
# Power/Capacity Scaling: Energy Savings With Simple Fault-Tolerant Caches

**Mark Gottscho**[1]
Abbas BanaiyanMofrad[2]
Nikil Dutt[2]
Alex Nicolau[2]
Puneet Gupta[1]

[1] NanoCAD Lab
UCLA Electrical Engineering
*nanocad.ee.ucla.edu*

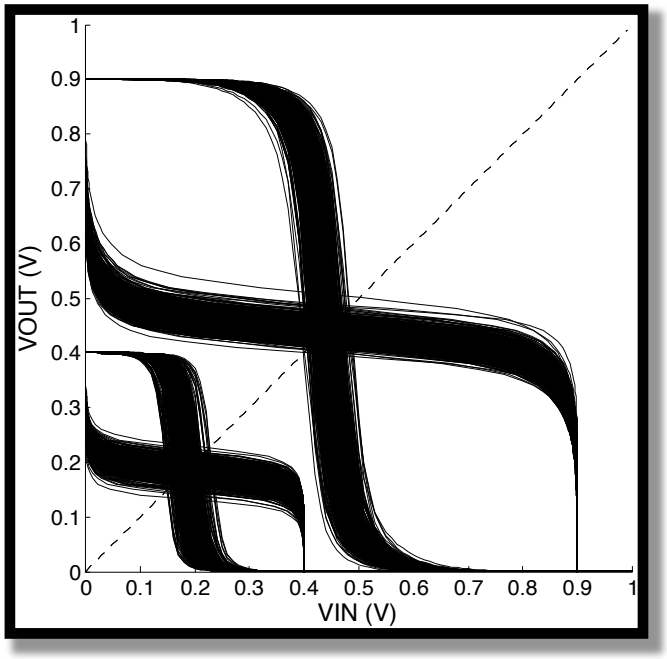
[2] UC Irvine Computer Science
*ics.uci.edu*

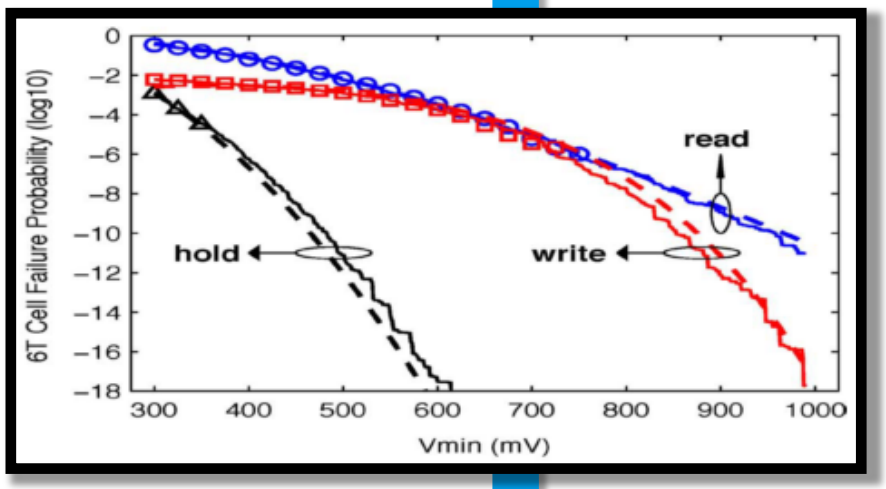
Design Automation Conference, San Francisco, CA
June 4, 2014

# Motivation: Increasing Process Variability Limits SRAM Voltage Scaling

Process variability ↑ → $V_T$ variations ↑

SRAM $\sigma_{SNM}$ ↑




J. Wang and B. Calhoun
TVLSI 2011

VDD ↓ → SRAM BER ↑ exponentially

# Question

**How to optimize SRAM**
**for the "best" *system-level* tradeoffs in**
**energy, reliability, performance, & area?**

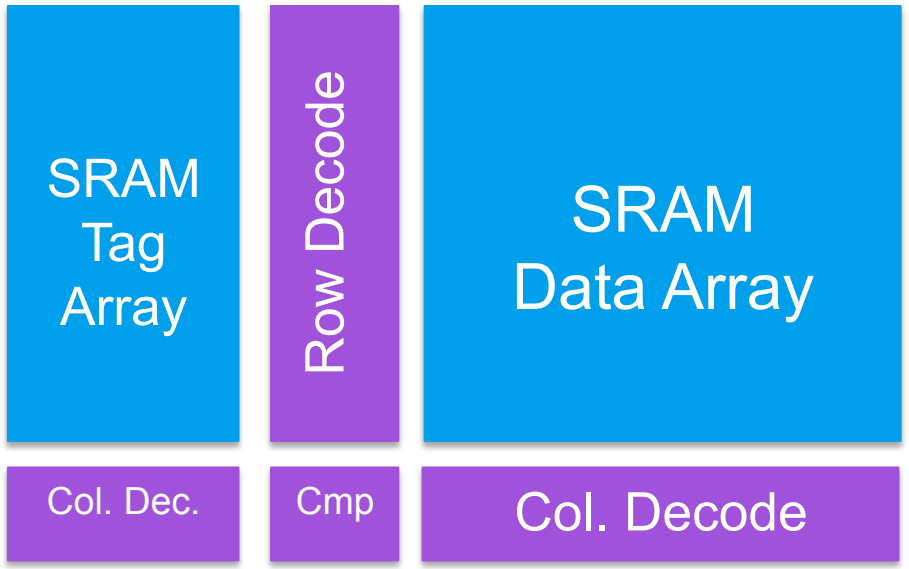
# Using Fault Tolerance to Achieve Lower min-VDD

Many fault-tolerant, voltage-scalable (FTVS) approaches lower min-VDD using sophisticated fault tolerance methods

**PURPLE** = periphery @ full VDD

**BLUE** = SRAM cells (bright is higher VDD)

Baseline Cache @ Nominal VDD – No Fault Tolerance

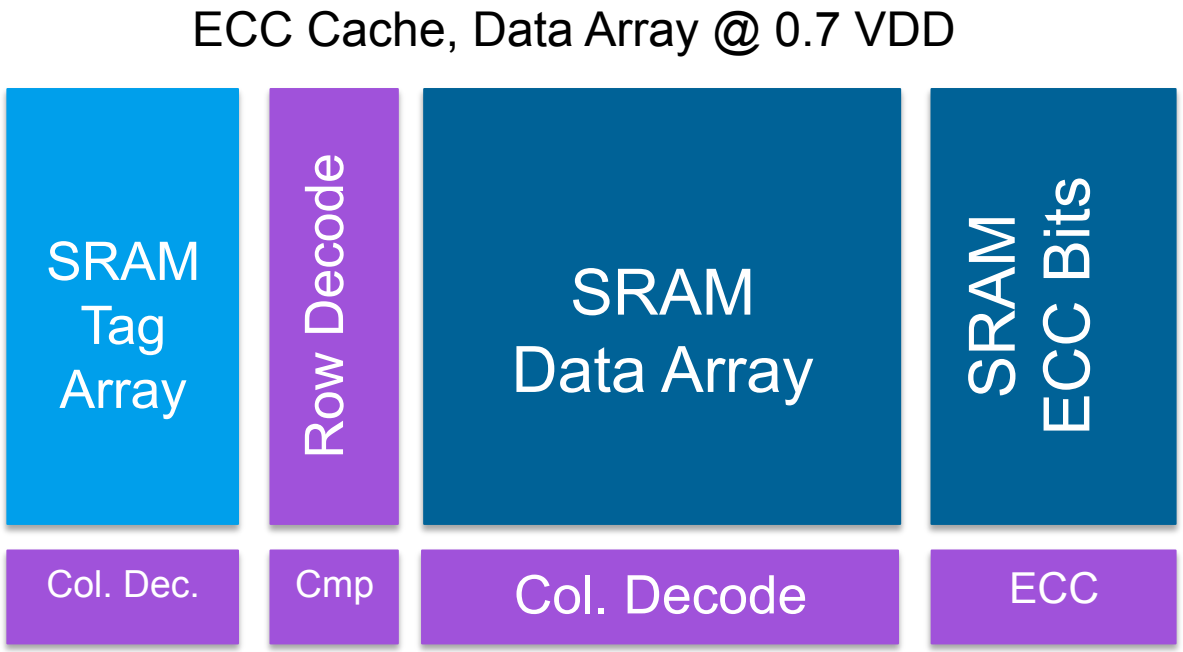


# Using Fault Tolerance to Achieve Lower min-VDD

Many fault-tolerant, voltage-scalable (FTVS) approaches lower min-VDD using sophisticated fault tolerance methods

**PURPLE** = periphery @ full VDD

**BLUE** = SRAM cells (bright is higher VDD)

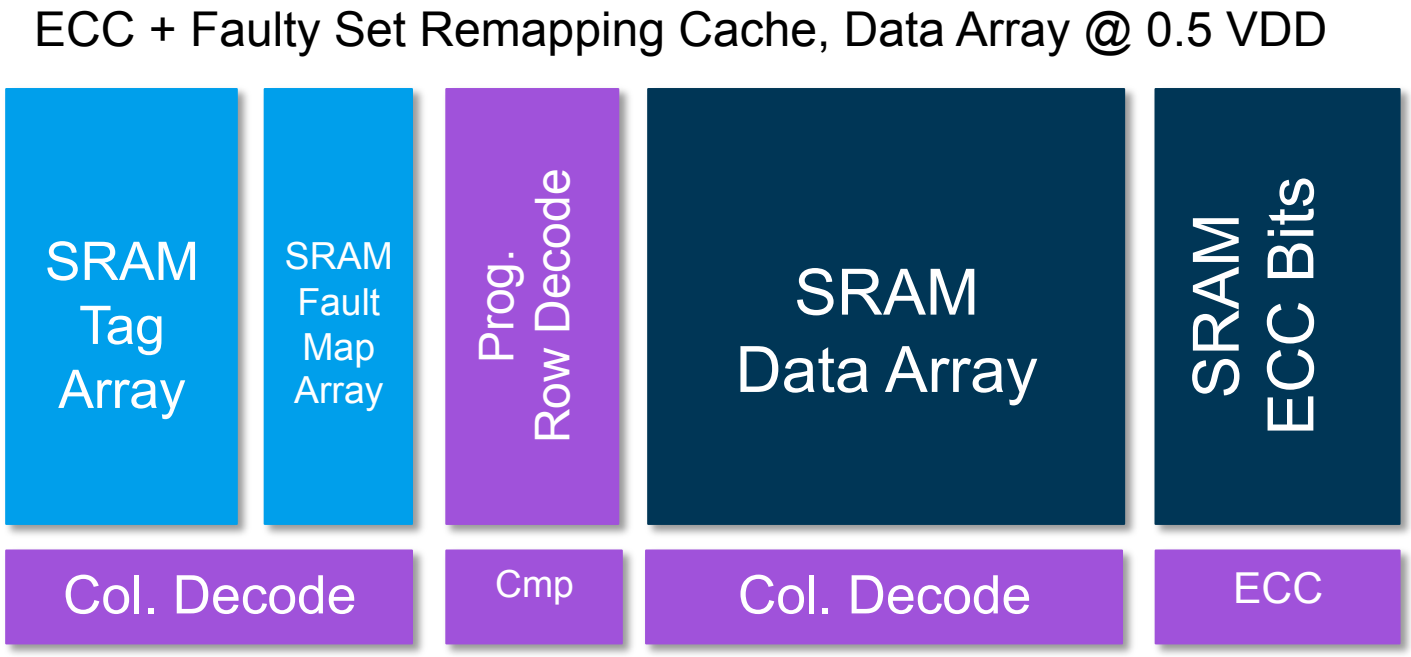


# Using Fault Tolerance to Achieve Lower min-VDD

Many fault-tolerant, voltage-scalable (FTVS) approaches lower min-VDD using sophisticated fault tolerance methods

**PURPLE** = periphery @ full VDD

**BLUE** = SRAM cells (bright is higher VDD)



# This Work

**Save energy via**

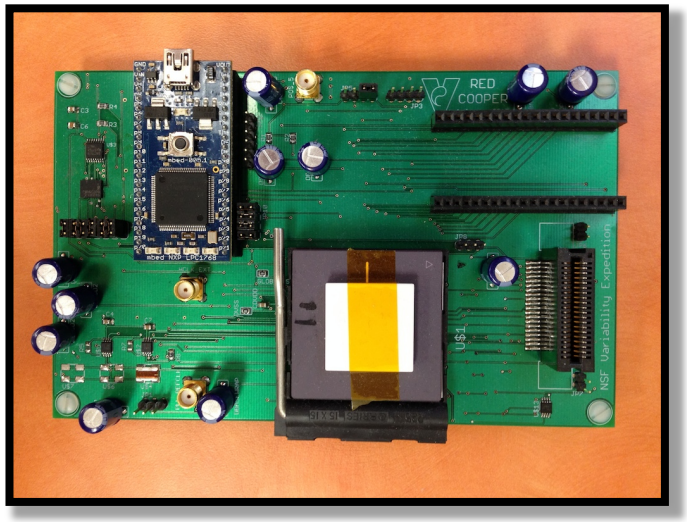
***simple & low-overhead***

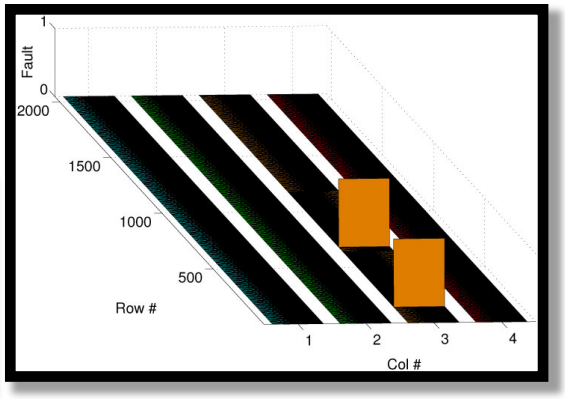
**fault-tolerant, voltage-scalable (FTVS)**

**SRAM cache architecture**

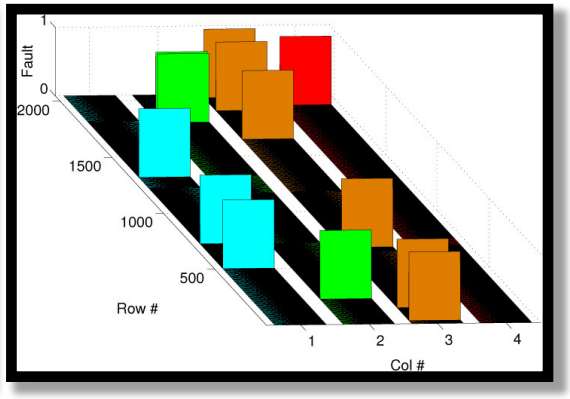
# SRAM “Fault Inclusion Property”

NSF Variability Expedition
“Red Cooper” test chips[1]
based on ARM Cortex M3

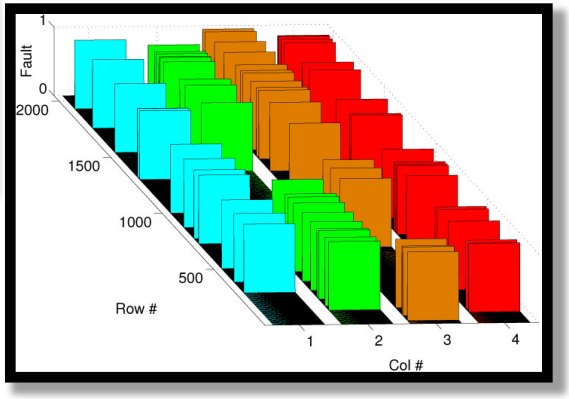


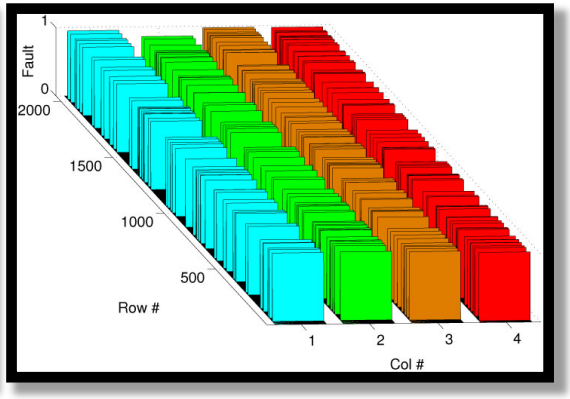

550 mV




525 mV



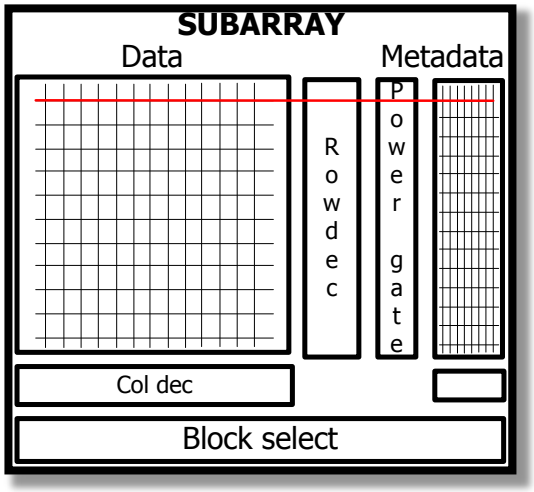

500 mV

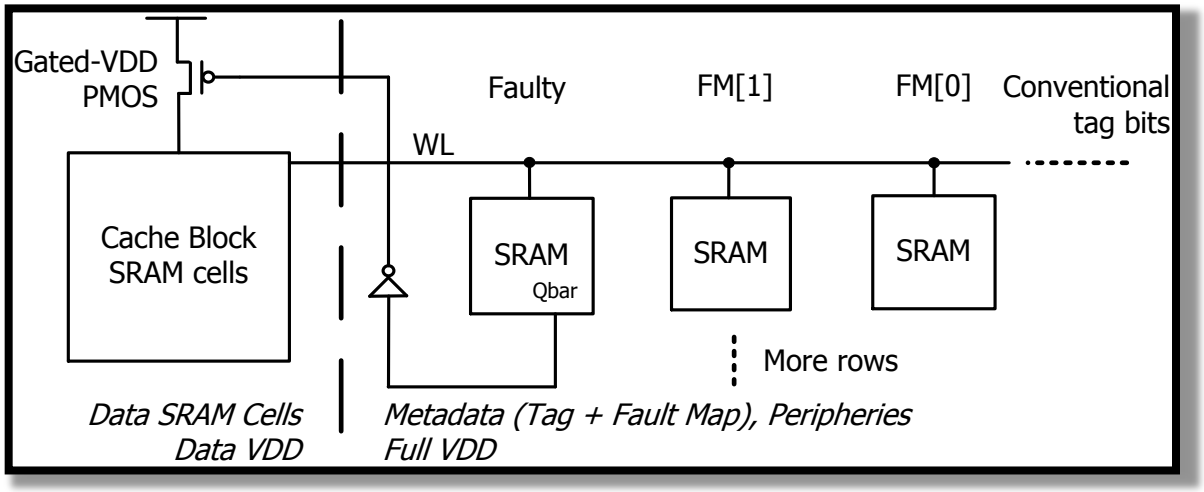



475 mV

[1]L. Lai ASP-DAC’14

# Architectural Mechanism




- No redundancy – just sacrifice faulty blocks as VDD scales
  - # good blocks fall off a "cliff" anyway
    - Redundancy can only do so much
  - Negligible area overhead

# Power/Capacity Scaling

- To adjust data array VDD:
  - Temporarily stall accesses
  - Cache controller finds the blocks that will become faulty at next VDD using *FM* bits
    - Flush those blocks that are also *Valid* & *Dirty*
    - Then set *Faulty* bits, power gating them
  - Adjust VDD, wait for voltage to settle
  - Resume operations

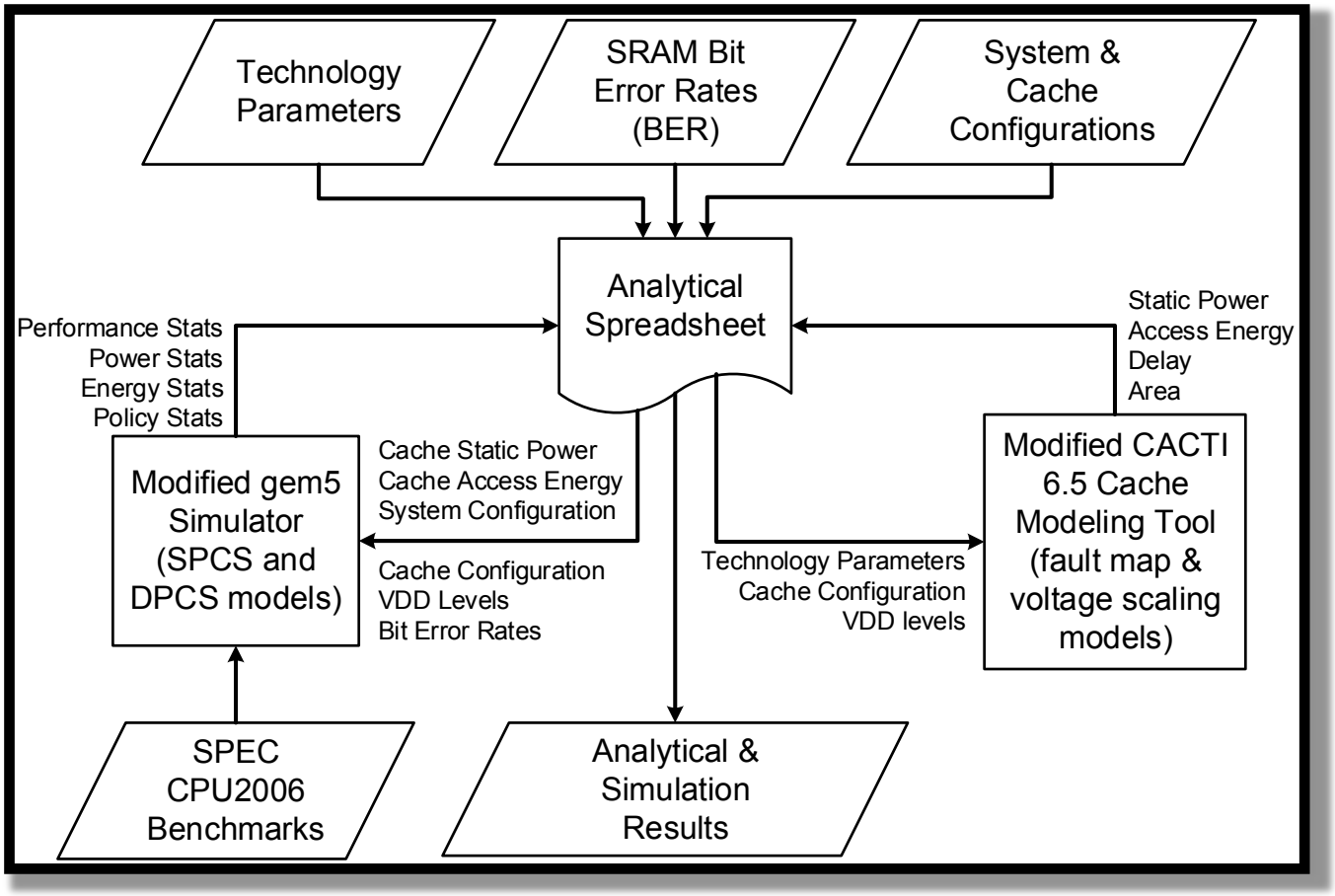
# Static & Dynamic Power/Capacity Scaling Policies

- *Static (SPCS) Policy*: Choose single optimal VDD at design, test, or boot time

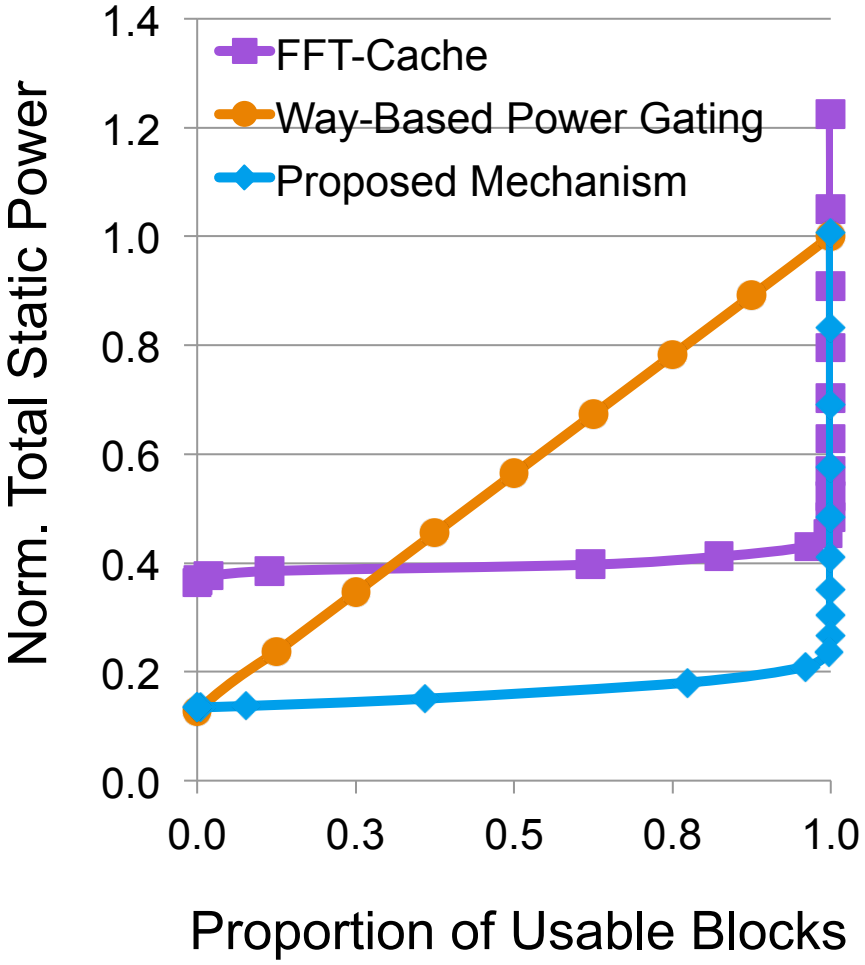
- *Dynamic (DPCS) Policy*: Decide on optimal VDD periodically during runtime

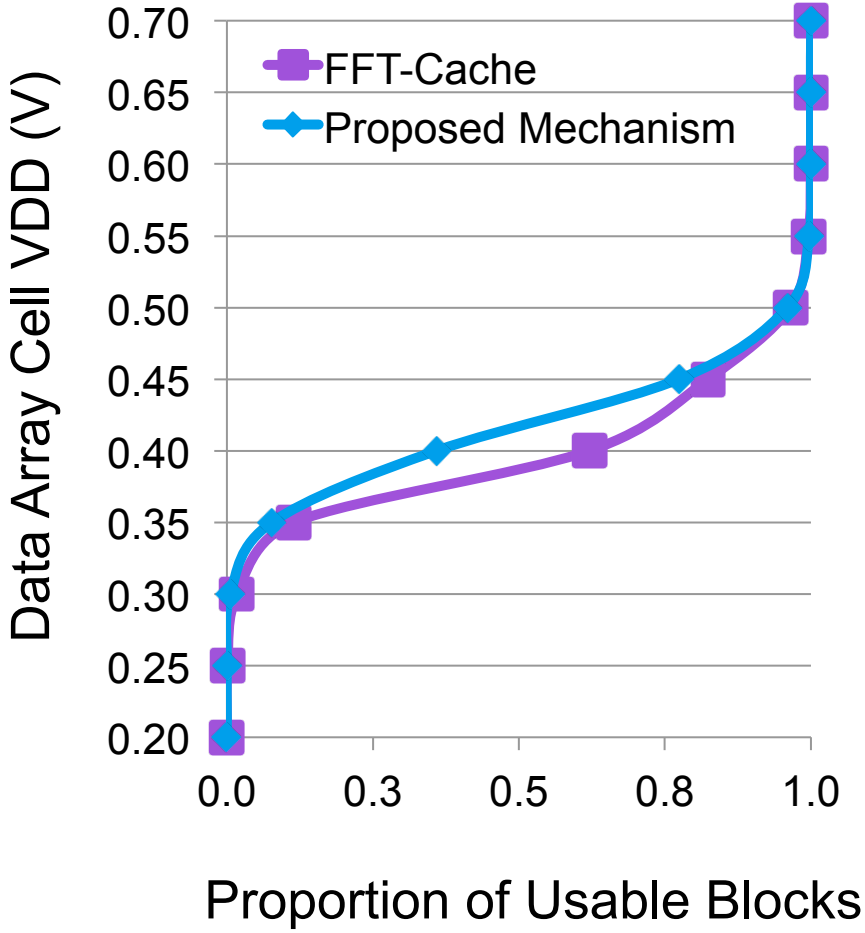
# Evaluation Setup

- 45nm SOI
- 2 system/ cache configurations for L1 & L2
- 3 permitted VDD levels
- SPEC CPU2006

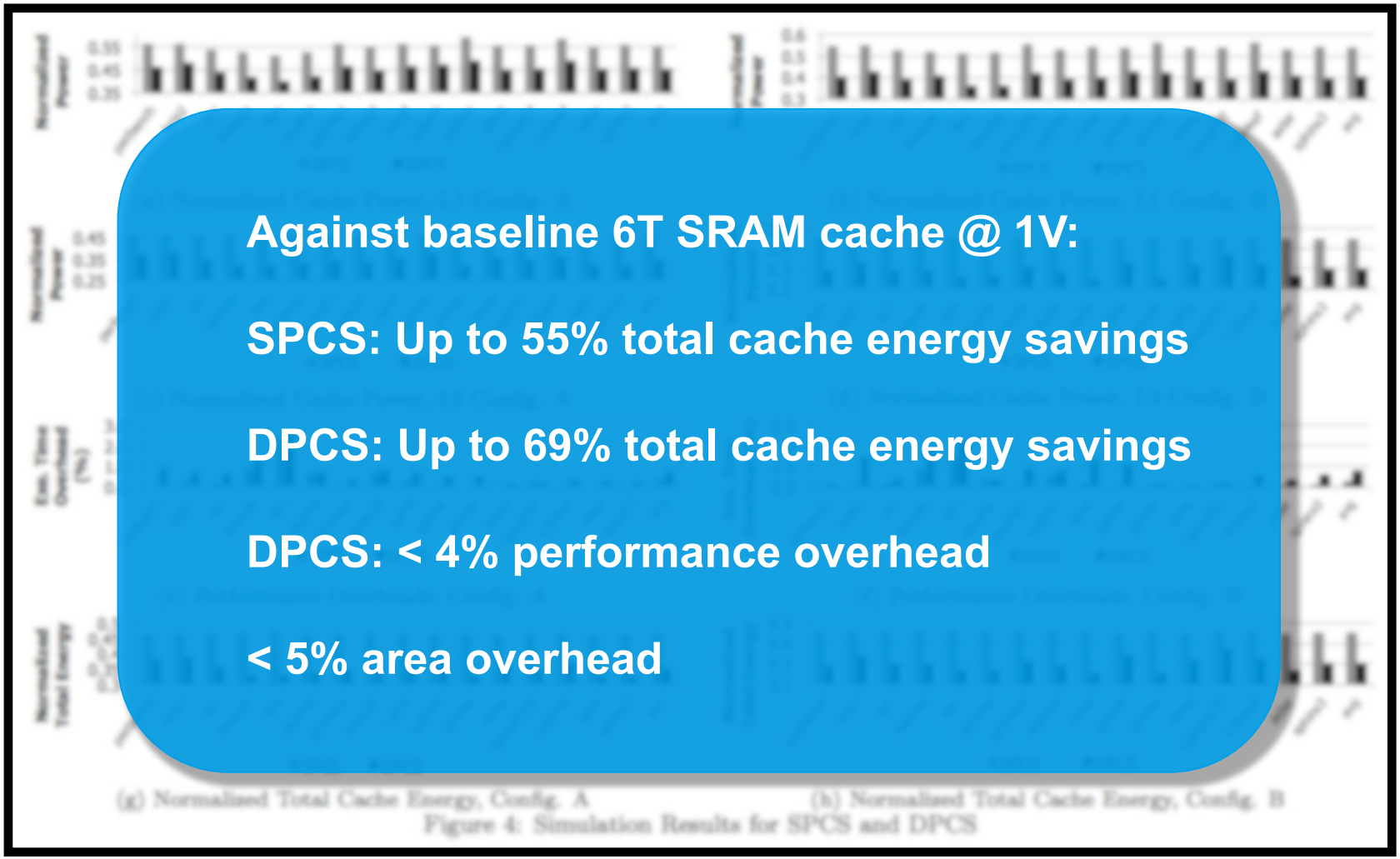


# Sample Analytical Results: L2 Cache Config. 1



# Simulation Results Summary




(a) Normalized Total Cache Energy, Config. A (b) Normalized Total Cache Energy, Config. B

Figure 4: Simulation Results for SPCS and DPCS

# Summary

Power vs. effective capacity

Fault Inclusion Property

Keep it simple

# Directions for Future Work

Optimality study of DPCS policies

DPCS for shared memory MPSoCs

DPCS-aware software strategies

# Q&A

**This work was supported by the NSF Variability Expedition Grant Numbers CCF-1029783 and CCF-1029030**

***www.variability.org***