Revisiting the Perfect Memory Paradigm: Designing Reliable systems in an Imperfect World

Fadi J. Kurdahi
Center for Embedded Computer Systems
University of California
Irvine, CA USA
The downside: an “Energy Gap”

[Ref: Delagi ISSCC 2010]
And also across the spectrum of electronics
Reducing power through $V_{dd}$ scaling

- We would like to lower $V_{dd}$ as much as possible
  - Dynamic power has square relationship with supply voltage ($V_{dd}$)
  - Leakage power has exponential relationship with $V_{dd}$
- However, lowering $V_{dd}$ too much results in erroneous operations

![Graph showing power saving vs. $V_{dd}$](image)

Almost 50% power saving

More than 50% power saving
Another Challenge: Variability

- Transistors on a chip are no longer identical
- Smaller features, low supply voltage make circuits more susceptible to noise
- Models used are increasingly inaccurate
- Temperature, etc…

Becoming more acute with advanced fabrication technologies

Lower Performance, yield, MTTF

Logic

Flip-flops

Memory

Bad

Worse

Worst!
How bad can Variability be?

Source: SR Nassif – IBM

26K recent IBM 65nm CPU

Straightforward solution:
Increase the supply voltage ($V_{dd}$)
- More noise immunity
- Faster

~50% variation
~10x variations!
So now we have a conflict!

- **Increase** $V_{dd}$ to compensate for variability, reliability, and performance
- **Decrease** $V_{dd}$ to reduce power

Programmable Systems have it worse!
- More memory
- More power
- Need high performance
What happens if I try to have my cake and eat it too?

**Conventional wisdom:** when you reduce $V_{dd}$, you should run slower. What happens if I reduce $V_{dd}$ and keep running at the same frequency?

Take a memory array, for example:
- Variability -> variable read, write delay across memory cells
- $V_{dd}$ reduced -> increased probability of access failure/cell -> more cells failing

![Memory Access Time Graph](image)
How bad are errors?

- For system designers: depends:
  - Systems are designed for worst case conditions
    - .... Which occur quite infrequently
  - Many systems are designed to inherently tolerate faults, but up to a limit
    - Wireless
    - Multimedia
  - Yet the realizations of those systems must be 100% correct wrt specs!
- For chip designers: catastrophic!
Error-Aware Design Concept

Co-design system and hardware to relax the 100% correctness requirement on the underlying hardware.

– If needed, fix the errors at the system/application level
– Improved power consumption by allowing aggressive voltage and frequency scaling
– Improved yield by relaxing the specifications defining a working chip.
Implications on the design space

- Proposed paradigm allows a new dimension for system optimization by encapsulating fault tolerance
- Traditional design space trades off power for performance
Existing work

- ERSA (Software level) [Mitra2010]
- Large body of work on RAM BIST/BISR [D&T2004]
- Breuer et. al. [D&T2003]
- Shanbhag et. al. [TVLSI2003]
- Roy et. Al [TVLSI2005]
- Timing-aggressive, error tolerant NoC design [Benini, DeMicheli 2007]
- EnerJ [2010] Approximation language
- Razor[2007]
- Shafique[2012] Error-resilient video codec
- Marwedel[2012]
- Many others.....
On chip SRAM contains 50-90% of total transistor count
- Xeon: 48M/110M
- Itanium 2: 144M/220M

SRAM is a major source of chip power dissipation
Nature of Memory Defects

• **Fixed**
  – Manufacturing errors
  – Predominant in above 100nm technologies
  – Redundancy solutions

• **Transient**
  – Alpha particles

• **Operating condition**
  – Voltage, frequency, temperature
  – Predominant in sub 100nm technologies
  – Defects are due to:
    • Gate Length Variation (GLV)
    • Random Dopant Fluctuation (RDF)
  – Manifest themselves at the circuit level as inter-die variation in Vt

Our focus
Understanding memory behavior

Berkeley Models

SPICE Monte Carlo simulation

Gaussian Vt

Value Flipping Time
Read Time
- DRF Success/Fail Border
- RAF Success/Fail Border

Time (ps)

\[ V_{dd} \]

\[ P_e \]

35ps
45ps
55ps
65ps
75ps
85ps

\[ 0 \quad 0.7 \quad 0.75 \quad 0.8 \quad 0.85 \quad 0.9 \]
CASE STUDY: WIRELESS SYSTEMS
Fault Tolerant Adaptation

Memory can be considered as another channel that contributes supply voltage dependent noise.
The Equi-Noise Concept

Data\textsubscript{in} \rightarrow \text{Modulation} \rightarrow \text{Filter} \rightarrow \text{AWGN} \rightarrow f_x(x) \rightarrow \text{Buffering Memory} \rightarrow f_y(y) \rightarrow \text{FIR} \rightarrow f_z(z) \rightarrow \text{Demodulation} \rightarrow \text{Data}_{\text{out}}

Error free

Data\textsubscript{in} \rightarrow \text{Modulation} \rightarrow \text{Filter} \rightarrow \text{Equi-Noise} \rightarrow f'_x(x) \rightarrow \text{Buffering Memory} \rightarrow f_y(y) \rightarrow \text{FIR} \rightarrow f_z(z) \rightarrow \text{Demodulation} \rightarrow \text{Data}_{\text{out}}
Memory failure is modeled as individual bit flips.

Error at $n^{th}$ location:

$$Y_n = X \pm 2^{n-r}$$

Unified Noise Model

$$f_Y(y) = \sum_{k=0}^{N} P(k)f_Y^k(y)$$

$$f_Y^k(y) = \frac{1}{k!} \sum_{n_1=0}^{N-1} \sum_{n_2=0, n_2 \neq n_1}^{N-1} \ldots \sum_{n_k=0, n_k \neq n_{k-1}}^{N-1} f_{Y_{n_1,n_2,\ldots,n_k}}(y)$$
Effect of FIR (1)

\[ z(n) = \sum_{k} h(k)y(n - k) \]
Effect of FIR (2)

Find PMF by obtaining DFT

\[ f_Y(y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega y} \varphi_Y(i\omega) d\omega \]

\[ \varphi_Z(i\omega) = E[e^{i\omega Z}] = E[e^{i\omega \sum_k h(k)y(n-k)}] \]

\[ \varphi_Z(i\omega) = E\left[ \prod_k e^{\sum_k i\omega h(k)y(n-k)} \right] \]

\[ \varphi_Z(i\omega) = \prod_k \varphi_Y(i\omega h(k)) \]

PMF of output data is a convolution of multiple scaled version of the input PMF

\[ f_Z(z) = \frac{1}{\prod_k |h(k)|} f_Y\left( \frac{y}{h(0)} \right) * f_Y\left( \frac{y}{h(1)} \right) * f_Y\left( \frac{y}{h(2)} \right) * \ldots \]
Simulation Results

Using the proposed model results in 375X speed up in simulation time
Propagating through FFT [Globecom 2010]
FFT Processing Through Low Power Faulty Memories

- BPSK Modulation
- 8 bits Memory (3,5)
  - 2048 FFT
- $P_e = 5 \times 10^{-3}, 10^{-3}, 10^{-4}$
- 135x speedup
Case Study

- BPSK, 2k FFT
- We assume N=8, d=2 and r=5, 1 sign bit
- Memory error: $P_e = 10^{-3}$
- Memory size: 158KB
  1. 35% chip area
  2. 25% chip power

* Lei-Fone Chen et al., “A 1.8V 250mW COFDM Baseband Receiver for DVB-T/H Applications”

Kurdahi ME AoW Talk 2013
Case Study

DVB-T, 2k FFT, Gl=1/8, QPSK, code rate 1/2

Over 40% power saving in memories which corresponds to 10% total chip power saving
Error-Aware Processor Caches

What is done today

Can we do this?

Nominal Vdd

Dynamic Vdd-Frequency Scaling

Nominal Vdd

Vdd Scaling

Access time

Access time

Power

Power

Errors

Errors

What is done today

Can we do this?
Comparing different techniques for resilient cache design
Error-Aware Video Decoder Design

• Without correction:
  – Can save 40% power
  – Serious degradation in image quality (20db drop in PSNR)
Error-Aware Video Decoder Design

With correction (median filters):
• About 35% savings in power
• Video quality almost fully restored

![Graph showing power savings and PSNR (dB) vs. Memory Vdd](image)

- **Nominal Case**
- **Reduced Voltage** ~40% less
- **Adaptive Reduced Voltage** ~35% less
What’s next?

• Logic modeling [ICCD 2012]
  – Modeling error propagation in RTL, system components

• Software level [Engel 2011]: Reliability annotations

```c
unreliable int u, x;
reliable int y, z;
...
x = y - ( z + u ) * 4
```
Concluding Thoughts

Caveats:

• What about other applications?
  – Network processors, sensor networks,…

• A general methodology?
  – Still needs research

• A paradigm shift?
  – Yes (CAD flow!!), but worth the effort

Take away messages:

1. *Think (un)reliability*: System designers should assume underlying SoCis unreliable
2. *Talk (un)reliability*: SoC designers should bring reliability modeling and consideration to the system/software level
Thank You