# Energy consumption in embedded systems; abstractions for software models, programming languages and verification methods

Florence Maraninchi — orcid.org/0000-0003-0783-9178

*thanks to M. Moy, L. Mounier, L. Maillet-Contoz, D. Barthel, J. Cornet, C. Helmstetter, T. Bouhadiba, N. Berthier, L. Samper, ...*

`www-verimag.imag.fr/~maraninx`

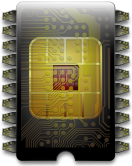
Communauté UNIVERSITÉ Grenoble Alpes

Verimag

EMSOFT'16, Pittsburgh

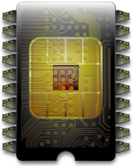
# Lessons Learned in Various Contexts...

Modeling energy consumption and temperature at the transactional level for systems-on-a-chip (with STMicroelectronics) - Validation of low-level software that implements power-domain control

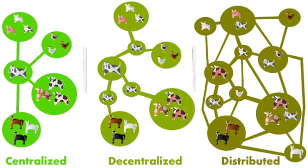


# Lessons Learned in Various Contexts...

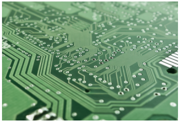
Modeling energy consumption and temperature at the transactional level for systems-on-a-chip (with STMicroelectronics) - Validation of low-level software that implements power-domain control



Modeling energy consumption in sensor networks (with Orange Labs) - trade-offs between energy consumption and security at the routing level, precise modeling of idle-listening in MAC protocols, ...



# From Physics to (Application) Software

| | |
|---|---|
| Application SW | Decide what to switch on/off |
| OS | control sleep modes<br>real–time scheduling and adjusting V, F |
|  | Components' operational modes<br>Power Domains and DVFS<br>Temperature Sensors<br>Static+Dynamic Energy Consumption |
|  | Battery behaviour and Discharge time |

# Discharge time is not a Simple Function of Power Consumption

Estimating energy consumption does not give easily an estimate of the battery discharge time.

More details available if needed.

*see "rate-dependency effect" in David Linden et Thomas B. Reddy — Handbook of batteries. McGraw-Hill 2002*

*Ravishankar Rao, Sarma Vrudhula et Naehyuck Chang — Battery optimization vs energy optimization: which to choose and when?. ICCAD'05*

# Sources of Power Consumption

$$P = \begin{array}{l} P_{\text{static}} \ \textit{due to leakage currents} \ + \\ P_{\text{dynamic}} \ \textit{due to the switching of transistors} \end{array}$$

$$P_{\text{static}} = V \times K_1 \times g(T) \ \nearrow \ \text{when transistor size} \ \searrow$$
$$P_{\text{dynamic}} = F \times V^2 \times \alpha \times K_2$$

$V$: Voltage, $F$: Frequency, $T$: Temperature

$g$: increasing function

$\alpha$: activity ratio, or amount of computation performed

$K_i$s: various "constants" depending on the module area and on the synthesis technology

# Power Control in Modern Circuits

- Clock Gating (turn off the clock):
  $P_{\text{dynamic}} = 0$, but $P_{\text{static}}$ unchanged
- Dynamic Voltage and Frequency Scaling (DVFS) reduces $V$, hence $F$ has to be reduced too. A circuit can have a (small) number of operating points $(V, F)$. Switching between them has a cost.
- Power Gating (switch a component on/off); Switching is very costly (save/restore state); application-level information is needed (e.g., GPS is not longer used, switch the sub-circuit off).

# Complex Feedback Interactions

Consumption $\uparrow$

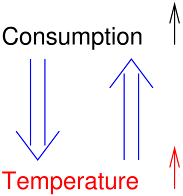
# Complex Feedback Interactions

Consumption $\uparrow$

$\Downarrow$

Temperature $\uparrow$

# Complex Feedback Interactions



# Complex Feedback Interactions

Consumption ↑

Temperature ↑

# Complex Feedback Interactions

Consumption $\uparrow$

Temperature $\uparrow$

Consumption $\uparrow$

Temperature $\uparrow$

# Complex Feedback Interactions

Consumption $\uparrow$

$\Downarrow \quad \Uparrow$

Temperature $\uparrow$

Consumption $\uparrow$

$\Downarrow$

SW switches component off $\Leftarrow$ Temperature $\uparrow$

# Complex Feedback Interactions

Consumption ↑

Temperature ↑

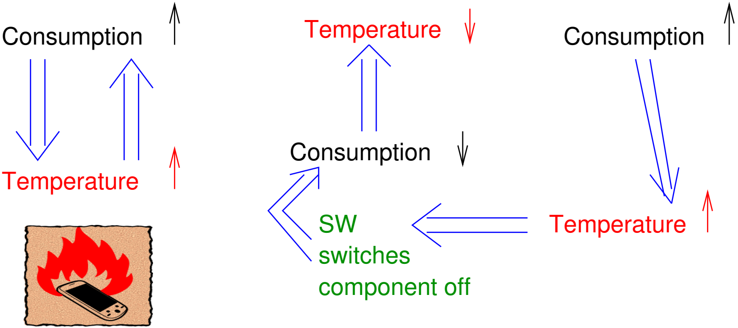
Consumption ↓

SW switches component off

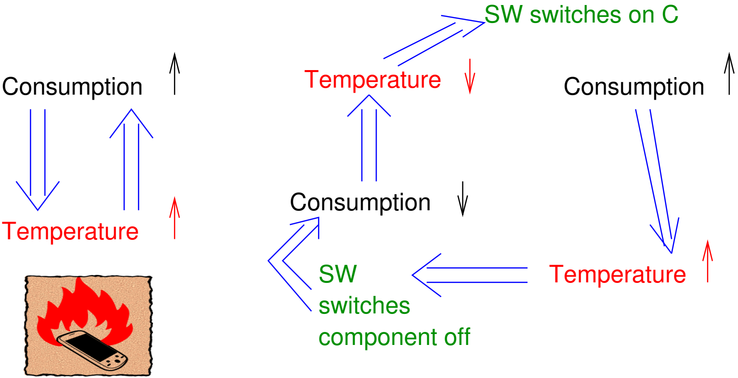
Consumption ↑

Temperature ↑

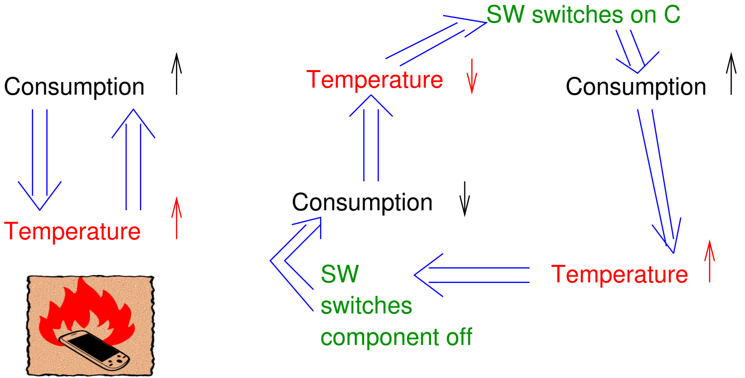
# Complex Feedback Interactions



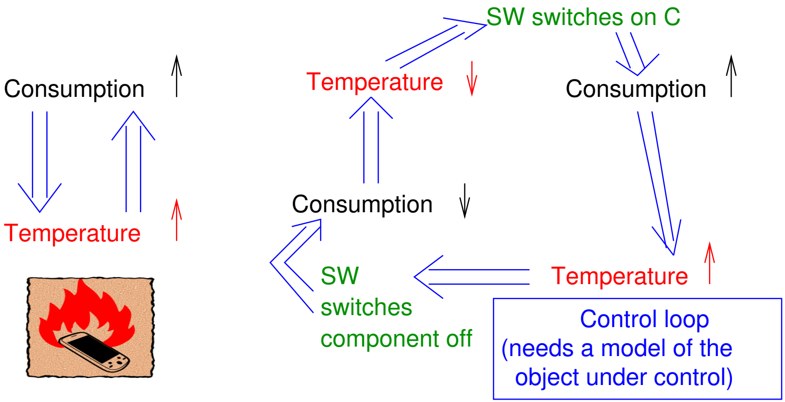
# Complex Feedback Interactions



# Complex Feedback Interactions



# Complex Feedback Interactions



# Models for What?

- Estimate/improve the discharge time of the battery
- Reduce temperature peaks and temperature gradient to improve the lifetime of the circuit
- Estimate the loss of QoS due to low-consumption operating modes
- Write control code that plays with the operating modes of the components, validate the power-management policies
- Detect "energy bugs" in applications (try to use a component that has been switched off; fail to switch off a component that is no longer needed...)
- Overall design space exploration

2 Models
- (Formal) State-Based Models
- Simulation Models

# Power-State Machines

**System-level Power Estimation And Optimization**

Luca Benini Robin Hodgson Polly Siegel
Hewlett-Packard Laboratories
benini@hpl.hp.com hodgson@hpl.hp.com polly_siegel@hp.com

ISLPED'98
International symposium on Low Power Electronics and Design

# Power-State Machines with Transition Penalties

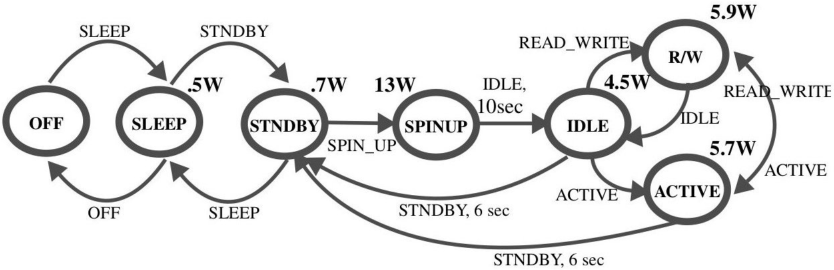



**Figure 3: Disk power state machine w/transition penalties**

States have an associated power consumption (per time unit)
Transitions have an associated penalty: transition time, power

# Linearly-Priced Timed-Automata

"LPTA are an extension of timed automata with prices on both transitions and locations: the price of a transition gives the cost for taking it and the price on a location specifies the cost per time-unit for staying in that location"

*Minimum-Cost Reachability for Priced Timed Automata; Gerd Behrmann Ansgar Fehnker, Thomas S. Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, Frits W. Vaandrager, 2001*

# Questions

- Where to use such power-state machines (or their LPTA counterpart)?
  Easy: for the DVFS operating points of the CPU, the operational modes of a sensor node radio (TX, RX, Idle...), ...
  Not so easy: the bus or NoC, the memories?
- What does it hide?
  What phenomena cannot be captured like that?

Let's look at "precise" and "complete" simulation models to try and understand what's not captured by those formal models.

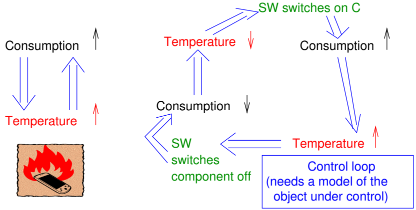
## 2 Models

- (Formal) State-Based Models
- Simulation Models

# Main Ideas

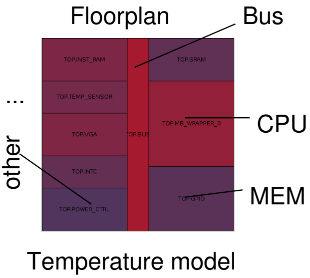
Capture all potential interactions between: voltage, frequency, consumption, temperature, software decisions, state of the battery, ...

Recall:



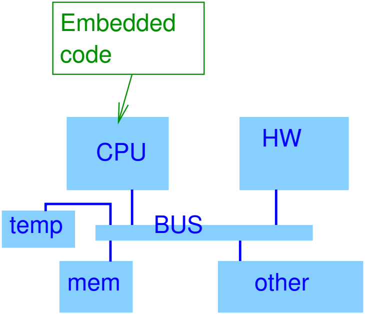

In the most detailed models one can play the actual software on top of a functional+extra-functional model of the hardware.

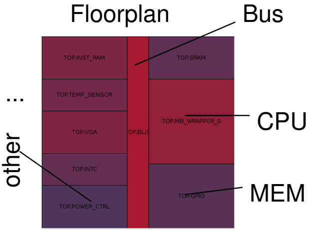
# Precise Simulation Models with Temperature Models and Actual Embedded Code



# Precise Simulation Models with Temperature Models and Actual Embedded Code

```
if (T > thr) { switch "other" off; turn CPU to (V1,F1) }
```

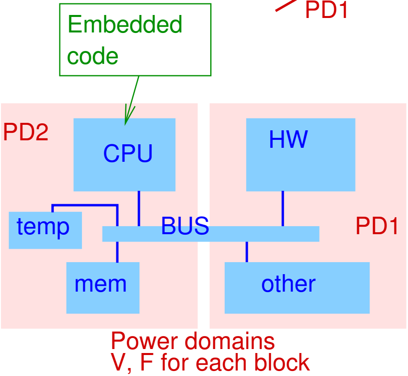


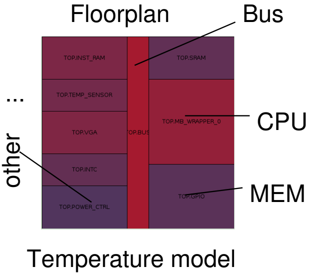




Temperature model

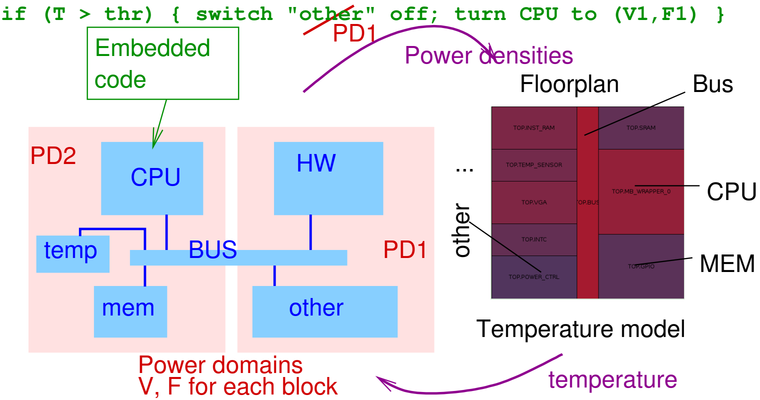
# Precise Simulation Models with Temperature Models and Actual Embedded Code

**if (T > thr) { switch "~~other~~" PD1 off; turn CPU to (V1,F1) }**



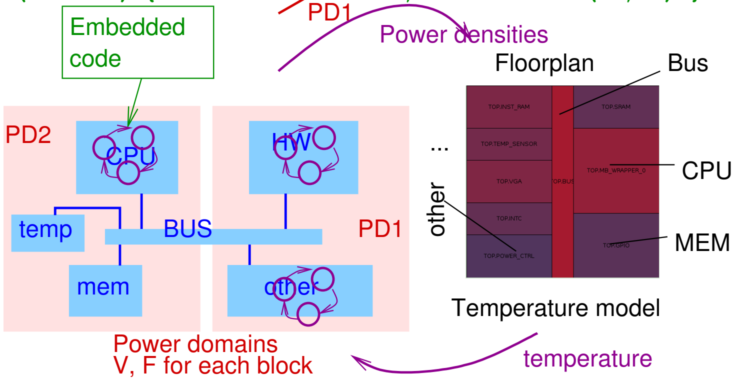



# Precise Simulation Models with Temperature Models and Actual Embedded Code



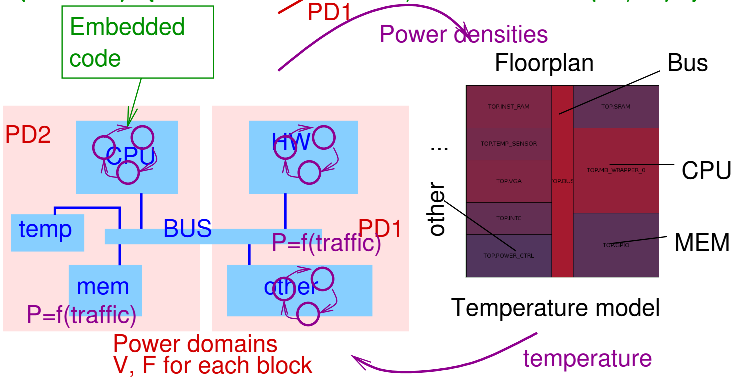
# Precise Simulation Models with Temperature Models and Actual Embedded Code

if (T > thr) { switch "~~other~~" off; turn CPU to (V1,F1) }



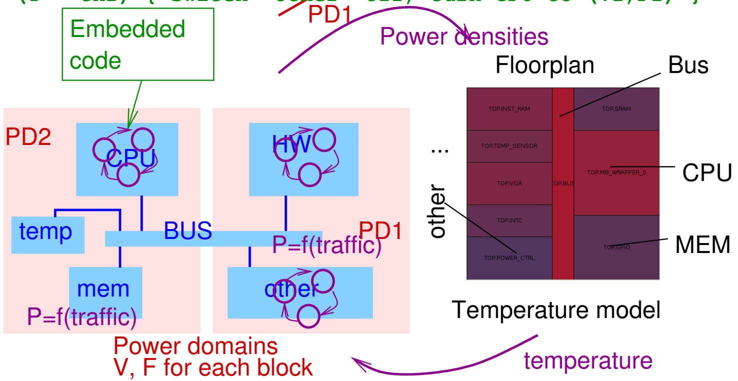
# Precise Simulation Models with Temperature Models and Actual Embedded Code

```
if (T > thr) { switch "other" off; turn CPU to (V1,F1) }
```



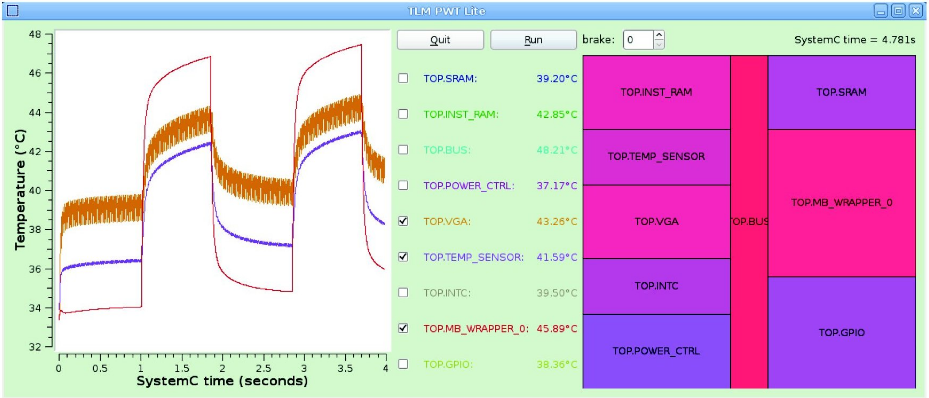
# Precise Simulation Models with Temperature Models and Actual Embedded Code

```
if (T > thr) { switch "other" off; turn CPU to (V1,F1) }
```




P=f(traffic) may take contention into account; the Joule–per–bit model cannot.

# Example Simulation Results




*Modeling Power Consumption and Temperature in TLM Models — Matthieu Moy, Claude Helmstetter, Tayeb Bouhadiba, Florence Maraninchi - Leibnitz Trans. on Emb. Syst. 2016*

# A Major Problem: Validation of the Models

A precise simulation is theoretically feasible (at gate level, or even below). But it's terribly slow.

Raising the level of abstraction to get reasonable-time SW-in-the-loop simulations implies accepting relative results only.

- Simulation, say 5%-precise w.r.t. real system: hopeless
- One objective can be to identify peaks, or the points that trigger the control policy in the SW.

+ what's the sensitivity of the overall model to small variations on the figures attached to states?

# A Hierarchy of Abstractions

- Forget about the battery model *(consider it as a bathtub)*
- Forget about temperature effects *(choose a fixed ambiant temperature in the equations)*
- Forget about static consumption *(or consider a fixed additional consumption)*
- Forget about anything else than computing elements *(or use the Joule-per-bit model for communication elements)*

# Examples (1)

Next talks at EMSOFT'16:

- Flexible Support for Time and Costs in Scenario-Aware Dataflow: LPTA-style costs in SADF + cost per token (i.e., Joule-per-bit model)
- Energy and Timing Aware Synchronous Programming: no battery model, no temperature effect, no static consumption, nothing else than computing elements, penalties for changing (V, F) in the CPU

# Examples (2)

- EMSOFT'10 - Energy-Aware Packet and Task Co-Scheduling for Embedded Systems: power-state machine for (CPU+radio transmitter)
- CODES+ISSS'11 - System-Level Power and Timing Variability Characterization to Compute Thermal Guarantees: based on real-time calculus, fixed or max ambiant temperature, influence of power on temperature as in abovementioned temperature simulators, power consumption (stat+dyn) as a function of the executing task, plus a system-level idle consumption.

# Recovering Some Interactions...

- Several power-state models (CPU, other HW components) and their implicit product, driven by the SW model
- Traffic models for communication elements, taking contention into account (needs to be precise on *which* components use the bus *and when*)
- Power-states machines for each component, modeling the electrical state (needs information on *power domains*)
- "states" in a battery model *See: Battery transition systems, Udi Boker, Thomas A. Henzinger, Arjun Radhakrishna, POPL'14*

The most difficult is to include the temperature effects, because you need the *floorplan* (not really usual software-level information)

# Not Really a Conclusion

Facts:

- Energy consumption (+ temperature) in embedded systems is a complex phenomenon, even more with the software taking control decisions
- Models that are usable "formally" are necessarily very abstract
- The distance between real life and models is quite big

So what? IMHO, there's no silver bullet, each situation requires a careful analysis of the aspects that can be safely ignored; understanding the nature of the interactions may help decide what to ignore, on purpose.

# Thank you.
# Questions?