Robust Low Power Computing in the Nanoscale Era

Todd Austin
University of Michigan
austin@umich.edu

Thanks

- Slide/concept contributions by:
  - David Blaauw, University of Michigan
  - Kypros Constantinides, University of Michigan
  - Kris Flautner, ARM Ltd.
  - Nam Sung Kim, Intel Corporation
  - Trevor Mudge, University of Michigan
  - Leyla Nazhandali, Virginia Tech
  - Dennis Sylvester, University of Michigan
  - Chris Weaver, Intel Corporation
Evolution of a 90’s High-End Processor

- Compaq’s Alpha
- 67 A @ 100 W
- Power density 30 W/cm²

<table>
<thead>
<tr>
<th></th>
<th>Power (Watts)</th>
<th>Freq. (MHz)</th>
<th>Die Size (mm²)</th>
<th>Vdd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha 21064</td>
<td>30</td>
<td>200</td>
<td>234</td>
<td>3.3</td>
</tr>
<tr>
<td>Alpha 21164</td>
<td>50</td>
<td>300</td>
<td>299</td>
<td>3.3</td>
</tr>
<tr>
<td>Alpha 21264</td>
<td>72</td>
<td>667</td>
<td>302</td>
<td>2.0</td>
</tr>
<tr>
<td>Alpha 21364</td>
<td>100</td>
<td>1000</td>
<td>350</td>
<td>1.5</td>
</tr>
</tbody>
</table>

High 90’s Digital Signal Processor

- Analog Devices 21160 SHARC
  - 600 Mflops @ 2W
  - 100 Mhz SIMD with 6 computational units
- Recognized that parallelism saves power
- Had the right workload to exploit this fact

[We will see that the story has become more complicated]
Why does power matter?

- “… left unchecked, power consumption will reach 1200 Watts for high-end processors in 2018. … power consumption [is] a major shows topper with off-state current leakage ‘a limiter of integration’.”


---

Total Power of CPUs in PCs

- Early ‘90s – 100M CPUs @ 1.8W = 180MW
- Early 21st – 500M CPUs @ 18W = 10,000MW
- Exponential growth
- Recent comment in a Financial Times article: 10% of US’s energy use is for computers
  - exponentially growth implies it will overtake cars/homes/manufacturing
- NOT! – why we’re here
What hasn’t followed Moore’s Law

- Batteries have only improved their power capacity by about 5% every two years

Also Important For Server Systems

Internet Service Provider’s Data Center

- Heavy duty factory – 25,000 sq. ft. ~8,000 servers, ~2,000,000 Watts
- Want lowest cost/server/sq. ft.
- Cost a function of:
  - cooling air flow
  - power delivery
  - racking height
  - maintenance cost
  - lead cost driver is power ~25%
Why does robustness matter?

- ... the ability to consistently resolve critical dimensions of 30nm is severely compromised creating substantial uncertainty in device performance. ... at 30nm design will enter an era of “probabilistic computing,” with the behavior of logic gates no longer deterministic...

- susceptibility to single event upsets from radiation particle strikes will grow due to supply voltage scaling while power supply integrity (IR drop, inductive noise, electromigration failure) will be exacerbated by rapidly increasing current demand

- new approaches to robust and low power design will be crucial to the successful continuation of process scaling ...


Why does robustness matter?

- Grove’s comments
  - SEUs
  - IR drop
  - inductive noise
  - Electromigration, etc.

- Increase in variability as feature sizes decrease

- Likely to be the next major challenge
  - strengthen interest in fault-tolerance
  - renew interest in self-healing
How are they related?

- The move to smaller features can help with power – with qualifications
- Smaller features increase design margins
  - reduce power savings
  - reduce performance gains
  - reduced area benefits

Challenges

- Power density is growing
- Systems are becoming less robust
- Can architecture help?
  - Lower power organizations – quick estimates of power
  - Robust organizations – quick estimates of robustness
- By one account we need a 2x reduction in power/generation from architecture
- Question where will the solution come from
  - process
  - circuits
  - architecture
  - OS
  - language
Tutorial Schedule

- **Power Issues: Dynamic and Static Power**
  - Dynamic Power Overview
  - Static Power Overview
  - Power Trends
- Low Power Design Techniques
- Reliability Issues: SER, Variability and Defects
- Break
- Fault Tolerant Design Techniques
- Robust Low Power Design Techniques

Power Sources

- Total Power =
  Dynamic Power + Static Power + Short Circuit Power

![Power Sources Diagram]
Dynamic Power Consumption

- Inverter initial state:
  - Input 1
  - Output 0
- No dynamic power

\[
E_{\text{swing}} = \int P(t) \cdot dt
\]
\[
= \int V_{\text{out}}(t) \cdot dt
\]
\[
= \int V_{\text{out}} \cdot C \cdot dV_{\text{out}}
\]
\[
= CV_{\text{out}}^2
\]

- Energy drawn from power supply:

- Energy consumed by PMOS:

\[
E_{\text{pass}} = \int P(t) \cdot dt
\]
\[
= \int (V_{\text{in}} - V_{\text{out}}(t)) \cdot dt
\]
\[
= \frac{1}{2} CV_{\text{in}}^2
\]

- Power is

\[
P = f \cdot E_{\text{pass}} = \frac{1}{2} f CV_{\text{in}}^2
\]
Dynamic Power Consumption

- **Input 0→1**
  - Energy drawn from supply: 0
  - Energy consumed by NMOS equals to the energy stored on the capacitance:
    \[ E_{\text{stored}} = \int V(t) \cdot \text{d}t = \frac{1}{2} CV_{\text{dd}}^2 \]
  - Power is
    \[ P = f \cdot E_{\text{stored}} = \frac{1}{2} fCV_{\text{dd}}^2 \]

---

Leakage Current Components

- **Subthreshold leakage (I_{sub})**
  - Dominant when device is OFF
  - Enhanced by reduced \( V_t \) due to process scaling

- **Gate tunneling leakage (I_{gate})**
  - Due to aggressive scaling of the gate oxide layer thickness (\( T_{ox} \))
  - A super exponential function of \( T_{ox} \)
  - Comparable to \( I_{sub} \) at 90nm technology
Dynamic and Leakage Power Trends

ITRS 2002 projections with doubling # of transistors every two years

Temperature Dependence

- Temperature across chip varies significantly
- Sub-threshold leakage a strong function of temperature
- Gate leakage less sensitive to temperature
- Greater than 10% variation /10 deg C

Source: R. Rao
Tutorial Schedule

- Power Issues: Dynamic and Static Power
  - Low Power Design Techniques
    - Dynamic Power Reduction Techniques
    - Static Power Reduction Techniques
    - Research Topic: Subthreshold Sensor Processors
  - Reliability Issues: SER, Variability and Defects
- Break
  - Fault Tolerant Design Techniques
  - Robust Low Power Design Techniques

How to Reduce Dynamic Power

- More generally

\[ P_{\text{dyn}} = \frac{1}{2} \alpha f CV_{\text{dd}}^2 \]

where \( \alpha \) is switching activity

- To reduce dynamic power, we can reduce

\( \alpha \) – clock gating
\( C \) – sizing down
\( f \) – lower frequency
\( V_{\text{dd}} \) – lower voltage
Dynamic Power Reduction - Parallel Computation

Energy = $\frac{1}{2} CV_{dd}^2$

- Energy reduced by 50%, but double the area and more leakage

Dynamic Power Reduction - DVS

- Given dynamic workload - scale frequency or voltage
  - Clock/power gating - linear energy saving with duty cycle
    $$\text{Energy} = P_{Vdd} \cdot t_{on} = P_{Vdd} \cdot t_{task} \cdot \text{(duty cycle)}$$
  - Just-in-time Dynamic Voltage Scaling (DVS) - cubic energy saving with duty cycle
    $$\text{Energy} = P_{Vdd scales} \cdot t_{task} = (f_{scaled} \cdot C_{task} V_{scaled}^2) \cdot t_{task} \propto f_{scaled}^3 \cdot (t_{task} \cdot \text{duty cycle})^3$$
How Far Can We Scale Down the Voltage?

- **Traditional DVS (Dynamic Voltage Scaling)**
  - Scaling range limited to less than $V_{dd}/2$

<table>
<thead>
<tr>
<th></th>
<th>Voltage Range</th>
<th>Frequency Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM PowerPC 405LP</td>
<td>1.0V-1.8V</td>
<td>153M-333M</td>
</tr>
<tr>
<td>Transmeta Crusoe TM5800</td>
<td>0.8V-1.3V</td>
<td>300M-1G</td>
</tr>
<tr>
<td>Intel XScale 80200</td>
<td>0.95V-1.55V</td>
<td>333M-733M</td>
</tr>
</tbody>
</table>

- **Minimum functional voltage**
  - For an CMOS inverter is [Meindl, JSSC 2000]:
    $$V_{dd, min} = 2V_T \ln\left(1 + \frac{S_I}{\ln(10)}\right) \approx 48\text{mV} \text{ for a typical } 0.18\mu\text{m technology}$$

LongRun Power Management [Transmeta]

![Transmeta™ LongRun™ Power Management](image_url)

Source: Crusoe™ LongRun™ Power Management White Paper
**SpeedStep Technology [Intel]**

- Next generation Speedstep supports more V,F settings
- 10ms performance switch time
- Software algorithms to dynamically change settings based on performance statistics

![Graph showing power consumption versus core voltage](image)

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6 GHz (HFM)</td>
<td>1.484 V</td>
</tr>
<tr>
<td>1.4 GHz</td>
<td>1.420 V</td>
</tr>
<tr>
<td>1.2 GHz</td>
<td>1.276 V</td>
</tr>
<tr>
<td>1.0 GHz</td>
<td>1.164 V</td>
</tr>
<tr>
<td>800 MHz</td>
<td>1.036 V</td>
</tr>
<tr>
<td>600 MHz (LFM)</td>
<td>0.956 V</td>
</tr>
</tbody>
</table>

**Reducing Static Power with Dual Vₜ Assignments**

- Transistor is assigned either a high or low Vₜ
  - Low-Vₜ transistor has reduced delay and increased leakage

<table>
<thead>
<tr>
<th></th>
<th>Low-Vₜ: 0.9V</th>
<th>High-Vₜ: 0.9V</th>
<th>Low-Vₜ: 1.8V</th>
<th>High-Vₜ: 1.8V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leakage (norm)</td>
<td>1</td>
<td>0.06</td>
<td>1</td>
<td>0.07</td>
</tr>
<tr>
<td>Delay (norm)</td>
<td>1</td>
<td>1.30</td>
<td>1</td>
<td>1.20</td>
</tr>
</tbody>
</table>

- Trade-off degrades for lower supply voltage
Dual $V_t$ Example

- Dual $V_t$ assignment approach
  - Transistor on critical path: low $V_t$
  - Non-critical transistor: high $V_t$

- State Dependence ($I_{\text{sub}}$)
  - Simulation results of a 0.13um process

<table>
<thead>
<tr>
<th>Input ABC</th>
<th>Output</th>
<th>Subthreshold Leakage (pA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1</td>
<td>8.0836</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>15.1577</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>13.5167</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>55.2532</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>13.4401</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>54.5532</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>64.259</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>191.2692</td>
</tr>
</tbody>
</table>

- Approach: rework FSM state assignment or logic
Balloon Latch [Shigematsu]

On-Chip Cache Leakage Power

leakage is 57% of total cache power

Caches design with 70-nm BPTM and sub-banking technique
On-Chip Cache Leakage Power

- Large and fast caches
  - Improving memory system performance
  - Consuming sizeable fraction of total chip power
    - StrongARM – ~60% for on-chip L1 caches
- More caches integrated on chip
  - 2x64KB L1 / 1.5MB L2 in Alpha 21464
  - 256KB L2 / 3MB(6MB) L3 in Itanium 2
- Increasing on-chip cache leakage power
  - Proportional to \(\exp(1/V_{TH}) \times \# \text{ of bits}\)
  - 1MB L2 cache leakage power – 87% in 70nm tech

Drowsy Caches [Mudge]

- Put cache lines into low-power mode when idle
- Cache energy reductions of 54% to 58%
- Run time increase by 0.41% for awake tag (drowsy tag)
Energy Efficiency: A Key Requirement

- They live on a limited amount of energy generated from a small battery or scavenged from the environment.
- Traditionally the communication component is the most power-hungry element of the system. However, new trends are emerging:
Performance Demands are LOW

<table>
<thead>
<tr>
<th>Platform</th>
<th>Voltage (V)</th>
<th>Speed (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM 720T</td>
<td>1.2</td>
<td>100M</td>
</tr>
<tr>
<td>ARM 7TDMI</td>
<td>1.2</td>
<td>133M</td>
</tr>
<tr>
<td>ARM 920T</td>
<td>1.2</td>
<td>250M</td>
</tr>
<tr>
<td>ARM 1020T</td>
<td>1.2</td>
<td>325M</td>
</tr>
<tr>
<td>1st-gen</td>
<td>1.2</td>
<td>114M</td>
</tr>
<tr>
<td>1st-gen</td>
<td>0.5</td>
<td>9M</td>
</tr>
<tr>
<td>1st-gen</td>
<td>0.232</td>
<td>168k</td>
</tr>
</tbody>
</table>

The Basics of Subthreshold Circuit Operation

A Short Animation 😊
Episode 1: Inverter operation in superthreshold domain
Superthreshold

Superthreshold
Episode 2: Inverter operation in subthreshold domain
Subthreshold

0.2V IN → P OUT 0.2V
0V IN ← N OUT 0V

Subthreshold

0.2V IN → P OUT 0V
0V IN ← N OUT 0V

P P
N N

OUTIN

Subthreshold

0.2V IN → P OUT 0.2V
0V IN ← N OUT 0V

P P
N N

OUTIN
Energy Per Instruction Analysis

\[ E_{ipst} = E_{cycle} CPI \]

- **EPI**: Energy per Instruction
- **CPI**: Cycles per Instruction
- **Energy per Cycle**

\[ E_{cycle} = N\left(\frac{1}{2}C_s V_{dd}^2 + V_{dd} I_{leak} t_{clk}\right) \]

- **Activity factor**: average number of transistor switches per transistor per cycle
- **Total circuit capacitance**
- **Supply Voltage**
- **Leakage current**
- **Clock period**

Effect of reducing the voltage:

- \[ \downarrow \text{quad.} \]
  - \( E_{dynt} \)
  - \( E_{cycle} \)

\[ \uparrow \sim \text{exp.} \]

- Subthreshold
  - \( \sim \text{const.} \)
  - \( \sim \text{lin.} \)
- Superthreshold
  - \( \sim \text{lin.} \)

Energy Per Instruction Analysis

\[ E_{ipst} = E_{cycle} CPI \]

- **EPI**: Energy per Instruction
- **CPI**: Cycles per Instruction
- **Energy per Cycle**

Graphs showing energy vs. tension and energy vs. supply voltage.
1st-gen General Microarchitecture Overview and Exploration Options

- Number of stages
- Harvard vs. Von-Neumann arch
- ALU width
- Presence of instruction prefetch buffer
- Presence of explicit register file

First Subliminal Chip

- Large solar cell
- Solar cell for processor
- Custom memories
- Test module
- Test memory
- Level converter array
- Discrete adders
- Subliminal processors
- Mux-based memories
- Mux-based memories
Pareto Analysis for Several Processors

Pareto Analysis of Sensor Network Processors
Lessons from 1st-generation Study (ISCA 2005)

- To minimize energy at subthreshold voltages, architects must:
  - Minimize area ⇒ To reduce leakage energy per cycle
  - Maximize Transistor utility ⇒ To reduce $V_{\text{min}}$ and energy per cycle
  - Minimize CPI ⇒ To reduce Energy per instruction

- As such, winning designs tend to be compromising designs that balance area, transistor utility and CPI
- The memory comprises the single largest factor of leakage energy, therefore, efficient designs must reduce memory storage requirements.

2nd Generation Sensor Network Processor

- IF/ID Stage
- EX/MEM Stage
- WB Stage

- Imem 4x16x2x12
- Register File 24x16
- 32-bit Timer
- ALU
- Jump Control
- External Interrupts
- Scheduler
- Fetch Control
- Write Control
- Register 128x8
Ongoing Work

- To be deployed in an intra-ocular pressure sensor
- Provides measurement of internal eye pressure
- Integrated with a MEMS pressure sensor, wireless communication, and energy scavenging facilities

Intra-ocular Pressure Sensor

Tutorial Schedule

- Power Issues: Dynamic and Static Power
- Low Power Design Techniques
- Reliability Issues: SER, Variability and Defects
  - Soft Error Radiation Overview
  - Variability Sources and Effects
  - Silicon Defect Trends
- Break
- Fault Tolerant Design Techniques
- Robust Low Power Design Techniques
Fault Classes

- **Permanent fault (hard fault)**
  - Irreversible physical change
  - Latent manufacturing defects, Electromigration

- **Intermittent fault**
  - Hard to differentiate from transient faults
    - Repeatedly occurs at the same location
    - Occurs in bursty manners when fault is activated
    - Replacing the offending circuit removes faults

- **Transient faults (Soft Errors)**
  - Neutron/Alpha particle strikes
  - Power supply and Interconnect noises
  - Electromagnetic interference
  - Electrostatic discharge

Introduction – Soft Errors

- **Soft errors**, also called **transient faults** and **single-event upsets (SEU)**
  - Processor execution errors caused by high-energy neutrons resulting from cosmic radiation and alpha particles radiation
  - Appears to be a reliability threat for future technology processors

- When a particle strikes a circuit element a small amount of charge is deposited
  - *Combinational logic node*: a very short duration pulse of current is formed at the circuit node
  - *State holding element (FF/SRAM cell)*: flip the stored value

- Unlike permanent faults the effects of soft errors are transient
Soft Errors (SER)

- Alpha particles stemming from radioactive decay of packaging materials
- Neutrons (cosmic rays) are always present in the atmosphere
- Soft errors are transient non-recurring faults (also called single event upsets, SEUs) where added/deleted charge on a node results in a functional error
  - Charge is added/removed by electron/hole pairs absorbed by source/drain diffusion areas

Source: S. Mukherjee, Intel

Soft Error Masking

- **Logic Masking**: the fault gets blocked by a following gate whose output is completely determined by its other inputs

- **Timing Masking**: the fault affects the input of a latch only in the period of time that the latch is not sensitive to its input
Soft Error Masking

- **Electrical Masking**: the fault’s pulse is attenuated by subsequent logic gates due to electrical properties, and does not affect any latch’s input

- **Microarchitectural Masking**: the fault alters a value of at least one flip-flop, but the incorrect values get overwritten without being used in any computation affecting the design’s output

- **Software Masking**: the fault propagates to the design’s output but is subsequently masked by software without affecting the application’s correct execution

How To Measure Reliability:
Soft Error Rate (FIT)

- **Failure In Time (FIT)**: Failures in $10^9$ hours
  - 114 FIT means
    - 1 failure every 1000 years
    - It sounds good, but
      - If 100,000 units are shipped in market, 1 end-user per week will experience a failure

- **Mean Time to Failure**: $1 / FIT
Soft Error Considerations

- Highly elevation dependent (3-5X higher in Denver vs. sea-level, or 100X higher in airplane)
- Critical charge of a node \( Q_{\text{crit}} \) is an important value
  - Node requires \( Q_{\text{crit}} \) to be collected before an error will result
  - The more charge stored on a node, the larger \( Q_{\text{crit}} \) is (\( Q_{\text{crit}} \) must be an appreciable fraction of stored \( Q \))
  - Implies scaling problems \( \rightarrow \) caps reduce with scaling, voltage reduces, so stored \( Q \) reduces as \( S^2 \) (~2X) per generation
    - Ameliorated somewhat by smaller collection nodes (S/D junctions)
    - But exacerbated again by 2X more devices per generation

Soft Error Rate Trends, ITRS03

Figure 14. SER/chip for SRAM/latches/logic
Impact of Soft Errors in Processors [Iyer]

- How do soft errors in processors propagate and impact applications?

- Approach
  - Fault injections (with i-Measure, hardware level fault injection framework) in combinational logic and flip-flops of MIPS and Alpha-like processors
  - Study fault propagation to the application level

- Major findings:
  - Nearly 5% of faults in combinational logic propagate to state of the processor
  - Errors in Control contribute to 79% of application hangs
  - Errors in Execution blocks a major factor in application crashes (45%) and silent data corruption (40%)
  - Faults in combinational logic can cause double and multiple bit errors

SER Analysis Tool [Shanbhag]

- Gate-level SER analysis point tool (available from GSRC web-site)
- Fast: Speed-up $\geq 10^6$ over Monte Carlo
- Accurate: $< 5\%$ error over Monte Carlo
- Captures SER dependence on: process, circuit and input vectors

![Multiple Bit-flip Distribution in Alpha processor]

- Single Bit-Flip Error: 83.11%
- Double Bit-Flip Errors: 15.10%
- Multiple Bit-flip Errors: 1.79%

![32x32 array multiplier]

- $\Delta V_{dd} = 20\% \rightarrow \text{SER} = 1.28X$
- $\Delta t_{\text{setup}} = 20\% \rightarrow \text{SER} = 50X$
Effects Of Variability

- High-performance processors are speed-binned
  - Faster == more $$$
  - These parts have small $L_{eff}$
- Exponential dependence of leakage on $V_{th}$
  - And $L_{eff}$, through $V_{th}$

Since leakage is now appreciable, parametric yield is being squeezed on both sides.

Printing in the Subwavelength Regime

Layout

0.25µ

0.18µ

0.13µ

90-nm

65-nm

Figures courtesy Synopsys Inc.
Variation: Across-Wafer Frequency

Random Dopant Fluctuations, Intel’s View

Figure courtesy S. Nassif, IBM
Inter-die vs. Intra-die Variation

Inter-die variation is not always larger than intra-die (ILD)

Design/EDA for Highly Variable Technologies

- Critical need: Move away from deterministic CAD flow and worst-case corner approaches
- Examples:
  - Probabilistic dual-Vth insertion
    - Low-Vth devices exhibit large process spreads; speed improvements and leakage penalties are thus highly variable
  - Parametric yield optimization
    - Making design decisions (in sizing, circuit topology, etc.) that quantitatively target meeting a delay spec AND a power spec with given confidence
  - Avoid designing to unrealistic worst-case specs
  - Use other design tweaks such as gate length biasing (next)
Noise Immune Layout Fabric

This layout style trades off area for:

- Noise immunity (both C and L)
- Minimizes variations (CMP)
- Predictable
- Easy layout
- Simplifies power distribution

Major area penalty (>60%)

Ref: Khatri, DAC99

Defects: The (Bumpy) Road Ahead for Silicon

- What is the failure model of silicon 2-3 generations out?
  - What the literature says…
    - “Expected failure rate of $10^{12}$ hours/device”, this would give a high end NVidia graphics part an expected lifetime of less than 1 year
    - “Failure rates higher than $10^{20}$ hours/device”, which eliminates the problem
  - What the experts say…
    - Intel [Borkar] and IBM [Bernstein]: critical problem for future silicon

- Key failure modes
  - Transistor wear-out (aggravated by scaling)
  - SER-related upsets (especially in logic)
  - Early transistor failures (due to ineffective burn-in)
  - Untestable defects (compounded by complexity)
Silicon Defects: Sources and Trajectory

- Sources: gate wearout, NBTI, hot electrons, electro-metal migration, etc...

![Diagram showing failure rate models and time periods]

Model Parameters:
- \( F_G \): grace period wear-out rate
- \( \lambda_L \): avg latent manufacturing defects
- \( m \): maturing rate
- \( b \): breakdown rate
- \( t_B \): breakdown start point

- Failures occur very soon and failure rate declines rapidly. Failures are caused by latent manufacturing defects.
- Failures occur with increasing frequency over time due to age-related wear-out.
- Failure rate falls to a small constant value where failures occur sporadically due to the occasional breakdown of weak transistors or interconnect.

Tutorial Schedule

- Power Issues: Dynamic and Static Power
- Low Power Design Techniques
- Reliability Issues: SER, Variability and Defects
- Break
- Fault Tolerant Design Techniques
  - Classical Techniques
  - SER Specific Techniques
  - Full-Spectrum Techniques
  - Research Topic: Self-Healing Systems
- Robust Low Power Design Techniques
Techniques For Improving Reliability

- Fault avoidance (Process / Circuit)
  - Improving materials
    - Low Alpha Emission interconnect and Packaging materials
  - Manufacturing process
    - Silicon On Insulator (SOI)
    - Triple Well design process to protect SRAM

- Fault tolerance (robust design in presence of Soft Error) : Circuit / Architecture
  - Error Detection & Correction relies mostly on “Redundancy”
    - Space : DMR, TMR
    - Time : Temporal redundant sampling (Razor-like)
    - Information : Error coding (ECC)

DMR Error Detection

- Context: Dual-modular redundancy for computation
- Problem: Error detection across blades
Triple Modular Redundancy (von Neumann)

Voter assumed reliable!
⇒ voter small
⇒ coarse-grained

Error Coding: Information Redundancy

- Coding: representation of information
  - Sequence of code words or symbols
  - Shannon’s theorem in 1948
    - In noisy channels, errors can be reduced to a certain degree
    - Golay(1949), Hamming(1950), Stepian(1956), Prange(1957), Huffman
- Overheads
  - Spatial overhead: Additional bits required
  - Temporal overhead: Time to encode and decode
- Terminology
  - Distance of code
    - Minimum hamming distance between any two valid codewords
  - Code separability (e.g., Parity Code)
    - Code is separable if code has separate code and data fields
**SER-Tolerant Circuit Design [Shanbhag]**

- Employed skewed CMOS for logic and dual sampling FF (DSFF).
- Both 0→1 and 1→0 errors are eliminated if skewing factor \( \geq 4 \).
- Speed penalty:
  - depends on \( \Delta \) (maximum SET width)
  - can be made a design parameter.
  - equals 300ps (for 0.18um process) if zero SER wanted.
- Power penalty: 17% (DSFF) + 20% (Skewed CMOS)

---

**Fingerprinting [Falsafi/Hoe]**

- Hash updates to architectural state
- Fingerprints compared across DMR pair
  - Bounded error detection latency
  - Reduced comparison bandwidth

<table>
<thead>
<tr>
<th>Instruction stream</th>
<th>Stream of updates</th>
<th>Fingerprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 \leftarrow R_2 + R_3 )</td>
<td>...001010101011010100101010...</td>
<td>( 0xC3C9 )</td>
</tr>
<tr>
<td>( R_2 \leftarrow M[10] )</td>
<td>( R_1 )</td>
<td>( R_2 )</td>
</tr>
</tbody>
</table>
**Recovery Model**

- Checkpoint n
- Soft error
- Error Undetected
- Recover to n
- Error undetected

➢ *Rollback-recovery to last checkpoint upon detection*

---

**Simultaneous Redundant Multithreading**

-[Reinhardt]-

*Logical boundary of redundant execution within a system*

- Trade-off between information, time, & space redundancy

**Sphere of Replication**

- Thread 1
- Thread 2
- Input Replication
- Output Comparison
- Rest of System

Compare & validate output before sending it outside the SoR
Full-Spectrum Fault Tolerance: DIVA Checker [Austin]

- All core function is validated by checker
  - Simple checker detects and corrects faulty results, restarts core
- Checker relaxes burden of correctness on core processor
  - Tolerates design errors, electrical faults, defects, and failures
  - Core has burden of accurate prediction, as checker is 15x slower
- Core does heavy lifting, removes hazards that slow checker

Checker Processor Architecture

- Core Processor Prediction Stream
  - Core PC
  - Core inst
  - Core regs
Check Mode

Core Processor Prediction Stream

Recovery Mode

51
How Can the Simple Checker Keep Up?

**Redundant Core**

- Slipstream effects reduce power requirements of trailing car
  - Checker processor executes in the core processor slipstream
  - Fast moving air ⇒ branch/value predictions and cache prefetches
  - Core processor slipstream reduces complexity requirements of checker
- Symbiotic effects produce a higher combined speed

**Advance Core**

---

How Can the Simple Checker Keep Up?

**Simple Checker**

- Slipstream effects reduce power requirements of trailing car
  - Checker processor executes in the core processor slipstream
  - Fast moving air ⇒ branch/value predictions and cache prefetches
  - Core processor slipstream reduces complexity requirements of checker
- Symbiotic effects produce a higher combined speed

**Complex Core**
Checker Performance Impacts

- **Checker throughput** bounds core IPC
  - Only cache misses stall checker pipeline
  - Core warms cache, leaving few stalls
- **Checker latency** stalls retirement
  - Stalls decode when speculative state buffers fill (LSQ, ROB)
  - Stalled instructions mostly nuked!
- **Storage hazards** stall core progress
  - Checker may stall core if it lacks resources
- **Faults** flush core to recover state
  - Small impact if faults are infrequent

Research Topic: Self-Repairing Systems

- Defect-tolerant self-repairing systems need to support:
  - Error Detection
  - System Diagnosis (locate the origin of the error)
  - System Repair
  - System Recovery
- Key idea:
  - Error detection must be performance efficient
    - Continuously check execution for errors
  - Diagnosis, repair and recovery are insensitive on performance
    - Get invoked only when an error is detected (rare scenario)
    - Trade-off performance for more cost efficient techniques
Fault Modeling & Analysis Infrastructure

- High-performance, high-fidelity, fault modeling simulation infrastructure
  - Asynchronous fault injection at the gate level
  - Fully models all the possible ways a fault can be masked

- Two different setups, one to evaluate the effects of transients, and one for permanent errors
- Monte Carlo modeling framework with realistic workloads

Self-Repairing BulletProof Silicon [Austin, Bertacco]

Goal: Single-defect tolerance for 5% area overhead

Key ideas:
- No expensive computation checking
- Protect computation and test Hw
- Repair by disabling redundant parts

Approach:
1. Execute and protect state
2. Test concurrently when Hw idle
3. If tests fails → roll back state
   → disable component
   → restart
### Tutorial Schedule

- Power Issues: Dynamic and Static Power
- Low Power Design Techniques
- Reliability Issues: SER, Variability and Defects
- Break
- Fault Tolerant Design Techniques
- Robust Low Power Design Techniques
  - Better-Than Worst Case Design Concepts
  - Example BTWC Designs
  - Research Topic: Razor Pipeline

### Power and Reliability: How are they related?

- The move to smaller features can help with power – with qualifications
- Smaller features increase design margins
  - reduce power savings
  - reduce performance gains
  - reduced area benefits
**Traditional Worst-Case Design**

Design-Time Verification and Optimization

- Time-to-Market: L → H
- Performance: L → H

**Better-Than-Worst-Case (BTWC) Design**

Typical Case Optimization

- Run-Time Verification
- Time-to-Market: L → H
- Performance: L → H
Algorithmic SER-Tolerance [Shanbhag]

- Voltage Overscale Main Block
- Error Control via Estimator
- Estimators: Prediction, Reduced Precision Replica, MAP, Error Canceller and others
- Employ two estimators in SEU/MEU scenario
- Robust to error frequencies up to:
  - 1 in 100 samples for SEU
  - 1 in 1000 samples for MEU

Timing Error Tolerant Links [De Micheli]

- Aggressively clock on-chips links with high frequency/low voltage
  - Double-sample link output
  - Once speculatively, then again with reliable timing
- Stall receiver for recovery data if samples disagree
  - Non-speculative if receiver incurs additional delay
  - Otherwise, receiver must perform internal recover
Research Topic: Razor Error Resilient Circuits [Austin/Blaauw]

**In-situ** detection/correction of timing errors
- Tune processor voltage based on errors
- Eliminate process, temperature, and noise margins (tune for near-zero errors)
- Purposely run below critical voltage to capture *data-dependent latency margins*

**Implemented with architecture and circuit support**
- Double-sampling metastability-tolerant Razor flip-flops validate pipeline results
- Pipeline initiates recovery after timing errors, forward progress is guaranteed

Razor Prototype Chip

- 4 stage 64-bit Alpha pipeline
  - 120 - 160MHz operation, 0.18μm
- Percentage of FF Razorized: 9%
  - Error free Razor overhead ~3%
- 54% energy reduction
Configuration of the Razor Voltage Controller

Configuration of the Razor Voltage Controller

Run-Time Response of Razor Voltage Controller

Runtime Samples

Percentage Error Rate

Voltage Output of Controller

E_{diff} = E_{ref} - E_{sample}

Voltage Control Function

Voltage Regulator

Pipeline

reset

Σ

120MHz

27C
**Energy/Performance Characteristics**

![Graph showing energy and performance characteristics](image)

**Conclusions**

- Power Issues: Dynamic and Static Power
- Low Power Design Techniques
- Reliability Issues: SER, Variability and Defects
- Break
- Fault Tolerant Design Techniques
- Robust Low Power Design Techniques
References

1. C. Constantinescu 'Trend and Challenge in VLSI Circuit Reliability' Intel
2. H. T. Nguyen 'A Systematic Approach to Processor SER Estimation and Solutions'
3. P. Shivakumar et. al, 'Modeling the effect of Technology trends on Soft Error Rate of Combinational Logic'
5. M. Nicolaidis 'Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies'
6. L. Anghel, et. al. 'Cost Reduction and Evaluation of a Temporary Faults Detecting Technique'
7. L. anghel, et. al. 'Evaluation of Soft Error Tolerance Technique based on Time and/or Space Redundancy' ICSD
8. I. Koren, University of Massachusetts ECE 655 Lecture Notes 4-5 'Coding'
9. ITRS 2003 Report
11. R. E. Lyons, et. al. 'The Use of Triple-Modular Redundancy to Improve Computer Reliability'
13. C. Weaver, et. al. 'A Fault Tolerant Approach to Microprocessor Design' DSN'01
16. S. Reinhardt, et. al. 'Transient Fault Detection Simultaneous Multithreading'

References

1. D. Siewiorek 'Fault Tolerance in Commercial Computers' CMU
3. T. Siegel et.al 'IBM's S/390 G5 Microprocessor Design'
5. D. Bossen et.al 'Fault tolerant design of the IBM pSeries 690 system using POWER4 processor technology'
6. 'Tandem HP Himalaya' White Paper
7. Fujitsu SPARC64 V Microprocessor Provides Foundation for PRIMEPOWER Performance and Reliability Leadership
8. D. J. Sorin, et. al. 'SafetyNet: Improving the Availability of SharedMemory Multiprocessors with Global Checkpoint/Recovery.'
10. J. Smolens, et.al 'Fingerprinting: Bounding SoftError Detection Latency and Bandwidth'
11. D. Sorin, et.al 'Dynamic Verification of End-to-End Multiprocessor Invariants'
Questions?

?      ?      ?      ?      