Developing Energy-Aware Strategies for the Blackfin Processor

Steven VanderSanden, David R. Kaeli
Dept. Electrical and Computer Engineering
Northeastern University
Boston, MA 02115 USA
{svanders,kaeli}@ece.neu.edu

Giuseppe Olivadoti, Richard Gentile
Analog Devices, Inc.
1 Technology Way
Norwood, MA 02062 USA
{giuseppe.olivadoti,richard.gentile}@analog.com

Abstract

Energy usage is becoming an increasingly important design constraint for all computer systems. This issue is particularly critical in battery powered, embedded designs. Although many embedded processors have developed sophisticated power management schemes, few have produced an accurate, easy-to-use energy estimation framework. In this presentation we will describe the development of an instruction-level energy modeling framework for the Analog Devices Blackfin family of processors. Using this model, we are able to accurately estimate the energy consumed when running this code. While our main goal is to demonstrate that we can perform accurate energy estimation, we also plan to develop a framework that is fully integrated with compilation in order to produce more energy-efficient binaries. In this abstract we briefly describe our methodology and show data that illustrate some of the difficulties encountered when attempting to statically model energy.

1 Introduction and Methodology

The design specifications of many embedded systems include strict energy budgets. In order to reduce the time to market (and still meet these constraints), a designer must be able to accurately predict the energy usage for the system. The goal of our work is to develop an energy estimation scheme for the Analog Devices Blackfin 533 (ADSP-BF533). Two of the more common modeling options employed for energy estimation are architectural-level and instruction-level estimation. Architectural-level tools, which include th Wattch [1] and SimplePower [2] power modeling frameworks, compute energy based on functional unit usage considering transitions of individual signals. Instruction-level tools calculate the energy budget by characterizing individual instructions and inter-instruction energy usage. Instruction-level tools can only be used when the microarchitecture of the underlying processor is simple (e.g., on embedded cores).

We have chosen to use instruction-level energy estimation for our work. This form of estimation was employed previously by Tiwari et al. at Princeton to develop models for a number of embedded processors [3, 4]. They developed accurate models for the Intel 486DX2 and the Fujitsu SPARClite. We are following a similar approach, but extending it to consider further power aspects of the microarchitecture and applying these extension to the ADSP-BF533.

As mentioned previously, an instruction-level estimation is constructed by characterizing the energy usage of individual instructions (i.e., base energy cost) and then computing the overhead that is incurred when two different instructions are executed consecutively (inter-instruction effects). The total energy for a program is computed by summing the base energy costs of the individual instructions and the total inter-instruction effects.

To capture the base energy cost for an instruction, we place several instances of that instruction in a loop, run the loop, and measure the average current produced. The base energy cost is directly proportional to this measured current multiplied by the number of cycles required for the execution of one instance of the instruction. Inter-instruction effects are those effects that cannot be captured in the base energy cost. Inter-instruction effects can be characterized as effects related to resource constraints and delays (e.g., pipeline stalls, cache misses, write buffer stalls, etc.) and circuit state overhead (the added cost of switching within the circuit when executing two different instructions in succesion). The circuit state overhead can be measured by placing many repetitions of a pair of instructions in a loop and measuring the average current. The inter-instruction overhead can be calculated by computing the difference between the measured current and the average of the two base costs of the instructions in the loop.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 FEB 2005	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Developing Energy-Aware Strategies for the Blackfin Processor	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Dept. Electrical and Computer Engineering Northeastern University Boston, MA 02115 USA**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

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

13. SUPPLEMENTARY NOTES
See also ADM001742, HPEC-7 Volume 1, Proceedings of the Eighth Annual High Performance Embedded Computing (HPEC) Workshops, 28-30 September 2004. , The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**24**	

	Instruction r7 = r3 + r4;	
r3 Value	r4 Value	Current(mA)
0x1	0x1	52.20
0x80000000	0x8000000	52.31
0x90B	0x371F	52.82
0xCCCCCCCC	0xCCCCCCCC	53.27
0x33333333	0x33333333	53.33
0xFFFF	0xFFFF	53.44
0x7FFFFFFF	0x7FFFFFFF	54.34
0xFFFFFFFF	0xFFFFFFFF	54.34

Table 1: Impact of data operand values

Instruction	**Initial Values**	**Current (mA)**
r6 = -r3;	r3 = 0x90B	51.98
r3 = -r3;	r3 = 0x90B	60.53

Table 2: Impact of toggling register values

To estimate the total energy consumed by various programs, the base energy cost must be measure for each instrucion in the instruction set and the circuit state overhead for a large number of instruction pairs needs to be computed. The estimation framework will use a table lookup strategy to sum the base energy costs and the inter-instruction effects of a program to estimate the energy usage. To reduce the amount of time to produce this tables, we can identify similaries in energy costs of similar instructions.

The goal of his work is to produce an instruction-level energy model for the ADSP-BF533 and to use that model as a base for an energy estimation framework. In addition to these goals, we are also trying to improve the methods currently used for energy profiling. The remainder of this paper will discuss some of the issues encountered during energy profiling and we will also provide an example result used to verify our approach.

2 Results

In this section we will show some examples of the measurements that need to be obtained to perform instruction level modeling. In addition collecting a large number of base energy cost measurements and circuit state overheads, we also looked at the role that data values play in our ability to accurately collect this data. In Table 1, we show the effects of using different data operand values for an add instruction. As we can see, using different data values can have a significant impact on the average current and therefore the overall energy consumed (4.0% in this example). One interesting observation is that since many values in a computer are typically close to zero, two-complement can be an inefficient representation when considering energy consumptions (due to the large number of bit flips when a value changes sign).

In addition to investigating the impact of data operand values, we also looked the the impact output operands and the cost of toggling destination register values. In this example, the input data values were kept constant, but the destination register was varied. In Table 2 we show the results of a simple negate instruction. As we can see, large changes in current occur for when the destination register value is toggled. We can see clearly how dependent current measurements are on the number bit flips performed in a cycle.

As an example of the fidelity of our approach, we provide a small example program. We have both utilized our profile data to produce an estimated energy budget for this snipit, as well as have measured the current drawn. The code of the program is shown in Table 3. We have run the program on the ADSP-BF533 and measured the average current during program execution. The energy estimation using our approach was computed to be 3.2 nJ, while the average energy on the BF533 was measured to be 3.3 nJ. This is only a 3% difference.

To date, our results have clearly demonstrated that we can utilize this approach and obtain accurate measurements. In the presentation of this work, we will discuss some further power issues related to leakage energy and temperature dependence. We will also discuss some of the difficulties of estimating energy in the memory hierarchy.

r1 *= r2;
r2 = [i1++];
r0 = r0 + r1 (ns);
r1 = [p2++];
nop;

Table 3: Simple program to validate our approach

References

[1] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proceedings of 27th International Symposium on Computer Architecture*, May 2000.

[2] W. Ye, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," *In the Proceedings of the Design Automation Conference*, June 2000.

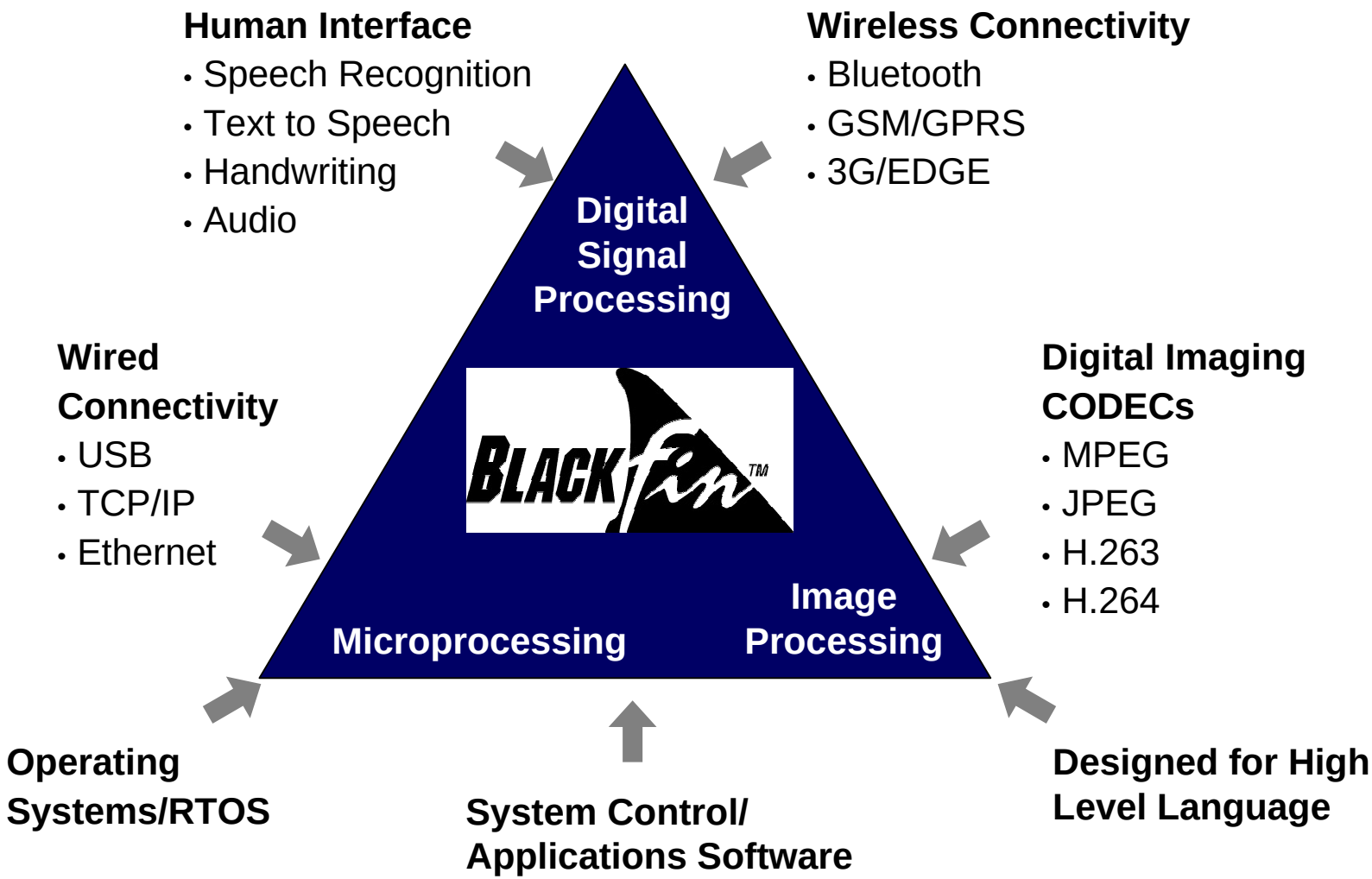
[3] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Toward Software Power Minimization," *IEEE Transactions on VLSI Systems*, pp. 437-445, Dec. 1994.

[4] M. T. Chien, V. Tiwari, S. Malik, and M. Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software," *IEEE Transactions on VLSI Systems*, pp. 123-135, Mar. 1997.

Introduction and Motivation

- Power consumption/density has become a critical issue in high performance processor design
- This issue is even more important on battery-powered embedded cores and systems
- The embedded processing market is growing at a very fast pace
- Application engineers must be able to accurately predict the energy usage for the core and the system when running their applications
- This project is targeted to improve the power analysis capabilities of the ADI Blackfin family of processors and systems

ADI Blackfin Family of Processors



Blackfin Family

- **Blackfin Core**
  - High-performance
  - 16-bit
  - Dual-MAC embedded processors
  - Equally adept at DSP, control processing, and image processing
- **Processor Features**
  - 400-756Mhz core capable of to 1.512 GMACs
  - 8, 16 and 32-bit fixed-point math support
  - Hierarchical reconfigurable memory systems
  - Dual core versions
  - High speed peripherals and DMA controller
    - Parallel Peripheral Interface (PPI) : dedicated 0-75Mhz parallel data port
    - SPORTS, SPI, External Port, SDRAM, UART (IrDA), etc
  - Control processing features
    - Very high compiled code density
    - Supervisor and user modes/MMU, watchdog timer, real-time clock

How does the Blackfin Processor help?

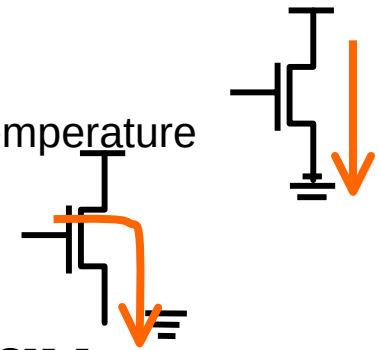
- **Speeds time-to-market and facilitates rapid product derivatives**
  - –High-performance software target
  - –Software-centric product development
- **Lowers BOM and R&D costs**
  - –Eliminates redundant DSP, MCU and hardware accelerator blocks
  - –Software reuse model enhances R&D productivity with each sequential product generation
  - –Processors begin at $5 (in quantities of 10K)
- **Reduces technical, market and schedule risks**
  - –Software support for multiple formats and evolving standards
  - –Development and debug within software—not ASIC—cycle times
  - –Signal processing capabilities along with a familiar RISC programming model
- **Enables end-product feature differentiation**
  - –2X to 4X performance advantage per dollar and per milliwatt

Blackfin Dynamic Power Management Overview

- Wide range of core frequencies supported (1.25M->756 MHz)
  - Programmable Core and System Clocks for maximum power savings
- Wide range of core operating voltages supported (0.8 -> 1.4 V)
  - Programmable internal voltage levels based on core frequency
- Full complement of power savings modes
  - Full-on, Active, Sleep, Deep-sleep and Hibernate
- “Voltage and frequency tuning” for minimum power
  - Ensures consistent, low power consumption across process
- Dual-core processor can be used for power savings
  - Lower voltage levels and lower frequencies provide additional power savings options with equivalent performance levels

Power Dissipation

- **Dynamic power dissipation**
  - Due to switching activity
- **Static power dissipation**
  - Due to leakage current – major paths are:
    - Subthreshold leakage
      - Exponentially dependent on Vdd, Vth, temperature
    - Gate leakage
      - Exponentially dependent on Vdd, Tox



Power vs. Energy

- **Important to distinguish between power and energy**
- **P = I * Vcc**
  - P – average power
  - I – average current
  - Vcc – supply voltage
- **E = P * T**
  - E – energy consumed
  - T – execution time
    - T = N * 1/f
    - N – number of cycles
    - f – clock frequency
- **Therefore**
  - $E \propto I * N$

Instruction-level Power Estimation Strategy

- **Develop an instruction-level energy model for the Blackfin processor (BF533 @ 1.2 V and 270 MHz, though our approach is re-targetable)**
  - **Core voltage operation between 0.8V and 1.4V from 0 to 756 MHz**
- **Leverage past work on instruction-level power profiling for embedded cores (Tiwari @ Princeton)**
  - Instruction-level estimation can be effective on cores with simple pipelines
- **We then build energy estimates, working with individual basic blocks, and then weight blocks based on the dynamic call graph traversal during program execution**

Instruction-level Power Estimation Strategy

- We consider variability due a configurable memory hierarchy
- We consider the impact of operand values and operand types on energy
- We consider environmental effects on measurements
- We will combine our instruction-level model with VisualDSP++ to provide power/performance framework

Instruction-Level Energy Modeling

Total Energy = Base Energy Cost + Inter-Instruction Effects

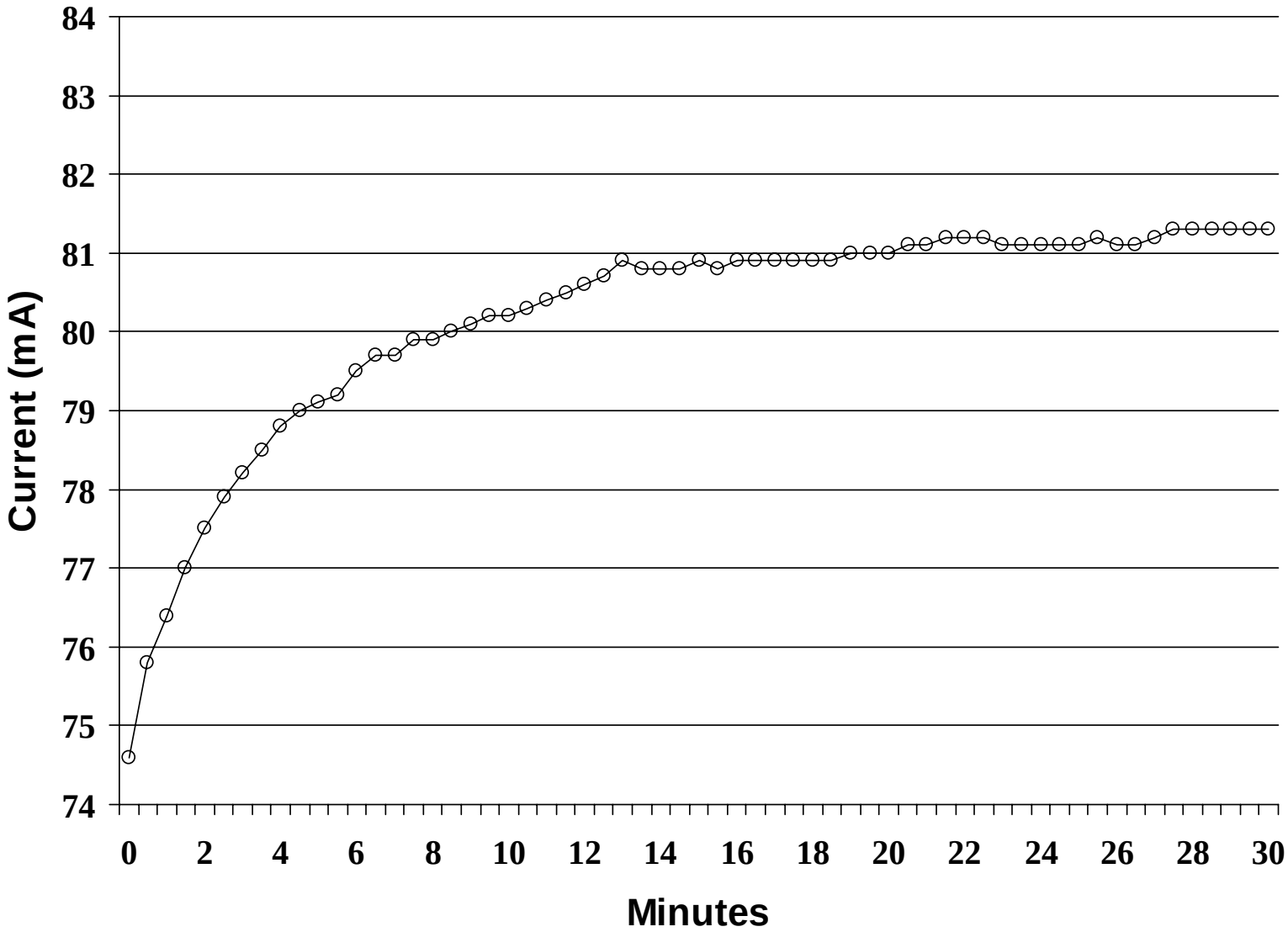
- **Base Energy Cost**
  - The energy cost to execute an individual instruction
- **Capture Base Energy Costs**
  - Construct loops containing several instances of the same instruction (now automated)
  - Measure the average current drawn while executing this loop
  - The base energy cost is directly proportional to this current, multiplied by the number of cycles needed to complete each instance of the instruction

Instruction-Level Energy Modeling

Total Energy = Base Energy Cost + Inter-Instruction Effects

- **Inter-Instruction Effects**
  - Energy contributions that are not considered in the base energy cost
  - Circuit state overhead
    - Added cost due to switching activity within the circuit when executing two different instructions in succession
    - Effect measured using a pair of different instructions in a loop and capturing the average current
  - Effects of resource constraints and delays
    - Common events - pipeline stalls, cache misses, write buffer stalls
    - These events increase the number of cycles required to complete an instruction
    - The average power per cycle often decreases, but the overall energy still increases due to the higher cycle count

Measurement Environment Warm-up



Impact of Operand Values

Instruction: r7 = r3 + r4;		
r3 Value	**r4 Value**	**Current (mA)**
0x1	0x1	93.8
0x3333	0x3333	94.7
0xFFFF	0xFFFF	95.6
0x33333333	0x33333333	95.6
0xFFFFFFFF	0xFFFFFFFF	97.5

Instruction	Initial Values	Current (mA)
r6 = -r3;	r3 = 0x90B	94.1
r3 = -r3;	r3 = 0x90B	108.5

- **Comments:**
  - **Input operand values have a significant impact on average current (range of 3.9 mA)**
  - **Power is dependent upon the number of bit flips performed in a cycle**
  - **Large variations in current are observed with changing destination register values**
  - **Presents challenges to our measurement assumptions**

Instruction Selection

Add

```
top_loop:
    r7 = r3 + r4;
    r7 = r3 + r4;
    r7 = r3 + r4;
    …
    jump top_loop;
```

Nop

```
top_loop:
    nop;
    nop;
    nop;
    …
    jump top_loop;
```

Combination

```
top_loop:
    r7 = r3 + r4;
    nop;
    r7 = r3 + r4;
    nop;
    …
    jump top_loop;
```

- **Average current**
  - Add: 94.7 mA
  - NOP: 90.9 mA
  - Combination: 108.7 mA
- **Comments:**
  - Circuit state overhead is significant (i.e., NOPs are not free)
  - Decode overhead is a major contributor to power consumption

Memory Configuration

- **Investigated current dissipation of L1 memory configured as SRAM vs. cache**
- **Cache overhead for Load instruction**
  - Instruction: 3.9 mA
  - Data: 11.8 mA
- **Comments:**
  - Cache maintenance operations increase current dissipation
  - Data cache consumes more current due to core layout and multi-port design

Example Program: Cache Disabled

```
r1 = [i0];
r7 *= r1;
r6 = r1 + r6 (ns);
r5 = r1 +|- r6;
[i1] = r7;
[i2] = r6;
[i3] = r5;
```

Measured

Average current: 116.4 mA

Number of Cycles: 9

E = 4.7 nJ

Estimated

E = 4.4 nJ

Percent Difference

5%

Example Program: Parallel Instructions

```
r1 = [i0];
r7 *= r1;
r6 = r1 + r6 (ns) || [i1] = r7;
r5 = r1 +|- r6 || [i2] = r6;
[i3] = r5;
```

Measured

Average current: 127.5 mA

Number of Cycles: 7

E = 4.0 nJ

Estimated

E = 3.8 nJ

Percent Difference

5%

Example Program: Multiple Basic Blocks

```
        r1.h = 0x5555;
        r1.l = 0xAAAA;
        r2.h = 0x3333;
        r2.l = 0xCCCC;
        jump label1;
label1:
        r7.h = r1.h*r2.h, r7.l = r1.l*r2.l;
        r6 = r1 & r2;
        r5 = ashift r1 by r2.l (s);
        jump label2;
label2:
        [i1++] = r7;
        [i1++] = r6;
        [i1++] = r5;
```

Measured

Average current: 114.2 mA

Number of Cycles: 20

E = 10.2 nJ

Estimated

E = 9.9 nJ

Percent Difference

2%

Summary

- Developed a retargetable method to produce an instruction-level energy model
- Constructed an instruction-level energy model for the Blackfin processor and used it to estimate programs with less than 6% error
- Developed a set of automated tools to drive test code generation and current measurements
- Studied the energy effects of the memory hierarchy, changes in operand values, and environmental factors

ANALOG DEVICES

NUCAR

Developing Power-Aware Strategies for the Blackfin Processor

Steven VanderSanden
David Kaeli
Northeastern University
Boston, MA

svanders@ece.neu.edu
kaeli@ece.neu.edu

Giuseppe Olivadoti
Richard Gentile
Analog Devices
Norwood, MA

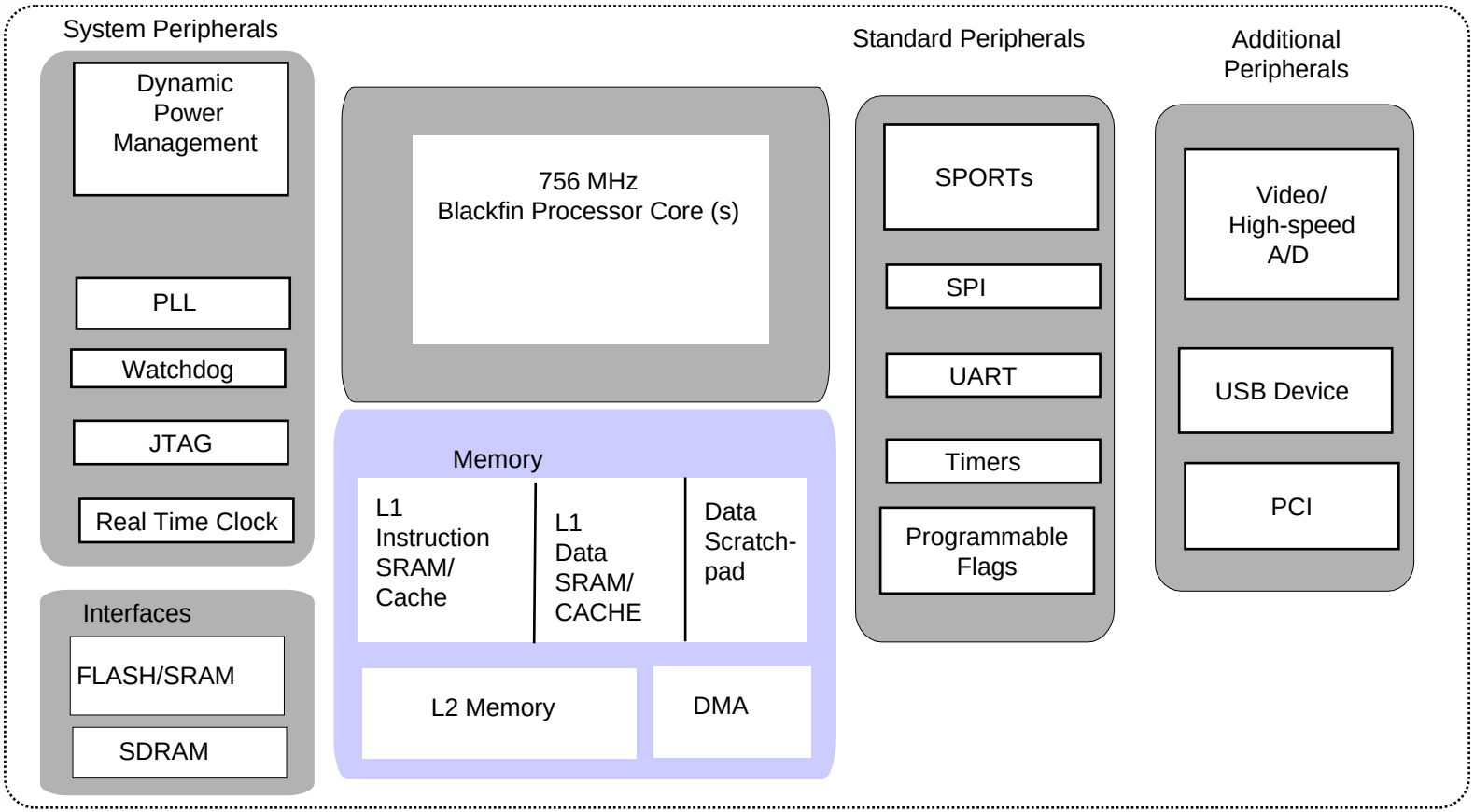
giuseppe.olivadoti@analog.com
richard.gentile@analog.com

The Need for Accurate Power Estimation

- Power management is particularly critical for portable embedded systems
- Power estimates will drive future core design decisions and impact battery design
- Present power estimation techniques utilize abstract architectural models
  - Good for predicting relative performance, but lack precision
  - Difficult to adapt across different core models
- Our work develops an instruction-level model
  - Profiles power/energy instruction-by-instruction
  - Utilizes statistical methods for estimating full program power
  - Methodology is portable to any embedded processor design
- This project is targeted to improve the power analysis capabilities of the ADI Blackfin family of processors and systems

Blackfin Processor

- **Built around Micro Signal Architecture, developed jointly with Intel Corp.**
- **Integrates DSP with features more typically found in an MCU**
- **Full suite of power management capabilities**



A DSP with a RISC instruction set and an MMU, an event controller and a wide range of peripherals

Our Approach

- Instruction-level power modeling
  - Computes energy budget by characterizing single instruction and inter-instruction power usage, combined with instruction execution time
  - Total energy = base energy cost + inter-instruction effects
- Profiling is used to construct a power/energy table for both base costs and inter-instruction effects
- We consider variability introduced by:
  - Operand types and operand values
  - Memory system configuration
  - Instruction selection
  - Measurement environment
- We then build energy estimates, working with individual basic blocks, and weight blocks based on the dynamic call graph traversal during program execution
- We are able to accurately estimate full program behavior (including memory access) within 6% of measured values