Welcome to the Low Power Robust Computing Tutorial

Todd Austin, David Blaauw, Krisztián Flautner, Nam Sung Kim, Trevor Mudge, Dennis Sylvester

Introduction

Trevor Mudge
tnm@umich.edu
The University of Michigan

Thanks to:
Shaun D'Souza, Taeho Kgil & Dave Roberts
Past Tutorials & Workshops

  D. Grunwald, S. Manne, T. Mudge
  D. Grunwald, S. Manne, T. Mudge
  D. Grunwald, M. Irwin, T. Mudge

Single thread performance was still king

Evolution of a 90's High-End Processor

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

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>Alpha 21164</td>
<td>50</td>
<td>300</td>
<td>299</td>
</tr>
<tr>
<td>Alpha 21264</td>
<td>72</td>
<td>667</td>
<td>302</td>
</tr>
<tr>
<td>Alpha 21364</td>
<td>100</td>
<td>1000</td>
<td>350</td>
</tr>
</tbody>
</table>
But there was another viewpoint

- ISLPED had been going strong for several years
- Design Automation Conference had fostered low power studies
- Manufacturers of untethered devices were acutely aware of power needs

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’.”


---

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 …

Power and Robustness

- ".. power has become a first order concern at the 90nm node."
- "The new paradigm for us as designers is that we are designing to a fixed performance instead of a fixed voltage,"
- "I know what kind of voltage I want to achieve, the question is 'what kind of voltage variation can I make and still achieve the required level of performance?'"
- "... EDA vendors need to develop technologies that allow designers to use multiple voltage domains and employ robust electrical rule checking ... tools need to better understand boundary conditions and variable Vdd .."
- "Tools also need to support multiple Vt libraries and need to help users apply "sleep" and "drowsy modes" on logic in addition to memory higher up in the design flow..."

Texas Instruments Fellow, Peter Rickert /ICCAD keynote Nov. 2004

Power is a 1st Class Design Constraint

- For untethered computing devices – Obvious
For Aggregated Systems too

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%

Total Power of CPUs in PCs
- Early ’90’s – 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

Low power has other implications …

- Low power has been the technology that defines mainstream computing technology
  - Vacuum tubes → silicon
  - TTL → CMOS
  - microprocessors
- 1950’s “supercomputers” created the technology
- 1980's supercomputer were the beneficiaries of microprocessor technology
- 1990's microprocessors led to PDAs/cell phones/etc
- Will the tethered computers of the 21st century be the beneficiaries of mobile computer technology
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

A System Challenge for the Near Future

- What the end-users really want: supercomputer performance in their pockets…
  - Untethered operation, always-on communications
  - Driven by applications (games, positioning, advanced signal processing, etc.)
- Mobile supercomputing

- All with very tiny batteries

<table>
<thead>
<tr>
<th>Workload</th>
<th>Performance Req’ed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft-radio</td>
<td>4x</td>
</tr>
<tr>
<td>Crypto-processing</td>
<td>4x</td>
</tr>
<tr>
<td>Augmented reality</td>
<td>4x</td>
</tr>
<tr>
<td>Speech recognition</td>
<td>2x</td>
</tr>
<tr>
<td>Mobile Applications</td>
<td>2x</td>
</tr>
</tbody>
</table>
Outline of the Presentations

- David Blaauw (U. Michigan)
  - *Physical basis for power consumption in CMOS*
- Kris Flautner (ARM Ltd.)
  - *System-Level energy management*
- Nam Sung Kim (Intel CRL)
  - *Low power memory systems*
- Dennis Sylvester (U. Michigan)
  - *Physical basis of variability*
- Todd Austin (U. Michigan)
  - *Robust computing*

Schedule

- 8:30 a – Start
- 10:00 a – Break
- 10:30 a – Resume
- Noon – Lunch
- 1:00 p – Resume
- 2:30 p – Break
- 3:00 p – Resume
- 6:00 p – Reception
Static and Dynamic Power Analysis
and Circuit Level Reduction Methods

David Blaauw
Bo Zhai
University of Michigan

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
- Removing safety margin using Razor
Power Sources

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

Dynamic Power Consumption

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

- **Input 1→0**
  - Energy drawn from power supply:
    \[ E_{\text{supply}} = \int P(t) \cdot dt \]
    \[ = \int V_d \cdot i_d(t) \cdot dt \]
    \[ = \int V_d \cdot C \cdot dV_o \]
    \[ = CV_o^2 \]
  - Energy consumed by PMOS:
    \[ E_{\text{PMOS}} = \int P(t) \cdot dt \]
    \[ = \int (V_{in} - V_o(t)) \cdot dt \]
    \[ = \frac{1}{2} CV_o^2 \]
  - Power is
    \[ P = f \cdot E_{\text{PMOS}} = \frac{1}{2} fCV_o^2 \]

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

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
- Removing safety margin using Razor

How to Reduce Dynamic Power

- More generally
  \[ P_{\text{dyn}} = \frac{1}{2} \alpha f C V_{dd}^2 \]
  where \( \alpha \) is switching activity

- To reduce dynamic power, we can reduce
  \( \alpha \) – clock gating
  \( C \) – sizing down
  \( f \) – lower frequency
  \( V_{dd} \) – lower voltage
Dynamic Power Reduction - Parallel Computation

\[ \text{Energy} = CV_{dd}^2 \]

\[ \text{Energy} = 2 \cdot C \left( \frac{V_{dd}}{2} \right)^2 = \frac{1}{2} CV_{dd}^2 \]

- 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
    \[ \text{Energy} = P_{on} \cdot t_{on} = P_{idle} \cdot t_{on} \cdot (\text{duty cycle}) \]
  - Just-in-time Dynamic Voltage Scaling (DVS) - cubic energy saving with duty cycle
    \[ \text{Energy} = P_{scaled} \cdot t_{on} = (f_{scaled} \cdot C \cdot V_{scaled}^2) \cdot t_{on} \approx f_{scaled} \cdot t_{on} \approx (f_{on} \cdot \text{duty cycle})^3 \]
How Far Should 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]:
    $$V_{dd,\text{min}} = 2V_{T} \ln(1 + \frac{V_{T}}{\ln(10 \cdot V_{T})}) \approx 48\text{mV} \text{ for a typical } 0.18\mu\text{m technology}$$

Is there a Minimum Energy Point?

- **Superthreshold region**
  - Active energy scales down quadratically with $V_{dd}$
  - Leakage power scales down linearly with $V_{dd}$, delay scales up almost linearly with $1/V_{dd}$, leakage energy stays approximately constant with $V_{dd}$.

- **Subthreshold region**
  - Active energy scales down quadratically with $V_{dd}$
  - Leakage power scales down linearly with $V_{dd}$, delay scales up exponentially with $V_{dd}$, leakage energy scales up almost exponentially with $V_{dd}$
  - Minimum Energy Point ($V_{\text{min}}$) takes place when leakage energy becomes comparable with active energy
Minimum Energy Point ($V_{\text{min}}$) Modeling

- Factors affecting $V_{\text{min}}$:

  when $\alpha \uparrow, n \downarrow, T \downarrow, V_{\text{min}} \downarrow \alpha, n, T, S$

  
  \[ V_{\text{min}} = [1.587 \ln(n \cdot n_{\text{eff}}) - 2.355] \cdot \frac{m}{q} \frac{kT}{q} \]

  less gates are leaking

  less time to leak due to path delay

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
- Removing safety margin using Razor
Leakage Current Components

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

- **Gate tunneling leakage (I\text{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

Dual $V_t$ Assignments

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

<table>
<thead>
<tr>
<th></th>
<th>Low-$V_t$; 0.9V</th>
<th>High-$V_t$; 0.9V</th>
<th>Low-$V_t$; 1.8V</th>
<th>High-$V_t$; 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
Standby Leakage Estimation for Transistor Stacks

- Leakage current of a gate depends on input state
- Consider a 4-input NAND
  - For \langle 1111 \rangle, the leakage current is determined by the pull up network
    \( V_{DD} = 1.5V, V_T = 0.25V \)
  - For other combinations, the leakage current is determined by the pull down network
  - So called *stack effect*

\[ I_{\text{leak}} = \sum_{i=0}^{4} I_{\text{sub}_i} \]

\[ I_{\text{ddq}} = 9.96\text{nA} \]

[Chen, et al., ISLPED98]

State Dependence (\( I_{\text{sub}} \))

- Simulation results of a 0.13um process
- Three OFF transistors in stack
- One OFF transistor in stack
- 8X increase in leakage

Source: F. Najm
State Dependence of Leakage Current

- Circuit state is partially known or unknown in sleep state
- Leakage variation is less for entire circuit than for individual gates

<table>
<thead>
<tr>
<th>Component</th>
<th>Leakage Current (nA)</th>
<th>Max / Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data path</td>
<td>Min 11.42</td>
<td>Mean 21.36</td>
</tr>
<tr>
<td></td>
<td>Adder1 256.8</td>
<td>283.1</td>
</tr>
<tr>
<td></td>
<td>Control 33.8</td>
<td>45.97</td>
</tr>
<tr>
<td></td>
<td>Decoder 1702.5</td>
<td>1914.3</td>
</tr>
<tr>
<td></td>
<td>Nand4 0.07</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>OAI21 0.84</td>
<td>7.73</td>
</tr>
<tr>
<td></td>
<td>Tinv 0.37</td>
<td>1.89</td>
</tr>
<tr>
<td></td>
<td>AOI21 2.44</td>
<td>8.51</td>
</tr>
</tbody>
</table>

Leakage Current Profile

- Distribution of leakage states
- Distribution strongly dependent on circuit topology
Average Leakage Measure

- Battery life is more directly related to average leakage than maximum leakage
  - Device enters standby mode many times over battery life time
- Approaches
  - Apply random vectors at input
  - Accurate results for circuit level leakage with limited number of random vectors
- For gate/transistor optimization, accurate leakage current measurement on each gate is needed
  - Leakage current varies dramatically on individual gates
  - Random vectors not effective in computing average leakage of individual gates in circuit

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
- Removing safety margin using Razor
Gate Oxide Leakage in an Inverter

- When input = $V_{dd}$
  - NMOS: maximum $I_{gate}$
  - PMOS: maximum $I_{sub}$, reduced $I_{gate}$
- When input = 0V
  - NMOS: $V_{gd}$ = negative
    $\Rightarrow I_{gd}$: restricted to reverse gate tunneling
    - maximum $I_{sub}$, reduced $I_{gate}$
  - PMOS: small $I_{gate}$
- $I_{gate}$ & $I_{sub}$
  - can be independently calculated and added for total leakage

Leakage Modeling (switch level)

- Scenario 1: Transistor positioned
  - Above 0 or more conducting transistors
  - Below 1 or more non-conducting transistors
    - $I_{gate}$ of transistor added to $I_{sub}$ of stack
- Scenario 2: Transistor positioned
  - Above 1 or more non-conducting transistors
  - Below 0 or more conducting transistors
    - Adjacent nodes are near VDD and thus gate leakage can be ignored
- Scenario 3: There is a non-conducting transistor above and below
  - $I_{gate}$ depends on $I_{sub}$
  - Increases in $I_{gate}$ pinch off $I_{sub}$
State Dependence ($I_{\text{gate}}$)

NAND3

<table>
<thead>
<tr>
<th>Input ABC</th>
<th>Output</th>
<th>Subthreshold Leakage (pA)</th>
<th>Gate Leakage (pA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1</td>
<td>8.0836</td>
<td>200.0241</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>15.1873</td>
<td>131.8958</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>13.5167</td>
<td>192.9729</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>55.2532</td>
<td>95.4877</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>13.4401</td>
<td>327.9802</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>54.5532</td>
<td>256.4272</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>64.259</td>
<td>455.7905</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>191.269</td>
<td>486.6814</td>
</tr>
</tbody>
</table>

- Gate Leakage is minimized when
  - The bottom transistor in a stack is OFF
    - This forces intermediate nodes in the stack to be near VDD
  - All other transistors in the stack are ON
    - This allows the complementary pull-up network transistors, which are in a parallel structure, to be OFF

Leakage Current Trends

$I_{\text{OFF}}$ @ 25°C (nA/μm)

Source: F. Najm
Leakage Projection

Year


Current [\(\mu A/\mu m\)]


Subthreshold current

Effective gate tunneling current

High-k dielectrics expected to reach mainstream

Technology node [nm]

Current leakage projection:

- Subthreshold current
- Effective gate tunneling current

Gate vs. Sub-threshold Leakage

- Leakage contribution heavily topology dependent
- Gate leakage contribution: ~30%
  - Expected to be 50% by next generation
- Gate leakage greater for Nand structures
  - Wider NMOS stack
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

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
  - MTCMOS
  - Dual Vt
  - State Assignment
  - VTCMOS
- Removing safety margin using Razor

Source: R. Rao
Leakage Reduction Overview

MTCMOS

- Dual Threshold
- State Assignment
- Variable $V_t$

Source: [Johnson, et al., DAC99]

MTCMOS Overview

- MTCMOS (Multi Threshold CMOS)
- Active mode
  - Low $V_t$ circuit operation
- Standby mode
  - Disconnect power supplies through high $V_t$ devices
- For fine grain sleep control
  - Sequential circuits must retain state
- Dual sleep devices are needed for sneak paths in state retaining latches

Source: [Mutoh, et al., JSSC 8/95]
State Retaining MTCMOS Latch

(Low $V_{th}$ Inverter)

High $V_{th}$ Inverters for State Retention

Setup Time Penalty

[Mutoh, et al., JSSC 8/95]

Sneak Leakage Path with Single Sleep Transistor

(Low $V_{th}$ Inverter)

- Need for both polarity high $V_t$ sleep devices

[Mutoh, et al., JSSC 8/95]
Balloon Latch

Retaining State through Scan

- Scan out state before entering standby mode
  - No state retaining flip-flop necessary
  - Single footer is sufficient
- Non-power gated memory needed
- Use existing scan circuitry
  - Slower transition to/from standby mode
Addressing $I_{\text{gate}}$ in MTCMOS

- Use header instead of footer sleep transistor
  - Relies on lower $I_{\text{gate}}$ in PMOS transistor

![Diagram of Addressing $I_{\text{gate}}$ in MTCMOS](image)

[Hamzaoglu, et al., ISLPED02]

Boosted Gate MOS (BGMOS)

- Use a thick oxide, high $V_t$ sleep transistor
  - Suppress both $I_{\text{sub}}$ and $I_{\text{gate}}$
- During active mode, overdrive sleep transistor gate input

![Diagram of Boosted Gate MOS (BGMOS)](image)

[Inukai, et al., CICC2000]
Sizing of Sleep Transistor

- Sleep transistor introduces additional supply voltage drop
  - Degradation in performance
  - Signal integrity issues
- Careful sizing of sleep transistor is needed
- Sharing virtual supply between gates reduces voltage fluctuation

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
  - MTCMOS
  - Dual Vt
  - State Assignment
  - VTCMOS
- Removing safety margin using Razor
Dual $V_t$ Example

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

---

V$_t$ Assignment Granularity

- **V$_t$ assignment can be at different level of granularity**
  - Gate based assignment
  - Pull up network / Pull down network based assignment
    - Single $V_t$ in P pull up or N pull down trees
  - Stack based assignment
    - Single $V_t$ in series connected transistors
  - Individually assignment within transistor stacks
    - Possible area penalty

- **Number of library cells increases with finer control**
  - Better leakage / delay trade-off
Example of Different $V_t$ Assignment Granularity

Source: [Wei, et al., DAC99]

Simultaneous $V_t$, Size and $V_{dd}$ Assignment - Result

- Adding $V_{dd}$ to $W/V_t$ resulted in average
  - 60% decrease over $W$ only
  - 25% decrease over $W/V_t$.

Benchmark Circuit

[Nguyen, et al., ISLPED03]
Increasing Device Length

- Increase in length decreases leakage, due to short channel effects
  - Delay penalty due to loss of device current and increased input loading

[Delay normalize w.r.t Hi-Vt transistor]
[Leakage normalized w.r.t Low-Vt transistor]

[Blauw, et al., ISLPED98]

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
  - MTCMOS
  - Dual Vt
  - State Assignment
  - VTCMOS
- Removing safety margin using Razor
Combining $V_t$ and Input State Assignment

- Given a known input state in standby mode, only “OFF” transistors set to high $V_t$
- All other transistors are kept at low $V_t$

[Lee, et al., DAC03]

Combining $V_t$ and Input State Assignment

- Optimal input state with $V_t$ assignment
  - Increased reduction of leakage current

[Lee, et al., DAC03]
Stack Order Dependence of $I_{gate}$

- Key difference between the state dependence of $I_{sub}$ and $I_{gate}$
  - $I_{sub}$ primarily depends on the number of OFF transistors in stack
  - $I_{gate}$ depends strongly on the position of ON/OFF transistors in stack

<table>
<thead>
<tr>
<th>Sta</th>
<th>$I_{sub}$</th>
<th>$I_{gate}$</th>
<th>$I_{total}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0.38</td>
<td>0.00</td>
<td>0.38</td>
</tr>
<tr>
<td>001</td>
<td>0.79</td>
<td>6.38</td>
<td>7.04</td>
</tr>
<tr>
<td>010</td>
<td>0.79</td>
<td>1.28</td>
<td>1.28</td>
</tr>
<tr>
<td>100</td>
<td>5.68</td>
<td>12.6</td>
<td>18.2</td>
</tr>
<tr>
<td>101</td>
<td>0.81</td>
<td>0.62</td>
<td>0.82</td>
</tr>
<tr>
<td>110</td>
<td>3.81</td>
<td>6.30</td>
<td>10.1</td>
</tr>
<tr>
<td>111</td>
<td>28.2</td>
<td>19.6</td>
<td>47.8</td>
</tr>
</tbody>
</table>

Source: [Lee, et al., DAC03]

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
  - MTCMOS
  - Dual Vt
  - State Assignment
  - VTCMOS
- Removing safety margin using Razor
VTCMOS

- Variable Threshold CMOS (from T. Kuroda, ISSCC, 1996)
- In active mode:
  - Zero or slightly forward body bias for high speed
- In standby mode:
  - Deep reverse body bias for low leakage
- Triple well technology required

Speed Adaptive $V_t$ CMOS

- Dynamically tune $V_t$ so that critical path speed matched clock period
- Reduces chip-to-chip parameter variations
- Reverse bias:
  - Operate only as fast as necessary (reduces excess active leakage)
- Forward bias:
  - Speeds up slow chips
  - Standby leakage with maximum reverse bias
- Also known as Adaptive Body Biasing (ABB)

Outline

- Power Consumption in CMOS Circuit
- Dynamic Power Reduction Methods
- Subthreshold Leakage Analysis
- Gate-Leakage Analysis
- Leakage Reduction Methods
- Removing safety margin using Razor

Impact of Process Scaling on Design

- Increasing uncertainty with process scaling
  - Inter- and intra-die process variations
  - Temperature variation
  - Power supply drop
  - Capacitive and inductive noise
- Robust Design increasing difficult
  - Reduced yield
  - Difficulty in design closure
  - Worst-case design requires large safety margins
  - High energy
- Alarming uncertainty in Nanotechnologies
Robust Design for Low Power Applications

Low power antagonistic to robust design
- Increased sensitivity to $V_t$ variation in low voltage operation
  - Dynamic voltage scaling
  - Subthreshold voltage operation
- Clock gating and low power modes increase power grid noise
- Power optimization equalizes circuit delay
  - Number of paths that can lead to chip failure dramatically increased

Fundamental challenge in nanometer design: **Robust and Low Power Design**

---

Robust Low Power Design

- **Worst-case conditions highly improbable**
  - Many sources of variability are independent (process, noise, SEU, supply drop)
  - Probability of all sources simultaneously having worst-case condition very low
  - "guaranteed correct" design highly inefficient
- **Common case design paradigm**
  - Significant gain for circuits optimized for common case
- **Efficiency mechanisms needed to tolerate infrequent worst-case scenarios**
  - In-situ error detection and correction
  - Dynamic runtime adjustment to silicon and environmental conditions
Self-Regulating DVS with Razor

- **Goal:** reduce voltage margins with *in-situ* error detection and correction for delay failures

- **Proposed Approach:**
  - Tune processor voltage based on error rate
  - Eliminate safety margins, purposely run below critical voltage
    - Data-dependent latency margins
    - Trade-off: voltage power savings vs. overhead of correction
  - Analogous to wireless power modulation

---

Razor Flip-Flop Implementation

- Compare latched data with *shadow-latch* on delayed clock

- Upon failure: place data from shadow-latch in main latch
  - Ensure shadow latch always correct using conservative design techniques

- Key design issues:
  - Maintaining pipeline forward progress - Recovering pipeline state after errors
  - Short path impact on shadow-latch - Meta-stable results in main flip-flop
  - Power overhead of error detection and correction
Centralized Pipeline Recovery Control

- Once cycle penalty for timing failure
- Global synchronization may be difficult for fast, complex designs
- Implementation currently being explored for ARM 926 commercial core

Distributed Pipeline Recovery Control

- Builds on existing branch / data speculation recovery framework
- Multiple cycle penalty for timing failure
- Scalable design since all recovery communication is local
- Prototype chip results available
**Trade-Off in Razor DVS**

- **Total Energy**
- **Processor Energy**
- **Pipeline IPC**
- **Supply Voltage**
- **Optimal Voltage**
- **Recovery Energy**

<table>
<thead>
<tr>
<th>Technology Node</th>
<th>0.18µm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Frequency</td>
<td>120 - 140MHz</td>
</tr>
<tr>
<td>DVS Supply Voltage Range</td>
<td>1.2-1.8V</td>
</tr>
<tr>
<td>Total Number of Transistors</td>
<td>1.58million</td>
</tr>
<tr>
<td>Die Size</td>
<td>3.3mm*3.6mm</td>
</tr>
<tr>
<td>Measured Chip Power at 1.8V</td>
<td>130mW</td>
</tr>
<tr>
<td>Icache Size</td>
<td>8KB</td>
</tr>
<tr>
<td>Dcache Size</td>
<td>8KB</td>
</tr>
<tr>
<td>Total Number of Flip-Flops</td>
<td>2408</td>
</tr>
<tr>
<td>Total Number of Razor Flip-Flops</td>
<td>207</td>
</tr>
<tr>
<td>Number of Delay Buffers Added</td>
<td>2498</td>
</tr>
<tr>
<td>Error Free Operation (Simulation Results)</td>
<td></td>
</tr>
<tr>
<td>Standard FF Energy (Static/Switching)</td>
<td>49fJ/124fJ</td>
</tr>
<tr>
<td>RFF Energy (Static/Switching)</td>
<td>60fJ/185fJ</td>
</tr>
<tr>
<td>Total Delay Buffer Power Overhead</td>
<td>3.7mW</td>
</tr>
<tr>
<td>% Total Chip Power Overhead</td>
<td>2.9%</td>
</tr>
<tr>
<td>Error Correction and Recovery Overhead</td>
<td></td>
</tr>
<tr>
<td>Energy of a RFF per error event</td>
<td>260fJ</td>
</tr>
</tbody>
</table>
Razor I - Prototype Testbed

Razor I - Prototype Testbed
Configuration of the Razor Voltage Controller

\[ E_{\text{diff}} = E_{\text{ref}} - E_{\text{sample}} \]

Configuration of the Razor Voltage Control System

Run-Time Response of Razor Voltage Controller

Percentage Error Rate

Voltage Output of Controller

Runtime Samples

120MHz

27°C
Questions

?? ?? ?? ?? ?? ?? ?? ??
System-Level Energy Management

Krisztián Flautner
krisztian.flautner@arm.com
ARM Limited

Why does energy efficiency matter?

<table>
<thead>
<tr>
<th>Year</th>
<th>Device</th>
<th>Battery Type</th>
<th>Weight</th>
<th>Features</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>Motorola DynaTAC 8000X</td>
<td>Lead Acid, 500g</td>
<td>800g</td>
<td>Talk for brokers</td>
<td>$3995</td>
</tr>
<tr>
<td>1995</td>
<td>Nokia 232</td>
<td>NiMh, 100g</td>
<td>205g</td>
<td>Talk for the masses</td>
<td>$500</td>
</tr>
<tr>
<td>2003</td>
<td>Nokia 6600</td>
<td>Li-ion, 21g</td>
<td>125g</td>
<td>Talk, play, web, snap, video, organize</td>
<td>$500</td>
</tr>
</tbody>
</table>

- The disappearing battery - despite only incremental capacity improvements: the rest of the system has become more power efficient!
- Power has major impact on form factor, features, cost ➔ marketability
### Smart-phone system power budget

<table>
<thead>
<tr>
<th>Smart-phone system power (W) during different operating modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak power</td>
</tr>
<tr>
<td>Gaming</td>
</tr>
<tr>
<td>Video playback</td>
</tr>
<tr>
<td>Voice recording</td>
</tr>
<tr>
<td>Camera</td>
</tr>
<tr>
<td>Phone call</td>
</tr>
</tbody>
</table>

- Backlight alone often uses as much as 0.2W to 0.3W
- Phone is mostly off: leakage is already important!
- Bigger battery is not a good option
  - Adds bulk, cost, compromises consumer sex-appeal...

### Higher performance, higher power

![Graph showing power consumption vs. Dhrystone MIPS for ARM7, ARM9, and ARM11 processors in 0.18um and 0.13um processes.](image-url)
ARM Power and silicon budgets

- High performance is achieved at ~constant Si and power budgets
  - Enabled by process scaling
- Transistors are not free: significant impact on Si and design cost
  - Architectural consistency is important to avoid legacy constraints

<table>
<thead>
<tr>
<th>Core</th>
<th>ARM940T™</th>
<th>ARM926EJ-S™</th>
<th>ARM1136J-S™</th>
<th>Tiger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>ARMv4T</td>
<td>ARMv5TEJ</td>
<td>ARMv6</td>
<td>ARMv7</td>
</tr>
<tr>
<td>Cache</td>
<td>4K+4K</td>
<td>8K+8K</td>
<td>16K+16K</td>
<td>32K+32K</td>
</tr>
<tr>
<td>Process</td>
<td>0.18</td>
<td>0.13</td>
<td>0.09</td>
<td>0.065</td>
</tr>
<tr>
<td>Power (W)</td>
<td>0.25</td>
<td>0.25</td>
<td>0.3</td>
<td>~same</td>
</tr>
<tr>
<td>Size (mm²)</td>
<td>4.2</td>
<td>4.2</td>
<td>4</td>
<td>~same</td>
</tr>
</tbody>
</table>

Some representative notebook specs

- Ultra-portables
  - Dell Latitude X300: 1.2 GHz, 640 MB DDR SDRAM 266 MHz
  - IBM ThinkPad X41: 1.4 GHz, 256 MB DDR SDRAM 266 MHz
  - Fujitsu LifeBook P5010: 500 MHz, 512 MB DDR SDRAM 266 MHz
  - Weight: 2.9 lbs, 3.5 lbs, 4 lbs
  - Battery: 14.4V, 1500mA, 16.8V, 4400mAh, 10.8V, 4600mAh
  - Lifespan: 2.5 hrs, 3.5 hrs, 4.5 hrs

- Desktop replacements
  - Dell Inspiron X90: 1.4 GHz, 9700 128MB
  - Lenovo ThinkCentre 7500 60GB: 1.2 GHz, 2000 rpm
  - Weight: 9 lbs, 9.5 lbs
  - Battery: 16.8V, 4400mAh (95W), 14.8V, 6400mAh (95W)
  - Lifespan: 2 hrs, 2 hrs
Notebook power consumption

Backlight consumes between 0.5W and 3.5W depending on brightness
- Hard drive consumes 1W-2W
- Memory consumes between
- Processor can be a significant fraction of total power consumed
- Misc. system components account for around 50% of power
- Factor of 10-20 higher power consumption than in mobile phones

Backlilghts are power hungry!

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Voltage</th>
<th>Current</th>
<th>Power</th>
<th>Pwr %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>VDD</td>
<td>3.3 V</td>
<td>0.8 mA</td>
<td>2.64 mW</td>
<td>0.4 %</td>
</tr>
<tr>
<td>LCD Driving</td>
<td>V0 ~ V5</td>
<td>30.5 V</td>
<td>2.4 mA</td>
<td>70.8 mW</td>
<td>9.7 %</td>
</tr>
<tr>
<td>CFL Backlight</td>
<td>VL</td>
<td>270 V</td>
<td>2.5 mA</td>
<td>675 mW</td>
<td>89.9 %</td>
</tr>
</tbody>
</table>

- Power consumption of a 3.8" Kyocera TFT LCD
  - http://americas.kyocera.com/kicc/Lcd/notes/powerconsump.htm
  - The power budget of the LCD + backlight is about 0.75W!
System vs. processor power

- Marketing doesn’t really care whether a feature is power hungry or not...
- … spec to sell (e.g. bright backlight, small battery)
- … optimize where you can, not necessary where it would have the biggest pay off
- One area where we can / have to do something about power consumption is the processor

Overview

- Dynamic Voltage Scaling background
- Processor support for DVS
- A role for asynchronous architectures?
- Software control of processor speed
- Is there more to speed setting than DVS?
- An example: ARM IEM Test Chip
CMOS Power and Energy in a Nutshell

- Power and Energy consumption trends of a workload running at different frequency and voltage levels.
- DFS: frequency scaling only, DVS: frequency & voltage scaling

\[ f \sim \frac{(v_\text{dd}-v_t)^\alpha}{v_\text{dd}} \]
\[ \alpha \approx 1.3 \]
\[ v_t / v_{\text{max}} \approx 0.3 \]

\[ P = Cv_\text{dd}^2f + v_\text{dd}I_{\text{leak}} \]
Avg. power ~ heat

\[ E = \int P \text{d}t \]
Need DVS to save energy

Must reduce voltage to save energy and extend battery life!

Performance scaling for energy efficiency

- Reduced processing rate enables more efficient operation
  - Use dynamic voltage scaling (DVS) and threshold scaling (ABB)
Run-time performance scaling = BIG payoff

- Run-time performance scaling enables energy reduction
  - Dynamic Voltage Scaling
  - Threshold scaling (ABB) + DVS

- Can be exploited in future process generations
  - Voltage is the only parameter that affects all types of power consumption: dynamic, static (leakage), gate-oxide

- Done under many different names
  - AMD PowerNow
  - ARM’s Intelligent Energy Manager (IEM)
  - IBM Dynamic Power Manager (DPM)
  - Intel SpeedStep, Wireless SpeedStep
  - Transmeta LongRun, LongRun2

Key: determining how fast a workload needs to run!

---

Transmeta’s Argument

Figure 6: Power Management Comparison (Performance vs. Power Consumption)

- Simplified cooling (no fan) = cheaper systems
- Performance on demand = smaller battery is sufficient
IEM Demonstration

Performance

100%
83%
66%
50%

4 performance (frequency and voltage) levels available in benchmarked system

MPEG video

Closest available performance level of system

Performance level requested by algorithm

2 seconds

ARM Intelligent Energy Manage™

LongRun Power Management

Transmeta™ LongRun™ Power Management

In Action

Task Runs & Finishes

Example: LongRun Operating Points

Performance Mode

Frequency Adjusted to Necessary Level in System Application

Source: Crusoe™ LongRun™ Power Management White Paper
Intel Enhanced SpeedStep

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

---

Intel Wireless Speedstep

- Extends XScale power modes
- Includes Power Manager (PM) software
- Modes:
  - Standby
  - Voice Communications
  - Data Communications
  - Multimedia (Audio, Video and Camera)
  - Multimedia + Data Comms (Video Conferencing)
- Emphasis on distinguishing CPU-bound from memory-bound operation

---

**Pentium M 1.6 GHz**

<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>
Intel Wireless Speedstep...

- **Software components**
  - Policy Manager - determines V and F settings based on mode and measured data
  - Idle Profiler - provides workload data to Policy Manager
  - Performance Profiler - Uses Performance Monitoring Unit (PMU) to determine if workload is CPU or memory bound
  - User Settings - Allows mode and user preference settings
  - OS Mapping - Allows PM to work on various operating systems

Further software features

- **Program states are**
  - Running - all data is available i.e. low likelihood of data stalls
  - Waiting - app. is idling or waiting for IO response
  - Memory Bound - app. is moving large blocks of data
  - Mem. And CPU bound - app is running complex software
IBM DPM

- Dynamic Power Management for IBM PowerPC 405LP
- 0.18μm process
- 1.0V-1.8V operating voltage
- Two main operating modes
  - CPU/SDRAM
  - 266/133 MHz above 1.65V
  - 66/33 MHz above 0.9V
- Glitch-free frequency scaling
- \((V, F)\) change latency is 13μs to 95μs under Linux

IBM DPM

- DPM software is an operating system module for power management
- Implemented in Linux
- Policies define allowed operating points
- On context switch, DPM invokes policy (frequency and voltage settings) associated with that task
- Policies include
  - (IS) Run slow when idle
  - (LS) Minimise idle time based on previous interval utilisation
  - (AS) Using application-specific deadline information e.g. for MPEG4 decode, slow down if ahead of deadline, speed up if behind

**Average Power Consumption for MPEG4 (in mW)**

<table>
<thead>
<tr>
<th>Strategy</th>
<th>System</th>
<th>Base Card (BC)</th>
<th>BC Normalized to Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>2762.7</td>
<td>941.6</td>
<td>100%</td>
</tr>
<tr>
<td>IS</td>
<td>2630.4</td>
<td>774.9</td>
<td>85%</td>
</tr>
<tr>
<td>LS</td>
<td>2538.2</td>
<td>747.4</td>
<td>95%</td>
</tr>
<tr>
<td>AS</td>
<td>2463.6</td>
<td>702.5</td>
<td>72%</td>
</tr>
<tr>
<td>FS</td>
<td>2401.2</td>
<td>700.8</td>
<td>74%</td>
</tr>
</tbody>
</table>
Asynchronous = Low Power ? Handshake Solutions HT-80C51

- 8-bit microcontroller
- Capable of operating in synchronous or asynchronous mode
- Low operational power consumption
- Zero stand-by power
  - Assuming no leakage
- Immediate wake-up
- Very low electromagnetic emission (EME)

Photon emission images of a clocked (left) and Handshake Technology (right) 80C51 microcontroller executing the same program. The red dots indicate the level and distribution of power dissipation, which is clearly lower and more localized in the HT-80C51.

Synch vs. Asynch
**Synch vs. Asynch Power**

<table>
<thead>
<tr>
<th>Processor</th>
<th>Clocked</th>
<th>Speed</th>
<th>Dynam retiring</th>
<th>Memory</th>
<th>Voltage</th>
<th>E'/ins (pJ)</th>
<th>mW / MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM11 [32]</td>
<td>Yes</td>
<td>133</td>
<td>4-4K</td>
<td>3V</td>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Xeon [32]</td>
<td>Yes</td>
<td>200</td>
<td>6.4-16GB</td>
<td>1.6-1.66V</td>
<td>650-1025</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic Voltage Scaled</td>
<td>Yes</td>
<td>5-4V</td>
<td>52</td>
<td>16K</td>
<td>1.6-1.86V</td>
<td>540-5800</td>
<td></td>
</tr>
<tr>
<td>CoreSonic [16]</td>
<td>Yes</td>
<td>1.6V</td>
<td>62</td>
<td>22K</td>
<td>2.4V</td>
<td>720</td>
<td></td>
</tr>
<tr>
<td>Cortex-M3 [8]</td>
<td>Yes</td>
<td>1.8V</td>
<td>56</td>
<td>48</td>
<td>1.0V</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>Asyp [24] [12]</td>
<td>No</td>
<td>2.0-4.2V</td>
<td>10</td>
<td>0.6V</td>
<td>1.1-2.1V</td>
<td>1000-3000</td>
<td></td>
</tr>
<tr>
<td>SNAP/LE [14]</td>
<td>No</td>
<td>2-MM</td>
<td>10</td>
<td>32</td>
<td>0.6V</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>SNAP/LE [14] [15]</td>
<td>No</td>
<td>2-MM</td>
<td>10</td>
<td>32</td>
<td>0.6V</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

The following table is valid at 25°C for fully scan-testable cores manufactured in a typical 3.3 V, 0.18 μm CMOS process.

<table>
<thead>
<tr>
<th>Area (Giga)</th>
<th>Performance</th>
<th>Power consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMIPS</td>
<td>MHz</td>
<td>E / ins (pJ)</td>
</tr>
<tr>
<td>HT-90CH</td>
<td>0.60</td>
<td>20.3</td>
</tr>
<tr>
<td>Standard clocked [16]</td>
<td>0.90</td>
<td>120</td>
</tr>
</tbody>
</table>

* DMIPS: DMIPS (integer); Mhz: MHz; E / ins (pJ): E / instruction (pJ); mW / MHz: mW / MHz
* Equivalent performance accounting for clock cycles per instruction cycle
* Equivalent performance accounting for clock cycles per instruction cycle

Below is an example of a synchronous core for small size (60K gates) and power efficiency:

Asynchronous designs have not demonstrated intrinsic power advantages over synchronous processors...

**Au contraire!**

---

**Intelligent Energy Manager SW**

- **Automatically derive required performance level**
  - Automatic monitoring to avoid missing deadlines
  - Sets frequency and voltage accordingly

- **Implemented as kernel modules for Linux**
  - Only few kernel hooks are required
  - Autonomous from most of the kernel: portable

- **No application modifications required**
  - But application-level power hints may be provided
  - Works with interactive applications
A utilization trace

Each horizontal quantum is a millisecond, height corresponds to the utilization in that quantum.

IEM accuracy: episode classification

Interactive (Acrobat Reader), Producer (MP3 playback), and Consumer (esd sound daemon) episodes.
Comparison with LongRun

- Sony PictureBook PCG-C1VN
  - Transmeta Crusoe 5600 processor

<table>
<thead>
<tr>
<th>Frequency (Mhz)</th>
<th>Voltage (V)</th>
<th>Power reduction</th>
<th>Energy reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>1.3</td>
<td>67%</td>
<td>34%</td>
</tr>
<tr>
<td>400</td>
<td>1.35</td>
<td>53%</td>
<td>30%</td>
</tr>
<tr>
<td>500</td>
<td>1.4</td>
<td>36%</td>
<td>23%</td>
</tr>
<tr>
<td>600</td>
<td>1.6</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

- Crusoe’s built-in LongRun policy used for comparisons.
- Implemented in Linux 2.4.4-ac18 kernel

IEM vs. LongRun

- LongRun: part of the processor firmware.
  - Interval based algorithm (guided by busy vs. idle time).
  - Min. and max. range is controllable in software.

- IEM: implemented in OS kernel.
  - Multiple algorithms (perspectives / interactive).
  - Takes the quality of the user experience into account.

- Comparisons on following graphs.
  - Repeated runs of interactive benchmarks are close but not identical.
  - Transitions to sleep are usually not shown.
No user activity

Performance level

Time (s)

LongRun

100% = 600Mhz @ 1.6V
Frequency range of the TM5600 processor.

50% = 300Mhz @ 1.3V

Emacs

Performance level

Time (s)

LongRun

IEM

Performance level

Time (s)
Acrobat Reader

Frequent transitions to/from sleep mode. Longer durations without sleeping.
Plaympeg: Red’s Nightmare

- Playback quality identical in both cases.
  - No dropped frames.
- LongRun: doesn't slow down the processor enough.
  - No feedback about interactive performance, must be too conservative (<50ms to "speculate").

<table>
<thead>
<tr>
<th>Execution statistics</th>
<th>MPEG decode</th>
<th>Mean performance level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Length (sec)</td>
<td>Idle</td>
</tr>
<tr>
<td>LongRun (320x240)</td>
<td>49.14</td>
<td>48%</td>
</tr>
<tr>
<td>IEM</td>
<td>49.23</td>
<td>32%</td>
</tr>
</tbody>
</table>

Classical interval-based algorithms (e.g. LongRun) are too conservative – choose higher performance than necessary.
Combining Threshold (ABB) Scaling with DVS

- Bias voltage can be applied to body to change the threshold voltage
- For a given frequency find optimum vdd, vbs combination
- Graph shows this trade-off for projected 70nm technology

Energy used in an inverter chain

- Energy consumed through 10 inverters (theory vs. Spice = 12.7% error)
- DVS+ABB: 54% better than DVS alone, 74% better than DFS
**Energy use on real workloads - 180nm**

- Data based on 0.18um TSMC models
- Performance scaling: 100% to 50% in 16% steps
- DVS+ABB: average energy reduction of 23% over DVS

![Normalized Energy Consumed for Various Energy Scaling Techniques](image)

**Energy use on real workloads - 70nm**

- Data based on projected 70nm process
- Performance scaling: 100% to 10% in 5% steps
- DVS+ABB: average energy reduction of 48% over DVS

![Normalized Energy Consumed for Various Energy Scaling Techniques with 100% - 10% Frequency Scaling in 5% steps](image)
IEM926 on the bench

IEM Test Chip Evaluation Board

- Development board for IEM test chip to facilitate:
  - verification of SoC design
  - benchmarking of full system IEM performance
Technical Specification

- Dynamic Voltage Scaling methodology test vehicle
- ARM926EJ-S core with retention-voltage TC RAMs
- 4 dynamic performance levels supported in prototype
  - 240/180/120/60 MHz (+ 0 MHz stopped)
- Pseudo-synchronous clock domains
  - Re-timed using latches (rather than fully asynchronous)
  - Interfaces synchronized to AMBA HCLK
- Linux OS base porting peripheral set
- Prototype IEC with DVS emulation control mode
- Functional Adaptive Voltage Scaling demonstrator
  - On-chip prototype PowerWise serial PSU interface
  - Off-chip FPGA control loop implementation

Core Voltage domains

- Dynamically scale voltage to both CPU and RAMs
  - But support state save to RAM and power-down of CPU
- Level-shifter cells interface to always-powered SOC logic
  - Clamps hold signals low when domain voltage “unsafe”
Adaptive DVS support

- Hardware performance monitor on CPU domain
  - Allow target clock frequency to determine voltage ‘headroom’
  - Support closed-loop power supply control
    - Plus standard open-loop DVS

Clock latency issues

- Individual System, CPU and RAM power domains
  - Level-shifters provide between SOC an CPU sub-system
  - CPU/RAM scaled together, or CPU off with RAM retained
  - IEM-ready cores will provide asynchronous bus interfaces
IEM test chip power domains

IEM926 testchip
IEM926 - more details

- ARM926EJ-S core
- Multiple power domains
- Voltage and frequency scaling of CPU, caches and TCMs
- First full DVS silicon with National Semiconductor PowerWise™ technology
- NSC Adaptive Power Controller (APC) implemented in FPGA
- Includes DVS emulation mode for comparative tests

- TSMC 0.13um - CL013G - April Cyber Shuttle
  - Packaged parts – 11 August 2003
- Developed by ARM, Synopsys and National Semiconductor using Synopsys EDA tools

Silicon Evaluation
IEM926 : Voltage Scaling Analysis

- Min voltage (room temp)
- Cached workload (Dhrystone)
- PLL settings:
  - 300MHz
  - 288MHz
  - 276MHz
  - 264MHz
  - 252MHz
  - **240MHz**
  - 228MHz
  - 216MHz

IEM926 : Power Analysis

- Measured V/I (room temp)
- Cached workload (Dhrystone)
- PLL settings:
  - 300MHz
  - 288MHz
  - 276MHz
  - 264MHz
  - 252MHz
  - **240MHz**
  - 228MHz
  - 216MHz
IEM926 : Energy Analysis

- Normalized to 1.2V nominal (room temp)
- PLL settings:
  - 240MHz
- DFS only:
  - 1.2V nominal
  - No energy savings
- DVFS:
  - Limiting voltage
  - Effect of +5, 10, 15, 20% V margins

Questions?!
Circuit and Microarchitectural Techniques Reducing On-Chip Cache Leakage Power

Nam Sung Kim
Microprocessor Research, Intel Labs.
Intel Corp.

Outlines

- Technology and on-chip cache leakage trends
- Leakage reduction circuit techniques
- Microarchitectural techniques for cache leakage power reduction
- Leakage optimization of multi-Level on-chip caches using multi-$V_{TH}$ assignment
- Q & A
Technology and On-Chip Cache Leakage Trends

Dynamic and Leakage Power Trends

*ITRS 2002 projections with doubling # of transistors every two years*
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 $\exp\left(\frac{1}{V_{TH}}\right) \times \#\text{ of bits}$
  - 1MB L2 cache leakage power – 87% in 70nm tech
Leakage Reduction Circuit Techniques

Two leakage paths via off-state devices
- In storage cell – cell leakage
- Connected to WL – bit-line leakage

6-Transistor SRAM Leakage Model
6-Transistor SRAM Leakage Model

- Off-state leakage current of inverter
  \[ I_{\text{off}} = I_{S0} \cdot e^{\frac{V_{DS}}{kT/q}} \left( 1 - e^{-\frac{V_{TH}}{kT/q}} \right) (1 + \lambda V_{DS}) \]

- Cell leakage current
  - Sum of two off-state PMOS / NMOS current
  \[ I_{Lkg} = (I_{SN} + I_{SP}) + (I_{SN} \lambda_N + I_{SP} \lambda_P) (1 - e^{-\frac{V_{TH}}{kT/q}}) \]

Increasing \( V_{TH} \) or voltage scaling reduces leakage super-linearly!

MTCMOS Principles

- Active mode
  - Low-\( V_{TH} \) operation

- Stand-by mode
  - Disconnect power supply through high-\( V_{TH} \) devices

- Sleep devices
  - Gate-drive decrease
  - Body effect increase \( V_{TH} \)
  - Ground bounce
VTCMOS Principles

- Adjusting $V_{TH}$ by varying body voltage $V_{sb}$
  - $V_{TH} = V_{TH0} + \gamma (\sqrt{\Phi_S} - V_{SB} - \sqrt{\Phi_S})$
  - Reverse-body biasing
    - increasing $V_{TH}$ of low-$V_{TH}$ transistors
  - Forward-body biasing
    - decreasing $V_{TH}$ of high-$V_{TH}$ transistors

- Body voltage control
  - Requiring a **triple-well** process
  - Decreasing body factor ($\gamma$) w/ tech scaling
  - Slow wake-up latency
Adaptive Body Bias VTCMOS SRAM

Dual-$V_{TH}$ CMOS Principles

- **Using**
  - Low- / high-$V_{TH}$ for critical / non-critical paths

- **Reducing both** active **and stand-by leakage power**

- **Leakage reduction**
  - More effective than VTCMOS
    - decreasing body factor ($\gamma$) w/ tech scaling
  - For $S = 85$mV/decade
    - reducing leakage by $\times 10$ for each $85$mV $V_{TH}$ increase
Dual-$V_{\text{TH}}$ CMOS SRAM

- Using
  - Low-$V_{\text{TH}}$ for peripheral circuit (e.g., decoders)
  - High-$V_{\text{TH}}$ for memory cells
- Unavoidable to use of high-$V_{\text{TH}}$ in critical path of memory cell

Gated-$V_{\text{DD}}$ CMOS SRAM

- MTCMOS variant
  - Using high-$V_{\text{TH}}$ device
  - Destroying states
  - $\times10$ leakage reduction
  - Access time impact
- Forced stacking variant
  - Using low-$V_{\text{TH}}$ device
  - Preserving state
  - 40% leakage reduction
  - Floated $V_{\text{SS}}$ – noise issue
**DVS CMOS SRAM**

- **Voltage Scaling**
  - Using $V_{DD}$ control devices
  - Preserving states
  - \(\times7\sim8\) leakage reduction
  - Fast wake-up
  - No access time impact
  - Stability and soft-error issues during sleep time

---

**Leakage Saving via Voltage Scaling**

![Graph showing leakage saving via voltage scaling](image)

- 96% Reduction w/o BL leakage
- 80% Reduction w/ BL leakage
- 96% Reduction
Minimum State-Preserving Voltage

6T-full cell

~95mV

Wake-up Latency and Energy

1.48% more area for 64×Lmin per 128-bit line
Soft Error Susceptibility

$Q_{\text{crit}}$ decreases linearly w/ voltage scaling

$\text{SER} \propto N_{\text{flux}} \times CS \times e^{\frac{Q_{\text{critical}}}{Q_s}}$

Leakage reduced super-linearly

Summary

Low-Leakage SRAM Ckt Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Active Leakage</th>
<th>Stand-by Leakage</th>
<th>Access Time</th>
<th>Wake-up Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTCMOS</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>VTCMOS</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>State preserving</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gated-$V_{\text{DD}}$</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>DVS</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Dual-$V_{\text{TH}}$</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Microarchitectural Techniques for Cache Leakage Power Reduction

Microarchitectural Techniques

- Incorporating w/ low-leakage ckt techniques
  - Gated-$V_{DD}$, VTCMOS, MTCMOS, DVS, etc.

- Basic microarchitectural controls
  - Exploiting generational cache access patterns
  - Switching cache line power-mode based on run-time decision from the access patterns
Data Cache Working Set Analysis

Inst Cache Working Set Analysis
Gated $V_{DD}$-Based Techniques

- **“Cache Decay”** – ISCA 2001
  - Turn-off unused data cache lines using gated-$V_{DD}$
    - unless accessed for a fixed interval
      - requiring 2-bit counter per line and 1 global counter

- **“DRI Cache”** – ISLPED 2000
  - Resize cache size using gated-$V_{DD}$ based on monitored miss statistics for a fix interval

---

Gated $V_{DD}$-Based Techniques

- **Pros**
  - reducing $\times10$ leakage power for cache lines in stand-by mode
  - reducing some active-mode leakage power due to stacking effects

- **Cons**
  - requiring sophisticated prediction techniques to minimize the penalties incurred by accessing wrongfully turned-off cache lines
  - causing excessive additional dynamic power / cycles (in bigger L2 caches) for inappropriate sleep intervals
DVS-Based Techniques

- “Drowsy Caches” – ISCA 2002
  - Put all cache lines into state-preserving sleep state using DVS and wake-up lines on-demand
    - requiring only 1 global counter
  - Pros
    - ~6× leakage power reduction w/ small performance loss
    - simple implementation w/ negligible access time impact
  - Cons.
    - complicate instr. scheduling for OOO processors when accessing sleeping cache lines

Dual $V_{TH}$-Based Techniques

- Asymmetric Dual-$V_{TH}$ Cache
  - Optimizing leakage power of SRAM cell for storing “0” using high-$V_{TH}$ devices in SRAM cells
    - exploiting highly biased memory bits to “0” in SPEC2K
    - requiring special sense-amplifier / slower access time
Gated Bit-line Precharge

[Diagram of a memory architecture with labels for tag arrays, data arrays, way-0, way-1, sbank-0, sbank-1, BL, WL, precharge, clock, gated clock signal, clock buffer, and 8k bytes / 256 lines.]
On-Demand Precharge (Instr Cache)

Source of Sub-Bank Transition
**Prediction-Based Technique**

![Diagram of prediction-based technique]

**GBP Accuracy vs. Run-Time Increase**

- **Configuration:** 32-KB, 2-way, and 8-sbanks
- **Bit-line leakage reduction:** 80%~
- **Run-time increase w/ 1K predictor:** 0.4%

![Graph showing GBP accuracy vs. run-time increase]
**Time-Based Gating Technique**

- **Gated-precharge – MICRO 2003**
  - Turn-off precharge devices of cache sub-banks unless accessed for a fixed time interval
    - accessing 20% of 64KB sub-banks
    - in ~100 cycle window

**On-Bank Fraction / Run-Time Increase**

![Graph showing on-bank fraction and run-time increase with various markers for different benchmarks (bzip2, crafty, gcc, parser, vortex, INT AVG)]
Summary

- Combined architectural & ckt techniques
  - Exploring temporal/spatial localities of L1 cache access patterns
  - Trade-off among leakage reduction, access time, power management complexity
    - more aggressive leakage power reduction requiring more sophisticated architectural controls and causing more performance/power penalties when prediction wrong
  - Reducing L1 cache leakage power by 6~10× w/ small avg. performance loss (~2%)

Leakage Optimization of Multi-Level On-Chip Caches using Multi-$V_{TH}$ Assignment
Cache Circuit Model

- Abus buffer w/ repeater
- Decoder
- Memory cell
- Sense-amp w/ I/O circuits
- Dbus buffer w/ repeater
- VTH1
- VTH2
- VTH3
- VTH4
- Bit-line pair
- Word-line
- Cache sub-bank organization

- 70nm Berkeley predictive technology model
- Interconnect R/C annotated
- Repeaters used to minimize interconnect delay

Leakage Optimization via Multi-VTH’s

- Future nanoscale CMOS technology
  - providing 2 or more $V_{TH}$’s for leakage / speed optimization
- Questions w/ more VTH choices
  - assignment of multi-$V_{TH}$’s for caches
  - trade-off between leakage and speed of caches
  - cost-effective number of $V_{TH}$’s
  - optimal L2 cache size considering leakage and avg. mem. access time (AMAT) of processor memory system
**Cache Access Time Model**

- **Measure** circuit delay at VTH points using HSPICE
- **Approx.** circuit delay using curve fitting

\[ T_{\text{delay}} = \frac{k \cdot L \cdot V_{DD}}{(V_{DD} - V_{TH})} \]

\[ T_{\text{delay}}(V_{TH}) \approx B_0 + B \cdot e^{V_{TH}/b} \]

\[ T_{\text{delay}}(V_{TH_1}, V_{TH_2}, V_{TH_3}, V_{TH_4}) = B_0 + \sum_{i=1}^{4} B_i e^{V_{TH_i}/b} \]

**Cache Leakage Power Model**

- **Measure** leakage power at VTH points using HSPICE
- **Approx.** leakage power using curve fitting

\[ P_{\text{leakage}}(V_{TH}) = A_0 + A \cdot e^{-V_{TH}/a} \]

\[ P_{\text{leakage}}(V_{TH_1}, V_{TH_2}, V_{TH_3}, V_{TH_4}) = A_0 + \sum_{i=1}^{4} A_i e^{-V_{TH_i}/ai} \]
Single Cache Leakage optimization

- Leakage optimization

Objective:

\[
\min \left( P_{\text{leakage}}(V_{TH1}, V_{TH2}, V_{TH3}, V_{TH4}) \right) = A_0 + A_1 e^{V_{TH1}/a_1} + A_2 e^{V_{TH2}/a_2} + A_3 e^{V_{TH3}/a_3} + A_4 e^{V_{TH4}/a_4}
\]

Constraints:

\[
T_{\text{target delay}}(V_{TH1}, V_{TH2}, V_{TH3}, V_{TH4}) = B_0 + B_1 e^{V_{TH1}/b_1} + B_2 e^{V_{TH2}/b_2} + B_3 e^{V_{TH3}/b_3} + B_4 e^{V_{TH4}/b_4}
\]

\[
0.2 \leq V_{TH1}, \ldots, V_{TH4} \leq 0.5
\]

VTH Assignment Approaches

- 1 high VTH – traditional
- 1 high VTH – a variant
- 2 high VTH’s – VTH1 / VTH2
- 4 high VTH’s
VTH Assignment Approaches

- 1 high VTH – traditional
- 1 high VTH – a variant
- 2 high VTH's – VTH1 / VTH2
- 4 high VTH's

VTH Assignment Approaches

- 1 high VTH – traditional
- 1 high VTH – a variant
- 2 high VTH's – VTH1 / VTH2
- 4 high VTH's
**VTH Assignment Approaches**

- 1 high VTH – traditional
- 1 high VTH – a variant
- 2 high VTH's – VTH1 / VTH2
- 4 high VTH's

**Single Cache Leakage Optimization**

- More leakage reduction w/ more VTH
- 80% leakage reduction w/ 10% delay increase
- Peripheral circuits responsible ~10% leakage
Optimized V\text{TH} trends

- Memory cell array – most leakage reduction, least delay impact
- Decoders – most delay impact, least leakage reduction

Optimizing L2 leakage at fixed L1 size

- Constraint – maintaining the same AMAT
- Optimization – use larger but less leaky L2 caches
**Summary**

- **Cost-effective # of V<sub>TH</sub> for cache leakage reduction**
  - Depending on the target access time, but 1 or 2 extra high V<sub>TH</sub>'s is enough for leakage reduction
  - 80% leakage reduction w/ 10% access time increase

- **L2 Cache leakage**
  - Another design constraint in processor design
  - Trade-off among delay / area / leakage
  - Small overall performance impact w/ slower but less leaky L2 caches
  - Larger but slower L2 caches at a fixed performance
Q & A
Physical Basis of Variability in Modern ICs

Dennis Sylvester
University of Michigan

Some slides courtesy: Nagib Hakim (Intel), Kerry Bernstein (IBM), Andrew Kahng (UCSD), David Blaauw (UM)

Outline

- **Definitions (classes) of variability**
  - Intra vs. inter-die, systematic vs. random, impact of each, functional vs. parametric yield

- **Variability sources**
  - Critical dimensions (CD)
  - Vth fluctuations
  - Capacitive coupling
  - Environmental: Power supply noise, temperature, etc.

- **Single event upsets (soft errors)**
  - Definitions, trends, some simple techniques to combat

- **Goal:** Take you to the last section of the tutorial where Todd will describe robust design techniques to cope with all of this
Bringing Robustness Into The Picture

- 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 $V_{th}$

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.

**ITRS 2003**

- **CROSSCUTTING CHALLENGE 5—ERROR TOLERANCE**
  - "Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test."
  - "SEUs severely impact field-level product reliability" both for memory and logic beyond 90nm
  - "Automatic insertion of robustness into the design will become a priority" including redundant logic, adaptive and self-correcting or self-repairing circuits, etc.
Printing in the Subwavelength Regime

Variation: Across-Wafer Frequency

Figures courtesy Synopsys Inc.

Figure courtesy S. Nassif, IBM
DAC-2003 Nanometer Futures Panel:
Where should extra design automation R&D $ be spent?

- Variability/Litho/Mask/Fab
- Low Power/Leakage
- Power Delivery/Integrity
- Tool/Flow Enhancements/OA
- IP Reuse/Abstraction/SysLevel Design
- DSM Analysis
- P&R and Opt
- Others (Lotto)

Robustness vs. Low-Power

- Power is reduced by slowing non-critical paths (exploiting slack)
- When power reduction is highly effective (good), many paths become critical (bad)
  - Implies difficulty in timing verification and optimization
  - Parametric yield reduction

Fig source: A.B. Kahng
Robustness vs. Low-Power, 2

- $V_{dd}$ reduction yields quadratic dynamic power reductions + marked leakage improvement
- But: enhances susceptibility to single event upsets (SEUs) due to charge reduction

- Robust design practices include redundancy, widening devices/wires to limit variability
  - Larger total capacitance, power

Motivation

- Concurrent technology and design development.
  - Surprises are the norm
  - Issues are identified late
- Non-uniformity and uncertainty are having increased impact
  - Power
  - Performance
  - Reliability
  - Cost
- Possible solutions:
  - Process: e.g. performance/control tradeoff
  - Design: e.g. robustness/area (power) tradeoff
  - Modeling and CAD improvements: Shift from uncertainty to modeled non-uniformity.

Courtesy N. Hakim (Intel)
Sources of Uncertainty in Design

- **Design model**
  - Approximations
  - Estimation errors in model assumptions
  - Changing reqs, etc.

- **Operation**
  - Applied signals
  - Power supply voltage
  - On chip voltage
  - Self heating
  - Device degradation
  - etc.

- **Manufacturing and packaging**
  - Process change and drift
  - Systematic variation
  - Unassignable causes
  - etc.

Limiting Factors in Modeling Uncertainty

- **Concurrency between process and product development**
  - Many systematic effects cannot be modeled
    - Requires additional knowledge about the design/process.
  - Impact mitigated through design rules and other collateral

- **Sequential and iterative nature of design**
  - Limits available information for better modeling
    - E.g. placement, layout, etc.

- **Design Methodology and Tools:**
  - Design efficiency:
    - Mitigating uncertainty requires additional design efforts, or a change in methodology
  - Established practices evolve slowly
    - Requires tools, global perspective, added risk

- **Solution must attack problem at all 3 levels:**
  - More interactive process/product development
  - Top-down design approach
  - Tools and methodologies for a practical way to account for uncertainty in design
Types of Variation

- **Random**
  - Modeling consists of approximating the random effect by a normal distribution
  - Knowing mean and $\sigma$, use statistical approaches (Monte Carlo, worst-case) to account
  - Example: random dopant fluctuations which impact device $V_{th}$

- **Systematic**
  - This type of effect should be studied and modeled deterministically to allow for design with variation in mind
  - Includes environmental variations such as IR drop, thermal gradients, crosstalk noise-on-delay effects

More Categories of Variation

- **Inter-die (die-to-die, D2D)**
  - Across the wafer or between wafers
  - Larger length scale (~8 inch) gives rise to larger potential process-induced variation
  - Example: Thermal gradient in furnace leads to variation in $T_{ox}$ across the wafer

- **Intra-die (within-die, WID)**
  - Each device on the chip is affected differently
  - Length scale (typically mm), magnitude of variation is often smaller than inter-die
  - But impact of variation can be greater!
  - Example: Proximity effects where minimum pitch features exhibit different width bias than isolated features
Inter-die variation is not always larger than intra-die (ILD)

Uncertainty or Non-Uniformity

Random variations \rightarrow \text{Systematic effects} \\
\begin{array}{c}
\text{Modeled deterministically?} \\
\quad \begin{array}{c}
\text{Y} \\
\text{N}
\end{array}
\end{array} \\
\text{Uncertainty} \rightarrow \text{Non-uniformity}

Random effects \rightarrow \text{Systematic effect uncertainty} \rightarrow \text{Non-uniformity} \rightarrow \text{Random effects}

Modeling non-uniformities allows reducing the uncertainty interval

Courtesy N. Hakim (Intel)
Yield

- Functional
  - Chip doesn’t work
  - Short and open circuits in metal levels, pinholes in gate oxide
  - Electromigration failure (time-dependent)

- Parametric
  - Chips run at different speeds
  - Binning of parts, sell at different prices if possible
  - Crosstalk noise, ILD variation, Idsat variation (L_{eff}, T_{ox}, V_{th})

- Parametric yield loss has become dominant over defect-based yield loss as processing conditions improved

- We are concerned with parametric effects in this discussion

Outline

- Definitions (classes) of variability

- Variability sources
  - Critical dimensions (CD)
  - Vth fluctuations
  - Capacitive coupling
  - Environmental: Power supply noise, temperature, etc.

- Single event upsets (soft errors)
Main Sources of Process Variations

- **CD variation**
  - Systematic and random die-to-die and within-die sources

- **Width variation**
  - Impact on narrow transistors

- **Vth fluctuations**
  - Most impact on short, narrow devices

- **Interconnect**
  - Pattern density effects from polishing, dishing

---

Decomposition of CD Variation Patterns

[Images showing the decomposition of CD variation patterns]

Total CD Variation | Random component
Within-Die component | Within Wafer component

Courtesy N. Hakim (Intel)
Sources of CD Uncertainties

- **Die-to-die variation**
  - From wafer non-uniformity

- **Long range within-die variation**
  - Stepper non-uniformity, lens aberration, flare
  - Density non-uniformity

- **Short-range WID variation**
  - From patterning limitations, mask alignment, line edge roughness, etc.

Modeling Poly CD WID Variation

- **Long-range WID CD variation**
  - CD variation between two devices separated by a distance $d$ can be modeled by a spatial correlation function such as:
    \[ \text{Var}(\Delta CD_d) \sim 2\text{Var}(CD) \left(1 - \exp\left(-\frac{d}{dl}\right)\right) \]
  - Where Var(CD) is the total CD variance of a single device, and $dl$ is a characteristic distance for a particular technology.
  - Affects large circuits (> 1mm spread)

- **Short-range variation**
  - May have a deterministic component from proximity
  - Generally modeled as a random component.
  - Multi-fingered devices see statistical averaging of the random component for $I_{on}$, less clear for $I_{off}$
    \[ \text{Var}(CD_{mult}) = \text{Var}(CD_{sin gle}) \times \text{# legs} \]
  - Averages out quickly for several gates deep paths
  - Affects matched pairs, reference circuits, etc.
Sources of Width Variation

- Lithography sources:
  - Poly and diffusion rounding
  - Compounded by mask alignment

- Polishing:
  - Unequal polish of Si and STI material
  - Density dependent
  - Impacts both $I_{dsat}$ and $C_{gate}$

Impact and Mitigation of Width Variation

- Circuit impact
  - Width variation affects both $I_{dsat}$ / $R_{ds}$ and $C_{gate}$
  - Affects only narrow devices:
    - Analog circuits, SRAM, register files, standard cells,

- Mitigation by:
  - Guardbanding
  - Layout and density design rules
    - But may also unnecessarily impact large devices
  - Device matching design rules

 Courtesy N. Hakim (Intel)
Vth Variation Sources

- **Die-to-die**
  - From wafer level uniformity (Tox, Implantation, etc)
- **Random WID component (dominant)**
  - Random Channel Dopant Fluctuations $f(W, L)$
  - Random Poly Dopant Fluctuations
  - Random Fixed Oxide Charge
- **Strong device size dependency**

![Graph showing $\sigma(\Delta Vt)$ versus technology generation](image)

**Random Dopant Fluctuations, Intel’s View**

![Graph showing mean number of dopant atoms versus technology node](image)

Courtesy N. Hakim (Intel)
Discrete Dopant Effects

- Average doping well controlled but fluctuations occur (only ~100 dopants in channel in small scaled devices)
  - 45nm device with W/L of 5 has 3σ Vth ~ 33mV from this effect alone
- Other issues: undoped channels – if we can set Vth by modifying gate workfunction rather than through doping
- Fully depleted SOI has further trouble with Vth fluctuations since Vth is set by body thickness which is difficult to control very precisely

Vth Modeling: (Pelgrom, Stolk, …)

Stolk’s formulation:

$$\sigma_{Vt} = \left( \frac{4q\phi\varepsilon_{Si}}{3} \right)^{\frac{1}{2}} + \frac{kT}{q} \frac{1}{\sqrt{4q\phi\varepsilon_{Si}N}} \frac{\sqrt{N}}{W_{off}L_{off}}$$

$$\sigma_{Vt} \sim \frac{1}{\sqrt{W_{off}L_{off}}}$$
Impact and Mitigation

- Largest impact is on analog circuits, memories, bandgap references
  - Impacts ability to match devices
  - Cannot be reduced by layout design rules
- Impact on delay averages out for long paths

- Mitigation by device engineering
  - Graded wells
  - Tip engineering
- Mitigation by device upsizing
  - Impact on cell area

Interconnect-Induced Variations

Repeater circuit

Variation is systematic and depends on neighboring layout:
1. Layout, Proximity
2. Density

Sources:
- **Metal Thickness**: Etch (density), Polish (density, width)
- **Dielectric**: Etch (density)
- **Metal width/spacing**: Litho (proximity), etch
- **Vias**: Lithography, dielectric thickness

Courtesy N. Hakim (Intel)
**CMP & Area Fill**

Chemical-Mechanical Planarization (CMP)
Polishing pad wear, slurry composition, pad elasticity make this a very difficult process step

Pattern density effects result where dense and sparse regions have very different dielectric thicknesses

Coping with Pattern Density Effects

Area fill feature insertion
Decreases local density variation and hence decreases the ILD thickness variation after CMP

Figs: Mehrotra, Nakagawa
Crosstalk Noise Impact on Delay

Goes by many names:
- Dynamic delay, delay degradation/deterioration, noise-on-delay
- Impact of neighboring signal activity on switching delay
- Neighboring lines switch in opposite direction of victim line, delay increases

Miller Effect
- Both terminals of capacitor are switched in opposite directions (0 → Vdd, Vdd → 0)
  - Effective voltage is doubled, additional charge is needed (Q=CV) [simplified model]

Impact of neighboring signal activity

Intel 1GHz Coppermine – 50MHz drop in timing due to capacitive crosstalk effects

Ref: Intel, ISSCC00
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

Impact of Interconnect Variations

Impact on circuit delay depends on:
1. Driver / receiver sizing
2. Interconnect length: density uniformity

<table>
<thead>
<tr>
<th>Relative impact of interconnect and transistor variations</th>
<th>Impact expressed as % of total variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC length (% of max repeater length)</td>
<td>100%  66%  33%</td>
</tr>
<tr>
<td>IC%</td>
<td>98%  66%  14%</td>
</tr>
<tr>
<td>Xtr%</td>
<td>2%   34%  86%</td>
</tr>
</tbody>
</table>

→ Need to consider both device and interconnect variation
→ Need to simulate multiple segments to assess overall impact
→ IC variation dominates long lines, device dominate short ones

Courtesy N. Hakim (Intel)
Interconnect Reliability Scaling

- *New low-k materials have worse thermal properties than* $SiO_2$.
- Global wiring is more susceptible to thermal effects (self-heating) due to larger separation from substrate.
- Polyimides yield ~30% lower allowable current density in 0.1 $\mu$m global wiring.
- Self-heating effects lead to worsened electromigration reliability (even in Cu) since the metal temperature is increased over the local ambient temperature.

<table>
<thead>
<tr>
<th>Metal</th>
<th>Oxide</th>
<th>HSQ $k$</th>
<th>Polyimide $k$</th>
<th>Oxide</th>
<th>HSQ $k$</th>
<th>Polyimide $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>M5</td>
<td>1.2</td>
<td>1.02</td>
<td>0.786</td>
<td>1.55</td>
<td>1.39</td>
<td>1.16</td>
</tr>
<tr>
<td>M6</td>
<td>1.17</td>
<td>0.994</td>
<td>0.765</td>
<td>1.52</td>
<td>1.37</td>
<td>1.14</td>
</tr>
<tr>
<td>M7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.19</td>
<td>1.01</td>
<td>0.775</td>
</tr>
<tr>
<td>M8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.17</td>
<td>0.980</td>
<td>0.766</td>
</tr>
</tbody>
</table>

Table gives max. allowable peak current density (MA/cm²)

Ref: Banerjee, DAC99

Power distribution challenges

- Power distribution requires low IR drop and $L \cdot di/dt$ noise across the die.
  - Supply currents and current transients get much worse with scaling.
- Pentium 3 power density distribution shown.
  - Hot spots require more aggressive power grid topologies.
  - Memory stays cool, integer execution units run hot.
  - Peak power density ~ 4-8X uniform density.

Ref: Pollack, Intel
Temperature Variation Effects

- Variation:
  - Placement / program dependent
  - Varies slowly across the die, with a gradient of possibly several degrees per mm
  - Some correlation with IR drop in power grid

- Variation Effects:
  - Device on-current; speed
  - Interconnect resistance
  - Leakage (exponential)
  - Strong impact on reliability (EM)
    - Impact doubles for each 5 degree increase

Supply Voltage Variation Effects

- Common mode (e.g. global droop):
  - E.g. from large-scale L di/dt
  - Path delay mis-tracking
    - Interconnect-dominated paths vary less than gate-dominated
    - High Vt and Low Vt may have different dependencies.

- Differential mode (transient gradient):
  - E.g. from localized IR drop
  - Spatial separation of paths
    - Point of divergence analysis (skew)
  - Transient effects
    - Program specific
    - Point of divergence fails to capture (e.g. jitter)
Environmental Variation: Supply Voltage

\[ T_d \propto \frac{C_L L^{0.5} T_{\text{on}}^{0.5}}{V_{dd}^{0.3} \left(0.9 - \frac{V_{th}}{V_{dd}}\right)^{1.3}} \left(\frac{1}{W_m} + \frac{2.2}{W_p}\right) \]

10% reduction in \( V_{dd} \) for \( V_{th}/V_{dd} = 0.25 \) yields 9.2% rise in delay
\( V_{th}/V_{dd} = 0.3 \), rise = 18.4%

Power Supply Noise

- Parasitic resistance and inductance in power network and package cause supply voltage to switching devices deviate from clean supply voltage
  - IR-drop
  - \( \frac{L}{di/dt} \)
- Excessive supply variation affects
  - Signal integrity
  - Performance
  - Reliability

Off-chip Package/Pad Interconnect + Devices
Clean Supply

I
Erosion of Noise Margin

- Noise margin is at a premium in today's designs
- Low Voltage Low Power circuits
  - Supply voltage < 1V
  - Threshold voltage lower (to recover performance)

⇒ A margin that is safe under normal operating conditions may be inadequate during transient conditions

Induced Noise

- Functional failure
  - Power rail fluctuation appears as noise at the output of a gate and is propagated further
  - Combined with other noise conditions, could result in functional failure.

⇒ Long signal lines are particularly vulnerable
Clock Jitter

[Larsson, CICC 99] [Hussain, et. al. CICC 99]

- Finite power supply rejection (PSR) of VCO
  - Cycle-to-cycle jitter in clock
  - Jitter accumulation over sustained supply noise

- Eg. Early arrival of 2nd clock edge leads to incorrect evaluation of logic

Performance Degradation

- Circuit design assumes a budgeted supply voltage variation (5 - 10%).
- When voltage drop exceeds this limit, speed of the circuit is affected.
  - Performance guarantee not met
  - Delay failures

\[
\text{Path Delay} \propto \frac{1}{(V_{dd} - V_{ss} - V_f) t_d}
\]
Situation getting any better?

- Peak power dissipation ↑
- Supply voltage ↓
- Power transition rate ↑

\[ \text{Supply voltage} \]

\[ \text{Peak power dissipation} \]

\[ \text{Power transition rate} \]

Example:

- 10W
- 2.5V
- 200MHz
- Chip

- 100W
- 1.5V
- 2GHz
- Chip

Total

170x increase in \( \text{di/dt} \)

- Need help from
  - Power grid designer
  - Package designer
  - Architect

IR Drop Simple Model

Grid structure yields low IR drops but wirebonding constrains power to be supplied from chip periphery.

Middle of die sees large IR drops due to \( D_c/2 \) maximum wirelength.

Top layer voltage drop is given by:

\[ V_{\text{top}} = I_{\text{top}} R_{\text{top}} = J_{\text{avg}} \frac{D_c}{2} \frac{P_{\text{top}}}{c} \frac{R_{\text{int}}}{c} \]

\[ = \frac{I_{\text{chip}}}{8} P_{\text{top}} R_{\text{int}} \]

With flip-chip, worst-case resistive path drops from \( D_c/2 \) to \( P_{\text{bump}} \) (bump pad pitch, ~ 200 um)

\[ V_{\text{top}} = I_{\text{top}} R_{\text{top}} = J_{\text{avg}} 2 P_{\text{bump}} R_{\text{int}} \]

\[ = J_{\text{avg}} P_{\text{bump}}^2 R_{\text{int}} \]

Compared to IBM S/390 (flip-chip), expression (max) = 32 mV, experiment (avg) = 23 mV.
AC power supply noise, 1

- L*di/dt noise has traditional scaling properties (perimeter wirebonding) of:
  \[
  \frac{(L*di/dt)}{V_{dd}} \sim S^2 S_c
  \]

- S = 1.4, S_c = 1.06 (given 20%/4 years, 2.5yr generations)
- Fully exploiting pad arrays reduces this to just S though
  - Inductance limited by use of many parallel bumps
- How do we get around this S factor?
  - Continue to increase decoupling capacitance
    - At same rate as on-chip switched capacitance \(\rightarrow \frac{(L*di/dt)}{V_{dd}} \text{ flat}\)
    - Traditionally, \(C_{\text{decoup}} \sim 10 \times C_{\text{switching}}\) : high-k gate dielectrics may help
- This requires the package resonant frequency to become larger than clock frequency
  - Potential noise accumulation when devices switch at resonant frequency
  - Add resistance in series with a very large (likely off-chip) damping capacitance to eliminate resonances

Ref: Larsson, CICC99

AC power supply noise, 2

- di/dt scaling in previous slide may actually be worse
  - Exacerbated by sleep modes which help power
- Differential or current-steering logic styles?
  - Internal logic and output buffers can both gain from this
  - One way to fight static power – use it

Ref: Viswanath
V\textsubscript{DD} and Temperature Mitigation strategies

- **Modeling:**
  - Long range correlated effect, or
  - Deterministic maps

- **Design mitigation:**
  - Power grid design
  - Dynamic voltage control
  - Functional unit block placement
  - Thermal solution

- **Variation accounting in tools:**
  - Worst-casing: use conservative process/voltage/temperature (PVT) conditions
  - Add statistical guardband for uncertainty

---

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)
Gate-length Biasing for Leakage Variability

- Reducing leakage due to Vth roll-off (well-known)

- Reduce leakage variability

Gate-length Biasing

- First proposed by Sirisantana et al.
  - Large biases used (20+%) \rightarrow significant speed penalty
- Better to use very small biases < layout grid resolution (Gupta et al.)
  - Little reduction in leakage beyond 10% bias while delay degrades linearly
  - Preserves pin compatibility: layout swappable
    \rightarrow Technique applicable as post-P&R step
  - No additional process steps
- Leakage reductions of up to 23% observed
  - But the main advantage is in tightening of distributions
Resulting Leakage Distributions

- Leakage distribution for the 13K cell benchmark (500 samples)
  - Unbiased circuit
  - Single biasing across all cells
  - Cell-level biasing (each cell unique)

Percentage Reduction in Leakage Spread

Major Manufacturing Problem Example: Intra-Chip $L_{\text{gate}}$ Variation

- ITRS: one of biggest challenges in lithography is $L_{\text{gate}}$ control
- Scaling worsens impact of lens aberration in litho process
  - Intra-chip $L_{\text{gate}}$ variability increased
- Need to study it
  - For modeling within CAD flow
  - For yield and performance improvement

Orshansky, ICCAD00
**Gate Classification by Local Layout Patterns**

- Spatial $L_{\text{gate}}$ variability depends on local layout patterns
  - Need to characterize different configurations separately

- Gates are classified by:
  - **A) Orientation**
    - (vertical vs. horizontal)
  - **B) Distance to neighbors**
    - (proximity effect)
  - **C) Left vs. right neighbor**
    - (comma effect)

**Spatial $L_{\text{gate}}$ Maps for Different Gate Categories**

- All spatial maps are statistically significant
- Mask-level gate $L_{\text{gate}}$ correction is feasible
Ring Oscillator Speed Gradient

- Delay of 151-stage NAND ring oscillator simulated
- 14% speed variation across chip
- Delay map consistent with $L_{\text{gate}}$ maps
- Chip timing properties depend on location within field
- *Shows ways to improve circuit performance*

Outline

- Definitions (classes) of variability
- Variability sources
- *Single event upsets (soft errors)*
  - Definitions, trends, some simple techniques to combat
Soft Errors

- 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

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

**Physical Solutions are Difficult**

- Shielding
  - No practical absorbent (e.g., approximately > 10 ft of concrete)
  - Alpha particles can be addressed with plastic coating techniques at package level (also removing lead from packaging helps)

- Technology solution: SOI
  - Partially-depleted SOI does better (IBM estimated: 5X) but not a scalable technology
  - Fully-depleted SOI (and dual-gate) will help significantly

- Radiation-hardened cells
  - 10X improvement possible with significant penalty in performance, area, cost
  - 2-4X improvement may be possible with less penalty

Some of these techniques will help alleviate the impact of soft errors, but not completely remove it

Source: S. Mukherjee, Intel
Soft Error Rate Trends, ITRS03

![Graph showing soft error rates for different technologies.](image)

Figure 14. SER/chip for SRAM/latches/logic

Reducing Soft Error Rates

- Several types of “masking”
  - Logical
  - Electrical
  - Temporal
- Logical: An error strikes a node X, causing a logical transition but downstream logic does not depend on the state of node X (similar to false path analysis)
- Electrical: Attenuation by downstream gates (e.g., very narrow voltage glitches will be filtered out by slow gates)
- Temporal: As errors are transient in nature, they must arrive at a latch or FF during a period of transparency so they can be captured and propagated
Some Design Techniques for Logic

- Redundancy
  - Ex: Majority voters work extremely well (dup/triplicate all latches)
  - Huge area penalties (2-3X) make this a last resort

- Intentionally increase node capacitances so $Q_{crit}$ rises
  - Obvious delay and power penalties
  - Capacitance can be increased in a number of ways
    1. Add weak latch structures at critical nodes (CCP: cross-coupled pairs)
    2. Re-allocate transistor width across stages or pull-up/pull-down networks
      - Possibly exploiting disparate state probabilities
    - 70% increase in mean time between failures using (1) and (2) above
      - With delay penalty <20% but power penalty of 80%
      - Need more work to reduce power penalty

SER-Focused EDA Tools Needed

- General idea:
  - Given a sized gate-level netlist
  - Determine nodes that are both vulnerable to soft errors (small $Q_{crit}$) but also have little masking of any kind
    - Complex cost function $\rightarrow$ depends on logic functionality, downstream gate sizing/topology, location along path (early vs. late)
  - Choose from a range of soft error rate reduction techniques
    - Sizing, Vth selection, CCP insertion, etc.
    - Based on sensitivity of critical path delay or total power
  - Apply and then update circuit timing, node sensitivities
- Some early work presented at DAC 2004 by Dey et al. (UCSD)
  - Much more to be done...
Low Power Robust Computing

Todd Austin  austin@umich.edu
Seokwoo Lee

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
Radiation Effecting Reliability

- **Primary effects**
  - Radiation dose
  - Single event

- **Total radiation dose affects long term device behavior and reliability**
  - Parasitic transistors, Leakage, Vt shift, Gate damage
  - Primary concerns in adverse environment (space shuttle)

- **Single event upsets two major sources in ground-level**
  - Radioactive decay in semi-conductor fabrication
  - Cosmic rays
    - Interaction with secondary particles from atmospheric molecules
    - Interaction with Boron dopant

---

Single Event Effects

- **SEE : particle radiation disturbed**
  - Single Event Upset (SEU) :
    - Disturbed storage element
  - Single Event Latch-up (SEL) :
    - Disturbance in PNPN structure, possibly leading to permanent damage
  - Single Event Transient (SET) :
    - Disturbance causes output from gate to change state temporarily

- **SEE may be source of Silent Data Corruption (SDC)**
  - SDC generates undetected Soft-errors
  - Error silently corrupts critical data
  - Most catastrophic case
Measuring Reliability: Soft Error Rate (FIT)

- **Failure In Time (FIT): Failures in 1 Billion hours**
  - 114 FIT means
    - 1 failure every 1000 years
    - It sounds good, but if 100,000 units is shipped in market, 1 end-user per week will experience a failure
- **Mean Time to Failure: 1 / FIT**

ITRS 2003

- **CROSSCUTTING CHALLENGE 5—ERROR TOLERANCE**
  1) Beyond 90 nm, single-event upsets (soft errors) severely impact field-level product reliability, not only for embedded memory, but for logic and latches as well. 2) Current methods for accelerated lifetime testing (burn-in) become infeasible as supply voltages decrease (resulting in exponentially longer burn-in times); even power demands of burn-in ovens become overwhelming. 3) Atomic-scale effects can demand new “soft” defect criteria, such as for non-catastrophic gate oxide breakdown. In general, automatic insertion of robustness into the design will become ......
Projected Trends in SER

![Graph showing projected trends in SER (Soft Error Rate) for different technology generations.](image)

Figure 14. SER/chip for SRAM/latches/logic

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)
Triple Modular Redundancy (von Neumann)

Voter assumed reliable!
⇒ voter small
⇒ coarse-grained

Fault Tolerance Technique (Overview)

- Circuit technique
  - SEU immune Latch
    - Tolerating transient pulses
  - Temporal redundancy
    - Temporal Sampling
    - Code word preservation

- Error coding
  - Redundant information

- Software technique
  - Compiler inserts redundant code & checks correctness
  - Checkpointing
Fault Tolerance Technique (Overview)

- Architectural techniques
  - Uni-processor
    - Pre-commit check
      - DIVA
      - Repeated Instruction Execution (RESEE, Dual Use)
  - Multiprocessor
    - TMR with voting
      - Forward Error Recovery (FER)
    - DMR with Lockstep
      - DMR: Error detection
      - Checkpointing: Backward Error Recovery (BER)

Circuit Techniques: Temporal Redundancy

(1) Temporal Sampling

- Assume a transient fault pulse will have short duration
- Three registers sampling with different delayed clocks and majority voting circuit provides fault tolerant output
Circuit Techniques: Temporal Redundancy

(1) Temporal Sampling: Optimization

- Triple redundancy is achieved through temporal sampling
- With appropriate $\Delta T$, can be immune to upset from double node strikes
- Immune to clock node transients

---

Circuit Techniques: Temporal Redundancy

(1) Temporal Sampling Design Tradeoffs

- Chip layout area penalties
  - Latch areas increase from $\sim 3x$ to $>5x$
- Operating frequency penalties
  - Setup time increases by twice the sampling $\Delta T$
- Evaluation
  - Introducing setup time penalty $2\Delta T$ may not be acceptable in high frequency architectures
  - Assume transient pulse will not amplify
  - Balance rise and fall time to prevent fault pulse spreading
Circuit Techniques: Temporal Redundancy  
(2) Code Word State Preservation: Concept

- Similar to temporal sampling, but only needs two types of signals
- The input encoded signal can pass through if they are valid, which means they are identical
  - Use concepts of CMOS transistor stack; passes only if inputs are same otherwise it preserves last logic value on the capacitance of output loading
- Put CWSP element only before registers; inputs are driven by original and either from duplicated logic block or its delayed version

![Figure 4. Duplicated circuit removal.](image)

Circuit Techniques: Temporal Redundancy  
(2) Code Word State Preservation: Design Tradeoffs

<table>
<thead>
<tr>
<th>Fault Tolerance Method</th>
<th>Area Overhead (%)</th>
<th>Perf. Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR</td>
<td>196</td>
<td>15</td>
</tr>
<tr>
<td>CWSP – Duplication</td>
<td>93</td>
<td>12</td>
</tr>
<tr>
<td>CWSP – Delay (δ = 0.45ns)</td>
<td>18</td>
<td>29</td>
</tr>
<tr>
<td>CWSP – Delay (δ = 0.15ns)</td>
<td>12</td>
<td>13</td>
</tr>
</tbody>
</table>

- Compared CWSP with two different delays (150ps/450ps)
  - Small delay version has lower overhead, but more vulnerable to fault
- Assumes transient pulse will not amplify
- Possibly sensitive to error pulse spreading effect
Circuit Techniques: Temporal Redundancy Evaluation

- **Advantage**
  - Provides fairly good logic SER protection with less area overhead compared to TMR
  - Easily applied to current systems with minimal change

- **Disadvantage**
  - Large delay introduced in circuit from temporal sampling
  - Won’t work for high frequency architectures
  - Razor a better solution for high frequency architectures

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
Coding

- Codes for storage devices and communication systems
  - Cyclic Codes
  - Checksum codes
- Codes for arithmetic
  - AN Codes
  - Residue codes
- Codes for control units (unidirectional errors)
  - m-out-of-n codes
  - Berger Codes

Cyclic Code

- Parity check code based on properties that a cyclic shift of the codeword generates a codeword
- Parity check code requires complex encoding, decoding circuits using arrays of EX-OR gates, AND gates, etc.
- Cyclic codes require much less hardware, in form of LFSR
- Cyclic codes are appropriate for sequential storage devices, e.g. tapes, disks, and data links
- An (n,k) cyclic code can detect single bit errors, and multiple adjacent bit errors affecting fewer than (n-k) bits, burst transient errors (typical in communication systems)
Arithmetic Code

- Parity codes are not preserved under addition, subtraction
- Efficient for checking arithmetic operations
- Used in STAR fault tolerant computer in space applications
- AN codes, Residue codes, Bi-residue codes

AN Code

- ‘A’ should not be a power of radix 2
  - Odd ‘A’ is best
    - Detects every single bit fault - such an error has a magnitude of 2
  - A=3 : least expensive AN-code enabling detection of all single bit errors
- Example: $0110_2 = 6_{10}$
  - Representation in the AN-code for A=3
    - $010010_2 = 18_{10}$
  - Fault in bit position 2 may give
    - $011010_2 = 26_{10}$
  - The error is detected easily
    - $26$ is not a multiple of $3$
Unidirectional Asymmetric Code

- Only 1 can be 0 or vice versa
- \( N(X,Y) \) number of crossovers from 1 to 0 in \( X \) to \( Y \)
  - \( X=1011, Y=0101, N(X,Y) = 2, N(Y,X) = 1 \)
- Hamming distance \( D(X,Y) = N(X,Y) + N(Y,X) \)
- Code C is capable of detecting all unidirectional errors if \( N(X,Y) > 0 \) for all \( X, Y \)
- Code C is capable of correcting \( t \)-symmetric errors and detecting multiple unidirectional errors iff it satisfies \( N(X,Y) > t \) for all \( X, Y \)
- ‘m out of n’ Code, Berger Code

Berger Code

- Let \( a_ka_{k-1}...a_1 \) be a given data word
  - Count number of zeros and append to data word
  - Detects all unidirectional errors
- Example: 1010100 100 (7 bit data, 3 bit code)
  - If error in data or check only, check won’t match
  - If error in both? Still the same
  - Errors in data bits increases # of zeros, but in code reduces count and vice-versa
- Berger code is the most optimal systematic code
  - For each data bit check bits must be separated -> \( \log(k+1) \)
Fault Tolerant Processors

REESE: A Method of Soft Error Detection in Microprocessors

Joel B. Nickel and Arun K. Somani
Dependable Computing & Networking Laboratory
Department of Electrical and Computer Engineering
Iowa State University
REdundant Execution using Spare Elements

- This approach is based on
  - Micro-architectural modification
  - Uses integrity checking in active-redundant stream, simultaneous multi-threading (AR-SMT) architecture
- Minimizes performance loss in AR-SMT
  - Two execution during a single cycle
  - Employs time redundancy and idle capacity
- Achieves low-cost fault tolerance
  - Small pipeline enhancement for error checking

REESE Pipeline

- R-stream queue
- Possible hardware enhancements:
  - Additional FUs
  - RUU/LSQ entries
  - Decode/Issue bandwidth
  - Memory ports
Analysis

- REESE causes 12-14% slowdown with no idle elements
- More hardware = Better REESE performance
- Memory ports are a critical factor, but not needed to meet the original goal
- ALUs are the essential idle elements

Fingerprinting: Bounding Soft-Error Detection Latency and Bandwidth
Jared C. Smolens, Brian T. Gold, Jangwoo Kim
Babak Falsafi, James C. Hoe, Andreas G. Nowatzyk

TRUSS
Computer Architecture Lab
Carnegie Mellon
http://www.ece.cmu.edu/~truss
DMR Error Detection

- Context: Dual-modular redundancy for computation
- Problem: Error detection across blades

Fingerprinting

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

Instruction stream

\[
\begin{align*}
R1 & \leftarrow R2 + R3 \\
R2 & \leftarrow M[10] \\
M[20] & \leftarrow R1
\end{align*}
\]

Stream of updates

\[
...001010101011010100101010...
\]

Fingerprint

\[= 0xC3C9\]
Recovery Model

- Rollback-recovery to last checkpoint upon detection

Full-state Comparison Bandwidth

- Differential comparison over interval

16-bit fingerprint < 150KB/s for 14K checkpoint intervals
- Full state bandwidth unreasonable for small checkpoint intervals
DIVA: Building Buggy Chips - That Work!

Chris Weaver (lead), Pat Cassleman,
Saugata Chatterjee (alum), Todd Austin,
Maher Mneimneh (FV), Fadi Aloul (FV),
Karem Sakallah (FV)

Advanced Computer Architecture Laboratory
University of Michigan

Dynamic Implementation Verification Architecture

- 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

Check Mode
Recovery Mode

How Can the Simple Checker Keep Up?

- **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**
How Can the Simple Checker Keep Up?

- 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

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
Transient Fault Detection via Simultaneous Multithreading

Steven K. Reinhardt
University of Michigan EECS
Shubhendu S. Mukherjee
Compaq Computer Corporation

Simultaneous Redundant Multithreadinging

Logical boundary of redundant execution within a system
- Trade-off between information, time, & space redundancy

Sphere of Replication

Thread 1
Input Replication
Thread 2
Output Comparison
Rest of System

Compare & validate output before sending it outside the SoR
Simultaneous & Redundantly Threaded Processor (SRT)

SRT = SMT + Fault Detection

- Sphere of replication
  - Output comparison of committed store instructions
  - Input replication via load value queue
- Less hardware compared to replicated microprocessors
  - SMT needs ~5% more hardware over uniprocessor
  - SRT adds very little hardware overhead to existing SMT
- Better performance than complete replication
  - Better use of resources
- Lower cost
  - Avoids complete replication
  - Market volume of SMT & SRT

Fault Tolerant Multiprocessor Platforms

SafetyNet
ReVive
End-to-end invariant checking
Outside of Processor

- Hardware faults in shared memory multiprocessors
  - Mostly transient, some permanent, not chipkill
  - Interconnection network
    - Example: dead switch
  - Cache coherence protocols
    - Example: lost coherence message
- Cost vs. Performance vs. Availability
  - Low Cost
    - Simple changes to a few key components
  - Low Performance Overhead
    - Handle frequent operations in hardware
  - High Availability
    - Fast recovery from a wide class of errors

Server System Hardware Design Space

- Existing systems get only 2 out of 3 features

[Diagram showing a triangle with High Availability at the top, Low Cost to the left, and High Performance to the right, with Backward Error Recovery (Tandem NonStop) and Forward Error Recovery (IBM mainframes) at the bottom.]
SafetyNet: Improving the Availability of Shared Memory Multiprocessors with Global Checkpoint/Recovery

Daniel J. Sorin, Milo M. K. Martin, Mark D. Hill, and David A. Wood

Computer Sciences Department
University of Wisconsin—Madison

SafetyNet Abstraction

Processor

Processor

Most Recently Validated Checkpoint

Active (Architectural) State of System

Recovery Point

Checkpoints Awaiting Validation
SafetyNet Checkpoint/Recovery

- SafetyNet: all-hardware scheme [ISCA 2002]
  - Periodically take logical checkpoint of multiprocessor
    - MP State: processor registers, caches, memory
  - Incrementally log changes to caches and memory
  - Consistent checkpointing performed in logical time
    - E.g., every 3000 broadcast cache coherence requests
  - Can tolerate >100,000 cycles of error detection latency

<table>
<thead>
<tr>
<th>CP 1</th>
<th>CP 2</th>
<th>CP 3</th>
<th>CP 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validated execution</td>
<td>Pending validation –</td>
<td>Still detecting errors</td>
<td>Active execution</td>
</tr>
</tbody>
</table>

Contribution of SafetyNet

- SafetyNet: global, consistent checkpointing
  - Low cost and high performance
  - Efficient logical time checkpoint coordination
  - Optimized checkpointing of state
  - Pipelined, in-background checkpoint validation

- Improved availability
  - Avoid crash in case of fault
  - Same fault-free performance
Overview of ReVive

- Entire main memory protected by distributed parity
  - Like RAID-5, but in memory
- Periodically establish a checkpoint
  - Main memory is the checkpoint state
  - Write-back dirty data from caches, save processor context
- Save overwritten data to enable restoring checkpoint
  - When program execution modifies memory for 1st time
Distributed N+1 Parity

- Allocation Granularity: page
- Update Granularity: cache line

Contribution of Revive

- **Low Cost**
  - HW changes only to directory controllers
  - Memory overhead only 12.5% (with 7+1 parity)
- **Low Performance Overhead**
  - Only 6% performance overhead on average
- **High Availability**
  - Recovery from: system-wide transients, loss of one node
  - Availability better than 99.999% (assuming 1 error/ day)
# High-Level Comparison Between ReVive and SafetyNet

<table>
<thead>
<tr>
<th></th>
<th>ReVive</th>
<th>SafetyNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward error recovery scheme</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fault model</td>
<td>Transient &amp; permanent</td>
<td>Transient &amp; some permanent</td>
</tr>
<tr>
<td>Processor modification</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Software modification</td>
<td>Minor</td>
<td>None</td>
</tr>
<tr>
<td>Fault-free performance</td>
<td>6-10% loss</td>
<td>No loss</td>
</tr>
<tr>
<td>Output commit latency</td>
<td>At least 100 milliseconds</td>
<td>No more than 0.4 milliseconds</td>
</tr>
</tbody>
</table>

---

## Dynamic Verification of End-to-End Multiprocessor Invariants

Daniel J. Sorin¹, Mark D. Hill², David A. Wood²  
¹Department of Electrical & Computer Engineering  
Duke University  
²Computer Sciences Department  
University of Wisconsin-Madison
Overview

- Goal: improve multiprocessor availability
- Recent work developed efficient checkpoint/recovery
  - But we can only recover from hardware errors we detect
  - Many hardware errors are hard to detect
- Proposal: Dynamic verification of invariants
  - Online checking of end-to-end system invariants
  - Checking performed with distributed signature analysis
  - Triggers recovery if invariant is violated
- Results
  - Detects previously undetectable hardware errors
  - Negligible performance overhead for error-free execution

Why Local Information Isn’t Sufficient

Neither P1 nor P2 can detect that an error has occurred!
Distributed Signature Analysis

- Reduces long history of events into small signature
  - Signatures map almost-uniquely to event histories

Event N at P1
  
  :  
  
  Event 2 at P1
  
  Event 1 at P1

Event N at P2
  
  :  
  
  Event 2 at P2
  
  Event 1 at P2

P1 Signature  P2 Signature

P1's signature  P2's signature

Checker

Check periodically in logical time (every 3000 requests)

Commercial Processors
Different Abstraction of Replication

Replicated lockstepped mirror processors
Replicated pipelines in same die

S/390 G5 CPU Fault Tolerant Approach

- Dual modular redundancy within microprocessor
  - Replicate and lockstepped pipelines (I and E-unit)
- Parity for cache data and data paths
- Error checking of control and ALU
- Dynamic CPU Recovery
  - R-unit
    - ECC-protected Register File; Checkpoint Array
    - Providing Backward Error Recovery (BER) by comparing results from replicated, lockstepped pipelines
- Dynamic CPU Sparing
  - Scan machine state information from failed CPU into spare CPU
  - System to be restored to full capacity in less than one second
S/390 G5 Memory System
Fault Tolerant Approach

- **L1**
  - write-through
  - Byte parity
  - Recover transient L1 failure by instruction retry
  - Recover permanent failure by deleting cache-line

- **L2**
  - Each L2 cache is shared by 6 microprocessors
  - Protected by SEC/DED ECC
  - Avoiding error from permanent fault by using cache-delete capability

- **Main memory**
  - Using SEC/DED ECC
  - Automatic on-line repair by using built-in spare chips

S/390 G5 I / O and Power
Fault Tolerant Approach

- **I/O Subsystem designed**
  - Redundant paths between all devices and main memory
  - Parallel Sysplex Provides Server-to-server Connection
  - 99.999% availability with two or more interconnected mainframe

- **Power supply**
  - Fully Redundancy
    - Battery
    - AC-to-DC Converters
    - DC-to-DC converters
    - Fan/Compressor assemblies
Fault Detection in Compaq Himalaya System

- Replicated Microprocessors + Cycle-by-Cycle Lockstepping

Tandem HP NonStop Servers

- Loosely coupled massively parallel computer
- Two replicated, lock-stepped MIPS R4400 RISC processors (mirroring) in each logical processor compare execution by external-chip comparison
- L2 Cache, main memory, and operating system are all independent
- Controlled by operating system
- 100% Design overhead
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' DSM/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 Soft Error Detection Latency and Bandwidth'
11. D. Sorin, et.al 'Dynamic Verification of End-to-End Multiprocessor Invariants'
Backup

Processor Core Fault Tolerance

- Adding redundancy into pipeline stages
- Observation
  - Modern microprocessor has support to recovery from exception / misprediction, before commit stage
  - Detect / recover from error by checking each instruction before in-order commitment
- Instruction re-execution (REESE, Dual Use)
  - Observation
    - Aggressive OoO. processor will not 100% utilize system resources
    - # of Committed instructions much less than # of fetched instruction on average
- Checker pipeline (DIVA)
  - Passing instruction to checker pipeline before commit stage
  - Complexity of checker pipeline is much less than that of main processor
    - Checker only deals with in-order retirement queue of the instruction from main pipeline
    - No need to deal with speculative instructions
Circuit Techniques: Circuit Techniques: (1) SEU Immune Latch

- The two extra inverters together with the normal gating transistors provide three independent delay stages for absorbing glitches.
- Glitches are absorbed whether generated internally, or whether coming in on the Data or clock (GB) lines, as long as the timing guidelines are followed. What is shown is a latch, which is 1/2 of the common D-flip-flop circuit.

Related Work: SafetyNet

- Types of recoverable errors
  - ReVive: Permanent (loss of a node)+Transient
  - SafetyNet: Transient; perm only w/ redundant devices
- HW modifications
  - ReVive: Directory controller only
  - SafetyNet: Memory, caches, coherence protocol
- Performance Overhead
  - 6% with ReVive, negligible with SafetyNet
Implementing Distributed Signature Analysis

- All components cooperate to perform checking
  - Component = cache controller or memory controller
- Each component contains:
  - Local signature register
  - Logic to compute signature updates
- System contains:
  - System controller that performs check function
- Use distributed signature analysis for dynamic verification
  - Verify end-to-end invariants

Two invariant checkers

- Message invariant
  - all nodes see same total order of broadcast cache coherence requests
  - Update: for each incoming broadcast, “add” Address
  - Check: error if all signatures aren’t equal
- Cache coherence invariant
  - All coherence upgrades cause downgrades
    - Upgrade: increase permissions to block (e.g., none \(\rightarrow\) read)
    - Downgrade: decrease permissions (e.g., write \(\rightarrow\) read)
  - Update: add Address for upgrade
    subtract Address for downgrade
  - Check: error if sum of all signatures doesn’t equal 0
Distributed Parity Update in HW

Family of Line X

Home of Line X

Home of parity for Line X