System Level Power-Performance Trade-Offs in Embedded Systems Using Voltage and Frequency Scaling of Off-Chip Buses and Memory

Kiran Puttaswamy[1], Kyu-Won Choi[2], Jun Cheol Park[1],
Vincent J. Mooney III[1], Abhijit Chatterjee[2] and Peeter Ellervee[3]
[1]Center for Research in Embedded Systems and Technology
[1,2]School of Electrical and Computer Engineering
Georgia Institute of Technology
{kiranp, kwchoi, jcpark, mooney, chat}@ece.gatech.edu
[3]Tallinn Technical University
lrv@cc.ttu.ee

ABSTRACT

In embedded systems, off-chip buses and memory (i.e., L2 memory as opposed to the L1 memory which is usually on-chip cache) consume significant power, often more than the processor itself. In this paper, for the case of an embedded system with one processor chip and one memory chip, we propose frequency and voltage scaling of the off-chip buses and the memory chip and use a known micro-architectural enhancement called a store buffer to reduce the resulting impact on execution time. Our benchmarks show a system (processor + off-chip bus + off-chip memory) power savings of 28% to 36%, an energy savings of 13% to 35%, all while increasing the execution time in the range of 1% to 29%. Previous work in power-aware computing has focused on frequency and voltage scaling of the processors or selective power-down of sub-sets of off-chip memory chips. This paper quantitatively explores voltage/frequency scaling of off-chip buses and memory as a means of trading off performance for power/energy at the system level in embedded systems.

Categories and Subject Descriptors

B.6.3 [Design Aids]: Automatic synthesis, Optimization, Simulation.

General Terms

Performance, Design, Experimentation.

Keywords

Voltage/frequency scaling, power-performance trade-offs, embedded systems, design space.

ISSS'02, October 2–4, 2002, Kyoto, Japan.

1. INTRODUCTION

A typical embedded system consists of at least three main components: a processor (often with L1 cache), an off-chip memory (called L2 memory) and an off-chip bus connecting the processor and memory. The off-chip components, being highly capacitive, may consume as much or more power than the processor. This suggests that we can gain significant reductions in power and energy by reducing the off-chip voltage and frequency. However, power reduction from voltage (and corresponding frequency) reduction could be compromised by an increase in execution time, thus resulting in overall increase in energy dissipation (note that the increase in execution time is due to increased memory access time and is a function of the cache misses). We demonstrate how the performance impact of voltage and frequency scaling can be reduced by implementing a known micro-architectural technique called a store buffer. While our approach can apply to dynamic voltage scaling, this paper only shows the tradeoffs between statically setting the off-chip bus and memory voltage at 3.3 Volts and frequency at 100 MHz versus 2 Volts and 50 MHz. As is evident from the above description, it is necessary to have an integrated framework in order to quantitatively explore the power-performance design space at the system level. Specifically, the contribution of this paper is as follows.

We have combined the techniques of frequency/voltage scaling of off-chip buses and memory (circuit level technique) with a store buffer (architectural technique) to realize reductions in both system power and system energy dissipation with a negligible impact on the execution time.

The rest of the paper is organized as follows: Section 2 discusses the motivation for this work. Section 3 gives an overview of the previous work. Section 4 describes the experimental infrastructure. Section 5 discusses the methodology. Section 6 presents the results and Section 7 concludes the paper.

2. MOTIVATION

In embedded systems, especially mobile applications, battery life is a significant concern. It has been established that the battery drains faster when power is drawn at a higher rate [1]. For example, a battery which lasts for 1000 hours when drawing 10 milliamps at 1.5 Volts will only last for 80 hours when drawing 100 milliamps at the same voltage [1]. Also, an old battery discharges faster than a new one. Currently, few hooks exist to trade-off per-

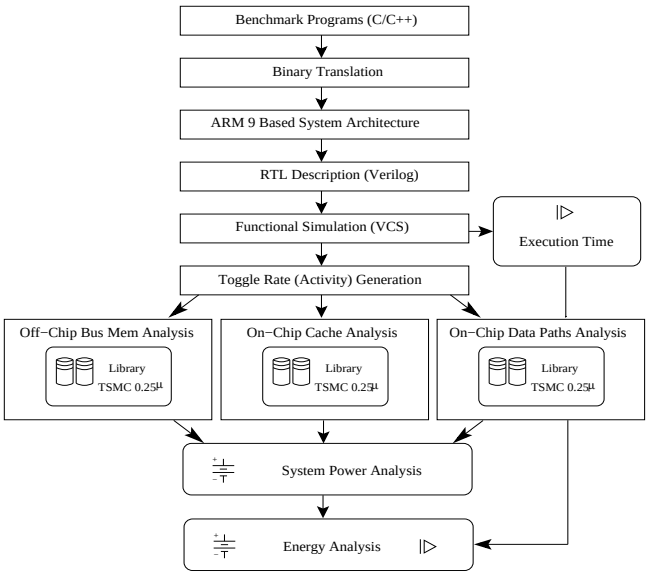

Figure 1: Experimental Infrastructure

formance for power to prolong the battery life of an embedded system. For example, when the user is executing a time-critical application like real-time video-conferencing, he might decide to operate at peak performance and high power. Or he might opt for low performance and very low power when executing low priority applications like checking e-mail. Or he might chose an intermediate power-performance point based on the existing battery capacity. Note that turning off memory chips may not be possible because, for example, video data might be stored in the memory and may need to be used. Also note that, as presented in this paper, this is a static technique and happens at the beginning of the program execution. We currently do not address dynamic (run-time) scaling of voltage/frequency of off-chip memory. This paper presents an approach that can allow the compiler and/or the user to decide at which power-performance point to operate, e.g., based on the knowledge of battery capacity and battery discharge pattern.

3. PREVIOUS WORK

There has been significant work in the field of voltage and frequency scaling. Voltage scaling techniques have been investigated at almost all levels of the design hierarchy from the system level to the device level due to the quadratic effect on the switching power dissipation. However, as the supply voltage becomes lower, the circuit delay increases and the performance degrades [2]. Techniques to improve performance fall into three main categories: reducing the threshold voltage to improve circuit speeds, introducing parallelism into the architecture while using slower device speeds, and using multiple supply voltages to choose the lowest supply voltage for different circuit components that still satisfies the speed requirements. Our approach falls into the third category.

At the system/architecture level, a number of memory optimization schemes for low power have been developed [3, 5]. Briefly, those approaches can be categorized as follows: cache optimization, memory access reduction (especially for off-chip memory), memory sizing/structuring and memory-intensive voltage scaling. Our work falls in the category of memory intensive voltage scaling.

There has also been a lot of work on system level power anal-

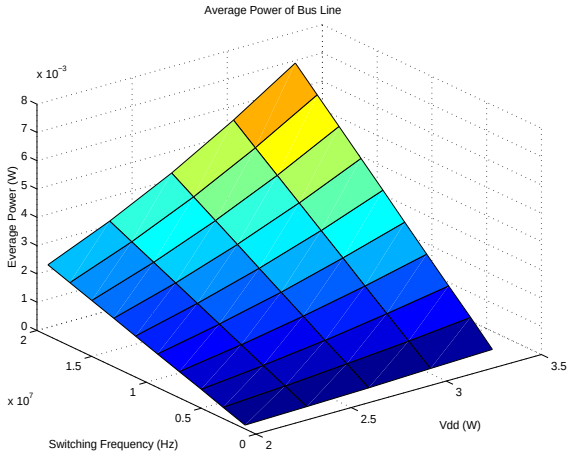

Figure 2: Bus Power Dissipation Pattern

ysis for software power dissipation. In [6], profiling hardware is used to identify tightly coupled regions of code and dynamically optimize the configuration of the microprocessor so as to minimize performance penalty. Our work is similar to this approach in the sense that we set the performance parameters, namely, voltage and frequency based on the requirements of the application. However, note that ours is a static technique rather than a dynamic technique. Also, there has been work on dynamically adjusting the speeds (i.e., either lower the frequency or shutdown the unused modules), whichever gives the best results [7]. At the technology level, various techniques have been researched such as gating the supply voltage to cache memories [14].

A similar framework to our power measurement infrastructure is introduced in SimplePower [4]. SimplePower is based on a subset of the Simplescalar Instruction Set Architecture. Currently, Simplepower does not capture the energy consumed by the control unit of the processor nor the clock generation nor the distribution network [13]. On the other hand, our work more accurately models the specific RISC processor as the measurements are based on cycle-accurate functional simulations and Register Transfer Level hardware models used along with an actual technology library. An additional aspect of our work is the inclusion of power models for both off-chip memory [19] and the Printed Circuit Board bus.

4. EXPERIMENTAL INFRASTRUCTURE

We consider an embedded system which consists of a classic five stage pipeline RISC processor core with 4 kilobytes of instruction cache, 4 kilobytes of data cache, a single off-chip synchronous SRAM memory of size 0.5 Megabytes organized as 128K X 4 bytes, and a bus interface consisting of a 32 bit address bus, 32 bit data bus and the read/write control signals between the processor core and the SRAM memory.

Our experimental infrastructure (Figure 1) consists of four main components: a C Compiler; MARS – obtained from the University of Michigan – a cycle-accurate Verilog Model of a RISC processor capable of running ARM instructions [12]; a power model for off-chip buses; and a power model for memory.

We use the GNU-gcc ARM cross compiler version egcs-2.91.66. For each benchmark we consider, we compile the benchmark to relocatable ARM assembly code using GNU-gcc ARM cross compiler. Then we use the GNU cross-assembler to generate a binary executable targeted towards ARM architectures. Then we translate the binary into an ascii format called VHX (Verilog HeX) [20]

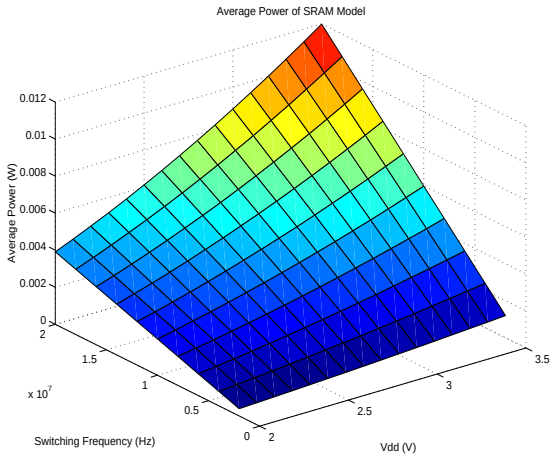

Figure 3: Memory Power Dissipation Pattern

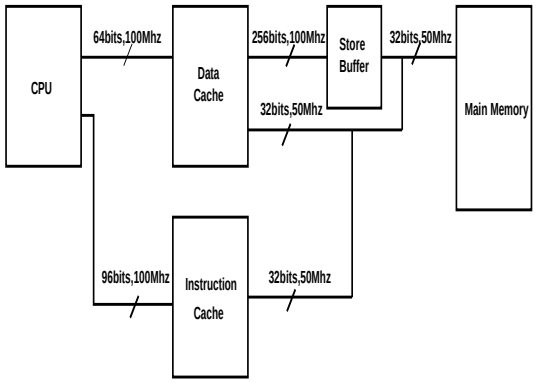

Figure 4: System Level Block Diagram

Benchmark	Size (bytes)	Explanation
bubble	8131	Bubble Sort Program
factorial	6641	Factorial Program
fib	6815	Fibonacci Sequence Calculation
matmul	8058	Matrix Multiplication
sort_int	7241	Integer Array Sort

Table 1: Benchmarks

which is suitable for being simulated on MARS using the Synopsys VCS simulator [8]. The simulation experiments are carried out in two modes. In the first mode, the CPU core and off-chip buses and memory are all operating at 100 MHz, a setup that is very similar to a hardware setup we have in the Hewlett-Packard "Skiff" Personal Server board [17] with a StrongARM SA-110 processor and 16 megabytes of off-chip memory [18]. (Note that we model a smaller off-chip memory of size 0.5 megabytes in accordance with the smaller applications we consider.) We obtained the simulation model for the off-chip memory from IDT Technologies, Inc. [15]. In the second mode, the processor core includes a Verilog description of a store buffer integrated into the core and interfaced to off-chip buses and off-chip L2 memory operating at 50 MHz. In both cases, using the set of benchmarks in Table 1, each benchmark is simulated and switching activity is collected for the processor core, off-chip buses and off-chip memory models. The switching activity is fed to the power models of the core, off-chip buses and off-chip memory along with the technology parameters of a TSMC 0.25μ CMOS technology standard cell library from Leda Systems [9] to obtain power and energy estimates.

4.1 On-chip Datapath Power Estimation

We use a synthesis based methodology for developing the power models for the submodules belonging to the datapath (we consider the datapath to consist of the fetch unit, decode unit, register file, arithmetic logic unit, data cache access unit and writeback unit). The synthesis infrastructure consists of two software tools from Synopsys, Inc.: the Design Compiler and Power Compiler [8]. Design Compiler generates the gate level netlist from the hardware description of the submodules, and Power Compiler generates power estimates for each of the synthesized netlists. The Verilog RTL description is given as input to the Synopsys Design Compiler. The output netlist is generated using a TSMC 0.25μ CMOS technology standard cell library from LEDA Systems [9]. The technology details include features such as transistor width, transistor length, gate capacitance, drain capacitance, transistor rise time and transistor fall time. The TSMC 0.25μ standard cell library is characterized for leakage power, thus enabling us to include both dynamic and static power and energy in our analysis.

The synthesis process was guided by fixing the maximum delay and maximum area. The maximum delay was set to 10 ns and the maximum area was fixed to infinity so as to get the fastest implementation. In our case, the modules were synthesized to operate at greater than or equal to 100 MHz (i.e., at less than or equal to a 10 ns cycle time).

We use Power Compiler from Synopsys to estimate the power of on-chip components. The Power Compiler obtains the switching activity of the various functional modules based on the simulation of benchmarks on MARS. Then, Power Compiler annotates this switching activity onto the synthesis environment and obtains estimates of the dynamic and static power dissipation for the particular technology chosen (in our case, TSMC 0.25μ CMOS technology).

4.2 Off-chip Bus Power Analysis

We use Spectre simulation to obtain the power for off-chip buses. The driver component is modeled by a series of inverters (buffers) of increasing size and the model is designed using TSMC 0.25μ CMOS process technology parameters. The parameters of TSMC 0.25μ process technology are available through MOSIS [11]. The bus line capacitance values are obtained from actual measurement on a PCB board using the Intel StrongARM processor [18]. The details of the measurement procedure has been explained in [20].

The graph in Figure 2 describes the dependence of the power dissipation on the switching frequency of the bus and also the power supply voltage V_{dd}. The power dissipation is found to decrease as the switching frequency and the supply voltage are decreased. (Switching frequency is how often a signal actually switches values. Clearly, switching frequency of a signal is data- or profile-dependent, i.e., dependent on a particular benchmark and data.) As the supply voltage decreases, the average power dissipation reduces quadratically. As the switching frequency decreases, the average power dissipation decreases almost linearly. Note that the simultaneous halving of both voltage and frequency results in cubic savings.

4.3 Memory Power Analysis

We use an analytical SRAM model [19] for the off-chip memory and cache power dissipation. For the off-chip memory power model, we updated the analytical model [19] using the TSMC 0.25μ process technology parameters. We use the switching activity from

simulations to obtain estimates for SRAM memory power dissipation. The variation of power for the memory with supply voltage V_{dd} and the switching frequency is thus found.

The graph in Figure 3 describes the dependence of the power dissipation of the memory with the power supply voltage V_{dd} and the switching frequency for a memory size of 0.5 megabytes (note that switching activity period is the reciprocal of switching frequency). As the supply voltage is decreased, the average power dissipation is found to decrease quadratically. Also the memory delay was found to double as we reduced the voltage from 3.3 Volts to 2 Volts, reducing the maximum clocking frequency possible from 100 MHz at 3.3 Volts to 50 MHz at 2 Volts.

For the on-chip cache power dissipation model, we use the same SRAM model [19]. The model is combined with the capacitance values obtained from the TSMC 0.25μ CMOS technology parameters. Note that the main difference between the off-chip memory model and on-chip cache model is that the off-chip model has significantly higher capacitance values (due to size) than the on-chip model.

4.4 System Level Power/Energy Model

We define the sum total of the processor core power, bus power and the memory hierarchy power as the system power P_{sys}.

$$P_{sys} = P_{cpu} + P_{bus} + P_{mem} \quad (1)$$

P_{cpu} is the power dissipated by the processor core,
P_{bus} is the power dissipated by the off-chip buses,
P_{mem} is the power dissipated by the memory.

Also, we calculate the system energy E_{sys} for each benchmark by multiplying the execution time collected by simulation and the corresponding sysem power P_{sys}.

$$E_{sys} = P_{sys} * t \quad (2)$$

P_{sys} is the power dissipated by the system,
t is the total execution time of the benchmark.

5. METHODOLOGY

We now explain the method we use to explore voltage and frequency scaling as a static technique for design space exploration in terms of power versus performance trade-offs.

5.1 Voltage/Frequency Scaling

In particular, we analyze the case where we reduce the voltage of the off-chip memory and off-chip buses from 3.3 Volts to 2 Volts (note that our original system consists of the MARS [12] processor powered at 2.75 Volts with the memory buses and memory chip powered at 3.3 Volts). The resulting increase in memory delay (it doubles) is taken care of by reducing the off-chip bus and off-chip memory frequency from 100 MHz to 50 MHz. Of course, reducing off-chip bus and off-chip memory frequency increases program execution time in proportion to cache misses. We help offset this effect by adding a store buffer to the processor model. The store buffer helps to reduce cache miss penalties generated due to "store" instructions (since the cache model follows a read-write allocation policy, cache miss servicing takes up a significant amount of time). On a cache miss on store, the processor stores the data into the store buffer. The store buffer assumes the responsibility of flushing the data into the main memory.

We achieve frequency scaling by simulating the off-chip components at the lower frequency of 50 MHz. We achieve voltage scaling by using the resulting switching activity with the reduced voltage value of 2 Volts to calculate the off-chip bus and memory power dissipation values.

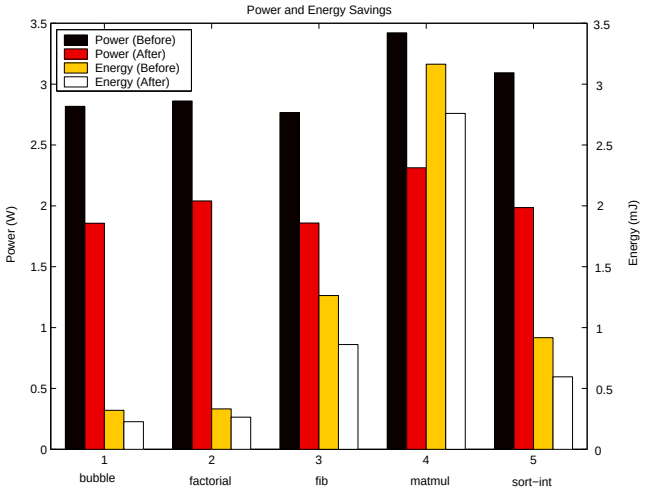

Figure 5: Power/Energy Saving Results

5.2 Store Buffer Technique

Figure 4 shows an architectural description of the embedded system with a store buffer included. Note that the store buffer is a 16 entry buffer. Stores to external memory are first placed in the store buffer and subsequently taken out when the off-chip bus is available. Thus, we reduce cache miss latency due to "store" instructions. The store buffer maintains synchronization with the on-chip data cache as well as with the off-chip memory. We use the Synopsys Power Compiler to estimate the additional power consumed due to the addition of store buffer into the processor core. Since we have integrated the description of the store buffer into MARS, the new system level power dissipation includes the store buffer overhead.

5.3 Design Space Exploration

In this section, we discuss the design space exploration of the power-performance space using voltage and frequency scaling. The Verilog model (MARS) is simulated in two modes. In the first mode, processor, off-chip buses and off-chip memory operate at 100 MHz. In the second mode, the processor operates at 100 MHz while the off-chip buses and off-chip memory operate at 50 MHz. In both cases, the Power Compiler collects the switching activity and uses the collected switching activity to give an estimate of dynamic and static power dissipation of all the modules within the core. We also collect the switching activity of off-chip memory elements and off-chip buses and feed the collected switching activity information to the analytical models to obtain the power estimates for off-chip bus and off-chip memory. We obtain the system level power/energy estimates as explained in Section 4.4.

Our calculations do not include the extra overhead of multiple supply voltage generation since we assume that the board already has the multiple supply voltages needed. For example, the "Skiff" Personnal Server Board from HP/Compaq [18] has 2 Volt, 3.3 Volt and 5 Volt power supplies. Also, currently there are memory chips from NEC Semiconductors where the chip component operates at 3.3 Volts and the I/O buffer component can operate at either 3.3 Volts or 2.5 Volts [16]. For example, the memory chip μPD4442361 Synchronous SRAM can operate at 3.3 Volt chip core voltage and either

benchmark	Executable size (kB)	Dynamic instruction count	Input data size	Data cache accesses	Data cache misses	Data cache miss %
bubble	34.852	7503	50 integers array	1675	107	6.39
factorial	34.634	6033	1 integer	2006	250	12.46
fib	34.651	30602	1 integer	11840	262	2.21
matmul	34.857	21642	0.5 kB	7358	4916	66.81
sort_int	34.763	23171	0.5 kB	7808	104	1.33

Table 2: Execution Statistics for Various Benchmarks

	Off-chip Bus, Memory at 100 MHz, 3.3 V				Off-chip Bus, Memory at 50 MHz, 2 V				% Improvement
Benchmark	cpu+cache (W)	bus (mW)	L2 memory(mW)	Total (W)	cpu+cache(W)	bus(mW)	L2 memory(mW)	Total (W)	
bubble	1.24	301.64	1276.49	2.817	1.22	96.14	541.08	1.857	34.07
factorial	1.18	444.35	1236.96	2.861	1.15	93.16	797.08	2.040	28.69
fib	1.25	287.68	1228.23	2.766	1.25	92.50	516.06	1.859	32.79
matmul	1.07	637.48	1713.34	3.421	1.04	129.04	1143.51	2.313	32.39
sort_int	1.27	336.78	1485.92	3.093	1.27	111.91	604.11	1.986	35.79

Table 3: System Level Power Estimates

3.3 Volt or 2.5 Volt I/O buffer voltage. The μPD4442361 is available in three speed grades of 133 MHz, 117 MHz and 100 MHz with the corresponding access times of 6.5 ns, 7.5 ns and 8.5 ns [16]. Therefore, we think it is reasonable to assume that our system will have at least three supply voltages readily available and that the system memory chips are capable of operating at dual voltages. The choice of frequency of off-chip buses and memory is currently assumed to be made statically by a mechanical or electrical switch.

6. RESULTS

The benchmarks in Table 1 are chosen to be computationally intensive. Also, the size of the data has been suitably modified so as to generate a significant number of L1 cache misses as can be seen from Table 2. For example, with matmul we increased the array sizes so that the 4KB L1 data cache could not hold the working set. Some of these benchmarks constitute the kernel of many signal-processing algorithms.

Table 2 shows the dynamic instruction count, cache access and miss statistics for the given benchmarks. Note that the final miss rates are smaller than the average miss rates at the beginning/middle of the execution of the program due to the temporal and spatial locality of the cache memories. Also note that the matmul benchmark has a very high miss rate. As a direct consequence of this, this benchmark experiences high off-chip traffic. As we will see later, benchmarks such as this, which need high off-chip bandwidth, show correspondingly lower improvement in terms of energy by our technique (although they still benefit in terms of power) as the power savings are nullified by the high increases in program execution time.

Table 3 presents the results related to system level power. The first three columns indicate the three components of power, namely, core power dissipation, off-chip bus power dissipation and off-chip memory power dissipation for the case where the MARS processor core operates at 2.75 Volts and 100 MHz and the off-chip system operates at 3.3 Volts and 100 MHz. The next three columns indicate the same three components of power for the case where the off-chip bus and memory operate at 2 Volts and 50 MHz while the MARS processor core still runs at 2.75 Volts and 100 MHz. The system level power estimate is obtained by adding up the core power, bus power and the memory power as shown in the fifth and ninth "Total(W)" columns in Table 3. Note that the proposed technique has reduced the power dissipation by an average of 32% on the five benchmarks. Also note that the average power reduction is almost uniform across all benchmarks irrespective of their execution characteristics like execution time and dynamic instruction mix. This highlights the dominant effect of voltage and frequency on the power dissipation.

Table 4 presents the statistics for the architectural and circuit level design space exploration where execution time represents the performance axis and system power represents the power axis. Note that, as expected, matmul benchmark has a higher penalty in terms of the execution time due to high off-chip traffic for loading and storing the data arrays.

The energy column combines both the design space axes, namely performance axis and the power axis and serves as a baseline for analyzing power-performance trade-offs. The trade-off shown, for example, the factorial benchmark, shows a performance penalty of 11.37% in return for a power reduction of 28.69% and an energy reduction of 20.48%. All benchmarks show improvements in both power and energy. However, as expected, the execution time increases. This shows that our technique reduces both power and energy by virtue of reducing the fraction of the off-chip bus and memory power consumed, at a (possibly small) penalty in increased execution time.

Figure 5 presents the overall results of the system level power and system level energy. Our technique of static voltage/frequency scaling combined with the store buffer is seen to reduce both power and energy at the system level.

7. CONCLUSION AND FUTURE WORK

In conclusion, we have shown how a simple trick of cutting off-chip voltage from 3.3 Volts to 2 Volts (note that this also cuts the voltage for the processor I/O pad driver logic), together with the enabled (due to extra latency available) reduction in frequency of off-chip buses and memory, reduces both power and energy. The basic point is that both the compiler and the programmer can take advantage of smart architectural and memory hierarchy features, which allow the reduction of power with some corresponding trade-offs in terms of performance. For example, the choice of 100 MHz versus 50 MHz for off-chip bus frequency could be made dynamically programmable (e.g., by writing to a special on-chip register or a memory-mapped location). In this case, then, code could be written which operates at high performance and high power during critical times, but scales down to lower performance and low power during non-critical time periods. This paper lays the groundwork for such a system where off-chip buses and memory and possibly

	Off-chip Bus, Memory at 100 MHz, 3.3 V			Off-chip Bus, Memory at 50 MHz, 2 V			Percent Change	
Benchmark	Execn Time (μs)	Power (W)	Energy (mJ)	Execn Time (μs)	Power (W)	Energy (mJ)	Execn Time increase (%)	Energy decrease (%)
bubble	113.945	2.817	0.321	122.265	1.857	0.227	7.3	29.3
factorial	116.115	2.861	0.332	129.325	2.040	0.264	11.37	20.48
fib	456.795	2.766	1.263	463.245	1.859	0.861	1.4	31.83
matmul	924.735	3.421	3.164	1192.98	2.313	2.759	29.0	12.8
sort_int	296.425	3.093	0.917	300.265	1.986	0.596	1.29	35.0

Table 4: System Level Design Space Exploration

more peripherals have their power dissipations modulated either by the user or by the compiler.

We will look at further hiding the load-instruction memory access latency caused by slowing down the off-chip buses and off-chip memory. We will explore other configurations for off-chip memory and also other memory technologies (e.g., SDRAMs). We will pursue making the voltage and frequency scaling dynamic.

8. ACKNOWLEDGEMENTS

This research was funded by DARPA under contract number F30602-00-2-0564. We also acknowledge donations received from Cadence, Hewlett-Packard, Intel, LEDA Systems, Mentor Graphics, Sun and Synopsys.

9. REFERENCES

[1] P. Horowitz and W. Hill, *The Art of Electronics*, Second Edition, Cambridge University Press, England, 1989.

[2] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, Vol. 27, April 1992.

[3] T. Ishihara and K. Asada, "A System Level Memory Power Optimization Technique using Multiple Supply and Threshold Voltages," *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 456-461, June 2001.

[4] W. Ye, N. Vijaykrishnan, M. Kandemir and M. J. Irwin, "The Design and Use of Simplepower: A cycle-Accurate Energy Estimation Tool," *Proceedings of 38th Design Automation Conference*, pp. 340-345, June 2000.

[5] L. Benini, A. Macii, E. Macii, and M. Poncino, "Synthesis of Application-Specific Memories for Power Optimization in Embedded Systems," *Proceedings of 38th Design Automation Conference*, pp. 300-303, June 2000.

[6] A. Iyer and D. Marculescu, "Power Aware Microarchitecture Resource Scaling," *Proceedings of Design Automation and Test in Europe*, pp. 190-196, March 2001.

[7] A. Acquaviva, L. Benini and B. Ricco, "An Adaptive Algorithm for Low-Power Streaming Multimedia Processing," *Proceedings of Design Automation and Test in Europe*, pp. 273-279, March 2001.

[8] Synopsys, Inc., Available HTTP: *http://www.synopsys.com*

[9] LEDA Systems, Inc., Available HTTP: *http://www.ledasys.com*

[10] TSMC, "IP Services," Available HTTP: *http://www.tsmc.com/design/ip.html*

[11] The MOSIS Service, Available HTTP: *http://www.mosis.org*

[12] The SimpleScalar-Arm Power Modeling Project, *http://www.eecs.umich.edu/~jringenb/power/*

[13] The SimplePower Energy Estimation Tool, *http://www.cse.psu.edu/~mdl/SimplePower.html*

[14] M. Powell, S. Yang, B. Falsafi, K. Roy and T. N. Vijaykumar, "Gated V_{dd}: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 90-95, July 2000.

[15] IDT Technologies, Inc., Available HTTP: *http://www.idt.com/products/pages/ZBT_Verilog_p.html*

[16] NEC Semiconductors, Inc., Available HTTP: *http://www.ic.nec.co.jp/memory/english/products/sram/ssram-4m.html*

[17] Hewlett-Packard, Inc., Available HTTP: *http://www.hp.com*

[18] HP Labs - Cambridge Research Laboratory Personal Server Project, Available HTTP: *http://crl.research.compaq.com/projects/personalserver/personal-server-spec.html*

[19] D. Liu and C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips," *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 6, pp. 663-670, June 1994.

[20] P. Korkmaz, K. Puttaswamy and V. Mooney, "Energy modeling of a Processor core using Synopsys and of the Memory Hierarchy using the Kamble and Ghose Model," Technical Report, CREST-TR-02-002, Georgia Institute of Technology, Feb. 2002.