Software strategies to handle hardware variability

IC Seminar Series – August 21, 2015
Lucas Wanner – IC/Unicamp
lucas@ic.unicamp.br
www.lucaswanner.com
Summary

Contemporary hardware suffers from variability, particularly in power consumption.

OS + Apps can adapt to variations, with benefits in quality of service and energy efficiency.
Outline

The variability problem
Power variation
Physically-Coupled Software
The Variability Problem

Scaling → reduced control over manufacturing

Variations across parts, time, and ambient conditions

Source: Synopsys
Variations across parts, time, and ambient conditions

- Maximum frequency
- Faults / Errors
- Power Consumption

Variability (%)

<table>
<thead>
<tr>
<th>Year</th>
<th>Total Power</th>
<th>Static (Sleep) Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2012</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2018</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2021</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2024</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Source: ITRS Roadmap
Variations in Contemporary Processors
ARM Cortex-M3 Active Power (Room Temperature)

Atmel SAM3U4E Cortex M3 Active Mode, 4MHz Internal Oscillator

~5x variation

Power (mW)

Processor Copy

(datasheet measured)
ARM Cortex-M3 Active Power (20-60°C)

~10% variation

Temperature (°C)

Processor Copies

Power (mW)

~10% variation
ARM Cortex-M3 Sleep Power (Room Temperature)

![ARM Cortex-M3 Sleep Power Graph](image)

- Processor Copy: P1, P2, P3, P4, P5, P6, P7, P8, P9, P10
- Power (μW)
- Measured vs Datasheet
- 8x variation
- Atmel SAM3U4E Cortex M3
  - Active Mode, 4MHz Internal Oscillator
ARM Cortex-M3 Sleep Power (20-60°C)

14x variation

Power (μW)

Processor Copies

Temperature (°C)
Some variation numbers from the literature

**Intel Core i5: 12-17%**
Balaji, HotPower, '12

**DDR3: 32%**
Gottscho, ESL 4(2), '12
The Software/Hardware Interface

Idealization: hardware has rigid specifications

Practice: guard-banding for illusion of rigidity
Physically-Coupled Software

Adaptation knobs
Parameter control
Dynamic recompilation
Hardware resources
Scheduling
Functional Adaptation

Elastic Quality

Software

App
App
App

Operating System

Hardware Abstraction Layer (HAL)

Hardware + Environment

Lucas Wanner – IC/Unicamp
Software strategies to handle hardware variability
13
Outline

The variability problem
Power variation
Physically-Coupled Software

14x static power variation
Sense and expose
Elastic quality

Work scheduling
Duty cycled embedded systems

\[ \Delta = \frac{c}{p} \]

\[ \uparrow \Delta \Rightarrow \uparrow \text{Quality} \]

\[ c \uparrow \quad p \downarrow \]
while(1) {
    do_something(duration);
    sleep(time);
}

What's the allowable $\Delta$ when power consumption is variable?

Can the OS help applications adapt?
Determining $\Delta$ for each instance

Objective: maximize active time for each instance subject to energy capacity and lifetime
Knobs for $\Delta$ control in VaRTOS

for \textit{knob} do
  sleep (constant - \textit{knob})  // computation time
  sleep (constant - \textit{knob})  // period

\textit{knob}: app variable shared with OS

↑ knob value ⇒ ↑ $\Delta$, ↑ quality
Knobs for $\Delta$ control in VaRTOS

```
xTaskCreate(..., &task_knob, min, max, priority);
```

↑ knob value ⇒ ↑ $\Delta$, ↑ quality
Δ control in VaRTOS

1) Requirements

**App:** knobs, lifetime, temperature profile

**Hardware:** power, temperature

Rough histogram

2) Model Training

T $\rightarrow$ PA, PS

knob $\leftrightarrow$ time

40 points: 2.5% error

3) Optimization

Maximize $\Delta$

Assign knob values

LP + Greedy Opt.
Greedy optimization of knob values

Global Δ

Global Utility

active

sleep

Task 1

Task 2

Task 3
Recap: Choices in determining $\Delta$

- Underestimated (Datasheet)
- Optimal (Variability-Aware)
- Sub-Optimal
- Guard-banded (worst case)
- Infeasible
Results: lifetime reduction with datasheet spec $\Delta$

Lifetime reduction (days)

Processor Copy

Average: 55 days

Lifetime: 1 year, Battery: 5400 mAh
Temperature: Stovepipe Wells, CA, 2009
Results: energy untapped by worst-case $\Delta$

Average: 63%

Lifetime: 1 year, Battery: 5400 mAh
Temperature: Stovepipe Wells, CA, 2009
Results: improvement over worst-case $\Delta$

Average: 22x

Average for multiple temperature profiles: 6x

Improvement (%)

Processor Copy

P1 P2 P3 P4 P5 P6 P7 P8 P9 P10

Lifetime: 1 year, Battery: 5400 mAh
Temperature: Stovepipe Wells, CA, 2009
VaRTOS vs. Oracle

Utility vs. Oracle (%)

- Best
- Nominal
- Worst

Temperature

Best
- 80
- 84
- 88
- 92
- 96
- 100

Nominal

Worst
- 80
- 84
Outline

The variability problem
- Power variation
- Physically-Coupled Software

Work scheduling

Functional adaptation

14x static power variation
- Sense and expose
- Elastic quality

6x improvement over worst-case
- 94% of oracle utility
How to graduate from college

Optimization problem

\[
\begin{align*}
\text{max} \quad & \text{party time} \\
\text{s.t.} \quad & \text{family pressure} < 10 \text{ atm} \\
& \text{graduation time} < \infty \\
& \text{cash} \geq R\$ 2 \\
& \text{job prospects} > 0
\end{align*}
\]

\text{w.l.o.g. party = gaming}

party time = f( courses chosen, study time, internships, lab work, connections)
Functional Adaptation

$f(x)$: cost and quality tradeoff

- best quality
- good quality
- approximation
- give up
ViRUS: Virtual Function Replacement Under Stress

**Function** $f()$

- **Best quality**
- **Good quality**
- **Approximation**
- **Give up**

### Table: Function Quality

<table>
<thead>
<tr>
<th>function</th>
<th>sensor</th>
<th>range</th>
<th>quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>temperature</td>
<td>[0, 40)</td>
<td>best, good</td>
</tr>
<tr>
<td>$f$</td>
<td>temperature</td>
<td>[40, 100)</td>
<td>approx</td>
</tr>
<tr>
<td>$g$</td>
<td>battery</td>
<td>[20, 100)</td>
<td>good</td>
</tr>
<tr>
<td>$g$</td>
<td>battery</td>
<td>[0, 20)</td>
<td>give up</td>
</tr>
<tr>
<td>$g$</td>
<td>temperature</td>
<td>[0, 60)</td>
<td>best</td>
</tr>
<tr>
<td>$g$</td>
<td>temperature</td>
<td>[60, 100)</td>
<td>good, approx</td>
</tr>
</tbody>
</table>
Memory overhead of multiple versions

- exp
- log
- pow
- sin
- cos
- tan
- lgamma

Memory Usage (KB)

- double
- single
- fast
- faster

full lib: 10% overhead
Energy usage across versions

- **blackscholes**: 50%
- **swapions**: 30%
- **whetstone**: 55%
Quality across versions: blackscholes

- NRMSE: 4.2%
- MAPE: 40%

- NRMSE: 0.003%
- MAPE: 0.62%

- NRMSE: 0.000005%
- MAPE: 0.00002%

Lucas Wanner – IC/Unicamp
Software strategies to handle hardware variability
Constant Quality vs. Constant Energy Policy

Policy: lower quality when power is high

12-20% savings
X-Ware: Mutant Computing

- Big Core
- Little Core
- Accelerator
- Cloud
- DSP

App

\( f() \)
CareDroid: from variability to context

Adapt functionality to match dynamic characteristics

CareDroid Dalvik extension for context aware apps: 10x more efficient, 50% fewer lines of code
Outline

The variability problem
Power variation
Physically-Coupled Software

14x static power variation
Sense and expose
Elastic quality

Work scheduling

6x improvement over worst-case
94% of oracle utility

Functional adaptation

20% savings, negligible quality loss
Efficient context adaptation
A career in physically-coupled software

Software controls the *physical* world!

↑ far from end users

↑ close ties to industry

… but with large impact potential

… not to “flavor of the month” tech
Summary

Contemporary hardware suffers from variability, particularly in power consumption.

OS + Apps can adapt to variations, with benefits in quality of service and energy efficiency.

14x variation in ARM processors [HotPower’10, DATE’11]

Scheduling Adaptation [TVLSI’13, TECS’15]

Mutant Computing [RSP’15]

Functional Adaptation [HotPower’14, Mobicom’15]
Software strategies to handle hardware variability

IC Seminar Series – August 21, 2015
Lucas Wanner – IC/Unicamp
lucas@ic.unicamp.br
www.lucaswanner.com

Thank you!