Modeling and Analysis of Power in Multicore Network Processors

S. Huang[1] Y. Luo[2] W. Feng[1]

[1]Department of Computer Science
Virginia Tech
Blacksburg, VA 24060
{huangs,feng}@cs.vt.edu

[2] Dept. of Electrical and Computer Engineering
University of Massachusetts Lowell
Lowell, MA 01854
yan_luo@uml.edu

Abstract

With the emergence of multicore network processors in support of high-performance computing and networking applications, power consumption has become a problem of increasing significance. Lower-power multicore processors, such as the Intel IXP network processors, have been employed in network devices to try to address this problem; however, effective tools are still needed to assist in the exploration of the "performance versus power" design space.

In this paper, we present a composite power model that simultaneously leverages three existing power-modeling tools (Cacti, Wattch, and Orion) to model the power consumption of Intel IXP2400. This model is then integrated with an open-source Intel IXP2400 NP simulator called NePSim2 and validated against its datasheet to within 5% of actual power consumption. We utilize the simulator and power model to explore how architectural parameters affect the performance and power consumption of a set of high-performance security applications, thus delivering valuable insights to chip architects for designing energy-efficient multicore network processors.

1 Introduction

Modern general-purpose microprocessors have recently incorporated multiple cores on a single chip in order to achieve scalable performance under a stringent power budget due to thermal and packaging requirements. Such multicore architectures have long been present in network processors (NPs). While NPs have their roots in embedded systems, recent trends point to the use of NPs in high-performance network cards in order to support network co-processing for high-performance computing (HPC) and real-time intrusion detection for high-performance network security [17].

NPs have even more cores on a chip than general-purpose processors to exploit packet-level parallelism. In addition, these programmable cores typically support hardware-based multithreading to hide memory latency. Such optimized multithreaded cores enable high-speed packet processing; this capability, combined with its programmability, distinguishes NPs as the core components in the next generation of high-performance routers and switches.

At the same time, power consumption has become a major concern in designing high-performance computer systems, including high-performance network devices, where the workload of processing network packets continues to increase in complexity. For example, it is now common for network routers to accommodate a large variety of protocols and carry out complex tasks, such as encryption and decryption for a virtual private network (VPN) [9].

In addition, network line rates at network endpoints are moving towards 10 Gbps and beyond. Consequently, the frequency and density of NPs has necessarily increased to keep pace, leading to increased power consumption and potential thermal dissipation issues. Specifically, the Intel IXP2800 NP contains 16 microengines (packet-processing cores), each operating at 1.4 GHz, and consumes 19-25 watts of power [6], while its predecessor, the IXP1200 NP, contained only six microengines operating at 232 MHz and consumed only 4.5 watts. In addition, typical routers mount a few racks containing groups of line cards (e.g., 8), each of which contains one or two NPs. Such routers are extremely dense in power dissipation (e.g., 375 watts per line card [2] and 200 watts per packet-processing module [15]), resulting in higher operating temperatures and requiring aggressive cooling facilities.

Both processor architects and system designers face the tradeoff between performance and power. The common goal is that the required performance should be guaranteed within a power budget and to do so without over-architecting the processor. It is highly desirable that this early exploration of the design space is done *before* the implementation of the system. In responding to the need of studying the performance and power consumption of multicore network processors, NePSim [11] was designed to model the performance and power of the Intel IXP1200 NP as a first step in this direction. With the release of the IXP2400/2800 family of NPs, NePSim necessarily evolved into NePSim2.0. However, until now, the power model in NePSim2.0 was still missing because the power model from NePSim could not be used for NePSim2.0 due to the dramatic architectural changes between the IXP1200 and IXP2400.

Thus, this paper focuses on the construction and evaluation of a new power model for the Intel IXP2400 NP. We then incorporate NePSim2.0 with this power model and use the performance-power model to explore the design space of network processors with a set of cryptographic applications. We observe that the simulation framework can provide valuable insights on the design of network processors. The specific contributions of this paper are as follows:

- Architectural analysis of the major components of the IXP2400 that contribute to the overall power consumption.
- Power models of the identified components employing existing power-modeling tools.
- Validation of the power models against Intel NP power data.
- Power evaluation of cryptographic benchmarks and micro-benchmarks.
- Performance-power analysis using the power model to guide the design of programmable network processors.

The remainder of the paper is organized as follows. Section 2 outlines the current research on the optimization of power dissipation. Section 3 then describes the technique and approach that we employed in our power modeling of Intel IXP2400. Section 4 presents how we validate our power model and simulation followed by some analysis. At the end of that section, we utilize the performance-power simulation tool to explore the design space of multicore network processors. Finally, we provide some concluding remarks in Section 5.

2 Related Work

Power is a critical concern in NP design from the modeling of performance-power tradeoffs to addressing static and dynamic power. With respect to the former, Franklin and Wolf developed an analytic performance-power model for typical NPs, explored the design space of NPs, and showed performance-power tradeoffs for different core and memory configurations [3]. With respect to the latter, many power-reduction techniques, both static and dynamic, have been proposed for modern NPs, as described below.

To address static power, researchers have focused on lowering the power for individual components in a NP. Kaxiras et al. proposed a set-associative memory approach for efficient IP lookup called IPStash to replace ternary content addressable memories (TCAMs). This approach effectively reduces the set associativity of TCAMs in order to signficantly reduce power consumption [8]. Mallik and Memik investigated the optimal operating frequency of data caches with the goal of reducing energy and improving performance but at the slight expense of reliability. This approach was justified by observing that errors in NPs could be fixed by higher levels of the network protocol stack [13]. In addition, techniques have been proposed to include dual threshold voltages (dual-Vt) and stacked transistors. With the dual-Vt technique [16], low-Vt devices are used in the critical path of a design while high-Vt devices are used to reduce the leakage in the non-critical parts of the design. Stacked transistors [20] reduce leakage through transistor stacks to maximize the number of transistors that are “off” during the idle mode.

To reduce dynamic power, many approaches have been proposed to reduce the NP’s switching activities, voltage and capacitance. Luo et al. proposed a clock-gating technique to reduce the power of NPs under fluctuating network traffic [12]; their results show that power reduction can reach up to 30%. Luo et al. [11] also presented how to apply dynamic voltage scaling (DVS) to reduce the NP’s power. Compared to clock gating, DVS incurs longer delay for adjusting the voltage and clock frequency.

Finally, thread migration and adaptive resource allocation are also common approaches towards improving power efficiency in NPs or general CMPs. For example, Kokku et al. present an analytical model for allocating the appropriate number of processors to each service on NPs [10]. However, their work made several assumptions to derive the estimate for the benefits in an ideal adaptation scheme.

3 Power Modeling of the Intel IXP2400 Network Processor

In this section, we describe our methodology for modeling the power consumption of the Intel IXP2400 network processor in NePSim2.0.

3.1 Overview of Methodology

The *power* model is built on the *performance* model of the IXP2400 that is implemented in NePSim2.0 [14]. NePSim2.0 is an open-source, cycle-accurate, execution-driven simulator that models the performance of IXP2400/2800 NPs. The simulator takes benchmark executables and packet traces as inputs, simulates the execution, and produces extensive performance statistics including access counters to individual functional units. As shown in Figure 1, we leverage interim results from the performance simulation and integrate our power model into the current release of NePSim2.0. Specifically, in the simulation, benchmark applications are preprocessed before being loaded into NePSim. The traffic generator generates the inputs to the loaded applications. When NePSim starts the execution-driven simulation, it begins recording the number of accesses to each individual unit on the NP chip. Our power estimator leverages the cycle-by-cycle access counters kept in NePSim and derives the power consumption of the NP.

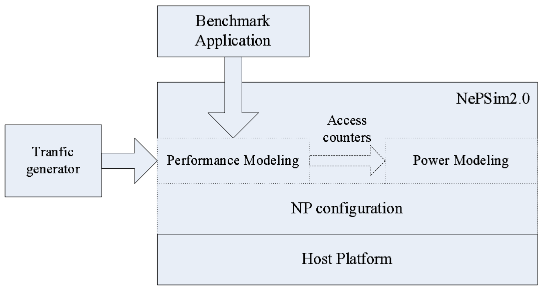

Figure 1. Power modeling infrastructure

3.2 Anatomy of the IXP2400 NP

Figure 2 presents the architecture of the IXP2400 NP. The main components shown in the figure are microengines (MEs or packet-processing elements), XScale core, memory controllers (one SDRAM channel and 2 SRAM channels), media switching fabric (MSF) and SHaC (short for scratch pad, hash unit, and control & status registers). The MEs are programmable processing elements used for packet processing, as depicted in more detail in Figure 3. In this paper, we denote architectural constituents like the MEs as "components" and denote the basic units that compose an individual component as "units" (e.g., control store or register file).

Of the architectural components in IXP NPs, we model the MEs, memory controllers (SRAM and DRAM controllers), media switching fabric (MSF), and SHaC. The XScale core and PCI controller are not simulated in NePSim2.0, primarily because they consume relatively a small amount of power. The XScale core handles primarily noncritical workloads such as logging and system initialization while the PCI controller is used only when the NP communicates with host CPUs, which is quite rare in standalone network systems. Thus, the XScale core and PCI controller are excluded from our power modeling. Although our current modeling implementation does not cover all the components, we validate the total power consumption by adding the power of unsimulated components to our simulated result.

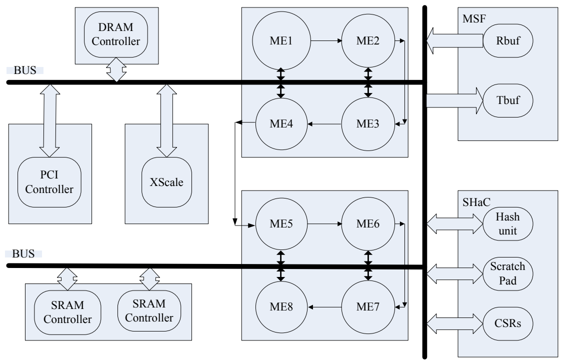

Figure 2. Architecture of the IXP2400

The IXP2400 NP contains eight identical microengines. Figure 3 shows the internal architecture of a microengine (ME). The datapath includes an ALU, a shifter, and several storage units. The control store keeps the instructions that the ME executes. The GPRs (General Purpose Registers), XFERs (Transfer Registers), and local memory serve as high-speed data storage, among which XFERs are used for SRAM and DRAM transactions. There are also 128 next-neighbor registers (not shown in Figure 3) in the datapath for direct communication between two neighboring MEs. Because the next-neighbor registers are structurally the same as GPRs, when we simulate their power, we simply treat them as GPRs. Due to page-length limitations, we refer readers to the IXP2400/2800 hardware reference manual [7] for more detailed information.

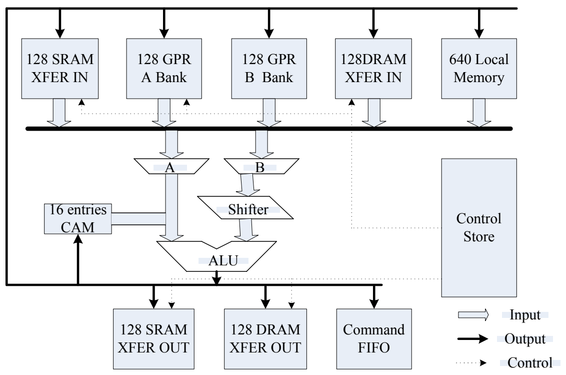

Figure 3. Architecture of a microengine

3.3 Power Modeling

We estimate the power consumption of each individual structural component in the IXP2400 at the architectural

Table 1. Model types and corresponding tools

Model type	Power-Modeling tool
register file	cacti
queue & CAM	wattch
array	cacti
arbiter	orion
ALU & shifter	wattch

level and sum them up to derive the total power consumption of the IXP2400. Similarly, we estimate the component power by summing up the power consumption of all its sub-level units. Furthermore, power consumption in digital circuits can be divided into static and dynamic power consumption. The static power is estimated with 2% of the peak power when assuming that all components are actively switched. We estimate the dynamic power by multiplying the number of voltage switches and the power consumption per voltage switch. In summary, our power modeling can be represented by the formula below.

$$P_t = \sum (P_s + P_d * C)$$

where P_t stands for the total power consumption of the IXP2400, P_s is the static power of a unit, P_d is the dynamic power of a unit per access, and C is for the access count for that unit.

In our experiments, before the performance simulation starts, the power consumption of each unit per access is calculated. Then we leverage the performance simulation to obtain the total access count for each particular unit. Finally, we apply the formula above to calculate the total power consumption.

3.4 Tools for Power Modeling

In our power-modeling process, we leverage and integrate three existing power-modeling tools: Cacti [19], Wattch [1] and Orion [18]. We categorize the structural units into several types and summarize the chosen power-modeling tools for each type. Such choice of tools is driven by the characteristics of the tools and the nature of the components to be modeled. Cacti models caches and other SRAM structures. Register files and array structures like local memory and control store fall into this category. Wattch estimates the power of the queue structures, ALU, shifter, and content-addressable memory (CAM). Orion models the power of the arbiters for buses and thread scheduling. Table 1 summarizes how we modeled the basic units.

Table 2 describes the specific parameters of the power model for the ME. Tables for the SRAM controller, DRAM controller, SHaC, and MSF are omitted due to space limitations. In these tables, the first column presents a unit in a structural component; the second column shows the type of model; the third column shows the configurations. Table 2 lists the units in MEs including GPRs, command FIFO, and context arbiter. Each ME has two identical GPR register files: A and B. Each register file has one read/write port and contains 128 registers that are 32-bit wide. The command FIFO in Table 2 is a queue structure that is used to store the SRAM, DRAM, and CSR accesses issued from the ME before sending them to the target memory units. Thus, we use Wattch to model it, and its settings are 64 bits with 4 entries reflecting the specification of the command FIFO. The context arbiter schedules the hardware-based threads in the ME, applying a round-robin scheduling policy. It is modeled using Orion.

Table 2. Decomposition and power modeling of ME

Unit	Model type	Configuration
ALU	ALU	32 bits
shifter	shifter	32 bits
GPR	register file	2 128-entry files 1 read/write port per file
XFER	register file	4 128-entry files 1 read/write port per file
command FIFO	queue	4 entries * 64 bits
control store	array	4k * 40 bits instruction
local CSR	register file	16-entry file 1 read/write port per file
context arbiter	arbiter	round-robin
local memory	array	640 entries * 32 bits
CAM	CAM	16 entries * 32 bits

4 Experiments and Analysis

With the power model developed in Section 3 integrated into the cycle-accurate NePSim 2.0 infrastructure, we now describe the evaluation methodology of the effectiveness of our power model for the IXP2400 network processor (NP). In particular, we study the power model with typical cryptographic benchmark applications because of the importance of security applications in high-performance network systems. We tune the major architectural parameters of the processor to investigate the performance and power tradeoff in designing multicore network processors.

Table 3. Parameters of NePSim 2.0

Components	Frequency
Microengine	600 MHz
SRAM controller	200 MHz
DRAM controller	150 MHz
MSF	133 MHz
SHaC	300 MHz

4.1 Benchmark Applications

In our experiments, we use eight cryptographic benchmark applications that are ported to the IXP2400/2800 NPs [21]: `aes`, `blowfish`, `des`, `idea`, `md5`, `rc5`, `rc6`, `sha`. They are well-known cryptographic algorithms, thus detailed descriptions of them are not included in this paper.

In addition to the cryptographic benchmark applications, we also employed two microcode applications in our evaluation: `meter` and `sram_dram`. The `meter` algorithm is the single-rate three-color marker, which gives three possible mark choices at the output for a packet flow that is being metered. The `sram_dram` application performs SRAM and DRAM read-write operations and SRAM atomic operations (test_and_set commands) involving read-modify-writes wherein pre-modified data is returned.

4.2 Configuration of NePSim2 Modeling

To estimate the power consumption of the IXP2400, we configure the NePSim simulator according to the specifications in the IXP2400 datasheet [5]. Table 3 shows the operating frequency that we use for different components when simulating these components. The operating voltage simulated in NePSim is set at 2.0V. The technology feature used in the model is 0.18 um.

The Intel IXP2400 contains eight powerful microengines to process network packets. In order to clearly observe how much power each application consumes, and thereafter, compare and analyze their power distribution, we load the same benchmark application on each microengine in a single run.

4.3 Validation of Power Estimation

Here we validate our power model within the cycle-accurate NePSim 2.0 infrastructure. Figure 4 shows the total power consumption generated using our simulator for IXP2400. The Intel datasheet [5] states that the typical power consumption of the IXP2400 core (B-stepping) at 600 MHz is 8.26 watts. Since we did not model the power consumption of XScale and PCI, we expect the estimated power consumption that we derive to be smaller than what is reported in the IXP2400 datasheet.

The XScale technical summary [4] reports that the typical power consumption of Intel XScale processor is 0.45 watts running at 600 MHz. In our simulation, the average power consumption of all the benchmark applications shown in Figure 4 is 7.558 watts. Thus, the total power consumption of the IXP2400 (including XScale) turns out to be 7.558 + 0.45 = 8.01 watts which is 97% of the measured power (8.26 watts). Considering that we did not model the power consumption of PCI controller (very small as stated before), the simulated power we derive is close to the power reported in [5].

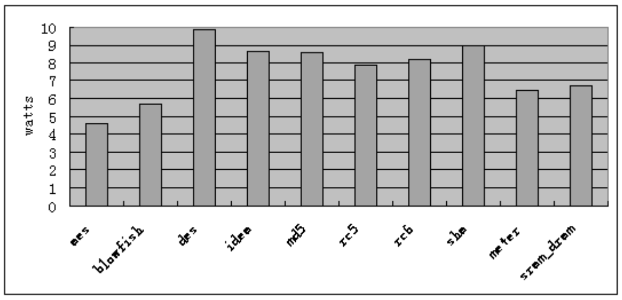

Figure 4. Total power consumption

4.4 Power Distribution within IXP2400

Figures 5 and 6 plot the power breakdown of the structural components within the IXP2400. From these figures, we make several observations. First, the microengines (MEs) in the IXP2400 consume the largest percentage of the total power. The reasons for this are two-fold.

- There are *eight* powerful microengines in the IXP2400 NP. Each one is a multithreaded processing unit.
- The cryptographic applications that we ran are computationally intensive. Thus, the data path and control store (instruction storage) of the microengines are heavily utilized. The I/O interfaces are idle for most of the running time, so generally, the I/O power consumption is due mainly to its static power consumption, which is relatively smaller than its dynamic power. Although there are some accesses to I/O, the dynamic power is negligible for those I/O components.

Second, `meter` and `sram_dram` consume relatively more power in SRAM than other benchmark applications, particularly as shown in Figure 5. We attribute this phenomenon to `meter` depositing all of its data structures and packet-descriptor data in SRAM. Thus, DRAM is not required in the execution of `meter`. With respect to `sram_dram`, it uses SRAM more heavily than DRAM, so that the SRAM controller power is more significant than

DRAM controller. In contrast, the cryptographic benchmark applications use DRAM more often than SRAM.

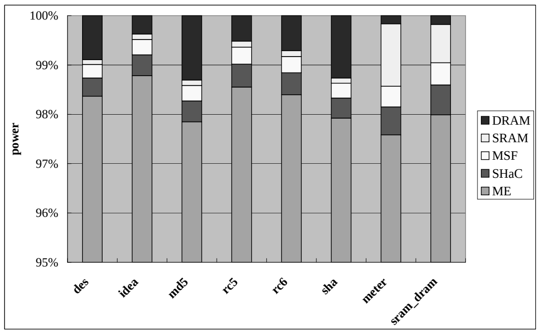

Figure 5. Power distribution over structural components

Third, for the aes and blowfish applications in the crypto benchmarks, the distribution of power behaves a bit differently. Figure 6 shows the power breakdown for these two crypto benchmark applications. While the power consumption of the microengines (MEs) in other crypto benchmark applications exceed 90%, the MEs for aes and blowfish applications consume less than 90% of the total power consumption. This is due to the fact that these applications possess a large number of I/O operations, which in turn, consume about 15%-20% of the total power consumption, as shown in the figure via the SHaC component. As a result of the queuing effect of the accesses to the shared I/O components (in this case SHaC), MEs idly wait for the completion of the I/O operations. During this idle time, MEs consume only static power.

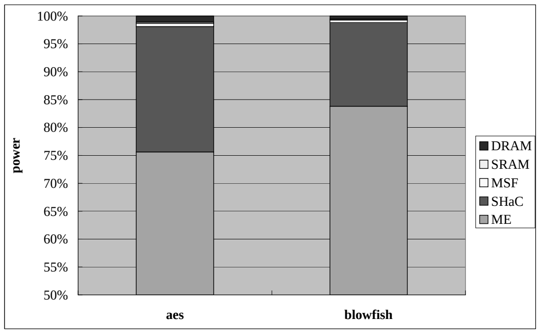

Figure 6. Power distribution over structural components

Fourth, referring back to Figure 4, aes, blowfish, meter, sram_dram consume less power than des, idea, md5, rc5, rc6, sha. The second and third points above show that aes and blowfish has relatively more I/O operations and meter and sram_dram has relatively more memory operations than others. Since MEs are idle waiting for I/O and memory accesses, MEs in aes, blowfish, meter, sram_dram should consume less power. Furthermore, an I/O operation usually takes longer than memory access. That explains why aes and blowfish consume further less power than meter and sram_dram.

Overall, our results show that MEs typically consume over 80% of the total power in the IXP2400. To understand what contributes to power consumption of an ME, we show the power breakdown inside one ME in Figure 7. For all of the applications that we ran, the control store and ALU were the two most power-consuming units. The control store is SRAM and holds the program that the ME executes. There are two reasons for the control store's large power consumption. First, since the control store holds the instructions a microengine executes, it will be accessed almost every cycle, leading to considerable dynamic power consumption. Second, the size of the control store (i.e., 4096 40-bit instructions) requires a large number of SRAM cells, thus significant power is consumed in an access. The ALU consumes the next most amount of power within an ME because the ALU is utilized in every instruction. Finally, the third most power-hungry component is the GPRs, where instruction operands and results reside in.

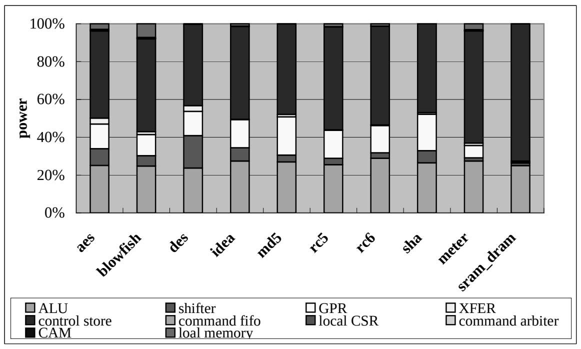

Figure 7. Microengine power distribution over units

4.5 Exploring Performance-Power Trade-offs

To help architects to design and optimize multicore network processors, we utilize our power model to explore the design space. The data obtained with the simulation framework and power model can guide chip architects to determine the architectural parameters under the constraints of power and performance. As shown in Section 4.4, MEs consume a significant amount of power in the IXP2400. Thus, we tune some of the key architectural parameters of the ME to study the power and performance characteristics. The parameters under study are the number of on-chip cores (microengines) and operation frequency. The base configuration of our experiment is as described in Section 4.4, in which the frequency is 600 MHz and the number of microengines is 8.

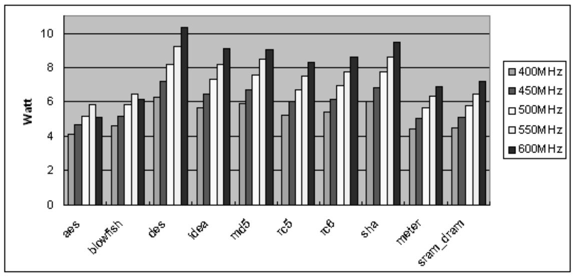

Figure 8. Power consumption on various frequencies

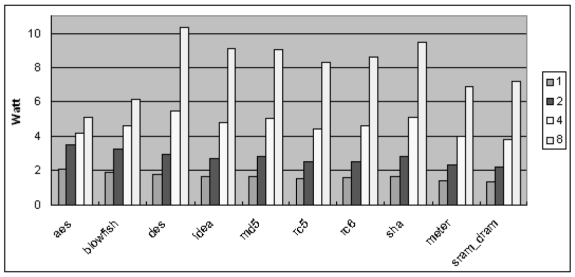

Figure 9. Power consumption on various number of microengines

Figures 8 and 9 show the power behavior while tuning the frequency and the number of microengines, respectively. As shown in Figure 8, the maximum power consumption across the cryptographic benchmarks is 10.30 watts (all the power is the total power including XScale hereafter) for `des` at 600 MHz. The average power consumption for all benchmarks at frequency levels of 400 MHz, 450 MHz, 500 MHz, 550 MHz, 600 MHz are 5.20 watts, 5.92 watts, 6.68 watts, 7.48 watts and 8.01 watts, respectively. Figure 9 shows that the maximum power consumption for the cryptographic benchmarks is also 10.30 watts for `des` for 8 microengines. The average power consumption for all benchmarks for 1, 2, 4, 8 microengines are 1.65 watts, 2.75 watts, 4.59 watts, 8.01 watts respectively. The higher the frequency or the larger the number of microengines, the more power is consumed.

Power efficiency is another important factor to evaluate in NPs. We choose Million Instructions Per Second Per Watts (MIPS/Watt) as the metric to measure the power efficiency of a multicore-based design, and the results are shown in Figures 10 and 11. The performance-power efficiency for security applications rises as the number of microengines increases and drops as frequency goes higher. This is because of the computationally intensive nature of cryptographic algorithms. On average, the power efficiency of multicore processors decreases by 5.9% when the operational frequency increases from 400 MHz to 600 MHz. In constrast, increasing the number of cores from 1 to 8 improves the power efficiency by 42.6%. This indicates that it is more energy efficient for these security applications to lower processor frequency or use more microengines to do parallel computation, thus indirectly supporting the low-power supercomputing approaches of Green Destiny and IBM BG/L.

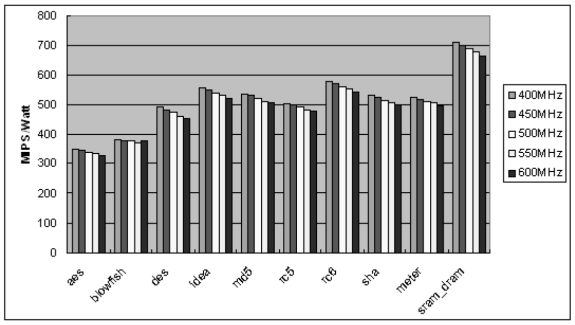

Figure 10. Performance-power trade-off across various frequencies (number of cores fixed to eight)

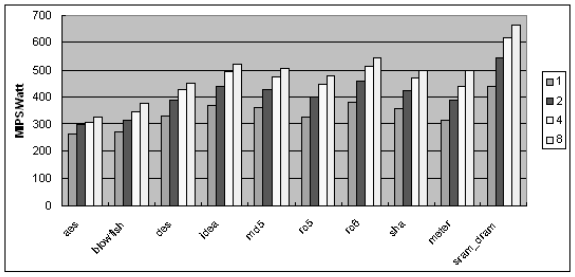

Figure 11. Performance-power trade-off across various number of microengines (core frequency is 600 MHz)

To better understand the impact of the number of cores and operational frequency on power and the performance-power tradeoff, we extensively study one of the crypto benchmarks `idea`. The behavior of `idea` should be typical since the analysis above shows consistent behavior among all benchmarks. Figure 12 shows the power consumption (left) and performance-power tradeoff (right) of `idea` for all combination of number of cores and frequencies. In the

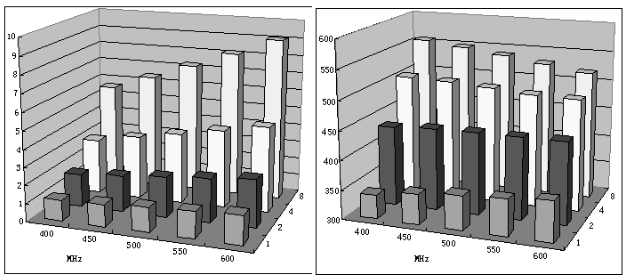

Figure 12. Power and performance-power tradeoff for `idea`

left graph, the power increases with an increase in the number of cores and frequency. In the right graph, power efficiency increases with an increase of frequency for a small number of cores, e.g., 1. On the other hand, power efficiency drops with the increase in frequency for a large number of cores, e.g., 8. For a moderate number of cores, e.g., 2 to 4, it does not have significant variation, since they are the transition steps. This further confirms our claim of using more cores and lower frequency to achieve high power efficiency.

5 Conclusion

In this paper, we present a power model that is integrated into the open-source Intel IXP2400 network processor simulator called NePSim2.0. In our experiments with a set of benchmark applications, we validate the estimated power against the processor datasheet. The breakdown of power consumption shows that microengines take up the most part of total power. Inside a microengine, the control store and ALU are the two most power-hungry components. Further experiments show that processor power consumption increases as the operational frequency and number of microengine number goes up. Finally, the power efficiency of the Intel IXP2400 increases as the number of MEs increases while it decreases as the frequency increases.

References

[1] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of International Symposium on Computer Architecture*, pages 83–94, 2000.

[2] Cisco. Cisco crs-1 carrier routing system 8-slot line card chassis system description.

[3] M. Franklin and T. Wolf. Power considerations in network processor design. In *Workshop on Network Processors in conjunction with Ninth International Symposium on High Performance Computer Architecture (HPCA-9)*, pages 10–22, Feb. 2003.

[4] Intel. Intel xscale microarchitecture technical summary. Intel Corporation, 2000.

[5] Intel. Intel ixp2400 network processor datasheet. Intel Corporation, 2004.

[6] Intel. Intel ixp2xxx product line of network processors. Intel Corporation, 2004.

[7] Intel. Intel ixp2xxx network processors hardware reference manual. Intel Corporation, 2005.

[8] S. Kaxiras and G. Keramindas. Ipstash: a power-efficient memory architecture for ip-lookup. *Proceedings of International Symposium on Microarchitecture*, page 361, 2003.

[9] S. Kent and R. Atkinson. Rfc 2401 security architecture for the internet protocol. RFC, 1998.

[10] R. Kokku, T. Riche, A. Kunze, J. Mudigonda, J. Jason, and H. Vin. A case for run time adaption in packet processing systems. *ACM SIGCOMM Computer Communication Review*, 34(1), 2004.

[11] Y. Luo, J. Yang, L. Bhuyan, and L. Zhao. Nepsim: A network processor simulator with power evaluation framework. *IEEE Micro*, sept 2004.

[12] Y. Luo, J. Yu, J. Yang, and L. Bhuyan. Low power network processor design using clock gating. In *IEEE/ACM Design Automation Conference (DAC)*, Anaheim, CA, 2005.

[13] A. Mallik and G. Memik. A case for clumsy packet processors. *Proceedings of International Symposium on Microarchitecture*, 2004.

[14] NePSim 2.0. http://www.cs.ucr.edu/%7Eyluo/nepsim/. NePSim 2.0 Source Code, 2006.

[15] Radisys. Promentum atca-7010 10gbps packet processing module datasheet. Radisys Corporation, 2006.

[16] J. Tschanz, Y. Ye, L. Wei, V. Govindarajulu, N. Borkar, B. Burns, T. Karnik, S. Borkar, and V. De. Design optimizations of a high-performance microprocessor using combinations of dual-vt allocation and transistor sizing. *Symposium on VLSI Circuits Digest of Techical Papers*, pages 218–219, 2002.

[17] N. Viljoen and D. McAuley. Intelligent network applications for 10gbps and beyond. Netronome White Paper, 2007.

[18] H.-S. Wang, X. Zhu, L. shiuan Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *The 35th International Symposium on Microarchitecture*, pages 294–305, 2002.

[19] S. J. E. Wilton and N. P. Jouppi. Cacti: An enhanced cache access and cycle time model. *IEEE Journal of Solid-state Circuits*, 31(5), 1996.

[20] Y. Ye, S. Borkar, and V. De. A new technique for standby leakage reduction in high-performance circuits. *Symposium on VLSI Circuits Digest of Techical Papers*, pages 40–41, 1998.

[21] Y. Yue, C. Lin, and Z. Tan. Npcryptbench: a cryptographic benchmark suite for network processors. *ACM SIGARCH Computer Architecture News*, 34(1):49–56, 2006.