched to the CM processor, it uses the request code to initiate the correct handler function in CM instruction memory. CM processing elements support a range of requests for key management, encryption, decryption, and CM

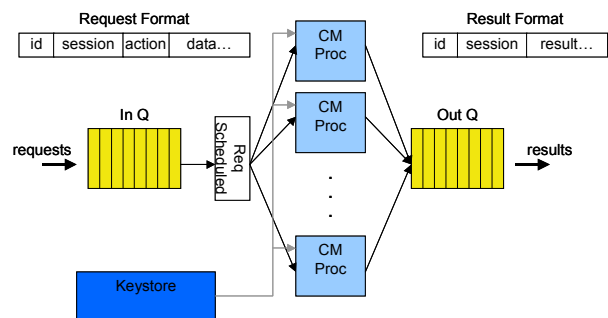


Figure 2 CryptoManiac System Architecture

administration. When CM processing is complete, the result of the computation is pushed into the OutQ for reception by the host processor. The host processor tags requests with a unique ID that it can use to identify requests as they complete and exit the OutQ. The unique ID permits requests to complete out of order as may be the case with varied processing demands on CM processors. CM requests also contain a session identifier, assigned by the host processor, that identifies the communication channel from which the request has been issued.

Requests arriving for CM processing are dispatched from the InQ to CM processing elements by the request scheduler. Requests are distributed first to a free CM processor. If multiple CM processors are free, the request is dispatched to the CM processing element that most recently processed a request in the same session. If there are no free CM processors in the same session, the request is assigned to the least recently used CM processor. Directing requests to CM processors in the same session reduces the setup time to service the request, and when multiple CM processors are free but not in the same session, the least recently used CM processor is likely to contain an unneeded session context.

The *keystore* is a high-density storage element that contains key-specific storage such as key data and substitution tables. The keystore permits the CryptoManiac to process simultaneous sessions on the same CM processor by storing key-specific data in the shared keystore. When a new context is loaded into a CM processor, key specific data is transferred over a high performance interface to internal CM storage. This data includes substitution data, permutation counters, and other internal cipher state.

2.2 CryptoManiac Processing Elements

Each CM processing element is a simple 4-wide 4-stage VLIW processor as shown in Figure 3. The front-end of

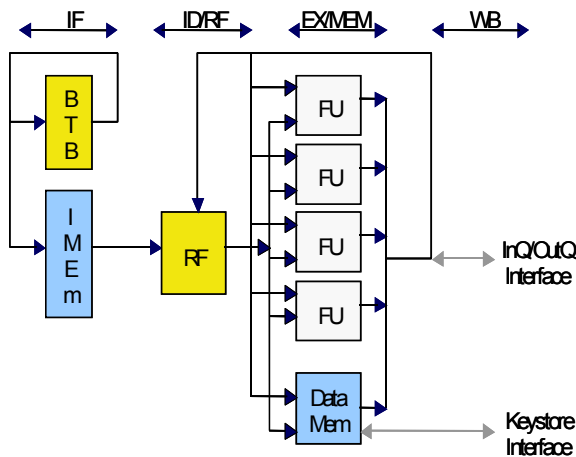


Figure 3 CryptoManiac Architecture

the CM pipeline contains a simple branch target buffer (BTB) used to predict branch targets. The BTB contains the target addresses of any branches. Any instruction that hits in the BTB is predicted taken. A VLIW instruction word is read from the CM instruction memory and accessed in parallel with the BTB probe. A VLIW instruction word contains four independent instructions (or NOPs if independent instructions are not available). During instruction decode, individual VLIW instructions access the register file. In the EX/MEM stage, instruction operations are executed, including loads and stores. The execute stage includes four functional units, each capable of executing arithmetic and logical operations, rotates, multiplication, or substitutions. A single data memory unit is available to execute loads and stores.

CM instructions have three inputs, one output, and support execution of up to two operations in a single cycle, a technique termed *instruction combining*. The CM instruction combining architecture divides operations into three classes: tiny, short, and long. Tiny operations include all logical operations and sign extension; short operations include arithmetic operations, rotates, and substitutions; multiplies are classified as long operations. Functional units contain datapath networks that allow any pair of tiny operations or short/tiny operations to execute together in a single cycle. As such, the CM allows general arithmetic instructions to combine with logical instructions, substitutions to combine with logical instructions, and finally rotate operations to combine with logical instructions.

2.3 Cryptographic Specific Optimizations

The CryptoManiac crypto-processor is significantly more efficient than a general purpose microprocessor design. In this section we list the specific program characteristics of cryptographic workloads that we identified, and the application specific optimizations that were applied to the CryptoManiac that rendered performance, power, and area improvements. We order the optimizations from least to most aggressive.

Application Domain Characteristic: Cryptographic workloads have well behaved data sets. Working set sizes are very small, and they exhibit high access locality.

Application Specific Optimization: A simple, small memory system suffices. The CryptoManiac contains a 512 byte instruction RAM, and a 4k byte data RAM. Caches or local DRAM is not required. This results in memory system design with fast deterministic access latency, low power consumption, and an area efficient implementation.

Application Domain Characteristic: Cryptographic workloads have few branches, and they are easy to predict. Most branches are used to implement the cipher kernel loop, and hence, are nearly always taken.

Application Specific Optimization: A simple, low-cost branch predictor suffices to eliminate control hazards in the CryptoManiac pipeline. The branch predictor is a simple branch target buffer (BTB), that is trained after the first branch. If a branch is encountered in the BTB, it is assumed taken. The design is fast, power frugal and area efficient.

Application Domain Characteristic: Cryptographic workloads are typified by simple kernels with well described dependencies and deterministic instruction latency.

Application Specific Optimization: The CryptoManiac microarchitecture can employ a simple VLIW architecture and a simple blocking scheduler. In addition, kernels are optimally scheduled through the use of a superoptimizing scheduler that is capable of examining all possible cipher kernel schedules, and choosing the best one given a set of microarchitectural constraints and performance goals.

Application Domain Characteristic: Cryptographic workloads feature heavy use of byte substitution operations. These operations implement a key-based function by extracting a byte from a code block, indexing a key-based substitution table, and then reassembling the block with the replaced byte.

Application Specific Optimization: A substitution instruction and a fast substitution functional unit are added to efficiently implement these operations. The CryptoManiac substitution instruction implements a byte substitution within a cipher block in one instruction, an improvement over the four instructions required to implement the same operation in the Alpha ISA. The optimized design eliminates address calculation by restricting all substitution tables to start on a page boundary, thus address generation into the table simplifies to simple bit concatenation. Byte extraction and reassembling operations are supported directly in hardware. The resulting optimized operation is faster, and reduces power by eliminating address calculation and multiple instruction fetches.

Application Domain Characteristic: Some cryptographic ciphers rely heavily on modular multiplication, a particularly powerful operation for bit diffusion.

Application Specific Optimization: A modular multiplication instruction and fast functional unit is added to implement this operation. Modular multiplication operations are implemented using one regular 16-bit multiplication followed by two 16-bit parallel additions and two levels of MUX'ing. This algorithm is derived from the Chinese remainder theorem detailed in [9]. Modular multiplication can be completed in just under three cycles.

Application Domain Characteristic: While typical ciphers have small amounts of instruction-level parallelism, it is possible to extract inter-session parallelism. This parallelism exists between cipher streams in different sessions. Since they are independent of each other,

and processed with different keys, the streams can be processed in parallel.

Application Specific Optimization: CryptoManiac can be implemented as chip multiprocessor to extract this parallelism. A central input scheduler directs requests to CM elements. There is locality in the stream contexts, attempting to keep a stream and its state (key tables) on the same CM processing element for as long as possible results in decreased CM initialization costs.

Application Domain Characteristic: Cryptographic workloads make heavy use of linear and non-linear instruction pairs. Arithmetic operations, such as additions and subtractions are often followed by logic operations, such as XORs, ORs or ANDs. The pairing provides effective bit-level diffusion while at the same time improving the strength of the cipher by reducing its susceptibility to linear cryptanalytic attacks. In addition, cryptography techniques such as whitening, chaining, and combining commonly pair arithmetic and substitution operations with XOR's, yielding similar instruction execution patterns.

Application Specific Optimization: CryptoManiac's cycle time is designed around the pairing of arithmetic/substitutions operations followed by a logical operation (XOR/OR/AND). The resulting design benefits from improved utilization of the processor clock cycle (no short XORs wasting most of a cycle), improved efficiency due to much fewer instructions fetched, and improved program density.

Application Domain Characteristic: Virtually all applications of crypto-processors are within communication environments where protocols employ low-cost correctness check mechanisms, such as cyclic redundancy codes (CRCs), to detect transmission errors.

Application Specific Optimization: We have begun experimenting with self-tuned circuit implementations of CryptoManiac. The technique couples a traditional core processing element with a checker unit that monitors the integrity of all core computation. The core accepts ciphertext packets and processes them using a computationally intensive cipher algorithm. The checker verifies the integrity of the core processor cipher processing by checking the CRC of the resulting plaintext. Control system software running on the checker increases frequency while monitoring system temperature and checker error rates until circuit performance is maximized. If the control system over steps the bounds of correct operation, the checker unit will detect the error, re-tune the system to a safe operating point and re-execute the operation on the core processor to recover the operation.

We evaluated the efficiency of the CryptoManiac through simulation-based analyses and timing analyses of detailed physical designs. Our design is extremely efficient; a 4-wide 360MHz CryptoManiac design implemented in a 0.25um CMOS technology with instruction combining runs Rijndael, the new AES

cipher standard, 2.25 times faster than a high-end Alpha 21264 based server. In addition, the design reduces area costs and power dissipation by nearly two orders of magnitude, compared to the Alpha 21264 in the same CMOS technology. To demonstrate the viability of self-tuning circuits, we have constructed a prototype self-tuned crypto-system using commercial off-the-shelf StrongARM SA-1110 microprocessors, which support frequency tuning across a wide range of frequencies (including overclocking). We currently have our testbed working with a rudimentary tuning control system. Passively cooled, self-tuned circuits provide an additional 26% increase in cipher processing speeds.

3 Conclusions and Future Challenges

We believe the CryptoManiac design makes a strong case for application specific architectures. The reason for its striking efficiency is simple - an application specific processor design can achieve a level of efficiency that is impossible for general purpose designs to obtain. Our application specific design contains none of the baggage necessary to execute non-cryptographic workloads, making the resulting design smaller and cooler. In addition, our limited application domain creates opportunities to optimize the implementation, yielding superior performance. If general purpose processor optimization derails from the trajectory defined by Moore's Law, application specific architectures hold great potential to provide 1-2 generations of additional performance in the same technology, while giving even larger power and cost benefits.

Two key challenges lie between today and realizing the potential of application specific architectures. The first challenge is to identify the scope and constituents of workloads for these designs. Some of these domains are obvious, such as graphics processing, signal processing, and cryptographic processing. Others domains certainly exist. Choosing the correct scope of these application domains will require a careful balance between specialization for performance and power and generality for economies of scale.

The second challenge is to reduce the design burden associated with application specific architectures. The lower the cost of producing a design, the more narrow its application scope can be, and the greater the potential benefits that it can render. Solutions to overcome this challenge, will require tools to assist in design and verification of hardware and software..

Going forward, we are working to assess the cost of programmability in the CryptoManiac. A dedicated Rijndael implementation is under development that will be compared to the design presented in this paper. In addition, we are developing application specific processors for other application domains. Through this work we hope to gather the experience necessary to begin crafting design tools.

References

- [1] V. Agarwal, M.S. Hrishikesh, S. W. Keckler, and D. Burger, "Clock rate versus ipc: The end of the road for conventional microarchitectures," In the 27th Annual International Symposium on Computer Architecture, May 2000.
- [2] Pentium 4 Productivity Benchmarks, Intel Corporation, <http://www.intel.com/procs/perf/pentium4/productivity/specint2000.htm>
- [3] Millind Mittal, Alex Peleg, and Uri Weiser, "MMX Technology Architecture Overview," Intel Technology Journal, Q3'1997.
- [4] SPARC V9 Architecture Book, www.sparc.org
- [5] L. Gwennap, "Tensilica shows speed," EETIMES.COM, March 7, 2001.
- [6] A. Wolfe, "HP lays foundation for embedded's future," EETIMES.COM, March 7, 1999.
- [7] L. Wu, C. Weaver, T. Austin, "CryptoManiac: a fast flexible architecture for secure communication," In the 28th International Symposium on Computer Architecture, June 2001.
- [8] Jerome Burke, John McDonald and Todd Austin, "Architectural support for fast symmetric-key cryptography," In the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Sept 2000.
- [9] Xuejia Laai, "On the Design and Security of Block Ciphers," Hartung-Gorre Verlag, 1992.