Adaptive Resource Management through Self-Awareness+

Nikil D. Dutt

Center for Embedded and Cyber-Physical Systems (CECS)
University of California, Irvine
dutt@uci.edu
http://www.ics.uci.edu/~dutt
https://duttgroup.ics.uci.edu

+Joint work with Tiago Mück, Bryan Donyanavard, Kasra Moazzemi, Amir Rahmani, Santanu Sarma, Biswadip Maity

Dutt Research Group

Research Partially Supported by the National Science Foundation
Self-Awareness

Self-awareness

From Wikipedia, the free encyclopedia

Not to be confused with Self-concept, Self-consciousness, Self-perception, or Self image.

This article has multiple issues. Please help improve [hide] it or discuss these issues on the talk page.

- This article may require cleanup to meet Wikipedia's quality standards. (March 2009)
- This article needs attention from an expert on the subject. (May 2009)

Self-awareness is the capacity for introspection and the ability to recognize oneself as an individual separate from the environment and other individuals.

The mirror test is a simple measure of self-awareness.
Computational Self-* Properties

• **Self-Awareness** [Hinchey2006]: System is aware of its *self* states and behaviors

• **Context-Awareness** [Parashar 2005]: System is aware of context – i.e., its *operational environment*

• **Self-configuring** -> capability of reconfiguring automatically

• **Self-healing** [Robertson2005] -> *self-diagnosing and self-repairing*

• **Self-optimizing** -> capability of *self-tuning* or **Self-adjusting**

• **Self-protecting** -> capability of detecting dangerous outcomes (e.g. security breaches) and recovering from their effects
Outline

• Computational Self-Awareness

• Why Self-Aware Chips?

• Cross-Layer Sensing & Actuation

• Towards Self-Aware Chips

• Supervisory Control & Coordination
Why On-Chip Self-Awareness (1)?
Why On-Chip Self-Awareness (1)?

Variability-induced challenges

Manufacturing

Frequency variation across 80-core processor within a single die in Intel's 65nm node at 1.2V and 0.8V [Dighe10]

[Source: NSF Variability Expedition Project]
Why On-Chip Self-Awareness (1)?

**Variability-induced challenges**

- **Environment**
  - Sleep Power (mW) vs. Temperature (°C)
  - Variation in sleep power with temperature across 5 ARM Cortex M3 processor instances

- **Manufacturing**
  - Frequency variation across 80-core processor within a single die in Intel's 65nm node at 1.2V and 0.8V [Dighe10]

Frequency variation: 25% and 50% variation in frequency across different cores at 7.3 GHz and 5.7 GHz.

Temperature impact: Significant variation in sleep power with increasing temperature.
Why On-Chip Self-Awareness (1)?

Variability-induced challenges

Applications: varying compute, memory, communication

Manufacturing

Environment

Applications:

- varying compute
- memory
- communication

Variability-induced challenges:

- Frequency variation across 80-core processor within a single die in Intel's 65nm node at 1.2V and 0.8V [Dighe10]
- Psleep variation with temperature across 5 ARM Cortex M3 processor instances

Triple Whammy!
Why On-Chip Self-Awareness (2) ?

• Chips must adapt to:
  – Performance, Power, Resilience, Security, ….

• Provide Guarantees

• Dynamically manage multi-dimensional trade-offs
  – Performance, Power/Energy, Thermal, …..
  – QoS, TDP, Wear-out, ….

Exploit Computational Self-Awareness
Outline

• Computational Self-Awareness

• Why Self-Aware Chips?

• Cross-Layer Sensing & Actuation

• Towards Self-Aware Chips

• Supervisory Control & Coordination
Cross-Layer Physical/Virtual Sensing & Actuation

- **Applications**
- **Operating System**
- **Network/Bus Communication Architecture**
- **Hardware Architecture**
- **Device/Circuit Architecture**

**Actuation (Act)**
- AA
- AO
- AN
- AH
- AC

**Adaptive Control (Decide)**

**Sensors (Observer)**
- SA
- SO
- SN
- SH
- SC
Examples of Virtual Sensors and Actuators Across Layers of CPSoC

<table>
<thead>
<tr>
<th>Layers</th>
<th>Virtual/Physical Sensors</th>
<th>Virtual/Physical Actuators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Execution Time, Workload Power, Energy,</td>
<td>Loop perforation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Algorithmic Choice</td>
</tr>
<tr>
<td>Operating System</td>
<td>System Utilization</td>
<td>Task Allocation, Scheduling, Migration, Duty Cycling</td>
</tr>
<tr>
<td></td>
<td>Peripheral States</td>
<td></td>
</tr>
<tr>
<td>Network/Bus Communication</td>
<td>Bandwidth; Packet/Flit status; Channel Status, Congestion, Latency</td>
<td>Adaptive Routing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dynamic Bandwidth Allocation Ch. no and direction</td>
</tr>
<tr>
<td>Hardware Architecture</td>
<td>Cache misses, Miss rate; access rate; IPC, Throughput, ILP/MLP, Core asymmetry</td>
<td>Cache Sizing; Reconfiguration, Resource Provision</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Static/Dynamic Redundancy</td>
</tr>
<tr>
<td>Circuit/Device</td>
<td>Circuit Delay, Aging, leakage Temperature, oxide breakdown</td>
<td>DVFS, DFS, DVS ABB, Clock and Power-gating</td>
</tr>
</tbody>
</table>

Examples of Virtual Sensors and Actuators Across Layers of CPSoC
Outline

• Computational Self-Awareness
• Why Self-Aware Chips?
• Cross-Layer Sensing & Actuation
• Towards Self-Aware Chips
• Supervisory Control & Coordination
Self-Reflection & Introspection

- Ability to create a **self-model (introspect)**

- Ability to model their own body/structure (usually known **self-modeling**)

- Ability to model their own **behavior**

- **Metacognition capacity**: ‘models one’s own thinking’, ‘think about thinking’

- **System with two/multiple minds**: one being modeled and other doing modeling
Reflex vs Reflect

Reflexive, Reactive

- Actions driven solely on external feedback
  - E.g., our autonomic nervous systems
Reflex vs Reflect

Reflexive, Reactive

- Actions driven solely on external feedback
  - E.g., our autonomic nervous systems

Reflection, Introspection

- Consider past and future outcomes
  - E.g., planning, strategies, policies, …
Towards Self-Aware Chips:
What we do now

Reflexive, Reactive

QoS/Goals

Simple Adaptation

Adaptive Polices / Controller / Governor

Self-monitoring chip

Temperature > 80C

[Sarma14, CODES+ISSS14]

Output
Towards Self-aware chips

*Beyond simple reactive models*

There is unusual activity in Core 27

System Behavior (Model Building)

Adaptive Policies / Controller / Governor

Self-Aware Adaptation

Simple Adaptation

Self-monitoring and **Self-modeling**

*QoS/Goals*

Input

Measurement

Output

Temperature > 80°C

[Sarma14, CODES+ISSS14]
Towards Self-aware chips

Beyond simple reactive models

Self-monitoring and Self-modeling

Adaptive Polices / Controller / Governor

System Behavior (Model Building)

Reflection, Introspection

QoS / Goals

Simple Adaptation

Self-monitoring chip

[Sarma14, CODES+ISSS14]
Today: “Reflexive” Resource Management

- Dynamic Voltage/Frequency scaling (DVFS)

  - Scale core frequency
  - Core load increased

- Observe-Decide-Adapt approaches
RefleXive vs RefleCTive Resource Management

- **RefleXive ODA**: decisions taken based on
  - past observations (purely reflexive) OR
  - predictions made from past observations
RefleXive vs RefleCTive Resource Management

- **RefleXive ODA**: decisions taken based on
  - past observations (purely reflexive) OR
  - predictions made from past observations

- **RefleCTive approach**: considers future events that could happen in the next iteration of the ODA loop
Adaptive Resource Management

- Use concept of **reflection**
  - **Reflection**: change your actions based on both external feedback and **introspection** (i.e., self-assessment)
Adaptive Resource Management

- Use concept of reflection
  - Reflection: *change your actions based on both external feedback and introspection (i.e., self-assessment)*

- **Reflective resource management combines:**
  - Current system state assessed from sensing information (e.g., readings from performance counters, power sensors, etc.)
  - Models to predict the behavior of other system components before performing an action
MARS: Our coordination approach

• Coordination though **reflective** resource management
  – **MARS**: Middleware for Adaptive Reflective Systems

What if I change x264 to a big core?

- Perf. will increase by 2x
- Power will increase by 4x
Do we have room for reflection?

- Systems actuations happen at different timescales

```
<table>
<thead>
<tr>
<th>Task mapping</th>
<th>DVFS</th>
<th>Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which core executes a task? (100’s – 10’s ms)</td>
<td>What’s the next frequency? (10’s ms – 10’s us)</td>
<td>Should I preempt a task? (&lt;1 us)</td>
</tr>
</tbody>
</table>
```

Coarser grained actuation

Finer grained actuation

- Some actuations happen quickly with little room for reasoning
- Other actuations can occur on larger timescales
  - Task mapping, Wear-leveling (for aging)....
MARS middleware for reflective resource management

Reflective System Model
- Partitioning models
- Kernel models
- HW models

Resource Management Policies
- Policy model 0
- Policy model n

Policy 1
- Policy n

System stack
- Applications
  - Linux kernel
    - Scheduling, resource allocation, DVFS, etc
  - HMP HW platform

Actuators
- AA
- PA
- KA

Sensors
- AS
- PS
- KS
MARS middleware for reflective resource management
MARS middleware for reflective resource management

Policy

Resource Management Policies

Policy 1

\[ \cdots \]

Policy n

Actuators

Applications

System stack

Linux kernel

Scheduling, resource allocation, DVFS, etc

HMP HW platform

Sensors

AS

PA

KA

KA

Decisions

Sensed data
MARS middleware for reflective resource management

Reflective System Model
- Partitioning models
- Kernel models
- HW models

Resource Management Policies
- Policy model 0
- Policy model n

Actuators
- AA
- PA
- KA

Sensors
- AS
- PS
- KS

System stack
- Applications
- Linux kernel
  - Scheduling, resource allocation, DVFS, etc
- HMP HW platform

Sensed data
Example: Reflective Task Mapping

Reflective System Model

Resource Management Policies

Task Mapping

System stack

Applications

Linux kernel

Scheduling, resource allocation, DVFS, etc

HMP HW platform

Actuators

KA

KA

PA

AS

PS

KS

Sensors

Task

Mapping

Reflective System Model

Resource Management Policies

Task Mapping

System stack

Applications

Linux kernel

Scheduling, resource allocation, DVFS, etc

HMP HW platform

Actuators

KA

KA

PA

AS

PS

KS

KS

Sensed data

Reflective middleware
Example: Reflective Task Mapping

Reflective System Model

Scheduler Model

Resource Management Policies

Task Mapping

Actuators

Sensors

System stack

Applications

Linux kernel

Scheduling, resource allocation, DVFS, etc

HMP HW platform

What if task migrated?
Example: Reflective Task Mapping

Reflective System Model

Scheduler Model

Freq. gov. Model

What if task migrated?

What if cpu utilization Increases?

Resource Management Policies

Task Mapping

Sensed data

System stack

Applications

Linux kernel

Scheduling, resource allocation, DVFS, etc

HMP HW platform

Actuators

AA

PA

KA

KA

Sensors

AS

PS

KS

KS

Reflective middleware

https://duttgroup.ics.uci.edu
Example: Reflective Task Mapping

Reflective System Model
Scheduler Model
Freq. gov. Model
HW Perf. model

Resource Management Policies
Task Mapping

Actuators
AA
PA
KA

System stack
Applications
Linux kernel
Scheduling, resource allocation, DVFS, etc

HMP HW platform

Sensors
AS
PS
KS
KS

Sensed data

What if task migrated?
What if cpu utilization Increases?
What if freq increases?
Example: Reflective Task Mapping

Reflective System Model
- Scheduler Model
  - What if task migrated?
- Freq. gov. Model
  - What CPU utilization increases?
- HW Perf. model
  - What if freq increases?
  - Task perf is X

Resource Management Policies
- Task Mapping

System stack
- Applications
  - Linux kernel
    - Scheduling, resource allocation, DVFS, etc
- HMP HW platform

Actuators
- AA
- PA
- KA

Sensors
- AS
- PS
- KS

Sensed data

Reflective System Model

Resource Management Policies

HMP HW platform

Applications

Linux kernel

Scheduling, resource allocation, DVFS, etc
Example: Reflective Task Mapping

- Reflective System Model
  - Scheduler Model
  - Freq. gov. Model
  - HW Perf. model
- Task Mapping
  - What if task migrated?
  - What cpu utilization Increases?
  - What if freq increases?
  - Task perf is X
- Resource Management Policies
- New task-to-core assignment

System stack
- Applications
  - Linux kernel
    - Scheduling, resource allocation, DVFS, etc
- Sensed data
  - Sensors
    - AS
    - PS
    - KS
- Actuators
  - AA
  - PA
  - KA
  - KS

Reflective middleware

Copyright © 2018 Dutt Research Group
SPARTA improvements

- 8-core big.LITTLE Exynos SoC
  - 4x big
  - 4x LITTLE
- Workload mixes (4 tasks each)
  - Mibench
  - x264 (Parsec)
- SPARTA vs Linux's GTS
- Avg. improvements of 16% in energy efficiency without performance degradation

Donyanavard, B., Mück, T., Sarma, S., & Dutt, N., SPARTA: Runtime Task Allocation for Energy Efficient Heterogeneous Many-cores. CODES+ISSS ’16
MARS: Middleware for Adaptive Reflective Computer Systems

- Framework and tools for developing reflective resource/power management policies
  - Use models to predict system behavior
  - Enable easy adaptation to runtime changes
  - Case studies show promise

MARS framework is open source

https://github.com/duttresearchgroup/MARS
Outline

• Computational Self-Awareness

• Why Self-Aware Chips?

• Cross-Layer Sensing & Actuation

• Towards Self-Aware Chips

• Supervisory Control & Coordination
Goals and Autonomy

Goal

- Single, straightforward objective
  - E.g., hit the pin
Goals and Autonomy

Goal

- Single, straightforward objective
  - E.g., hit the pin

Model Imperfection

- What happens when we introduce unpredictability?
  - E.g., balls with different sizes, shapes, weights; uneven or damaged surfaces
Goals and Autonomy

Supervision

- Constrain behavior so we are always headed toward the goal
  - E.g., bumpers
Goals and Autonomy

Supervision

- Constrain behavior so we are always headed toward the goal
  - E.g., bumpers
- Bonus: what about when we have more complex or multiple goals?
SPECTR: Our Supervisory Approach

- Autonomy and robustness through supervisory control

SPECTR: Our Supervisory Approach

- Autonomy and robustness through supervisory control

Low-level controllers satisfy objective

SPECTR: Our Supervisory Approach

• Autonomy and robustness through supervisory control

Supervisor bounds behavior of controllers, manages goal

Low-level controllers satisfy objective

SPECTR: Our Supervisory Approach

- Autonomy and robustness through supervisory control
- Case Study*

Supervisor bounds behavior of controllers, manages goal

Low-level controllers satisfy objective

SPECTR: Our Supervisory Approach

- Autonomy and robustness through supervisory control
- Case Study*

Supervisor bounds behavior of controllers, manages goal

Low-level controllers satisfy objective

ODROID-XU3 platform contains an Exynos 5422 Octa-core SoC

Example: Power Capping

- Specify desired behavior via accepted and forbidden states to restrict the behavior of the system.
Example: Power Capping

- Specify desired behavior via accepted and forbidden states to restrict the behavior of the system.

If power is in safe region, prioritize QoS.
Example: Power Capping

- Specify desired behavior via accepted and forbidden states to restrict the behavior of the system.

If power exceeds threshold,
Example: Power Capping

- Specify desired behavior via accepted and forbidden states to restrict the behavior of the system.

If power exceeds threshold, reduce power.
Example: Power Capping

- Specify desired behavior via accepted and forbidden states to restrict the behavior of the system.

If power exceeds threshold, reduce power...until it lowers again.
Example: Power Capping

- Specify desired behavior via accepted and forbidden states to restrict the behavior of the system

If power exceeds threshold, reduce power...until it lowers again
SPECTR Demonstration

**QoS Task:** x264

---

**Threshold (critical)**
- safePower
- decreaseCriticalPower

**AboveCapping**
- aboveTarget
- bellowTarget

**UnderCapping**
- maintain

**SwitchPower**
- switchQoS

---

Measured FPS
Reference FPS

---

Measured Power
Reference Power

---

[C] Rahmani18] ASPLOS '18
SPECTR Demonstration

QoS Task: x264

Safe Phase: QoS app only
SPECTR satisfies FPS with minimum power

Copyright © 2018 Dutt Research Group
https://duttgroup.ics.uci.edu
[Rahmani18] ASPLOS ‘18
SPECTR Demonstration

QoS Task: x264

Emergency Phase: TDP reduced in response to thermal event
SPECTR satisfies FPS and power

[Diagram showing FPS and power graphs]

[Rahmani18] ASPLOS ‘18
Disturbance Phase: TDP returned to normal, background tasks introduced SPECTR prioritizes power capping

QoS Task: x264
SPECTR Demonstration

QoS Task: x264

SPECTR meets FPS target when possible, while honoring power cap

[Rahmani18] ASPLOS ‘18
Outline

• Computational Self-Awareness

• Why Self-Aware Chips?

• Cross-Layer Sensing & Actuation

• Towards Self-Aware Chips

• Supervisory Control & Coordination

• Wrap-up
Key Take-Away 1:
Cross-Layer Physical/Virtual Sensing & Actuation

Applications
Operating System
Network/Bus Communication Architecture
Hardware Architecture
Device/Circuit Architecture

Adaptive Control (Decide)

Actuation (Act)

Sensors (Observer)
From today’s chips

Reflexive, Reactive

Simple Controller

QoS/Goals

Simple Adaptation

Self-monitoring and simple adaptation

Self-monitoring chip

output
Key Take-Away 2: Towards on-chip self-awareness

Reflection, Introspection

Simple Adaptation

Self-monitoring and Self-modeling

Adaptive Policies / Controller / Governor

System Behavior (Model Building)

Self-Aware Adaptation

QoS/Goals

measurement

input

output

Self-monitoring chip

[Sarma14, CODES+ISSS14]
Key Take-Away 3: Supervisory Control & Coordination

Supervisory Control

Supervisory Control & Coordination [Rahmani18, ASPLOS 2018]

Adaptive Polices / Controller / Governor

QoS/Goals

Simple Adaptation

Self-Aware Adaptation

Supervision System Behavior (Model Building)

measurement

input

output

Self-monitoring chip

Supervisory Control & Coordination

Copyright © 2018 Dutt Research Group
https://duttgroup.ics.uci.edu
#64
Special Issue on Self-Awareness in Systems on Chip 2017

- Self-Awareness in Systems on Chip—A Survey
- Health Management for Self-Aware SoCs Based on IEEE 1687 Infrastructure
- KOCL: Power Self-Awareness for Arbitrary FPGA-SoC-Accelerated OpenCL Applications
- A Self-Aware Architecture for PVT Compensation and Power Nap in Near-Threshold Processors
- Self-Adaptive Timing Repair

Survey Paper

6 Trusted Analog/Mixed-Signal/RF ICs: A Survey and a Perspective
Angelo Antimisiaris, Christiana Kapatos, and Yiorgos Makris

8 Self-Awareness in Systems on Chip—A Survey
Axel Jantsch, Nikil Dutt, and Amir M. Rahmani

27 Health Management for Self-Aware SoCs Based on IEEE 1687 Infrastructure
Konstantin Shibin, Sergey Devadze, Artur Jutman, Martin Grabmann, and Robin Pricken

36 KOCL: Power Self-Awareness for Arbitrary FPGA-SoC-Accelerated OpenCL Applications
James J. Davis, Joshua M. Levine, Edward A. Stott, Eddie Hung, Peter T. K. Cheung, and George A. Constantinides

46 A Self-Aware Architecture for PVT Compensation and Power Nap in Near-Threshold Processors
Davide Rossi, Igor Los, Antonio Pullini, Christoph Müller, Andrea Burg, Francesco Comi, Luca Renozzi, and Philippe Flateau

54 Self-Adaptive Timing Repair
Hau Goswen, Raphael Robin, Benjamin Gojman, and André DeHon

Layout-Aware Optimized Prebond Silicon Interposer Test Synthesis
Katherine Shu-Mei Li, Syung-Iue Wung, Ruo-Ting Gu, and Bo-Chuan Cheng

Adaptive ECC for Tailored Protection of Nanoscale Memory
Dongyeob Shin, Jangwon Park, Jongsun Park, Somnath Paul, and Swarup Bhunia

Low-Power Sparse Hyperdimensional Encoder for Language Recognition
Mohsen Imani, John Hwang, Tajana Rosing, Abbas Rahimi, and Jan M. Rabaey

General Interest

77 KOCL: Power Self-Awareness for Arbitrary FPGA-SoC-Accelerated OpenCL Applications
James J. Davis, Joshua M. Levine, Edward A. Stott, Eddie Hung, Peter T. K. Cheung, and George A. Constantinides

84 Adaptive ECC for Tailored Protection of Nanoscale Memory
Dongyeob Shin, Jangwon Park, Jongsun Park, Somnath Paul, and Swarup Bhunia

94 Low-Power Sparse Hyperdimensional Encoder for Language Recognition
Mohsen Imani, John Hwang, Tajana Rosing, Abbas Rahimi, and Jan M. Rabaey

ISSN: 2168-2356
Call for Papers: Special Issue on Self-Awareness in Resource Constrained Cyber-Physical Systems

Inspired by biological examples, self-awareness has become a hot research topic in a variety of disciplines and its applicability has been explored in various application domains. The topic owes its attractiveness to its promise to facilitate highly resilient, adaptive and outstandingly efficient behaviors. Thus, self-awareness holds the promise to promote dependability in all types of smart gadgets and artificial agents in the interconnected world of future.

However, the challenges raised by these new promising features are also significant, not least because they have a profound impact on the way we design, validate and test incorporating awareness: if a system smartly adapts to changing needs and environment, how do we validate functionality at design time? How do we specify the correct functionality in the first place? How are the relevant trade-offs? How can we quantify uncertainties and variabilities in a meaningful way? These are only some of the pressing questions that addressed before these new features can be exploited.

The ACM Transactions on Cyber-Physical Systems seeks original manuscripts for a special "Self-Awareness in Resource Constrained Cyber-Physical Systems" which will cover recent developments on methods, architecture, design, validation and application of resource-constrained cyber-physical systems that exhibit a degree of self-awareness.

Submission Guidelines:
Authors should submit their journal version at Manuscript Central adhering to the formatting instructions on the TCPs Web page, and indicate that you are submitting to the Special Issue on Self-Awareness in Resource Constrained Cyber-Physical Systems on the first page and in the field "Author's Cover Letter" in Manuscript Central. For additional questions, please send an email to any of the guest editors: p.lewis@aston.ac.uk, axel.jantsch@tuwien.ac.at, dutt@uci.edu.

Submission Guidelines:
Submission deadline: 7 September, 2018
Notification of First Round: 7 December, 2018
Submission of Revision: 8 February, 2019
Final Notification: 12 April 2019
Final Paper Due: 23 May 2019

Guest Editors Contacts:
Peter Lewis, p.lewis@aston.ac.uk
Axel Jantsch, axel.jantsch@tuwien.ac.at
Nikil Dutt, dutt@uci.edu
Ongoing Efforts

- More heterogeneity (CPU+GPU+DSP+NPU+FPGA+…..)
  - Reconfigure workloads at runtime to freely migrate between resources
  - Complex predictive models

- Distributed management
  - Propagating sensing info across non-coherent processing units

- Non-compute resources
  - Memory and I/O
Ongoing Challenges

- **Self-trained models**
  - Add feedback for error correction
  - Challenging for models that are non-linear and/or based on heuristics

- **Machine learning**
  - Replacement for analytical/heuristic-based models?
  - Unsupervised machine learning to mine sensing data and find patterns for optimizing policies or creating new ones

- **Policy supervisors**
  - Provide formal or stronger guarantees
Acknowledgements

• Dutt Research Group

• Collaborating Faculty:
  – UCI: Fadi Kurdahi
  – Austria: TU Wien: Prof. Axel Jantsch
  – Germany: TUM: Prof. Andreas Herkersdorf, TUB: Prof. Rolf Ernst

• NSF Information Processing Factory (IPF) project
Questions?

Dutt Research Group:  http://duttgroup.ics.uci.edu/