Nonvolatile Computing: Towards Ubiquitous Low-Power Embedded Computing

Jingtong Hu (jthu@okstate.edu)

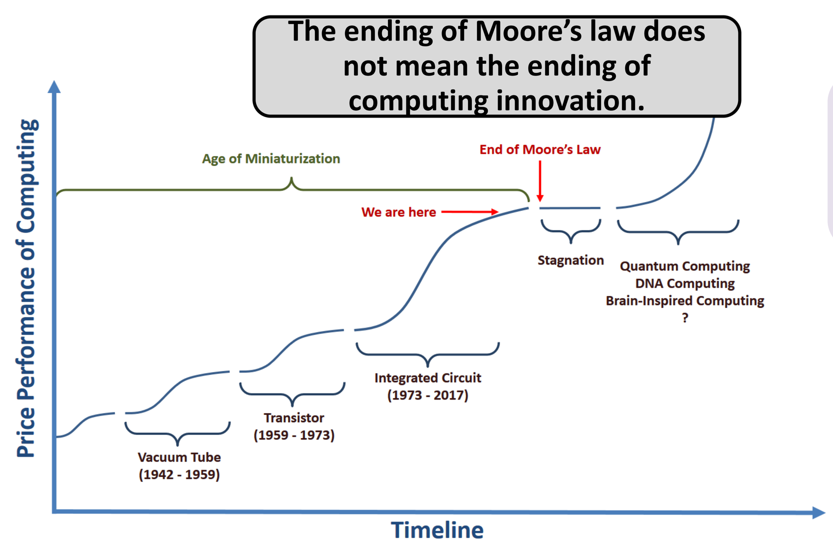
School of Electrical and Computer Engineering, Oklahoma State University

NSF CSR 1464429

Abstract

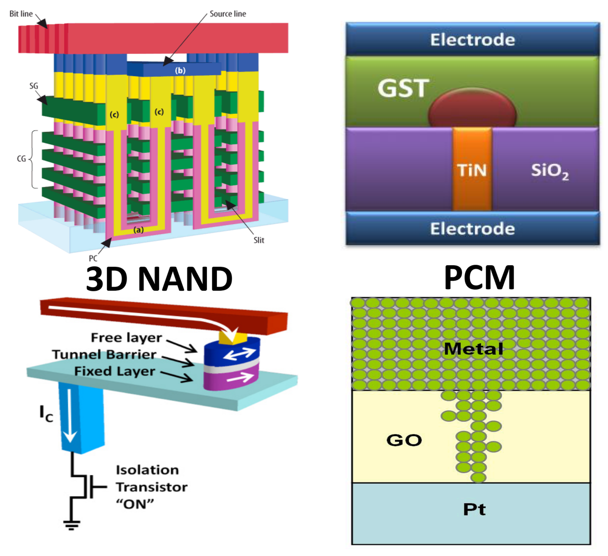
In the era of Internet of things, billions of embedded devices need to be powered with battery and tolerate the inconvenience come with recharging and replacing the battery. Energy harvesting is appealing to embedded systems especially in sensing since while the performance of embedded systems is improving every year, the battery development is lagging. However, the energy harvested from environment is usually weak and intermittent. With traditional CMOS based technology, whenever the power is off, the computation has to start from the very beginning. If we can save the intermediate computation and resume it when the power comes back, we can compute larger tasks with intermittent power. Compared with existing CMOS based memory devices such as SRAM which stores the data with charges, emerging Non-volatile memory devices such as PCM and STT-RAM, have the benefits of sustaining the data even when there is no power. These new devices bring promising opportunities to the computing paradigm since they have extremely low leakage power and better scaling than CMOS technology. With Non-volatile computing, we can turn off the processor and resume from where was left. In this way, we can either turn off processor on purpose to save energy or passively survive unstable power. This research focuses on achieving non-volatile computing for modern embedded systems.

Background

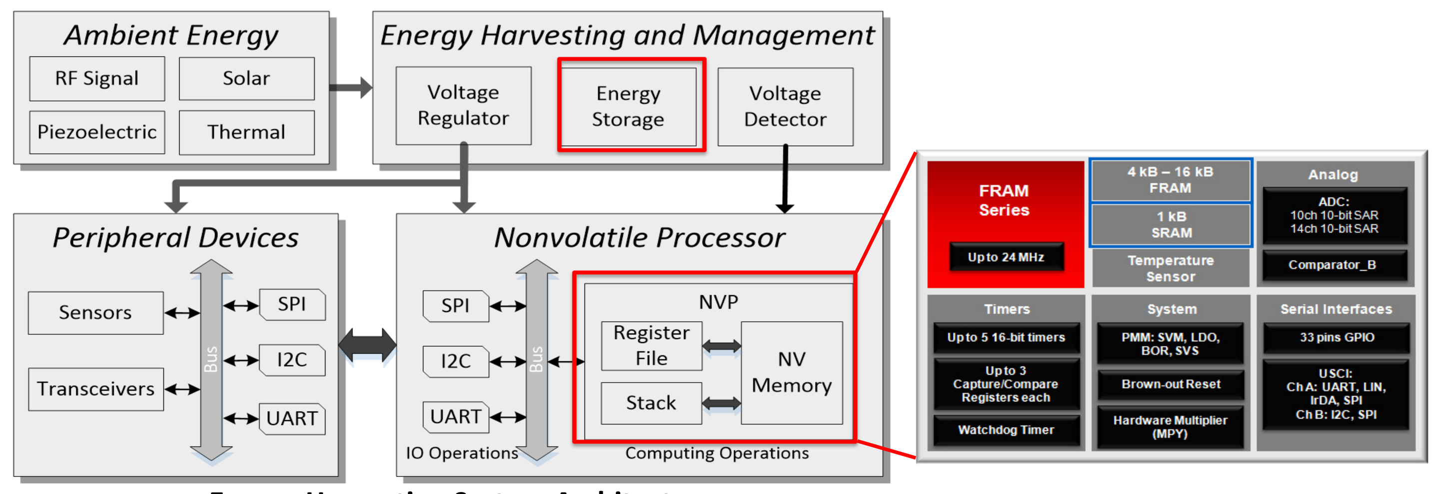


Moore's Law for CMOS Technology is Ending

Emerging Non-volatile memory devices bring promising opportunities to the computing paradigm.



Non-volatile memory technologies have extremely low leakage power and better scaling.

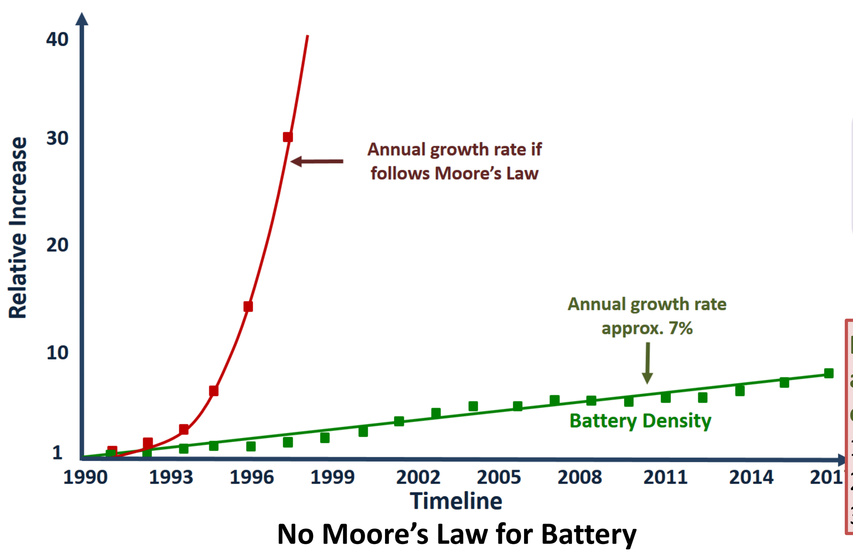


Energy Harvesting System Architecture

The energy harvested from environment is usually weak and intermittent. With traditional CMOS based technology, whenever the power is off, the computation has to start from the very beginning.

The length of computation is limited by the power you can obtain during this power cycle, which limits the application of energy harvesting powered computing.

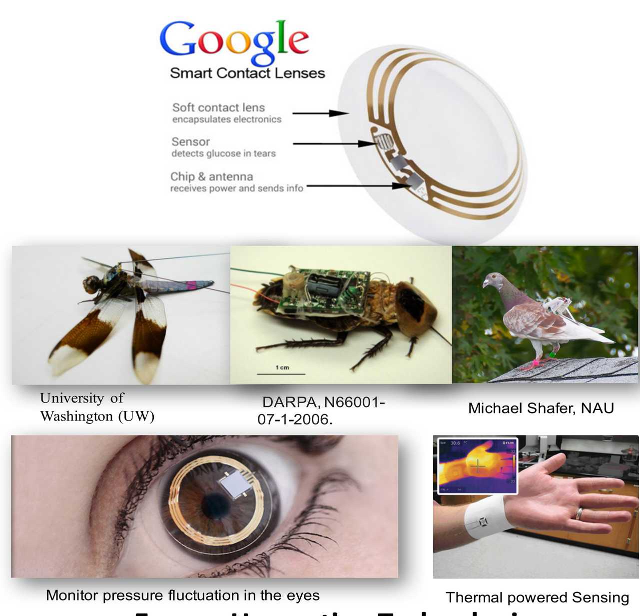
If we can save the intermediate computation and resume it when the power comes back, we can compute larger tasks with intermittent power.



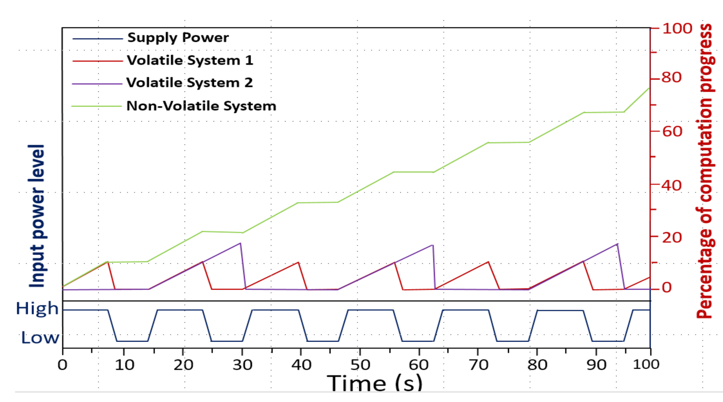
No Moore's Law for Battery

Energy Harvesting is a promising alternative to batteries.

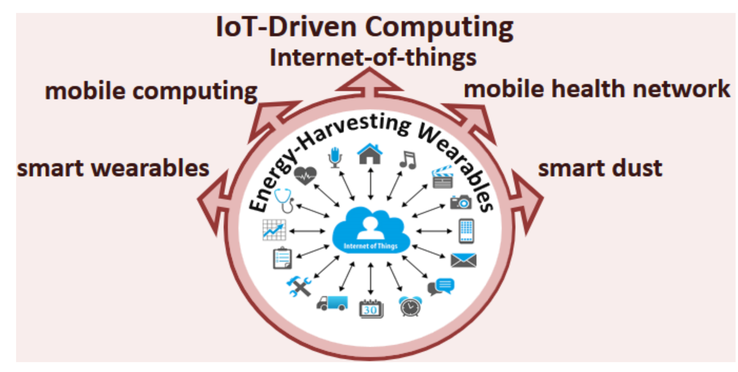
Energy harvesting has several advantages over batteries in the era of IOT:
1)Free of rechargeable concerns.
2)Environmentally friendly.
3)Pervasively available.



Energy Harvesting Technologies

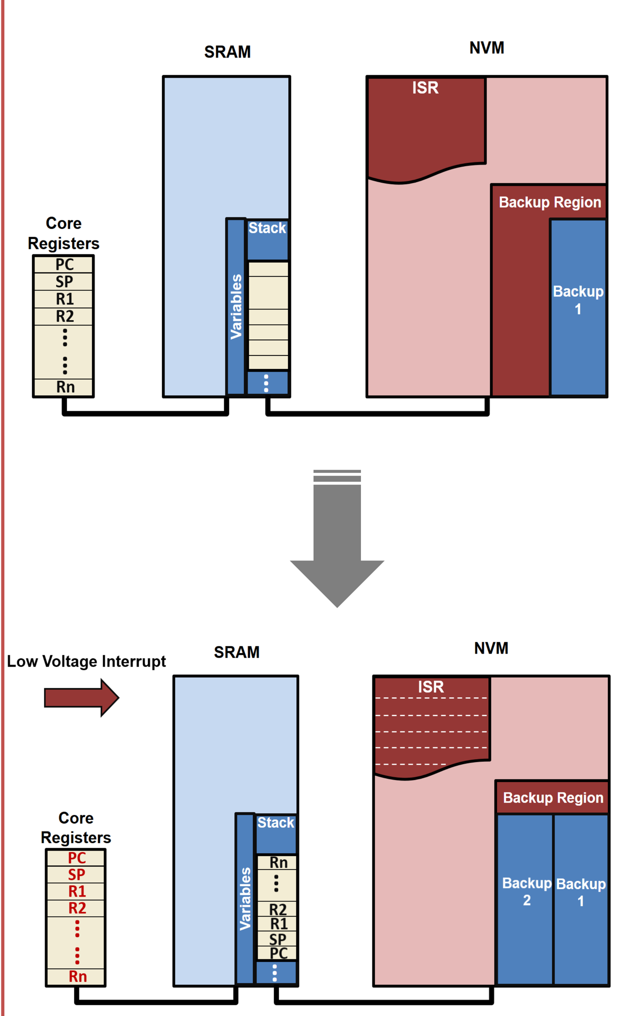


Volatile vs Non-Volatile in Energy Harvesting



Non-volatile Computing

Non-Volatile Processor



Checkpointing Processor States

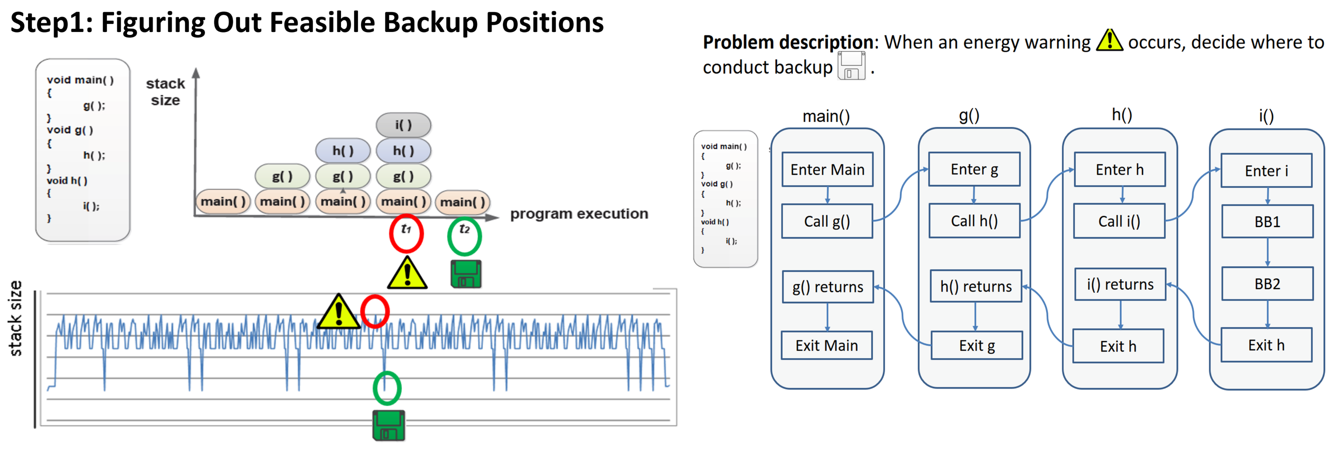
Stack Size Aware Checkpointing

Stack is a dynamic data structure, which increases and decreases along the program execution.

Main idea: instead of saving everything at the moment the interrupt signal is received, we can let the processor continue the execution to a point where the stack size is smaller. Then do the checkpointing.

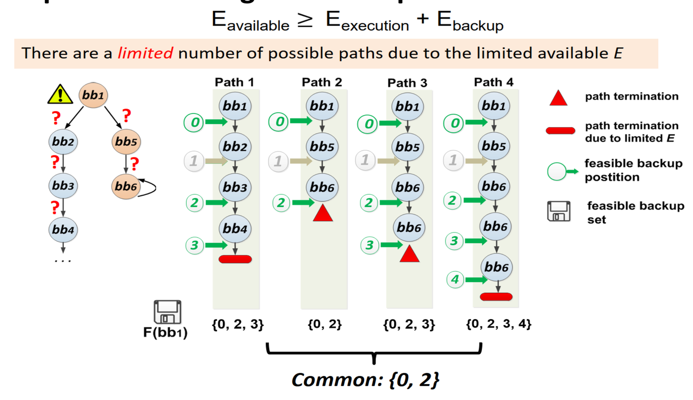
Step1: Figuring Out Feasible Backup Positions

Problem description: When an energy warning occurs, decide where to conduct backup.



Step2: Determining The Backup Position

$E_{available} \geq E_{execution} + E_{backup}$



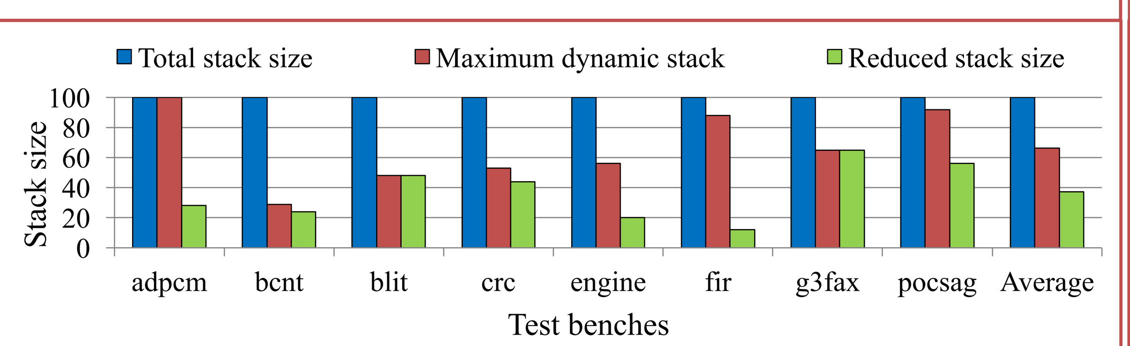
Common: {0, 2} , 0 or 2?

MinCom : choose the one with the *smallest* stack size

stackSize(0) *= max { stacksize(0 in path 1), stacksize(0 in path 2), stacksize(0 in path 3), stacksize(0 in path 4) };*

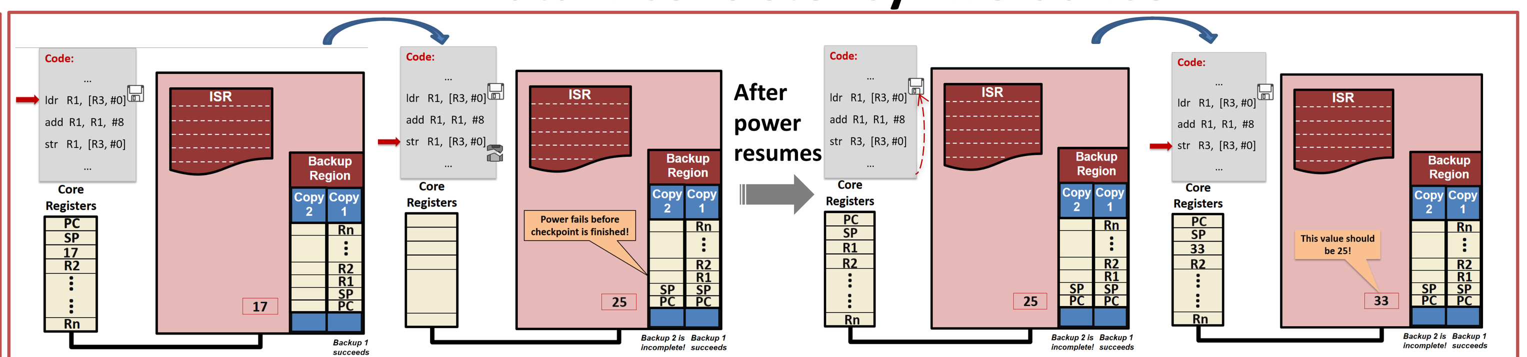
stackSize(2) *= max { stacksize(2 in path 1), stacksize(2 in path 2), stacksize(2 in path 4) };*

- Largest Stack Size: The maximum active data size in stack
- Immediate Checkpoint: The stack size if checkpoint immediately
- After Reduction: The required stack size after the proposed reduction

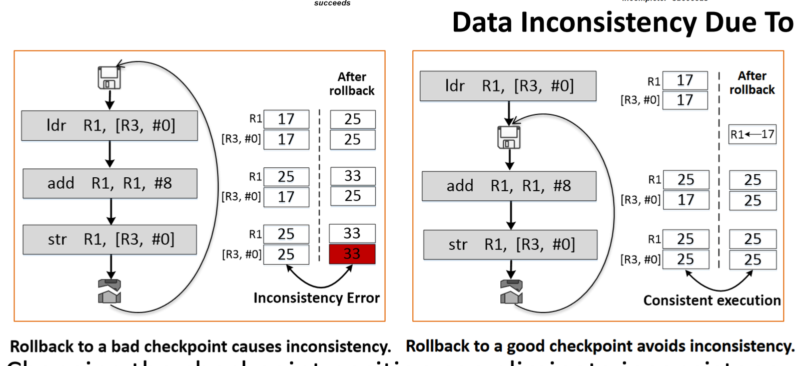


The benefits are twofold. First, we are making more progress. Second, we are saving less data and increase successful rate.

Data Inconsistency Avoidance



Data Inconsistency Due To Failed Checkpointing

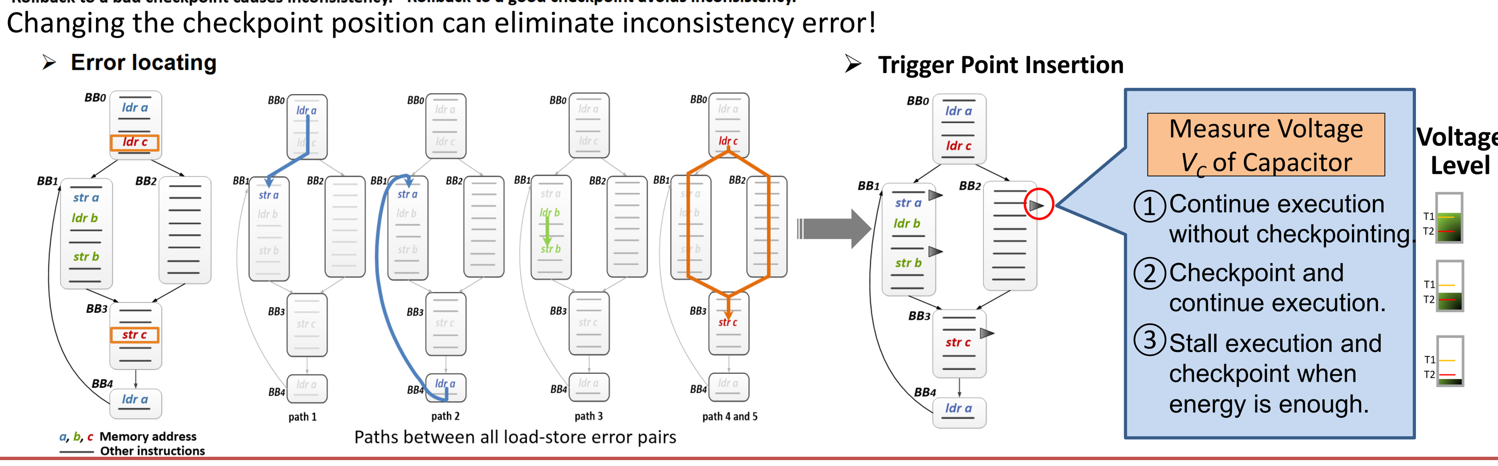


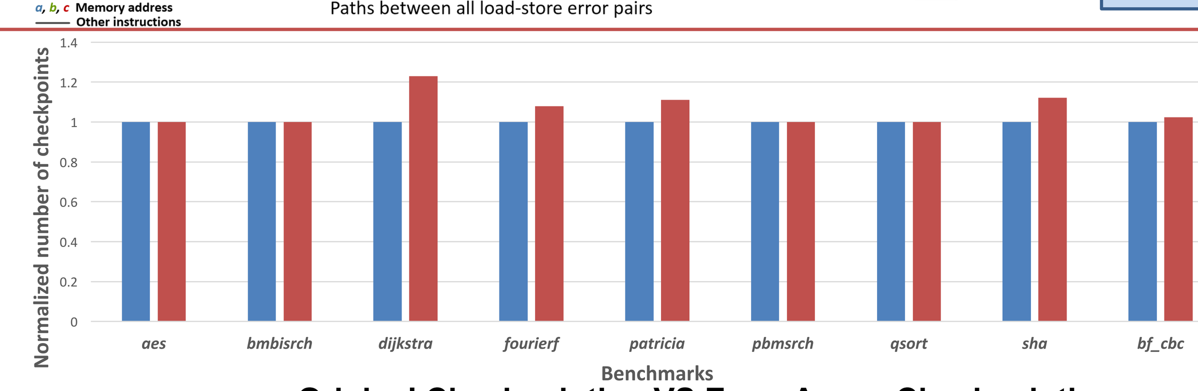
Rollback to a bad checkpoint causes inconsistency. Rollback to a good checkpoint avoids inconsistency.

Inconsistency may happen inside a basic block and across different blocks.

In order to reduce the overhead, energy-efficient checkpointing mechanism is proposed by setting two thresholds for voltage level: T1 and T2.

Changing the checkpoint position can eliminate inconsistency error!



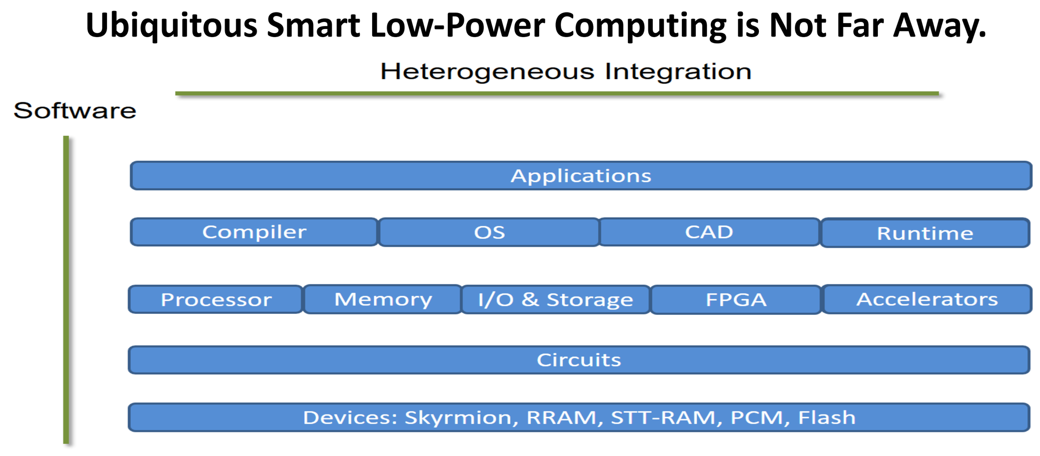


Eliminate all the errors with 6.3% more checkpoints

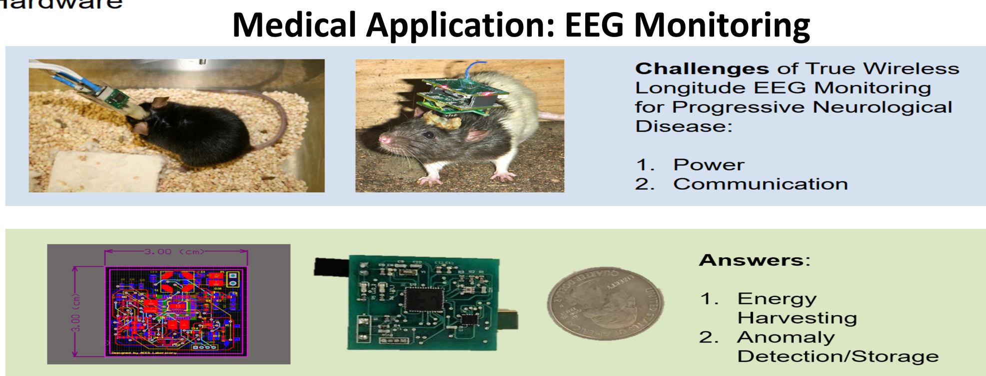
Original Checkpointing VS Error-Aware Checkpointing

Future Work

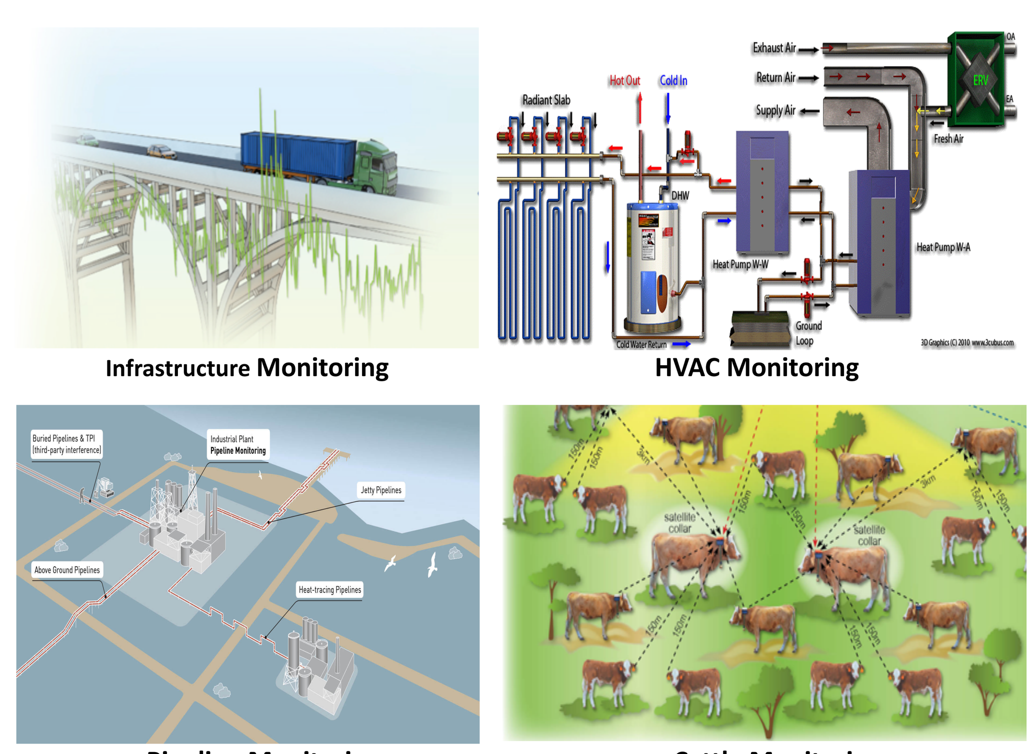
Ubiquitous Smart Low-Power Computing is Not Far Away.



Medical Application: EEG Monitoring



More applications…



NVM is Transforming Computing Paradigm