Dynamic Variability Management in Mobile Multicore Processors under Lifetime Constraints

Pietro Mercati[1], Francesco Paterna[1], Andrea Bartolini[2], Luca Benini[2] and Tajana Rosing[1].

[1]UC San Diego, [2]ETH Zurich & University of Bologna

Variability

Variability is the key issue in modern multiprocessors, resulting in performance and lifetime uncertainty, and high design margins.

- This has an impact on lifetime and on performance
- Due to scaling, variability increases and multiprocessors are more lifetime constrained.
- Mobile multicore processors run a varied workload in parallel with apps that may have critical requirements in terms of quality and user experience.
- Devices have been proposed for frequency degradation monitoring and degradation status estimation.

Contributions

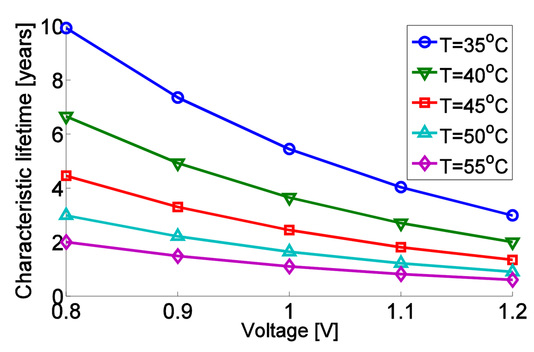
A novel Dynamic Variability Management policy for performance improvement of mobile multiprocessors under lifetime constraints that comprehensively accounts for:

1. Degradation rate variability
2. Frequency variability
3. Application-specific quality requirements for user experience
4. Lifetime constraint
5. Ambient temperature variation

Background

Environmental variation: Temperature Boost (*T-boost*)

- At high temperatures the dependence on voltage can be neglected



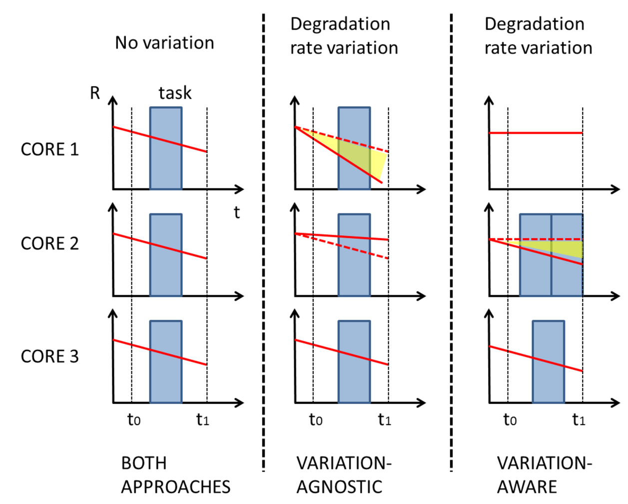
- Running at low temperature allows to reserve a time budget for running at maximum voltage/frequency at high temperature
- Such an approach allows for long term reliability banking when the device experiences temperature variations.

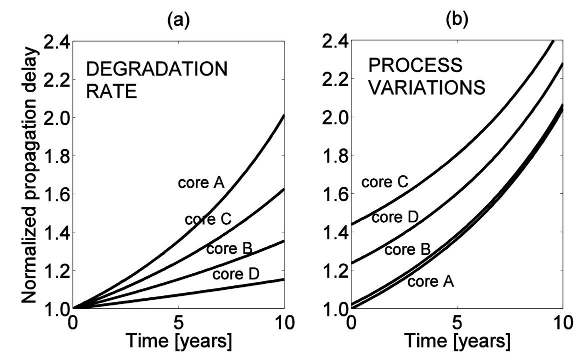
Frequency variation:

- Cores designed to be equal actually have different frequency
- Frequency degrades with time
- Time Dependent Dielectric Breakdown affects propagation delay
- Sensors have been proposed to monitor the execution frequency

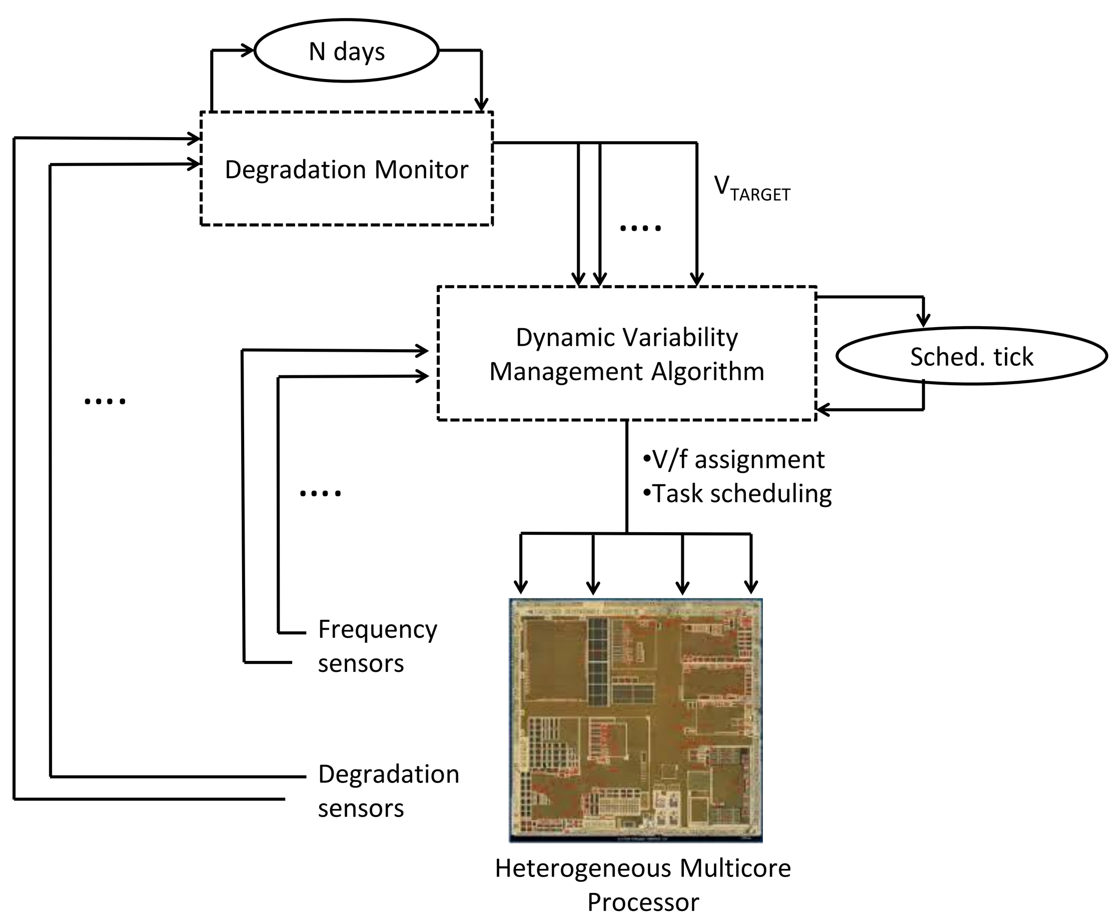
Degradation rate variation:

- Cores designed to be equal have different lifetime and reliability
- This is caused by process variations, unbalanced work allocation, environmental conditions.
- Cores are subject to TDDB degradation
- Sensors have been proposed to monitor degradation





Dynamic Variability Management



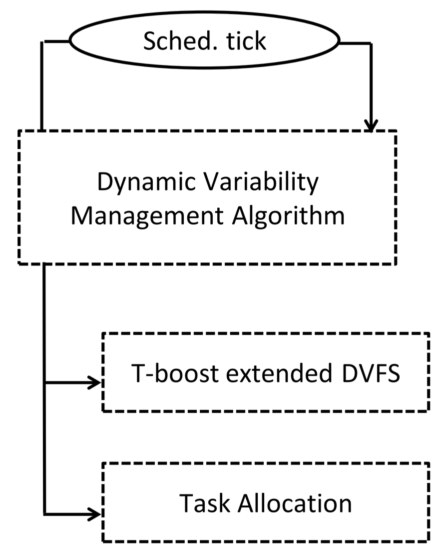
- Target platform: heterogeneous multiprocessor modeled as an array of cores $\Omega = [\omega_1, \omega_2, \dots \omega_N]$
- Each core is characterized by a type σ_i
- Each task can be assigned only to specific type of core.
- Cores have independent voltage and frequency settings, with fixed operating points
- Tasks can be Highly Critical (H) or Less Critical (L)

Degradation Monitor:

- Activates every reliability period (in the order of days)
- Reads degradation sensors and frequency sensors
- It monitors the degradation status of each core
- Outputs a reference voltage V_{TARGET}

Dynamic Variability Management Algorithm:

- It adjusts voltage and frequency of each core and it assigns tasks
- Voltage and frequency are assigned following the *Borrowing Strategy* and the *T-boost* mechanism.
- If the temperature is lower than T_{LOW}, the time budget t_{BOOST} is increased by one.
- If the temperature is over T_{HIGH} and t_{BOOST} is greater than zero, then *T-boost* activates and lowers t_{BOOST} by one unit.
- If the *T-boost* is off, the algorithm selects minimum V/f if the core is idle, V/f_REF if the core is running a L task and v/f_MAX if the core is executing a H task



- The algorithm takes the first task to be scheduled and considers whether it is H or L.
- In the first case, cores are ranked with respect to frequency variation
- In the latter case, cores are ranked with respect to reliability

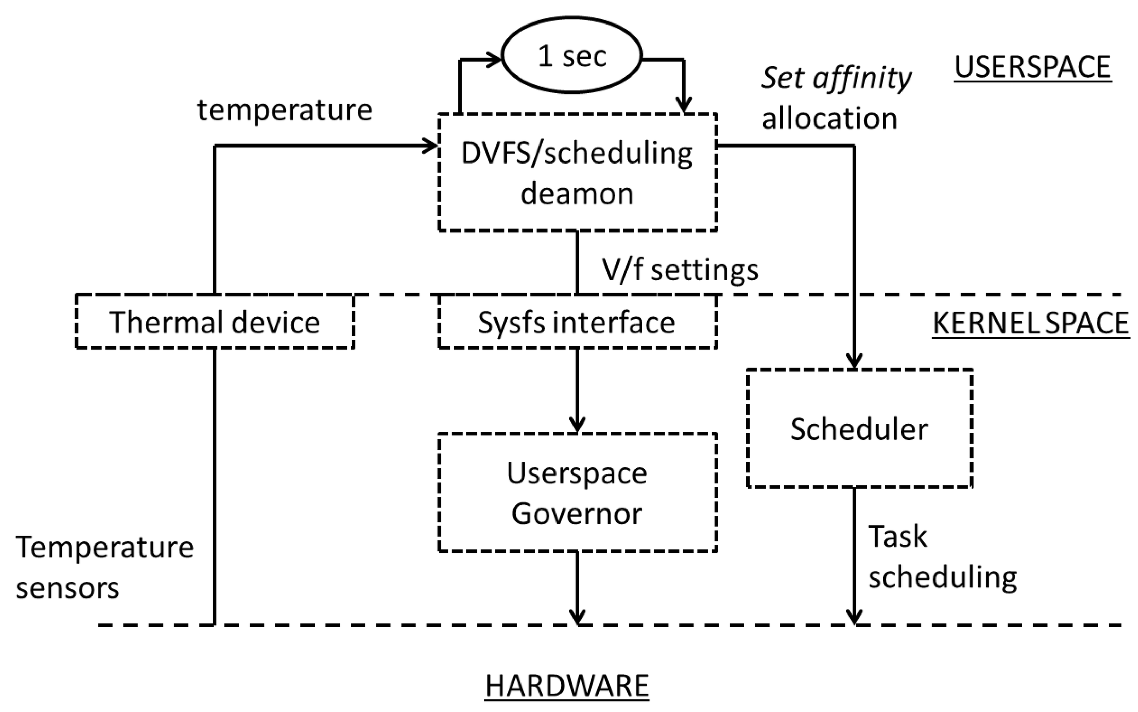
Results

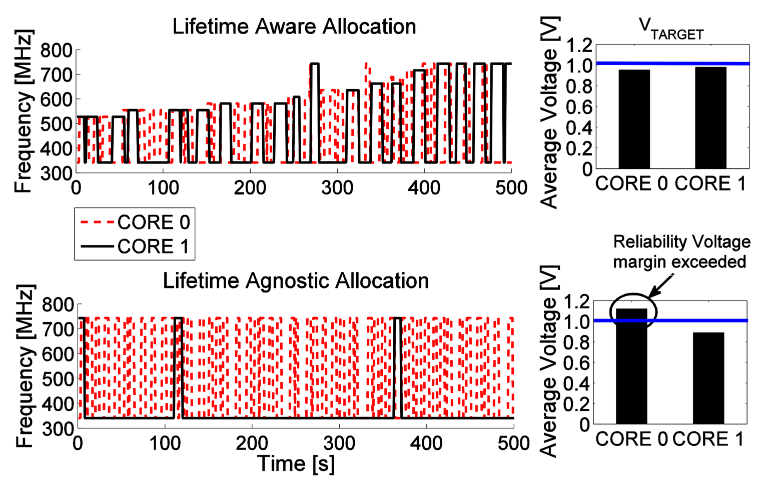
Experimental setup:

- Qualcomm MSM8660 mobile development smartphone
- Dual Scorpion core 45nm
- Adreno 220 GPU
- Voltage range: 0.8-1.2V , Frequency range: 810-1180MHz
- Android OS 4.0.3 with Linux Kernel 3.0.8
- Testbench computational kernels: DCT, IDCT, ROT

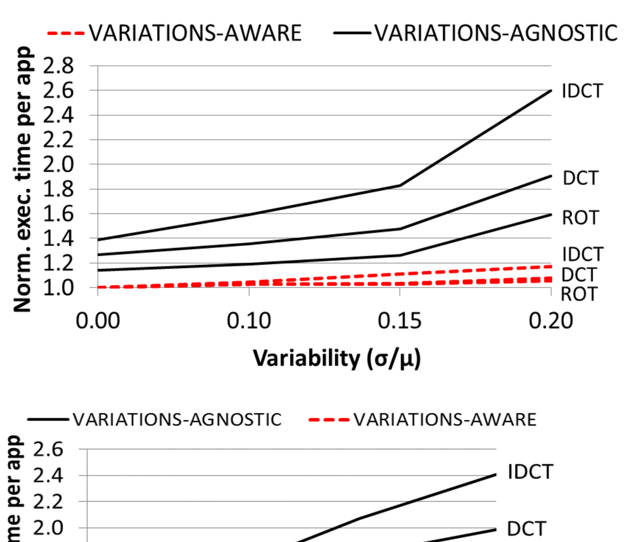
Implementation:

- The high level daemon exploits the Linux *set-affinity* mechanism to allocate tasks to cores
- It is also interfaced with the userspace governor
- The scheduling period is set at 1 second to approzimate the Linux reassignment granularity



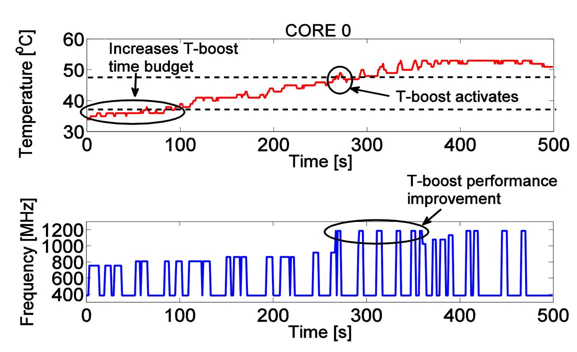


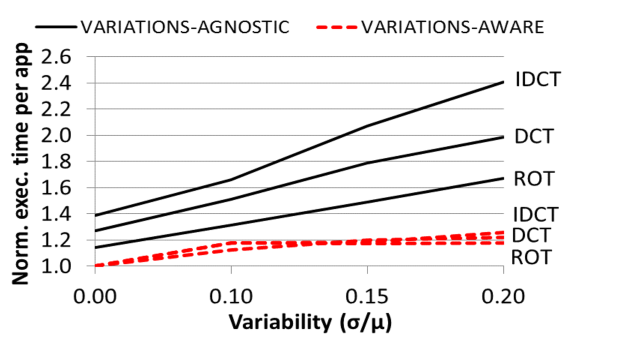
Lifetime-agnostic scheduling violates the constraint imposed by V_{TARGET}

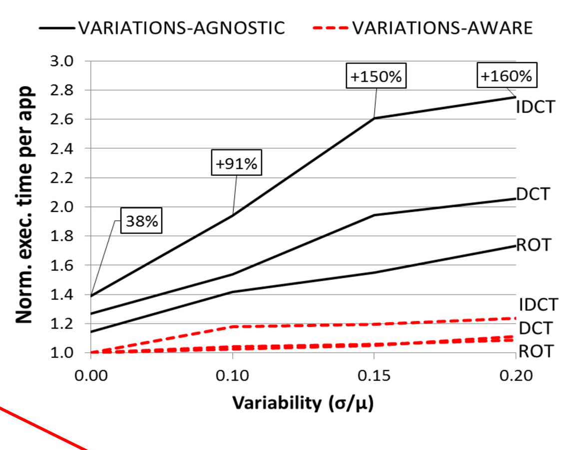


In case of variability, our technique considerably outperforms a strategy which does not exploit task scheduling

If *T-boost* is not present, the runtime management reacts to temperature increase by limiting the voltage, resulting in performance loss.







Our solution achieves up to 160% performance improvement over the state-of-the-arts

$\sigma/\mu = 0.2$	case 1	case 2	case 3	case 4
VARIATION-AWARE	1.75	1.40	0.81	0.98
VARIATION-AGNOSTIC	0.11	0.02	0.13	0.14
PI	164%	138%	66%	84%

Conclusion

In this work we propose a novel Dynamic Variability Management technique which uses task scheduling and per core performance control for achieving high performance in multiprocessors affected by variations, while meeting lifetime constraints. The technique has been implemented on a real Android smartphone. The presented experiments show that it achieves up to 160% performance improvement over state-of-the-art techniques.