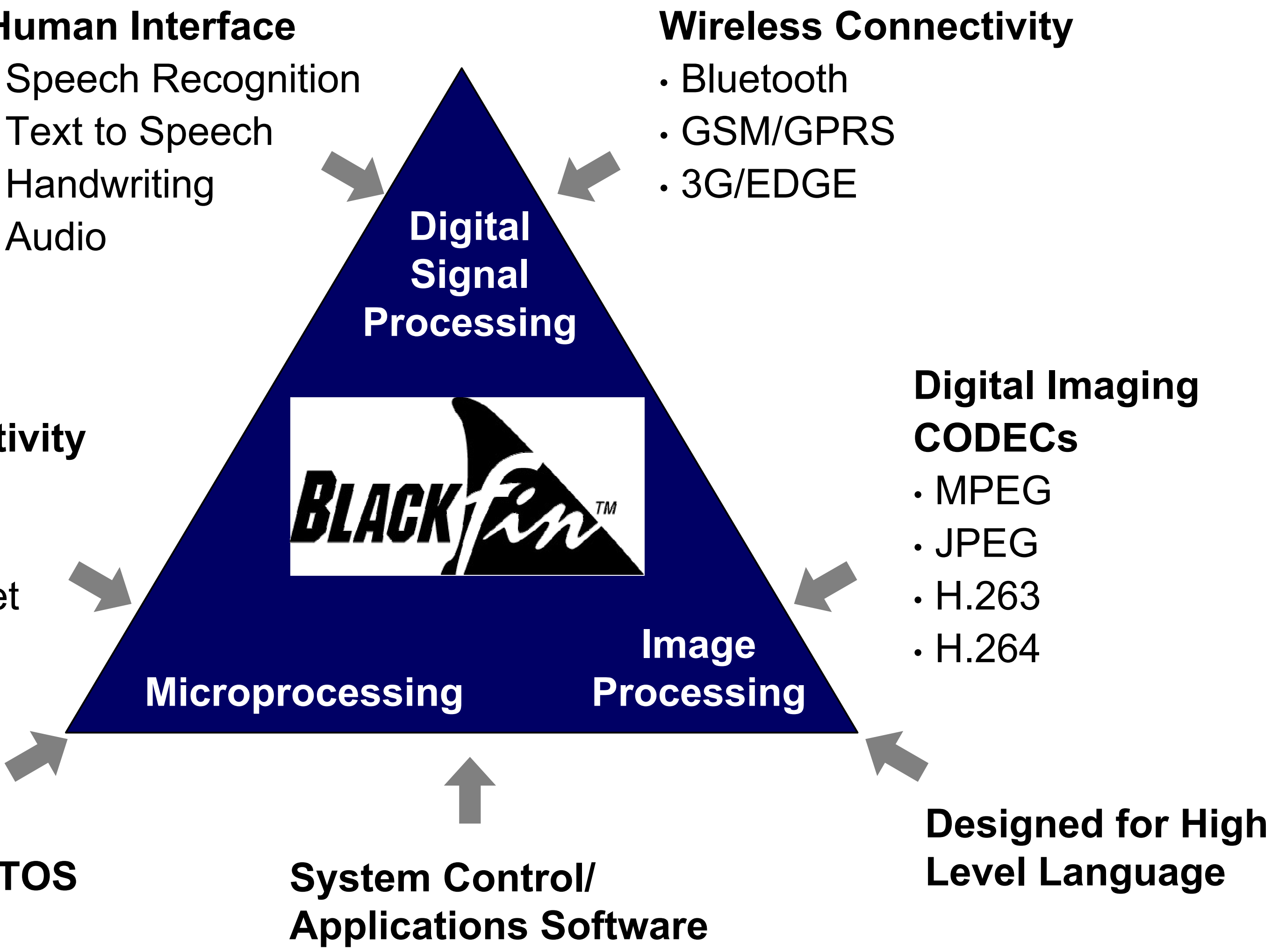
Introduction and Motivation

- 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?

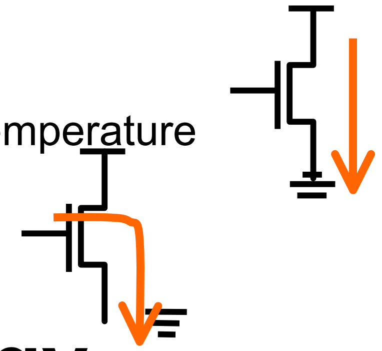
- **Speeds time-to-market and facilitates rapid product derivatives**
  - High-performance software target
  - Software-centric product development
- **Lowers BOM and R&D costs**
  - Eliminates redundant DSP, MCU and hardware accelerator blocks
  - Software reuse model enhances R&D productivity with 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

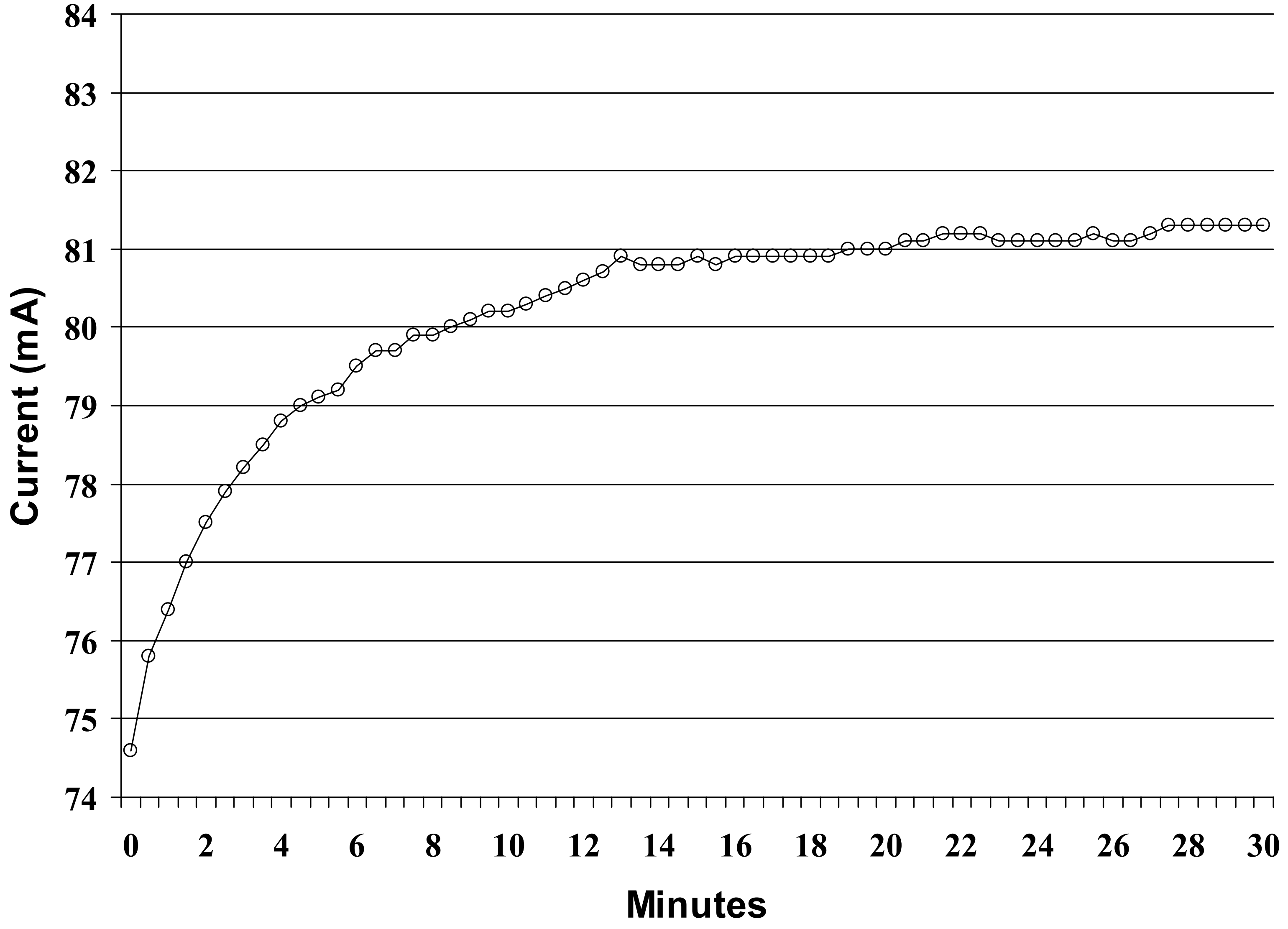
- **Base Energy Cost**
  - The energy cost to execute an individual instruction
- **Capture Base Energy Costs**
  - Construct loops containing several instances of the same instruction (now automated)
  - Measure the average current drawn while executing this loop
  - The base energy cost is directly 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