A Case for Opportunistic Embedded Sensing In Presence of Hardware Power Variability

L. Wanner, C. Apte, R. Balani, Puneet Gupta, and Mani Srivastava
University of California, Los Angeles
puneet@ee.ucla.edu

variability.org
nesl.ee.ucla.edu
nanocad.ee.ucla.edu
Outline

• A Variability Primer
• Hardware-Software Interface in Presence of Variability
• Variability in Modern Embedded Processors
• An Example Variability-Aware Software Stack
• Conclusions
Technology Scaling Problems

Figure courtesy Synopsys Inc.
Variability: Consequence of Shrinking Dimensions

Projections of Variability (ITRS)

Error is just *one* manifestation of variability!
Sources of Variability

Frequency variation in an 80-core processor within a single die in Intel’s 65nm technology

Semiconductor Manufacturing
WYS *is not* WYG

Frequency variation in five 512 MB DDR2-533 DRAM parts [Hanson07]

Vendor Differences
Multi-sourcing of parts

Variation in $P_{\text{sleep}}$ with temperature across five instances of an ARM Cortex M3 processor

Ambient Conditions
Extreme voltage/temperature environments especially for sensors

Normalized frequency degradation in 65 nm due to NBTI [Zheng09]

Aging
Temporal stress, e.g., due to NBTI
Outline

• A Variability Primer
• Hardware-Software Interface in Presence of Variability
• Variability in Modern Embedded Processors
• An Example Variability-Aware Software Stack
• Conclusions
The Hardware-Software Boundary

Idealization: hardware has rigid specifications

Application ••• Application

Operating System

Hardware Abstraction Layer (HAL)
The Hardware-Software Boundary

Practice: over-design & guard-banding for illusion of rigidity

Hardware Abstraction Layer (HAL)

Operating System

Application ••• Application

20X in Power
50% in Performance
The Cost of “Hiding” Variability

Variability Expedition

variability.org

130nm        90nm       65nm     45nm       32nm      22nm                  post-silicon

nominal scaling

variability aware software

guard-banded scaling
A New Hardware-Software Interface

Time or part

minimal variability handling in hardware

Operating System

Hardware Abstraction Layer (HAL)

Application

Application

Traditional Fault-tolerance

Opportunistic Software

Underdesigned Hardware

Variability Expedition
Opportunistic Software on Underdesigned Hardware

• Hardware signature measurement
  • One time for process/vendor variation
  • Periodic for ambient/aging.

• Advantages
  • Hardware can avoid overdesign as well as self-healing → lowered cost
  • Software leverages hardware maximally
Outline

• A Variability Primer
• Hardware-Software Interface in Presence of Variability
• Variability in Modern Embedded Processors
• An Example Variability-Aware Software Stack
• Conclusions
Variability in Contemporary Embedded Processors

Cortex M3 Active Power (Room Temperature)

Atmel SAM3U4E Cortex M3 Active Mode, 4MHz Internal Oscillator Room Temperature
Variability in Contemporary Embedded Processors

Cortex M3 Sleep Power (Room Temperature)

Atmel SAM3U4E Cortex M3 Sleep Mode, 32KHz Slow Oscillator Room Temperature
Sleep power vs. Temperature

Atmel SAM3U4E Cortex M3
Sleep Mode, 32KHz Slow Oscillator
Analytical Modeling of Sleep Power

• Sources of static (sleep) power:
  1. Sub-threshold Leakage
  2. Gate Leakage
  3. Reverse Biased Junction Leakage
  4. Gate Induced Drain Leakage

• Sleep power model (derived from BSIM4 compact device model)

\[ P_{\text{sleep}} = V_{dd} \left( AT^2 e^{B/T} + I_{gl} \right) \]

• A and B are technology-dependent constants
• \( I_{gl} \) is the temperature-independent gate leakage current
• T is the core temperature.
Measured vs. Modeled Sleep Current

These calibrated models are the *hardware variability signatures* passed to the software stack.
Outline

• A Variability Primer
• Hardware-Software Interface in Presence of Variability
• Variability in Modern Embedded Processors
• An Example Variability-Aware Software Stack
• Conclusions
A Software Stack for Variability-aware Duty-cycling

Scenario 1

- App
  - timer
- Poll $P_{\text{Sleep}}$
- $P_{\text{Sleep}}$

Scenario 2

- App
  - handler
- Events of Interest (type, severity)
- Variability Event

Scenario 3

- App
- $\text{dutycycling scheduler}$

Hardware Signature Inference

On/Off
- Read Sample
- Sampling Configuration

Sample, Event, Time-series...

Many Sensors: $P_{\text{Sleep}}$, $P_{\text{active}}$, Memory Speed, Temp, Battery,...
Energy-Aware Operating through Duty-Cycling

- Embedded sensing systems are typically duty cycled
  - Systems “sleep” for most of the time
  - “Wake up” periodically to acquire data or respond to events

- Often, duty cycle rate is very small (e.g. < 1%), so that the energy consumed in the sleep state accounts most of the energy consumption
Variability-Aware Duty Cycling

- The maximum duty cycle rate is a function of
  - Available Energy
  - Lifetime required for the application
  - Active mode power
  - Sleep Mode Power

\[
\text{MaxDutyCycle} = \frac{\text{EnergyBudget}}{\text{LifeTime}} - \frac{P_{\text{sleep}}}{P_{\text{active}} - P_{\text{sleep}}}
\]

- Sleep power (and active power, to a lesser extent) changes according to instance and temperature-dependent variation
- Implemented variation-aware duty cycling scheme in TinyOS
Implications of Variation for Duty Cycling

P1 can acquire 51% more data than P3

Active Mode: 48 MHz
Sampling Task: 10 s
Battery: 2xAA (5.4 A-h)
Lifetime: 20000 hours

Temperature + Instance performance improvement = 80%!
Opportunism Advantage vs. Lifetime

P2 can acquire 70% more data than P3

Active Mode: 48 MHz
Sampling Task: 10 s
Battery: 5.4 A-h
Room Temperature
Projecting Opportunism Benefit into Future

The graph shows the comparison between Conventional Non-Adaptive Sensor and Sensor with Signature Measurement and Variability Prediction. The y-axis represents Quality of Sensing, ranging from 0.01 to 1. The x-axis represents Technology (nm), ranging from 22 to 130. The graph indicates that the Sensor with Signature Measurement and Variability Prediction has a significantly higher benefit compared to the conventional sensor, as marked by the label "1.8x current benefit (measured)".
Outline

• A Variability Primer
• Hardware-Software Interface in Presence of Variability
• Variability in Modern Embedded Processors
• An Example Variability-Aware Software Stack
• Conclusions
Conclusions

• Growing variability $\rightarrow$ unmanageably high cost of preserving rigid hardware-software interface $\rightarrow$ Need for a software stack that opportunistically adapts to “as measured” hardware characteristics
  • Self-monitoring hardware as opposed to self-healing

• Variability-Aware opportunistic sensing systems
  • No adaptation $\rightarrow$ conservative specifications $\rightarrow$ untapped energy resources
  • Proof-of-concept variability-aware duty cycle scheduler
    • 1.8x improvement in quality of sensing for current generation hardware
      • Benefits will increase with scaling of technology

• Ongoing work (plenty!)
  • Alternative methods for exposing variation to software layers
  • Cheap variation monitoring strategies
  • Implications for hardware design
  • See the new NSF Variability Expedition (http://variability.org) with the goal of a fluid hardware-software interface
Thank you!

variability.org
From Crash-and-Recover to Sense-and-Adapt

- Variability manifestations:
  - faulty cache bits
  - delay variation
  - power variation

- Variability signatures:
  - cache bit map
  - cpu speed-power map
  - memory access time
  - ALU error rates

- Underdesign Mechanisms:
  - stochastic processor
  - fluid hw constraints
  - application intent

- Do Nothing:
  (Elastic User, Robust App)
  (Disabling parts of the cache, Changing V-f)

- Change Hardware Operating Point:
  (Codec Setting, Duty Cycle Ratio)

- Change Algorithm Parameters:
  (Alternate code path, Dynamic recompilation)

- Change Algorithm Implementation:
  (Dynamic linking to new library module)

- Change to Algorithm with Different Characteristics:
  (Dynamic linking to new library module)
Measuring Hardware Signatures

• Production test (static signatures)
  • Explore low-cost methods of rich, fine-grained binning
  • Spatial by leveraging correlations
  • Non-conventional axes such as error behaviors
  • Runtime sensing (dynamic signatures)

• Monitors: simple low-overhead test structures (e.g., ROs)
  • Error detection: E.g., Razor. Can allow direct tradeoff between error rate and power. May need offline calibration
  • Online Self-test: may be useful to detect functional problems
  • Software inference: insert test operations within software not requiring any hardware support.

• Optimizing measurement overheads
  • Use (compiler-inserted) application directives to change monitoring accuracy
  • Leverage alternative application configurations in deriving the optimal signature measurement points
  • Use smart, adaptive sampling methods
  • Example: H.264 optimal frequency
  • Signature sampling