

Robust Low Power Computing in the Nanoscale Era

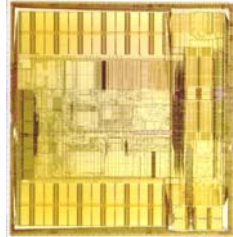


   **Todd Austin**  
University of Michigan
austin@umich.edu 

Thanks

- **Slide/concept contributions by:**
 - David Blaauw, University of Michigan
 - Kypros Constantinides, University of Michigan
 - Kris Flautner, ARM Ltd.
 - Nam Sung Kim, Intel Corporation
 - Trevor Mudge, University of Michigan
 - Leyla Nazhandali, Virginia Tech
 - Dennis Sylvester, University of Michigan
 - Chris Weaver, Intel Corporation

Evolution of a 90's High-End Processor



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

	Power (Watts)	Freq. (MHz)	Die Size (mm ²)	Vdd
Alpha 21064	30	200	234	3.3
Alpha 21164	50	300	299	3.3
Alpha 21264	72	667	302	2.0
Alpha 21364	100	1000	350	1.5

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’.”

Intel chairman Andrew Grove *Int. Electron Devices Meeting* keynote Dec. 2002

Total Power of CPUs in PCs

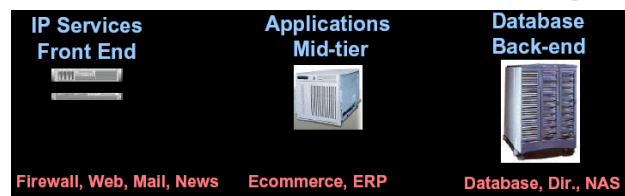
- Early '90's – 100M CPUs @ 1.8W = 180MW
- Early 21st – 500M CPUs @ 18W = 10,000MW
- Exponential growth
- Recent comment in a Financial Times article:
10% of US's energy use is for computers
 - exponentially growth implies it will overtake cars/homes/manufacturing
- NOT! – why we're here

What hasn't followed Moore's Law

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



Also Important For Server Systems



Internet Service Provider's Data Center

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

Why does robustness matter?

- ... the ability to consistently resolve critical dimensions of 30nm is severely compromised creating substantial uncertainty in device performance. ... at 30nm design will enter an era of “probabilistic computing,” with the behavior of logic gates no longer deterministic...
- susceptibility to single event upsets from radiation particle strikes will grow due to supply voltage scaling while power supply integrity (IR drop, inductive noise, electromigration failure) will be exacerbated by rapidly increasing current demand
- new approaches to robust and low power design will be crucial to the successful continuation of process scaling ...

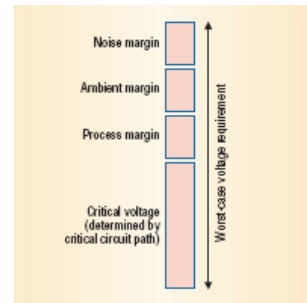
Intel chairman Andrew Grove *Int. Electron Devices Meeting* keynote
Dec. 2002

Why does robustness matter?

- **Grove’s comments**
 - SEUs
 - IR drop
 - inductive noise
 - Electromigration, etc.
- **Increase in variability as feature sizes decrease**
- **Likely to be the next major challenge**
 - strengthen interest in fault-tolerance
 - renew interest in self-healing

How are they related?

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



Challenges

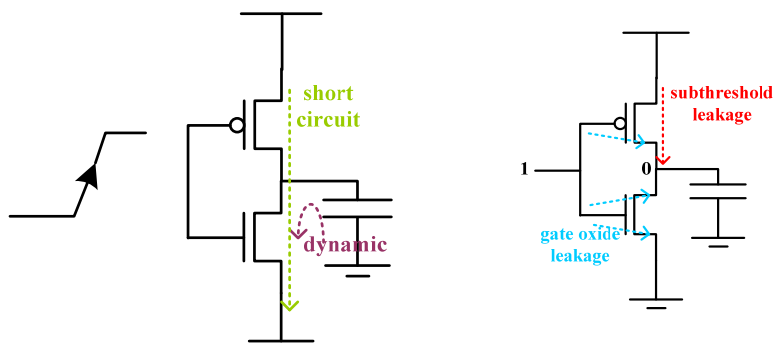
- Power density is growing
- Systems are becoming less robust
- Can architecture help?
 - Lower power organizations – quick estimates of power
 - Robust organizations – quick estimates of robustness
- By one account we need a 2x reduction in power/generation from architecture
- Question where will the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

Tutorial Schedule

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

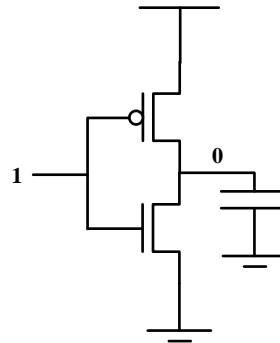
Power Sources

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



Dynamic Power Consumption

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



Dynamic Power Consumption

- Input 1 → 0
- Energy drawn from power supply:

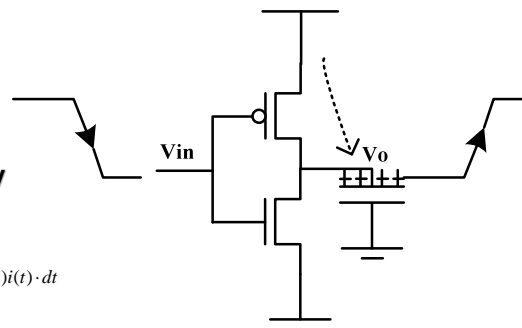
$$\begin{aligned}
 E_{supply} &= \int P(t) \cdot dt \\
 &= \int V_{dd} i(t) \cdot dt \\
 &= \int_0^{V_{dd}} V_{dd} C \cdot dV_o \\
 &= CV_{dd}^2
 \end{aligned}$$

- Energy consumed by PMOS:

$$\begin{aligned}
 E_{PMOS} &= \int P(t) \cdot dt \\
 &= \int (V_{dd} - V_o) i(t) \cdot dt \\
 &= \frac{1}{2} CV_{dd}^2
 \end{aligned}$$

- Power is

$$P = f \cdot E_{PMOS} = \frac{1}{2} f CV_{dd}^2$$



Dynamic Power Consumption

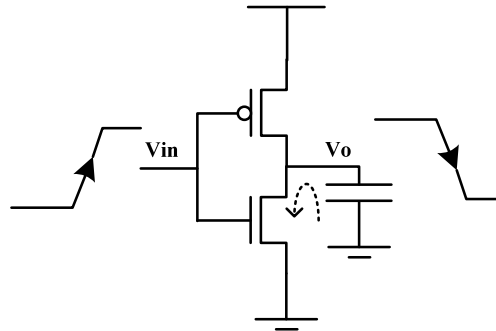
- **Input 0→1**

- Energy drawn from supply: 0
- Energy consumed by NMOS equals to the energy stored on the capacitance:

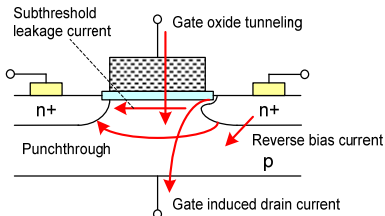
$$E_{NMOS} = \int V_{Gi}(t) \cdot dt = \frac{1}{2} CV_{dd}^2$$

- Power is

$$P = f \cdot E_{NMOS} = \frac{1}{2} f CV_{dd}^2$$



Leakage Current Components



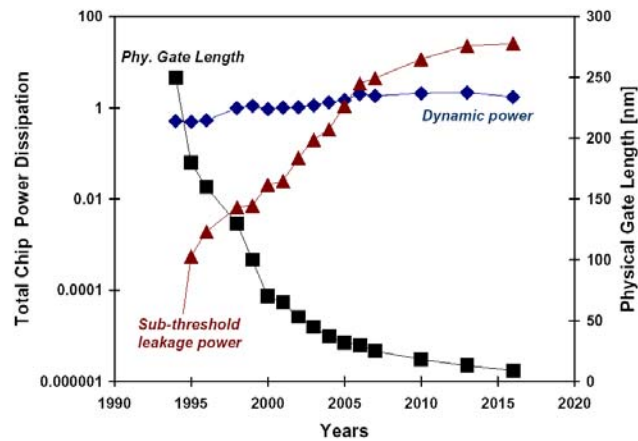
- **Subthreshold leakage (I_{sub})**

- Dominant when device is OFF
- Enhanced by reduced V_t due to process scaling

- **Gate tunneling leakage (I_{gate})**

- Due to aggressive scaling of the gate oxide layer thickness (T_{ox})
- A super exponential function of T_{ox}
- Comparable to I_{sub} at 90nm technology

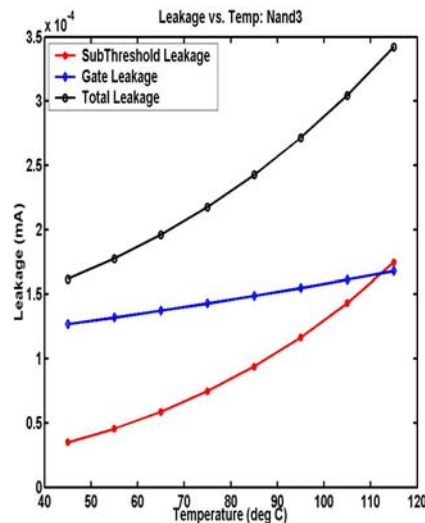
Dynamic and Leakage Power Trends



ITRS 2002 projections with doubling # of transistors every two years

Temperature Dependence

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



Source: R. Rao

Tutorial Schedule

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

How to Reduce Dynamic Power

- More generally

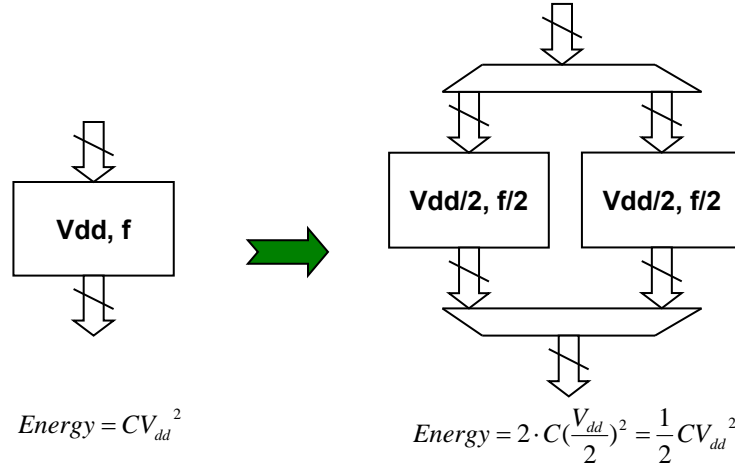
$$P_{dyn} = \frac{1}{2} \alpha f C V_{dd}^2$$

where α is switching activity

- To reduce dynamic power, we can reduce

α	– clock gating
C	– sizing down
f	– lower frequency
V_{dd}	– lower voltage

Dynamic Power Reduction - Parallel Computation



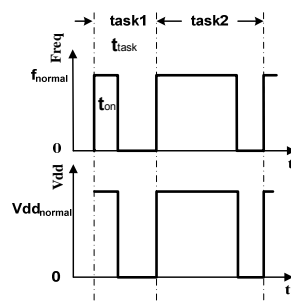
- Energy reduced by 50%, but double the area and more leakage

Dynamic Power Reduction - DVS

- Given dynamic workload – scale frequency or voltage

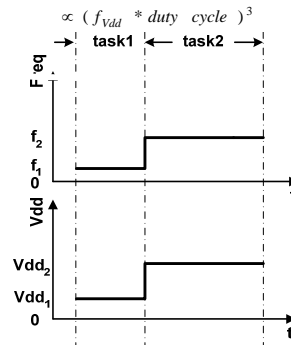
- Clock/power gating – linear energy saving with duty cycle

$$\begin{aligned}
 Energy &= P_{Vdd} * t_{on} \\
 &= P_{Vdd} * t_{task} * (duty\ cycle)
 \end{aligned}$$



- Just-in-time Dynamic Voltage Scaling (DVS) – cubic energy saving with duty cycle

$$\begin{aligned}
 Energy &= P_{Vscaled} * t_{task} \\
 &= (f_{scaled} C_s V_{scaled}^2) * t_{task} \\
 &\propto f_{scaled}^3 \\
 &\propto (f_{Vdd} * duty\ cycle)^3
 \end{aligned}$$



How Far Can We Scale Down the Voltage?

- **Traditional DVS (Dynamic Voltage Scaling)**

- Scaling rang limited to less than $V_{dd}/2$

	Voltage Range	Frequency Range
IBM PowerPC 405LP	1.0V-1.8V	153M-333M
Transmeta Crusoe TM5800	0.8V-1.3V	300M-1G
Intel XScale 80200	0.95V-1.55V	333M-733M

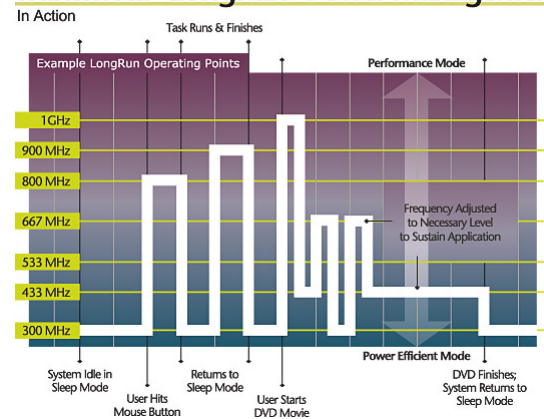
- **Minimum functional voltage**

- For an CMOS inverter is [Meindl, JSSC 2000]:

$$V_{dd,limit} = 2V_T \ln\left(1 + \frac{S_S}{\ln 10 \cdot V_T}\right) \sim 48\text{mV for a typical } 0.18\mu\text{m technology}$$

LongRun Power Management [Transmeta]

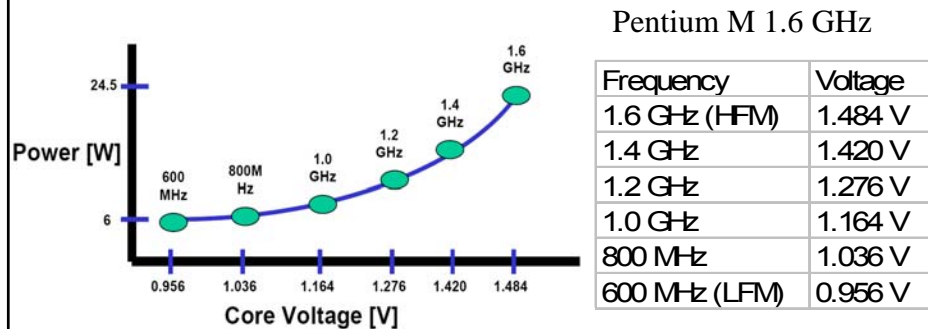
Transmeta™ LongRun™ Power Management



Source: Crusoe™ LongRun™ Power Management White Paper

SpeedStep Technology [Intel]

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



Reducing Static Power with Dual V_t Assignments

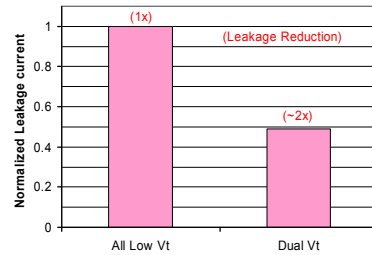
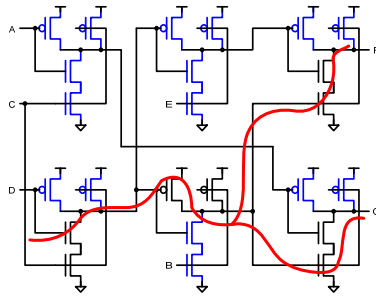
- Transistor is assigned either a high or low V_t
 - Low- V_t transistor has reduced delay and increased leakage

	Low- V_t ; 0.9V	High- V_t ; 0.9V	Low- V_t ; 1.8V	High- V_t ; 1.8V
Leakage (norm)	1	0.06	1	0.07
Delay (norm)	1	1.30	1	1.20

- Trade-off degrades for lower supply voltage

Dual V_t Example

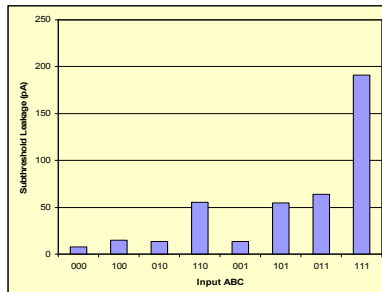
- Dual V_t assignment approach
 - Transistor on critical path: low V_t
 - Non-critical transistor: high V_t



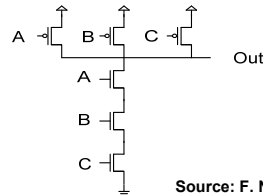
State Dependence (I_{sub})

- Simulation results of a 0.13 μ m process

Three OFF transistors in stack
One OFF transistor in stack
8X increase in leakage



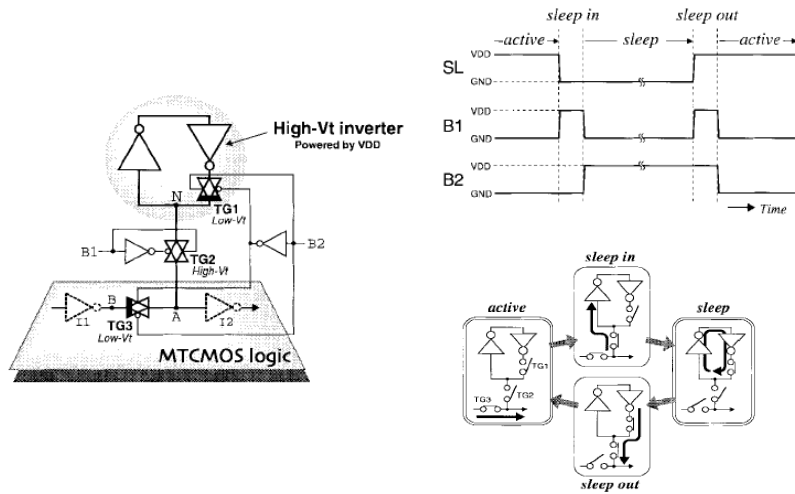
Input ABC	Output	Subthreshold Leakage (pA)
000	1	8.0836
100	1	15.1873
010	1	13.5167
110	1	55.2532
001	1	13.4401
101	1	54.5532
011	1	64.259
111	0	191.2692



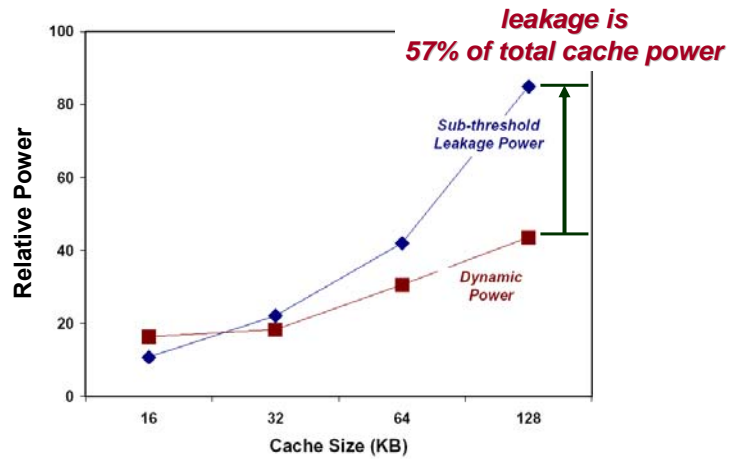
Source: F. Najm

- Approach: rework FSM state assignment or logic

Balloon Latch [Shigematsu]



On-Chip Cache Leakage Power



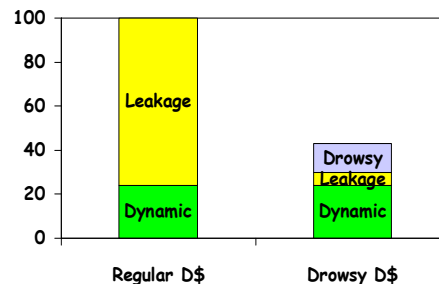
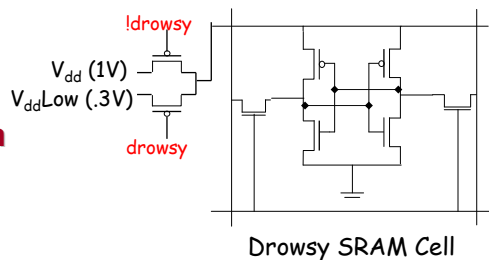
Caches design with 70-nm BPTM and sub-banking technique

On-Chip Cache Leakage Power

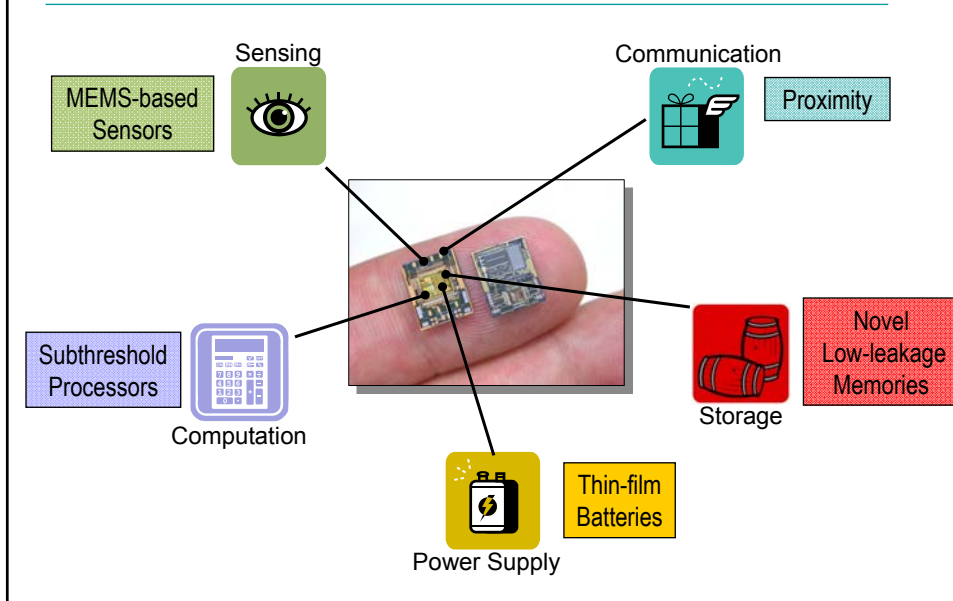
- Large and fast caches
 - Improving memory system performance
 - Consuming sizeable fraction of total chip power
 - StrongARM – ~60% for on-chip L1 caches
- More caches integrated on chip
 - 2x64KB L1 / 1.5MB L2 in Alpha 21464
 - 256KB L2 / 3MB(6MB) L3 in Itanium 2
- Increasing on-chip cache leakage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

Drowsy Caches [Mudge]

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

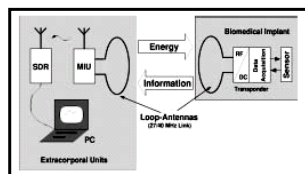


Research Topic: Subthreshold Sensor Processors [Austin/Blaauw]

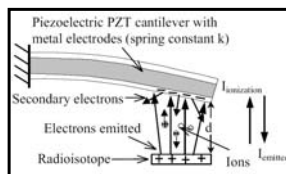


Energy Efficiency: A Key Requirement

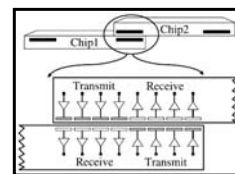
- They live on a limited amount of energy generated from a small battery or scavenged from the environment.
- Traditionally the communication component is the most power-hungry element of the system. However, new trends are emerging:



Passive telemetry

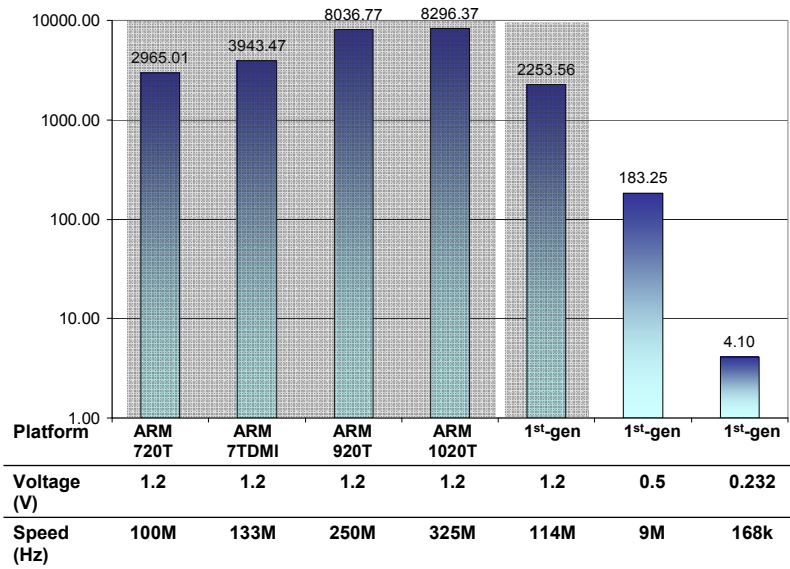


Self-powered RF



Proximity comm.

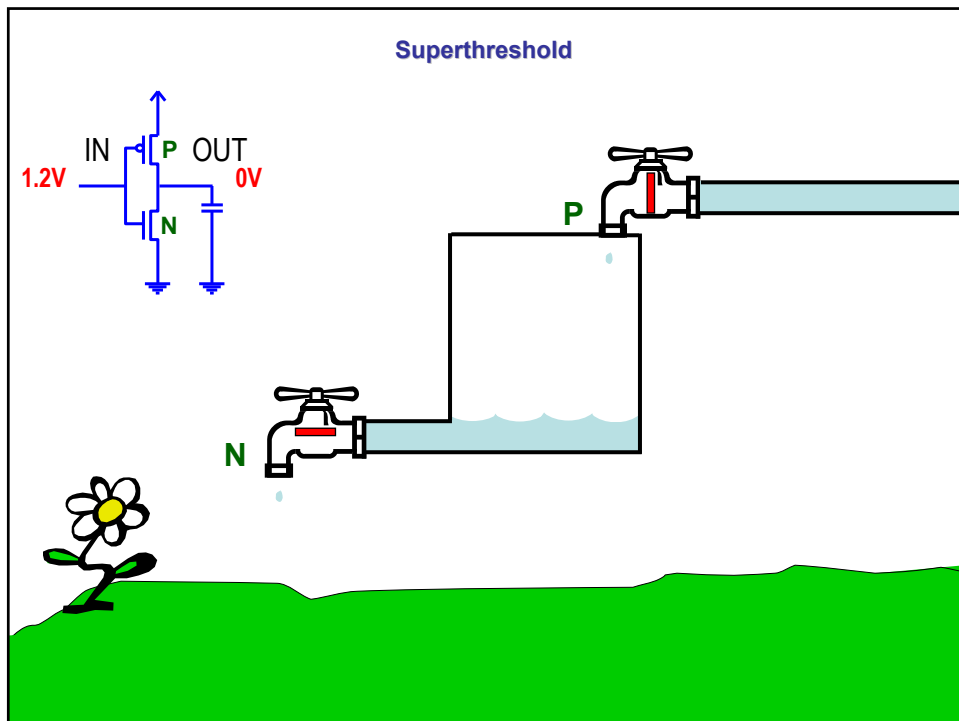
Performance Demands are LOW

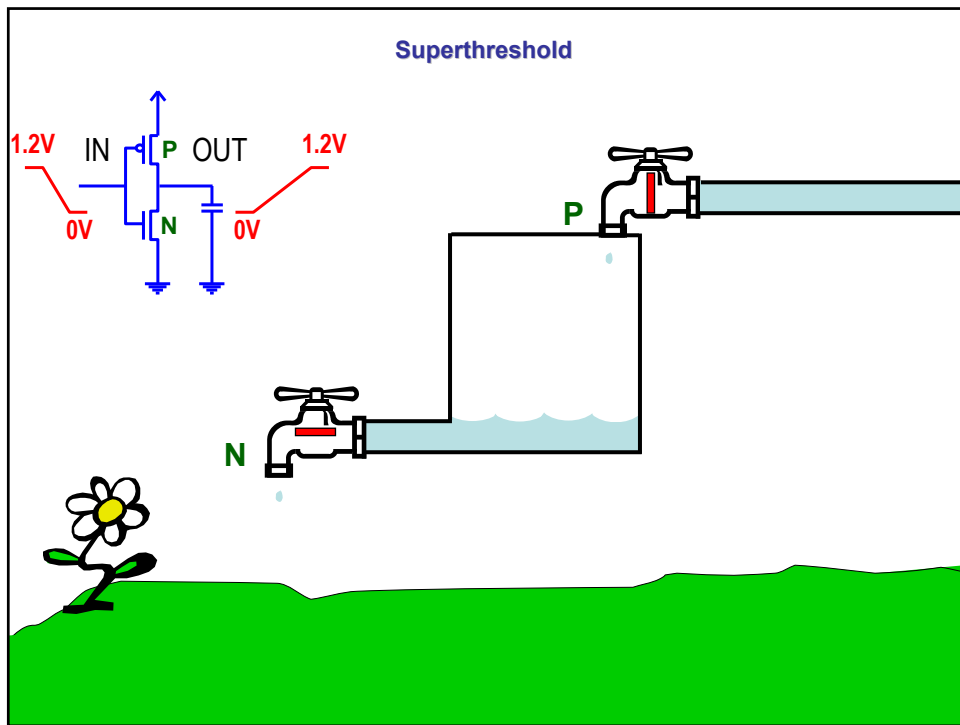
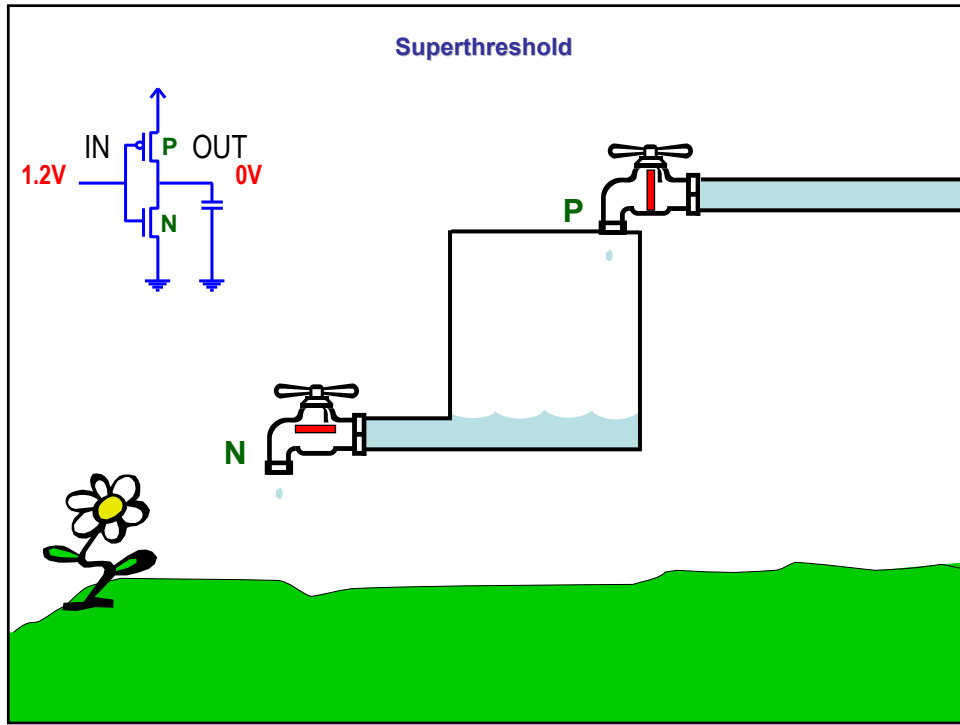


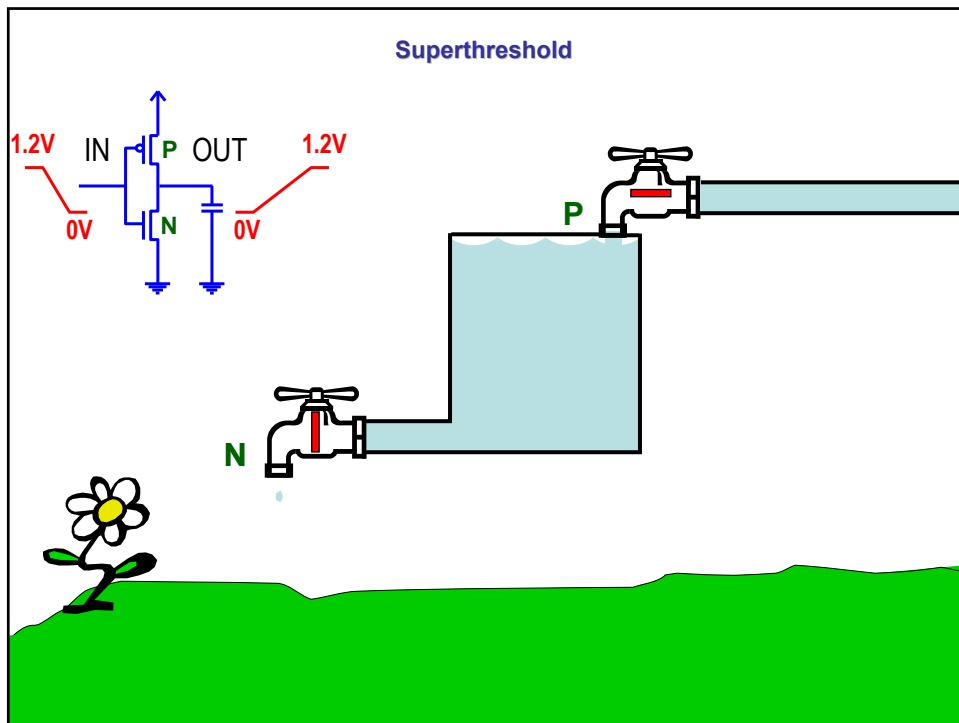
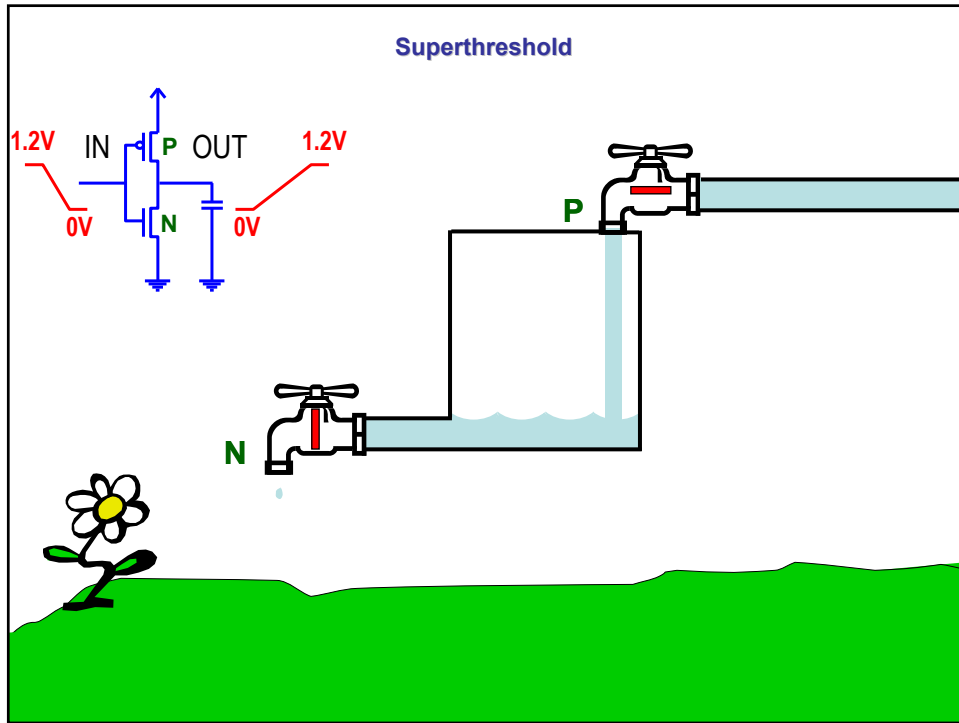
The Basics of Subthreshold Circuit Operation

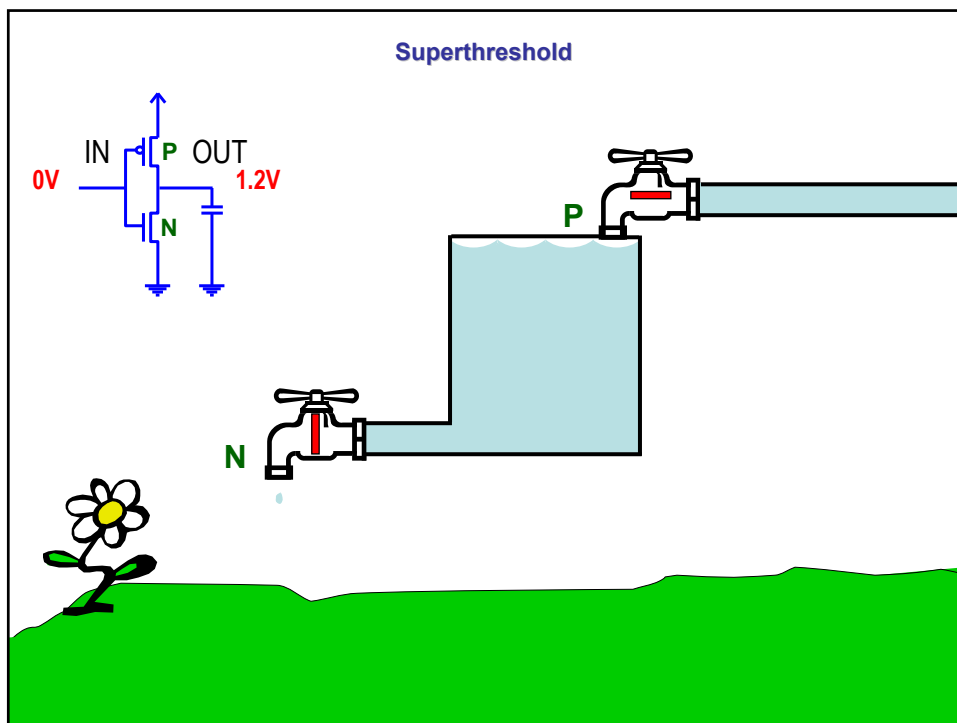
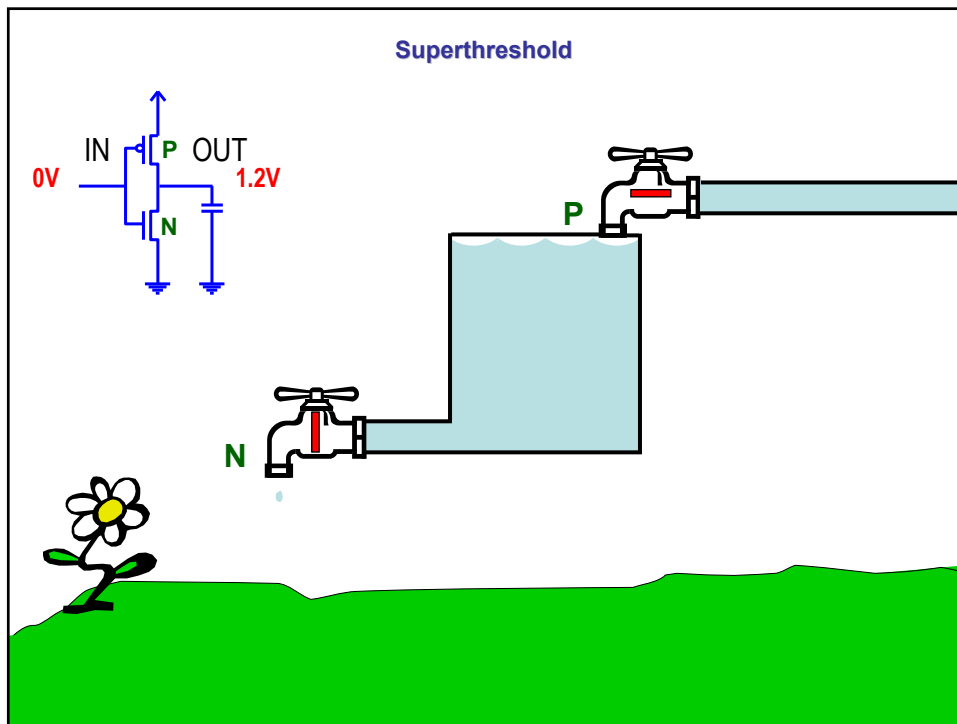
A Short Animation 😊

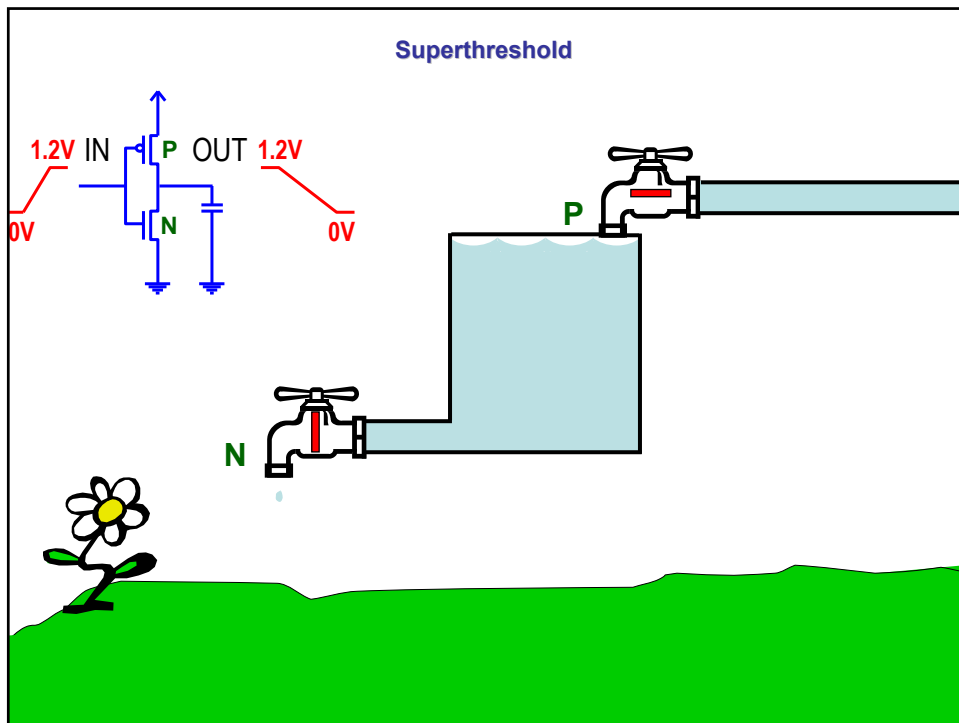
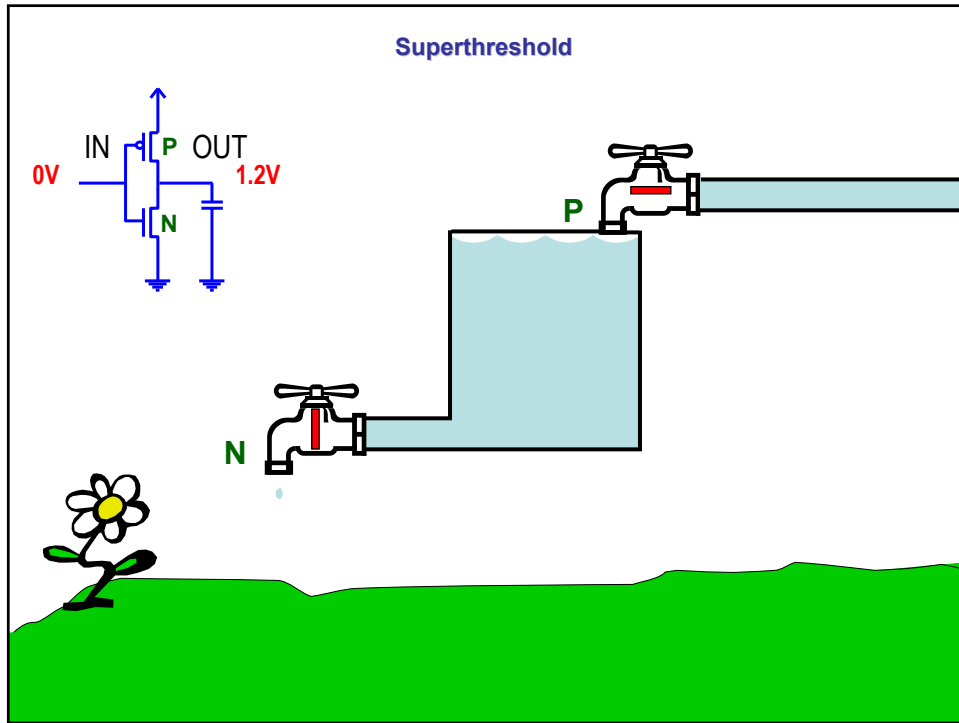
Episode 1: Inverter operation in superthreshold domain

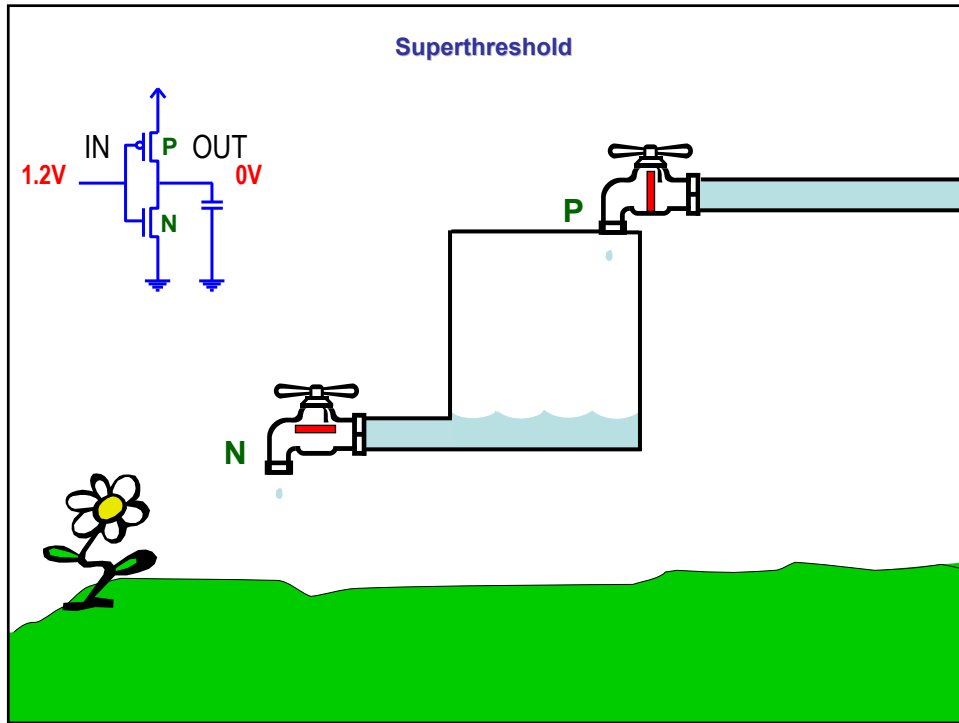




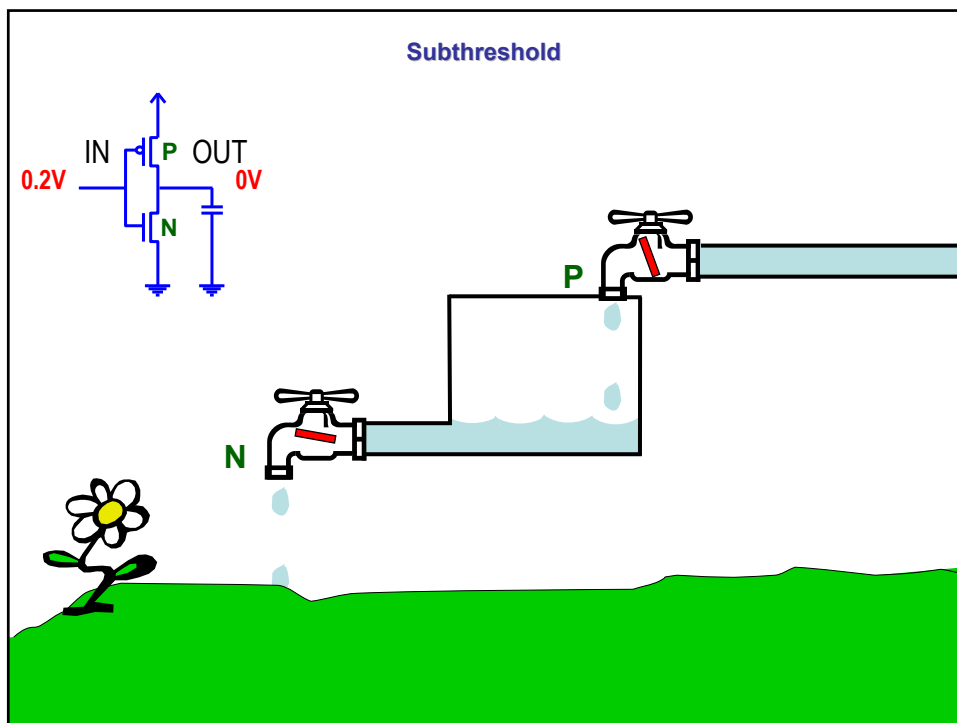
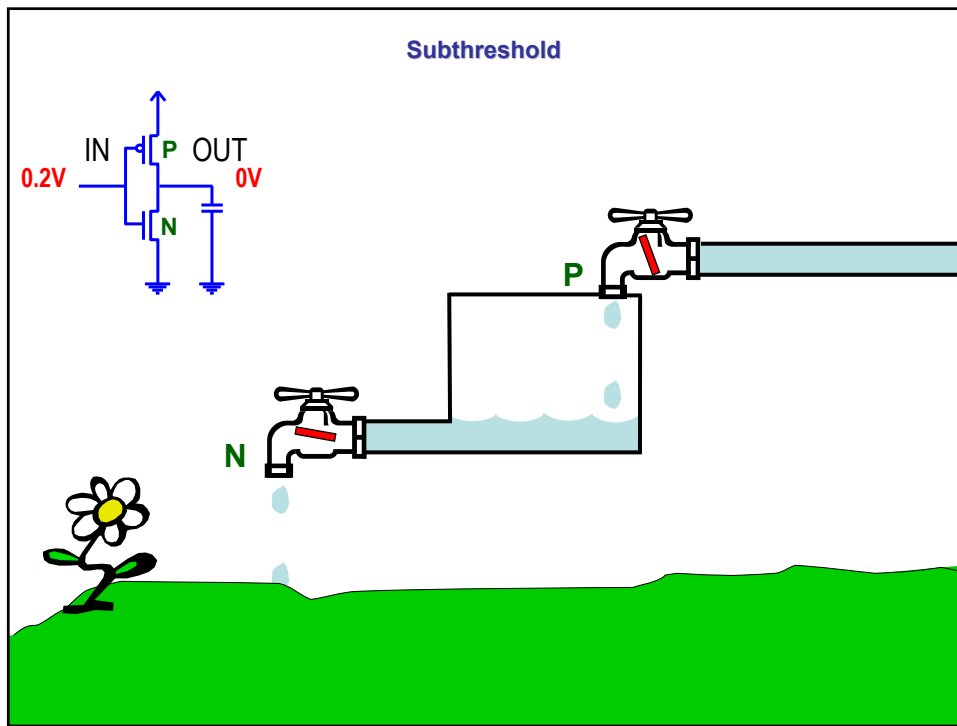


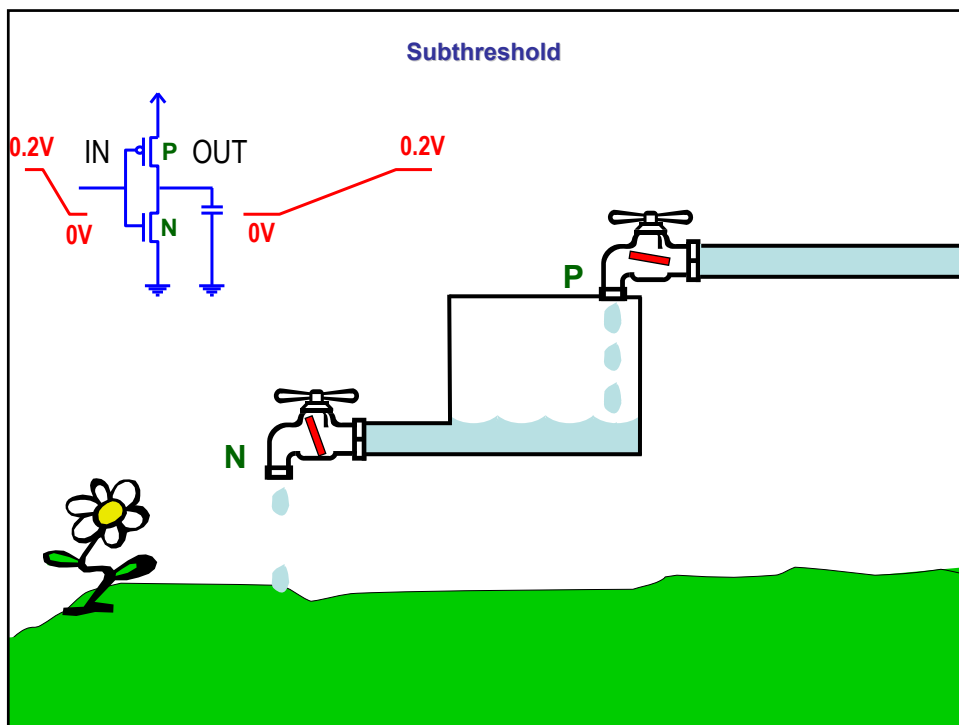
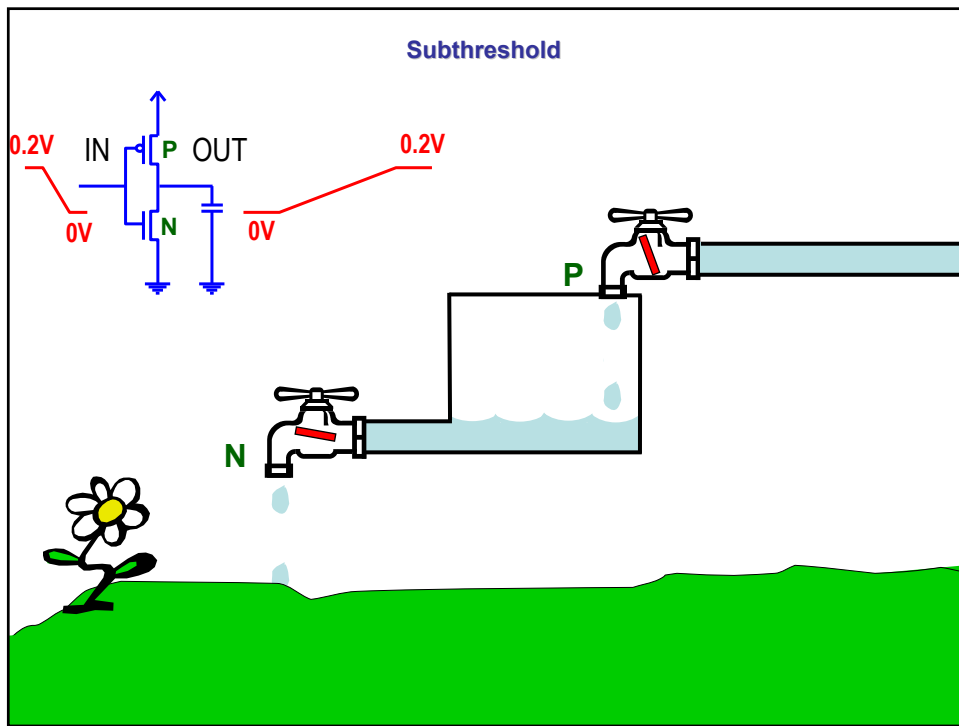


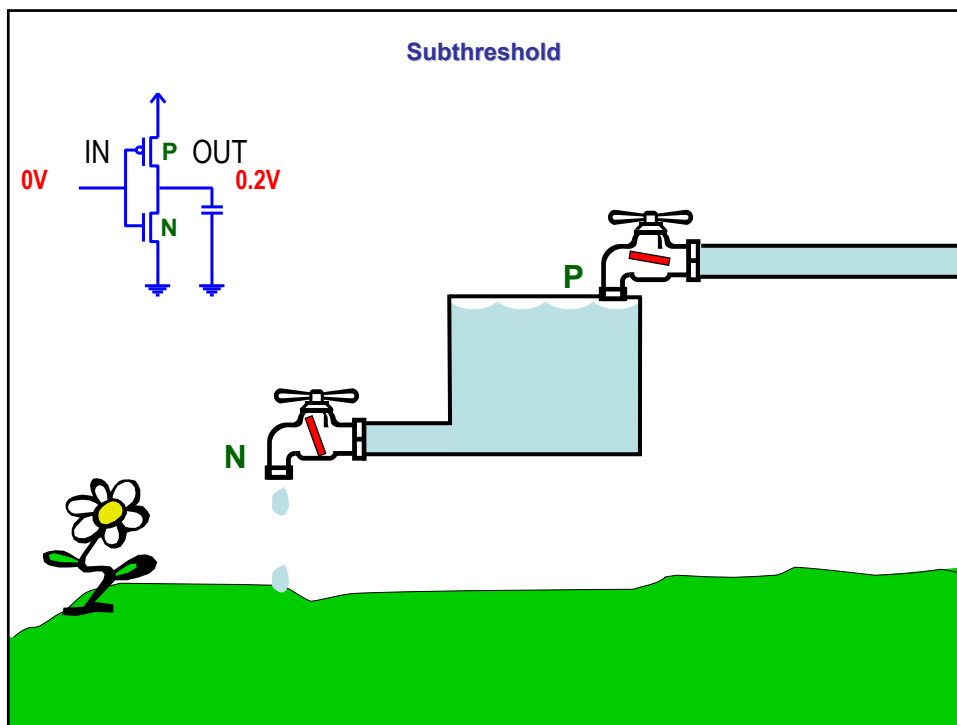
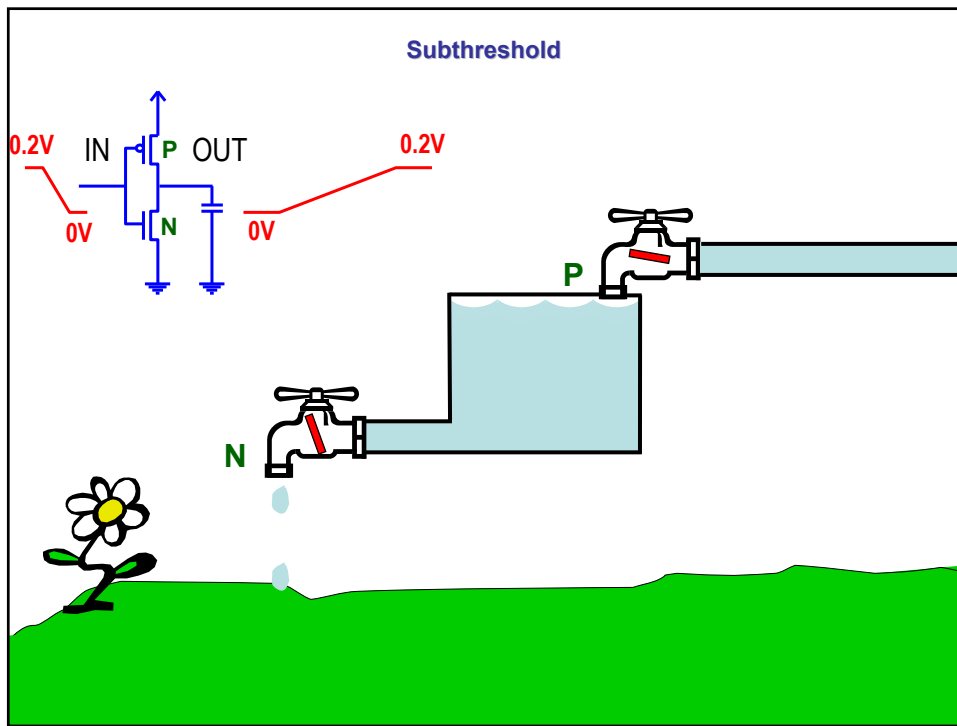


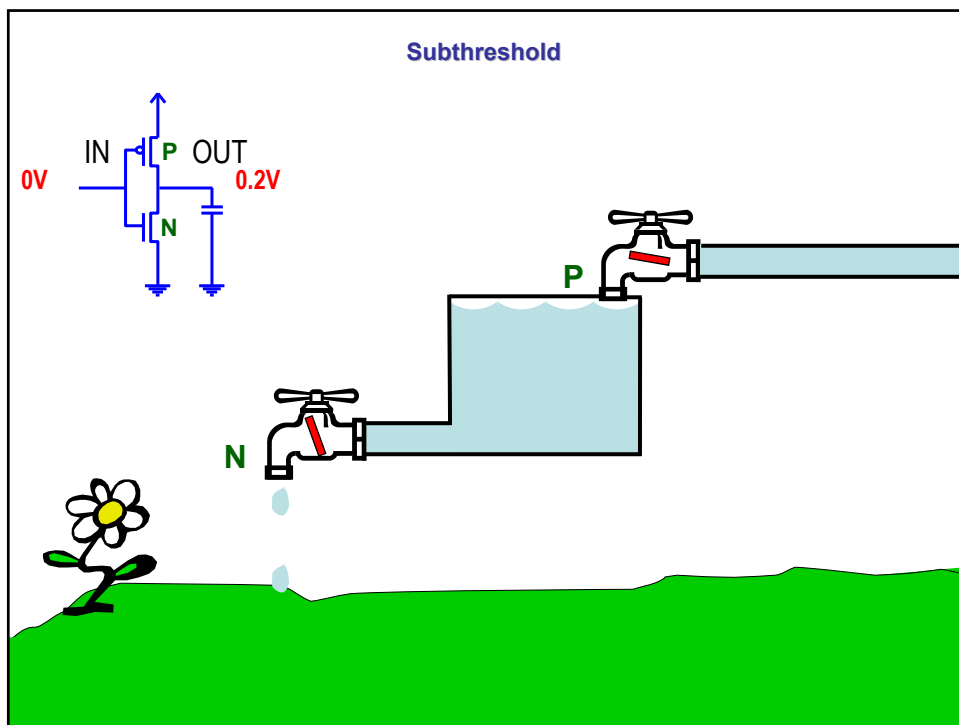
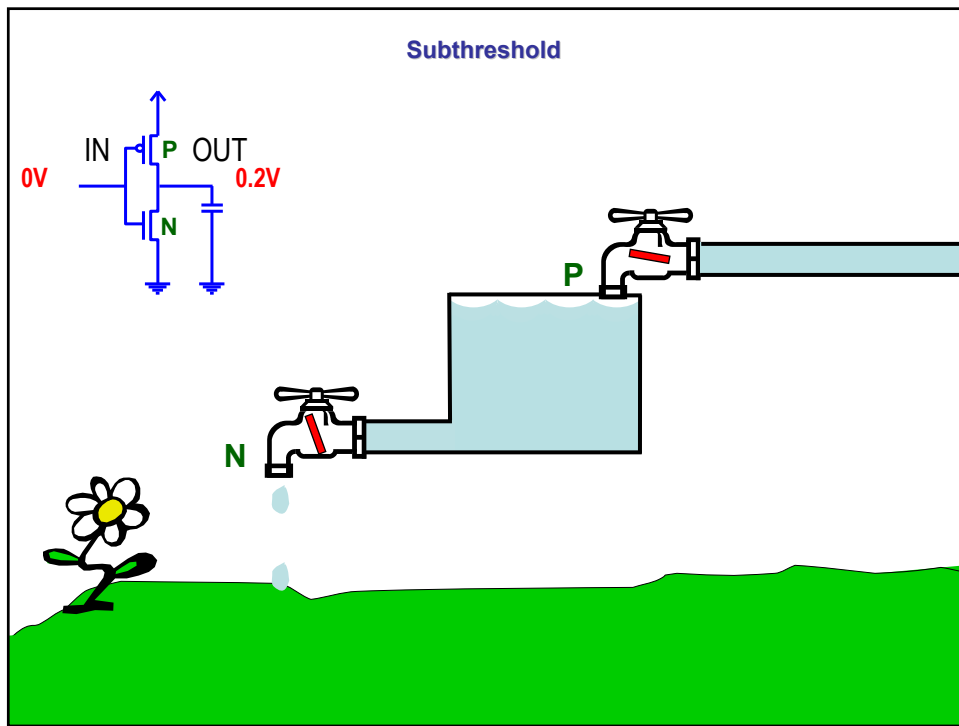


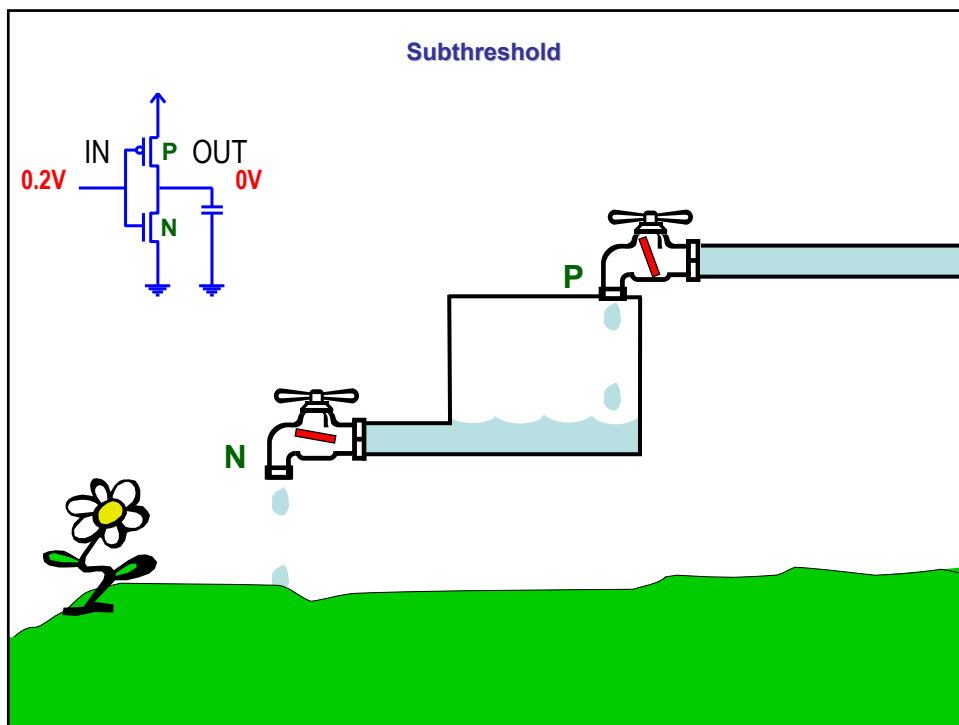
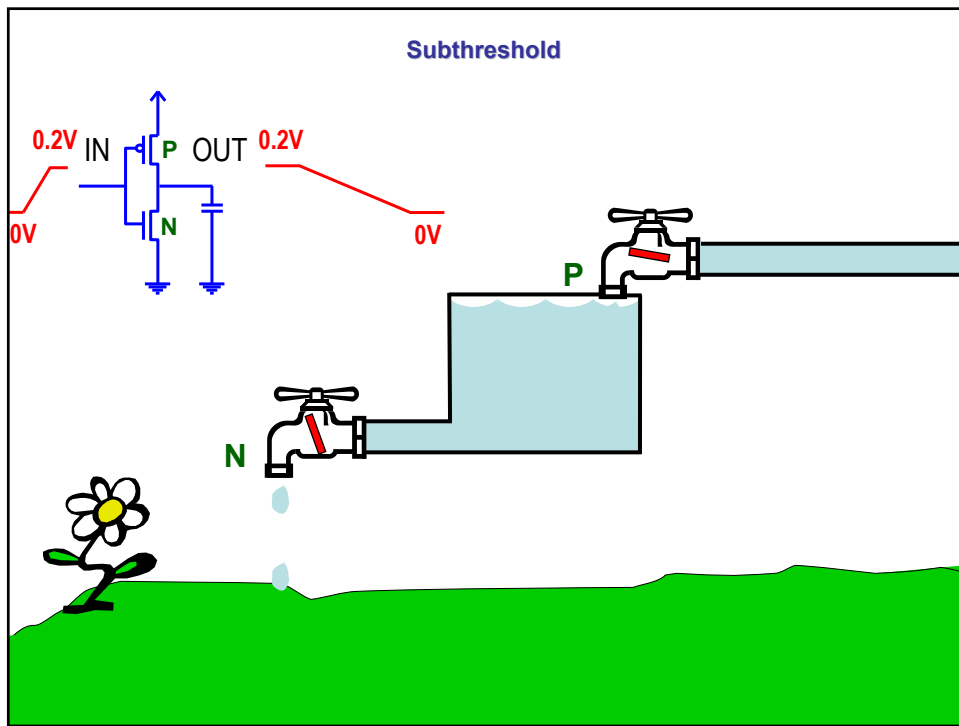
**Episode 2: Inverter operation in
subthreshold domain**









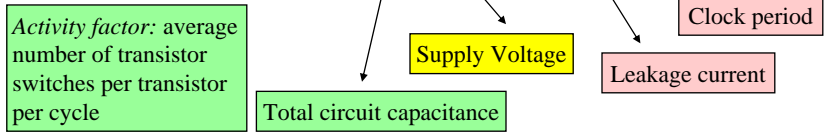


Energy Per Instruction Analysis

$$E_{inst} = E_{cycle} CPI$$

Cycles per Instruction
EPI: Energy per Instruction Energy per Cycle

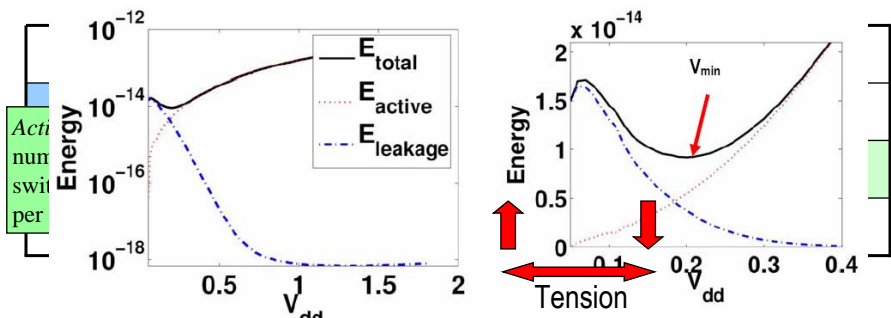
$$E_{cycle} = N \left(\alpha \frac{1}{2} C_s V_{dd}^2 + V_{dd} I_{leak} t_{clk} \right)$$



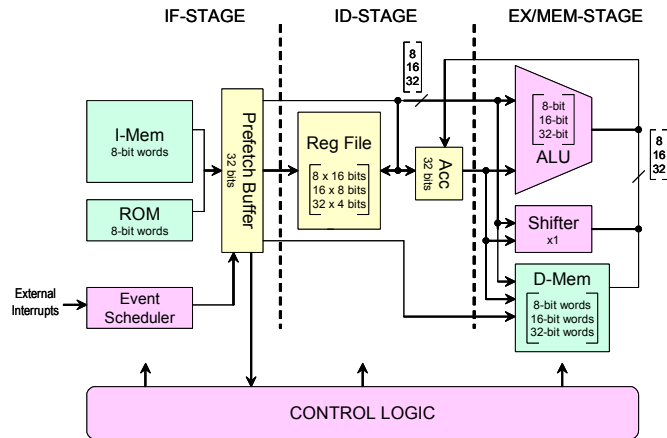
Energy Per Instruction Analysis

$$E_{inst} = E_{cycle} CPI$$

Cycles per Instruction
EPI: Energy per Instruction Energy per Cycle

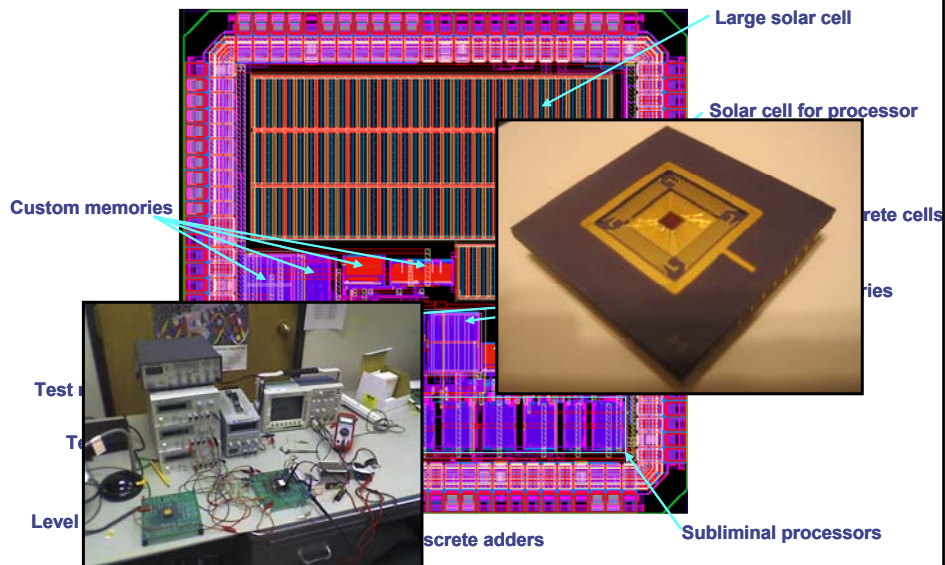


1st-gen General Microarchitecture Overview and Exploration Options

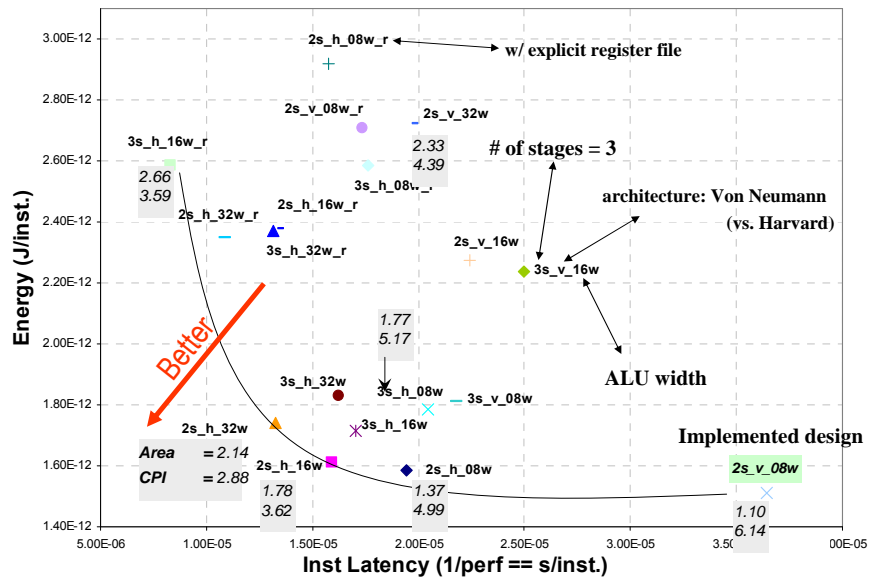


- Number of stages
- Harvard vs. Von-Neumann arch
- ALU width
- Presence of instruction prefetch buffer
- Presence of explicit register file

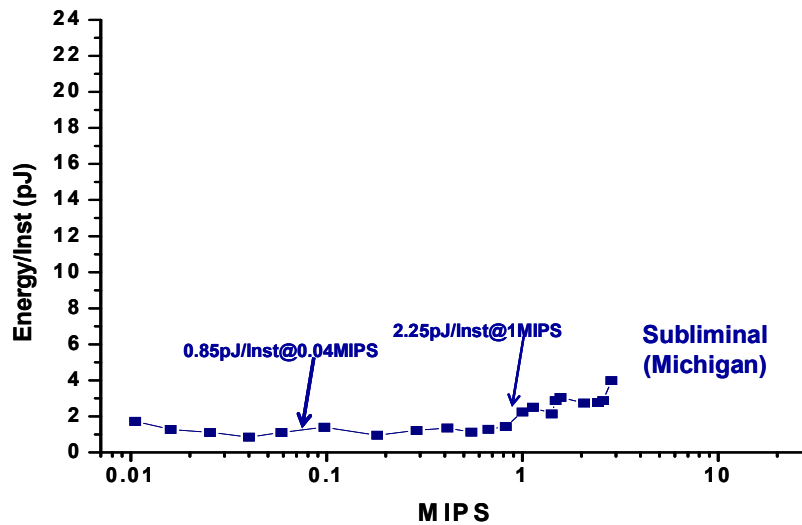
First Subliminal Chip



Pareto Analysis for Several Processors



Pareto Analysis of Sensor Network Processors



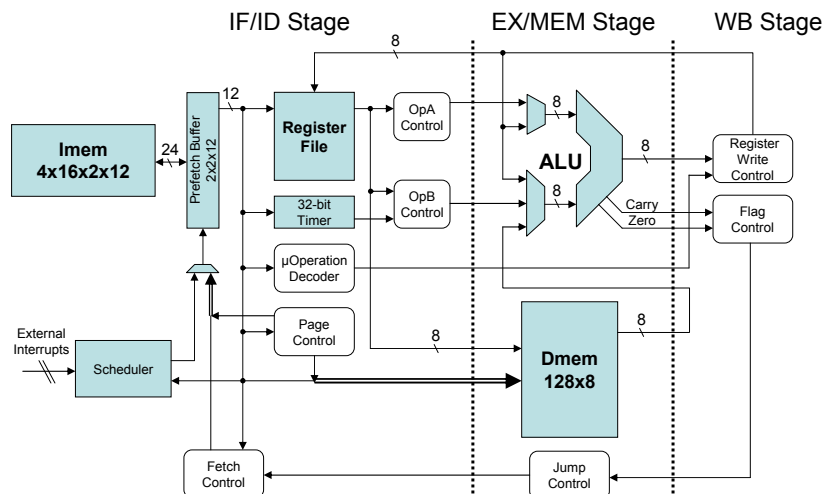
Lessons from 1st-generation Study (ISCA 2005)

- To minimize energy at subthreshold voltages, architects must:

Minimize area	⇒ To reduce leakage energy per cycle
Maximize Transistor utility	⇒ To reduce V_{\min} and energy per cycle
Minimize CPI	⇒ To reduce Energy per instruction

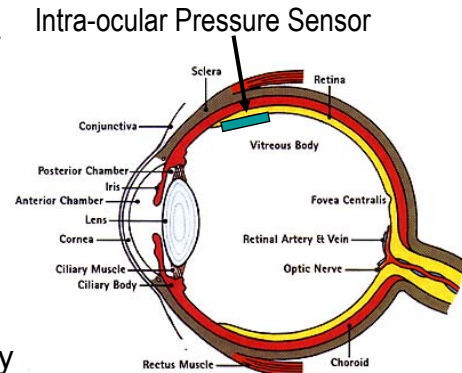
- As such, winning designs tend to be compromising designs that balance area, transistor utility and CPI
- The memory comprises the single largest factor of leakage energy, therefore, efficient designs must reduce memory storage requirements.

2nd Generation Sensor Network Processor



Ongoing Work

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



Tutorial Schedule

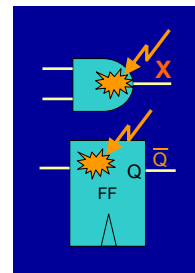
- **Power Issues: Dynamic and Static Power**
- **Low Power Design Techniques**
- **Reliability Issues: SER, Variability and Defects**
 - **Soft Error Radiation Overview**
 - **Variability Sources and Effects**
 - **Silicon Defect Trends**
- **Break**
- **Fault Tolerant Design Techniques**
- **Robust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

Fault Classes

- **Permanent fault (hard fault)**
 - Irreversible physical change
 - Latent manufacturing defects, Electromigration
- **Intermittent fault**
 - Hard to differentiate from transient faults
 - Repeatedly occurs at the same location
 - Occurs in bursty manners when fault is activated
 - Replacing the offending circuit removes faults
- **Transient faults (Soft Errors)**
 - Neutron/Alpha particle strikes
 - Power supply and Interconnect noises
 - Electromagnetic interference
 - Electrostatic discharge

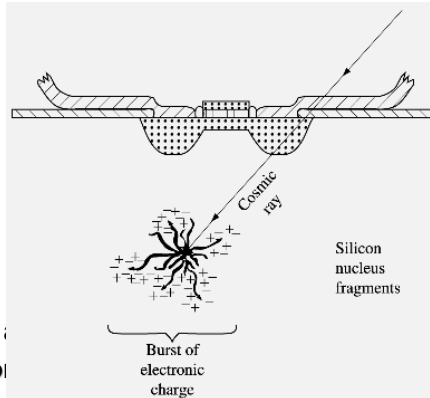
Introduction – Soft Errors

- ◆ *Soft errors, also called transient faults and single-event upsets (SEU)*
 - Processor execution errors caused by high-energy neutrons resulting from cosmic radiation and alpha particles radiation
 - Appears to be a reliability threat for future technology processors
- ◆ When a particle strikes a circuit element a small amount of charge is deposited
 - *Combinational logic node*: a very short duration pulse of current is formed at the circuit node
 - *State holding element (FF/SRAM cell)*: flip the stored value
- ◆ Unlike permanent faults the effects of soft errors are transient



Soft Errors (SER)

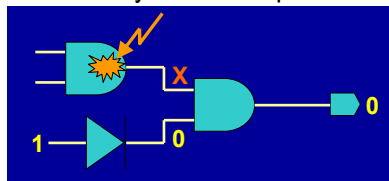
- Alpha particles stemming from radioactive decay of packaging materials
- Neutrons (cosmic rays) are always present in the atmosphere
- Soft errors are transient non-recurring faults (also called single event upsets, SEUs) where added/deleted charge on a node results in a functional error
 - Charge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



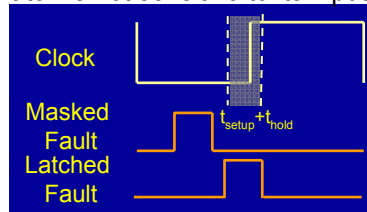
Source: S. Mukherjee, Intel

Soft Error Masking

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

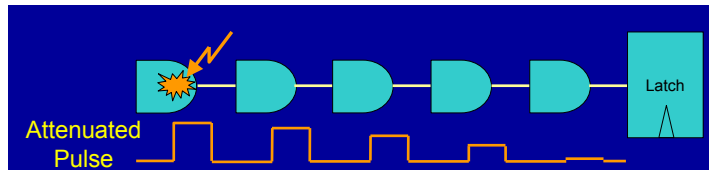


- ◆ *Timing Masking*: the fault affects the input of a latch only in the period of time that the latch is not sensitive to its input



Soft Error Masking

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



- ◆ *Microarchitectural Masking*: the fault alters a value of at least one flip-flop, but the incorrect values get overwritten without being used in any computation affecting the design's output
- ◆ *Software Masking*: the fault propagates to the design's output but is subsequently masked by software without affecting the application's correct execution

How To Measure Reliability: Soft Error Rate (FIT)

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

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_{crit}) is an important value
 - Node requires Q_{crit} to be collected before an error will result
 - The more charge stored on a node, the larger Q_{crit} is (Q_{crit} must be an appreciable fraction of stored Q)
 - Implies scaling problems → caps reduce with scaling, voltage reduces, so stored Q reduces as S^2 (~ 2X) per generation
 - Ameliorated somewhat by smaller collection nodes (S/D junctions)
 - But exacerbated again by 2X more devices per generation

Soft Error Rate Trends, ITRS03

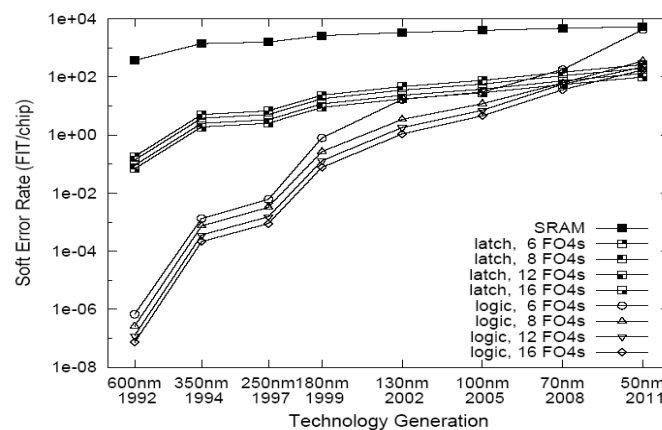


Figure 14. SER/chip for SRAM/latches/logic

Impact of Soft Errors in Processors [Iyer]

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

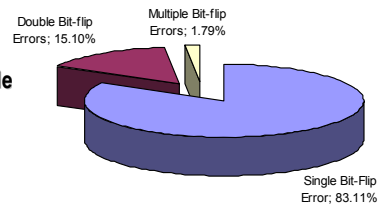
◆ Approach

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

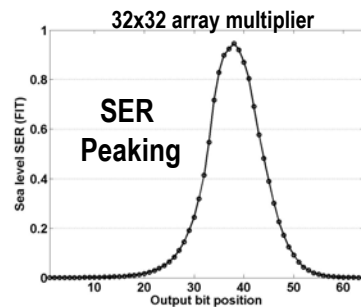
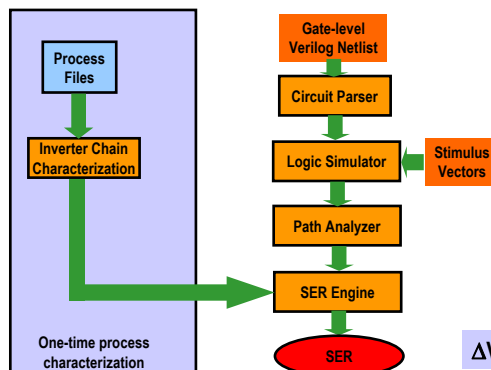
◆ Major findings:

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

Multiple Bit-flip Distribution in Alpha processor

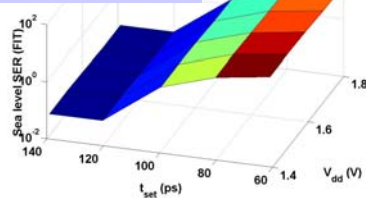


SERA SER Analysis Tool [Shanbhag]



$\Delta V_{dd} = 20\% \rightarrow \text{SER} = 1.28X$

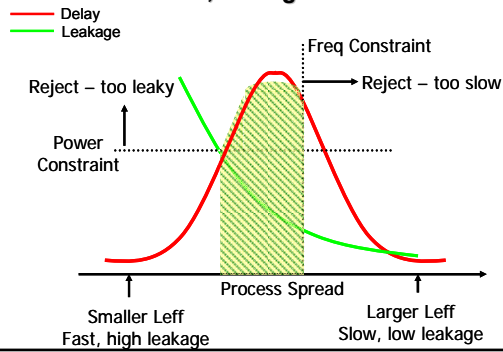
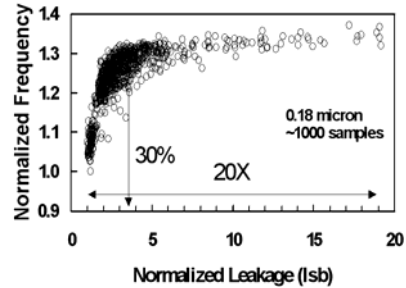
$\Delta t_{\text{setup}} = 20\% \rightarrow \text{SER} = 50X$



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

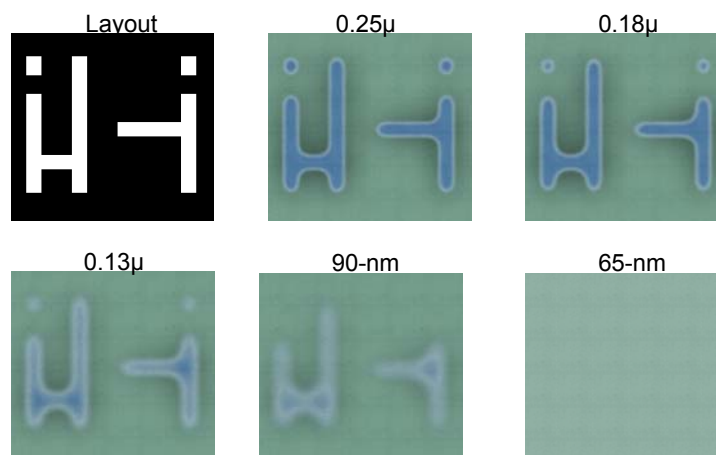
Effects Of Variability

- High-performance processors are speed-binned
 - Faster == more \$\$\$
 - These parts have small Leff
- Exponential dependence of leakage on Vth
 - And Leff, through Vth



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

Printing in the Subwavelength Regime



Figures courtesy Synopsys Inc.

Variation: Across-Wafer Frequency

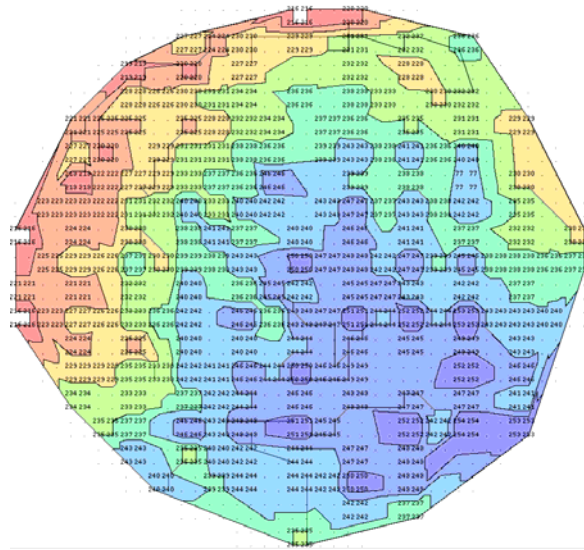
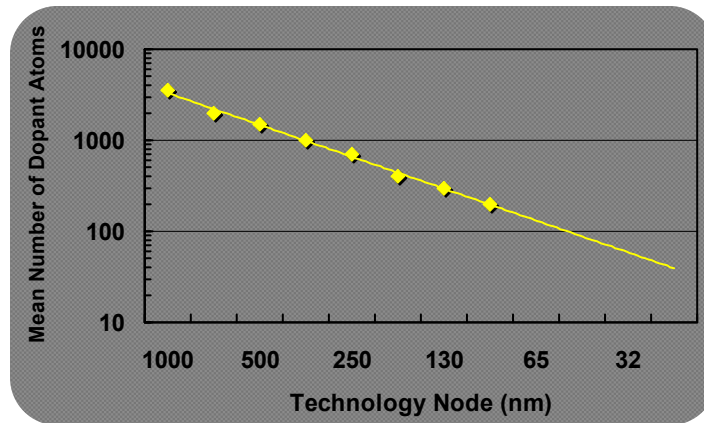
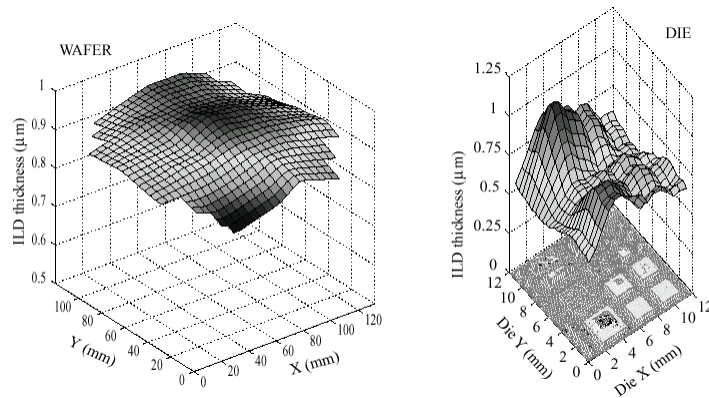


Figure courtesy S. Nassif, IBM

Random Dopant Fluctuations, Intel's View



Inter-die vs. Intra-die Variation

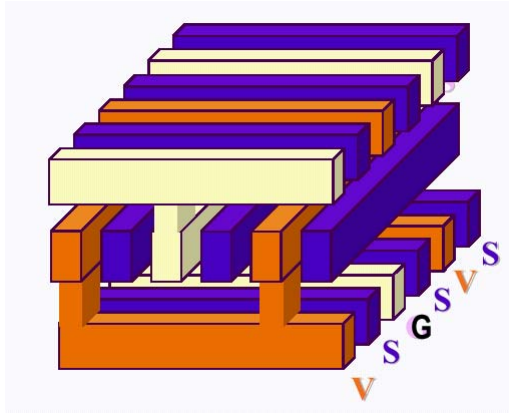


Inter-die variation is not always larger than intra-die (ILD)

Design/EDA for Highly Variable Technologies

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

Noise Immune Layout Fabric



Major area penalty (>60%)

This layout style trades off area for:

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

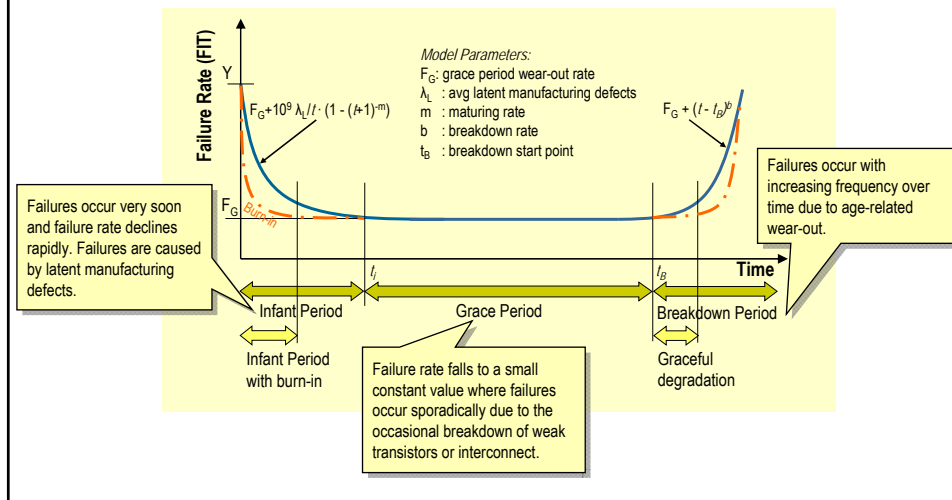
Ref: Khatri, DAC99

Defects: The (Bumpy) Road Ahead for Silicon

- ◆ What is the failure model of silicon 2-3 generations out?
 - ◆ What the literature says...
 - ◆ "Expected failure rate of 10^{12} hours/device", this would give a high end NVidia graphics part an expected lifetime of less than 1 year
 - ◆ "Failure rates higher than 10^{20} hours/device", which eliminates the problem
 - ◆ What the experts say...
 - ◆ Intel [Borkar] and IBM [Bernstein]: critical problem for future silicon
- ◆ Key failure modes
 - ◆ Transistor wear-out (aggravated by scaling)
 - ◆ SER-related upsets (especially in logic)
 - ◆ Early transistor failures (due to ineffective burn-in)
 - ◆ Untestable defects (compounded by complexity)

Silicon Defects: Sources and Trajectory

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



Tutorial Schedule

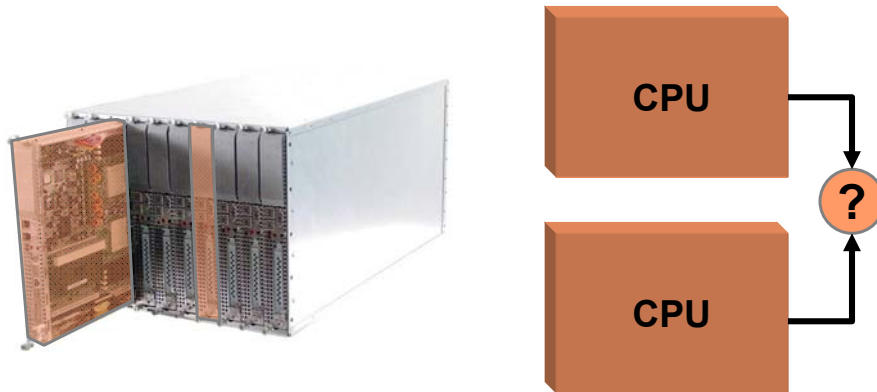
- **Power Issues: Dynamic and Static Power**
- **Low Power Design Techniques**
- **Reliability Issues: SER, Variability and Defects**
- **Break**
- **Fault Tolerant Design Techniques**
 - Classical Techniques
 - SER Specific Techniques
 - Full-Spectrum Techniques
 - Research Topic: Self-Healing Systems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

Techniques For Improving Reliability

- **Fault avoidance (Process / Circuit)**
 - Improving materials
 - Low Alpha Emission interconnect and Packaging materials
 - Manufacturing process
 - Silicon On Insulator (SOI)
 - Triple Well design process to protect SRAM
- **Fault tolerance (robust design in presence of Soft Error) : Circuit / Architecture**
 - Error Detection & Correction relies mostly on “Redundancy”
 - Space : DMR, TMR
 - Time : Temporal redundant sampling (Razor-like)
 - Information : Error coding (ECC)

DMR Error Detection

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

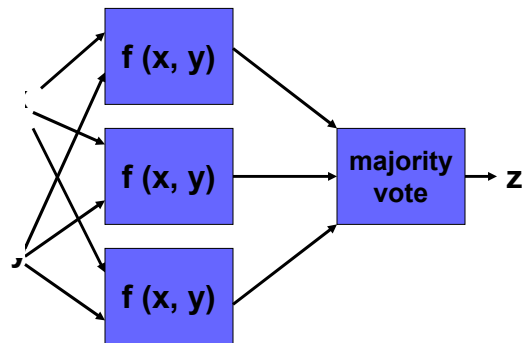


Triple Modular Redundancy (von Neumann)

Voter assumed
reliable!

⇒ voter small

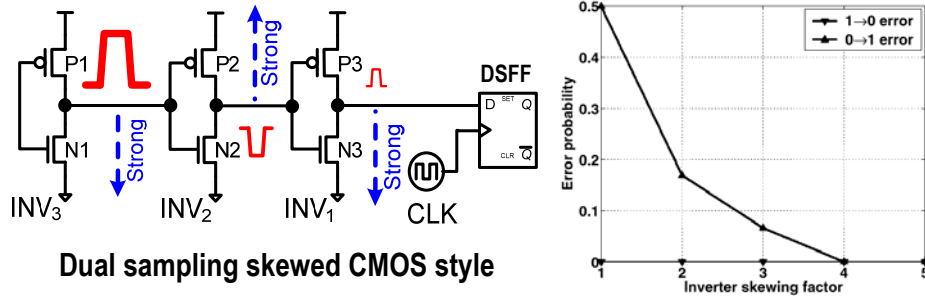
⇒ coarse-grained



Error Coding : Information Redundancy

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

SER-Tolerant Circuit Design [Shanbhag]

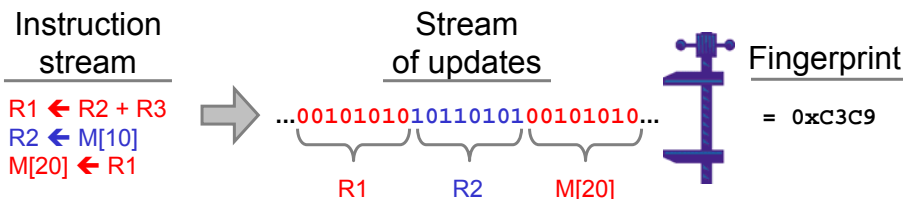


Dual sampling skewed CMOS style

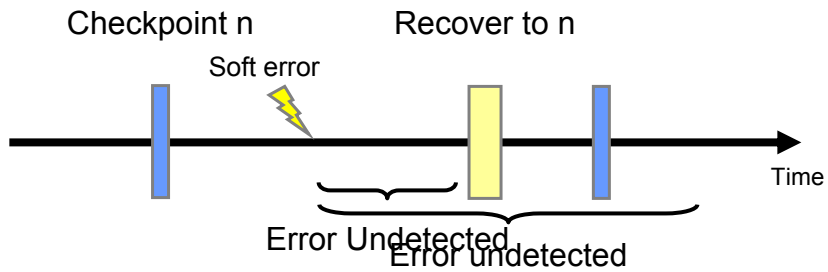
- ◆ Employs skewed CMOS for logic and dual sampling FF (DSFF)
- ◆ Both 0→1 and 1→0 errors are eliminated if skewing factor ≥ 4 .
- ◆ Speed penalty
 - depends on Δ (maximum SET width)
 - can be made a design parameter.
 - equals 300ps (for 0.18um process) if zero SER wanted.
- ◆ Power penalty: 17% (DSFF)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

Fingerprinting [Falsafi/Hoe]

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



Recovery Model

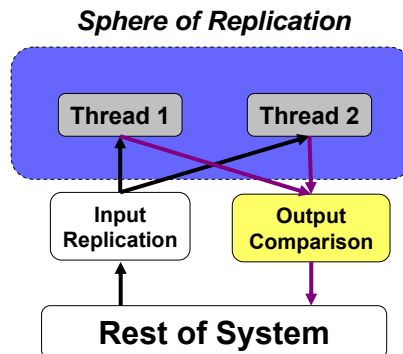


➤ **Rollback-recovery to last checkpoint upon detection**

Simultaneous Redundant Multithreading [Reinhardt]

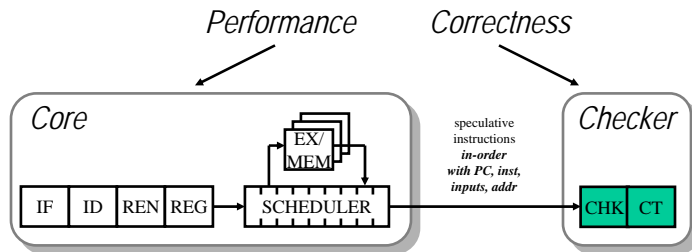
Logical boundary of redundant execution within a system

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



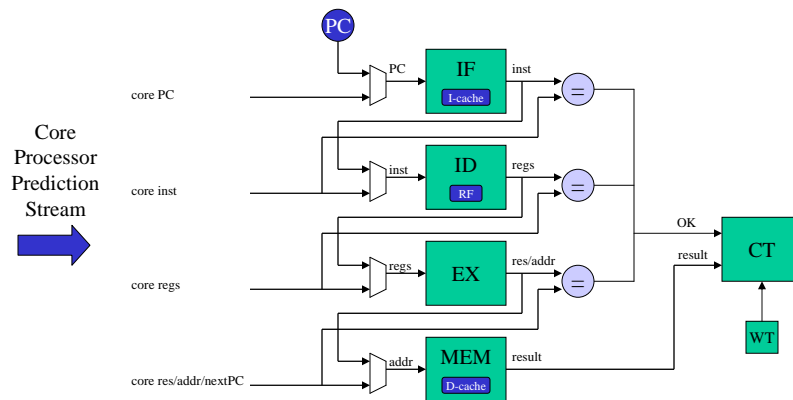
Compare & validate output before sending it outside the SoR

Full-Spectrum Fault Tolerance: DIVA Checker [Austin]

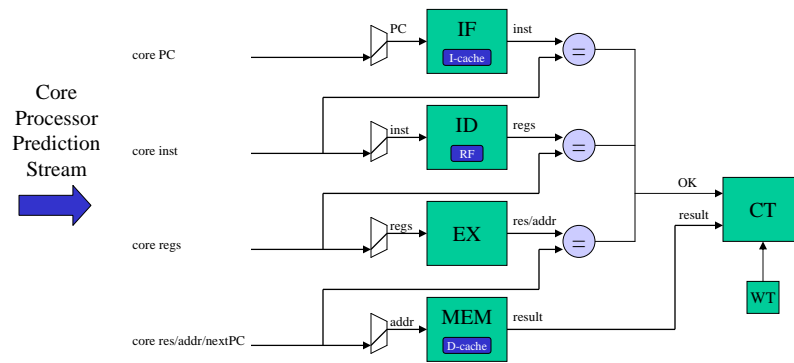


- **All core function is validated by checker**
 - Simple checker *detects* and *corrects* faulty results, restarts core
- **Checker relaxes burden of correctness on core processor**
 - Tolerates design errors, electrical faults, defects, and failures
 - Core has burden of accurate prediction, as checker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

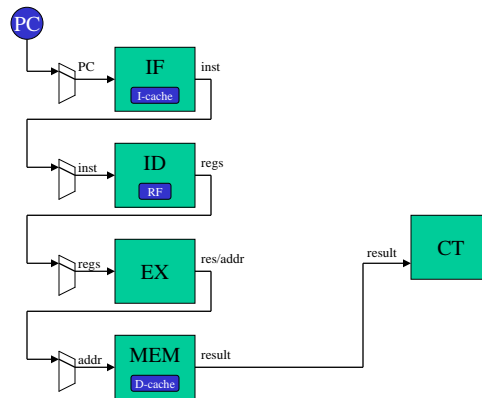
Checker Processor Architecture



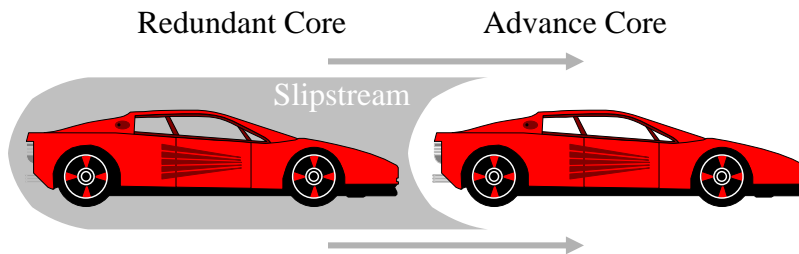
Check Mode



Recovery Mode

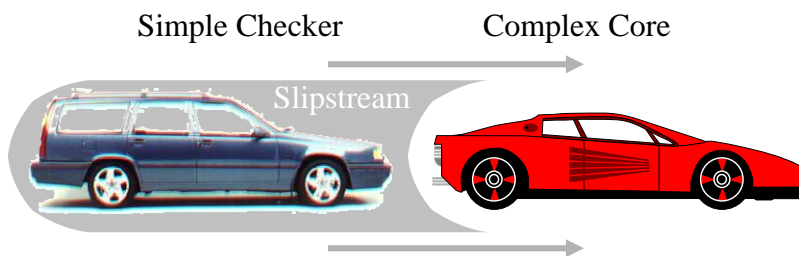


How Can the Simple Checker Keep Up?



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

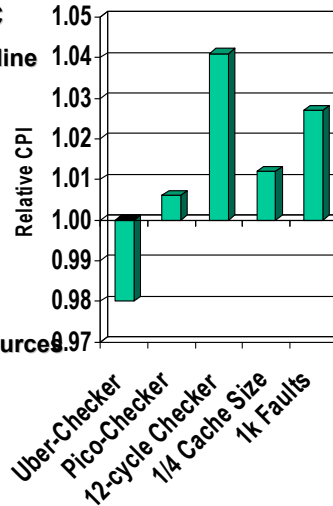
How Can the Simple Checker Keep Up?



- **Slipstream effects reduce power requirements of trailing car**
 - Checker processor executes in the core processor slipstream
 - fast moving air \Rightarrow 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



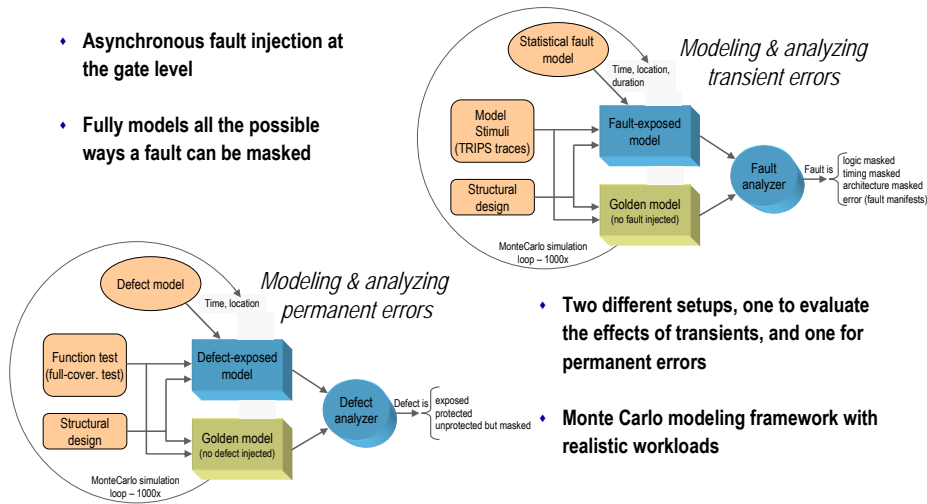
Research Topic: Self-Repairing Systems

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

Fault Modeling & Analysis Infrastructure

◆ High-performance, high-fidelity, fault modeling simulation infrastructure

- ◆ Asynchronous fault injection at the gate level
- ◆ Fully models all the possible ways a fault can be masked



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

Self-Repairing BulletProof Silicon [Austin, Bertacco]

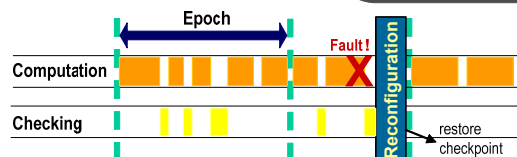
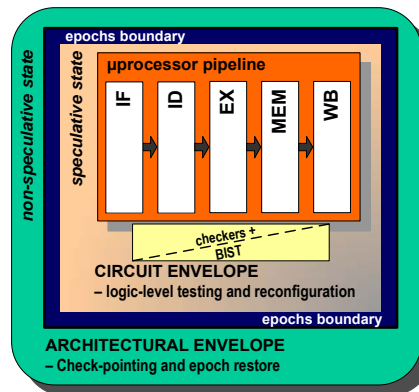
Goal: Single-defect tolerance for 5% area overhead

Key ideas:

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

Approach:

1. Execute and protect state
2. Test concurrently when Hw idle
3. If **tests fails** → roll back state
 - disable component
 - restart

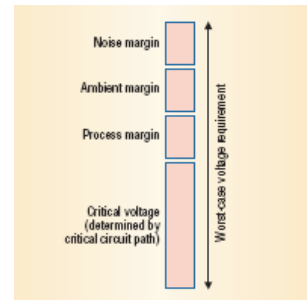


Tutorial Schedule

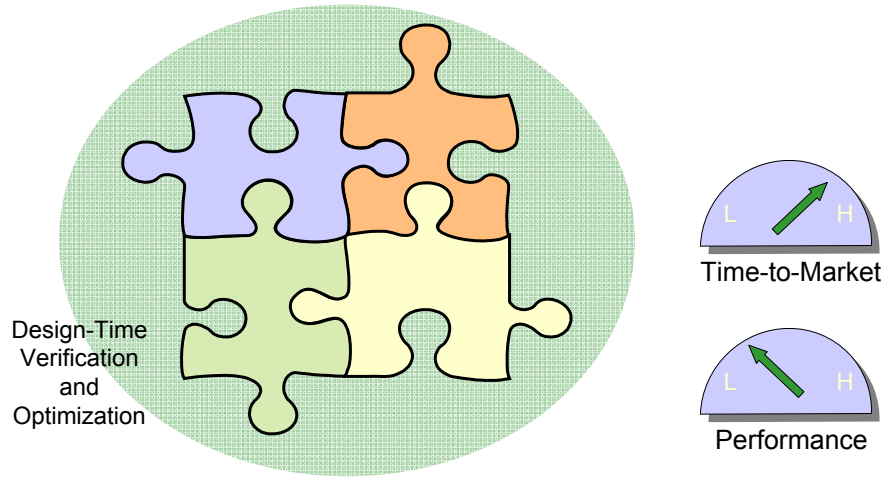
- Power Issues: Dynamic and Static Power
- Low Power Design Techniques
- Reliability Issues: SER, Variability and Defects
- Break
- Fault Tolerant Design Techniques
- Robust Low Power Design Techniques
 - Better-Than Worst Case Design Concepts
 - Example BTWC Designs
 - Research Topic: Razor Pipeline

Power and Reliability: How are they related?

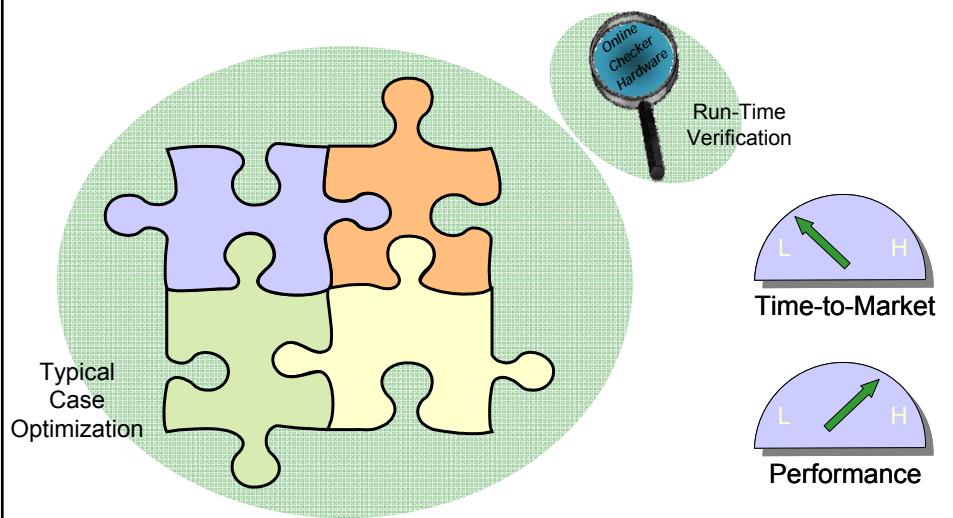
- ◆ The move to smaller features can help with power – with qualifications
- ◆ Smaller features increase design margins
 - reduce power savings
 - reduce performance gains
 - reduced area benefits



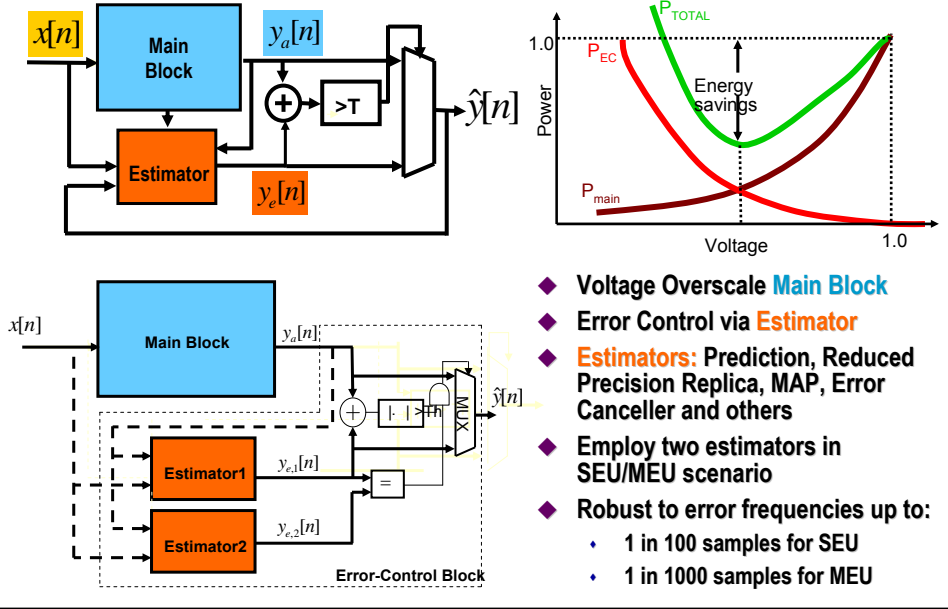
Traditional Worst-Case Design



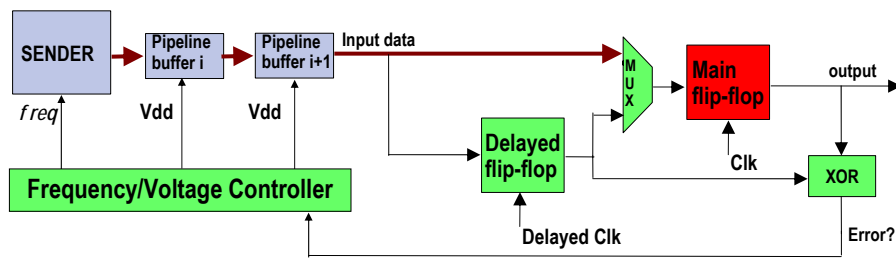
Better-Than-Worst-Case (BTWC) Design



Algorithmic SER-Tolerance [Shanbhag]



Timing Error Tolerant Links [De Micheli]



- ◆ Aggressively clock on-chips links with high frequency/low voltage
 - Double-sample link output
 - Once speculatively, then again with reliable timing
- ◆ Stall receiver for recovery data if samples disagree
 - Non-speculative if receiver incurs additional delay
 - Otherwise, receiver must perform internal recover

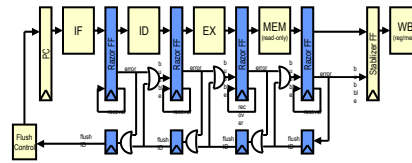
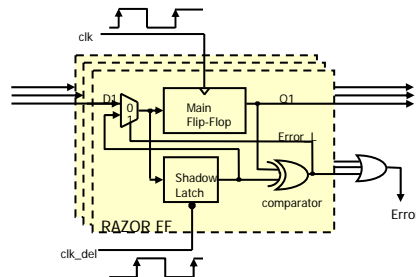
Research Topic: Razor Error Resilient Circuits [Austin/Blaauw]

◆ In-situ detection/correction of timing errors

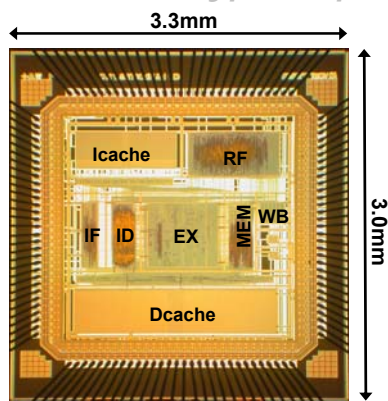
- Tune processor voltage based on errors
- Eliminate process, temperature, and noise margins (tune for near-zero errors)
- Purposely run *below* critical voltage to capture *data-dependent latency margins*

◆ Implemented with architecture and circuit support

- Double-sampling metastability-tolerant Razor flip-flops validate pipeline results
- Pipeline initiates recovery after timing errors, forward progress is guaranteed



Razor Prototype Chip



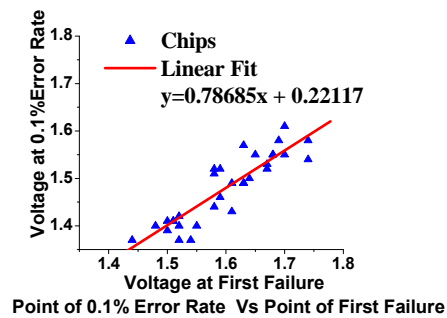
◆ 4 stage 64-bit Alpha pipeline

- 120 - 160MHz operation, 0.18 μ m

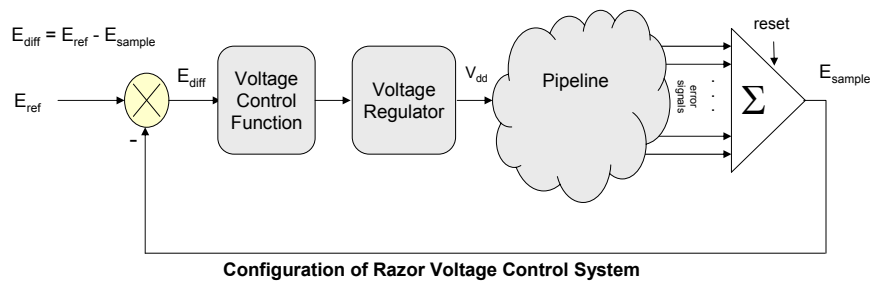
◆ Percentage of FF Razorized: 9%

- Error free Razor overhead ~3%

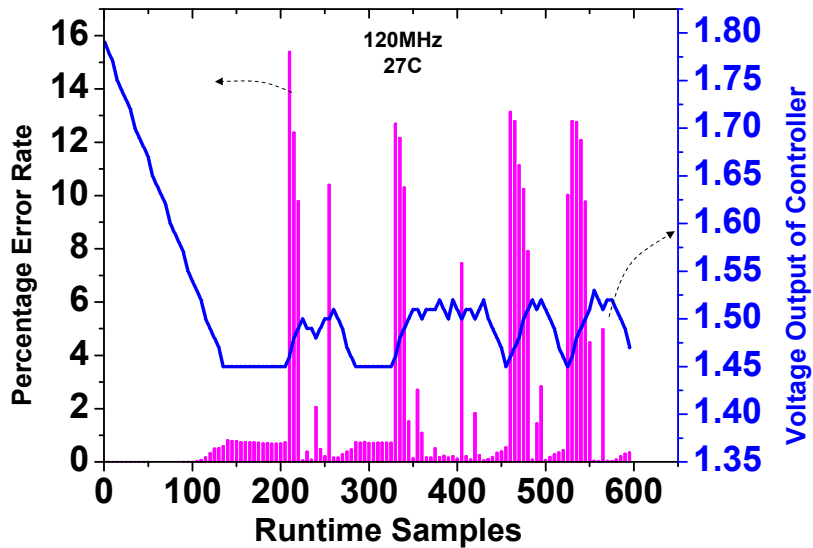
◆ 54% energy reduction



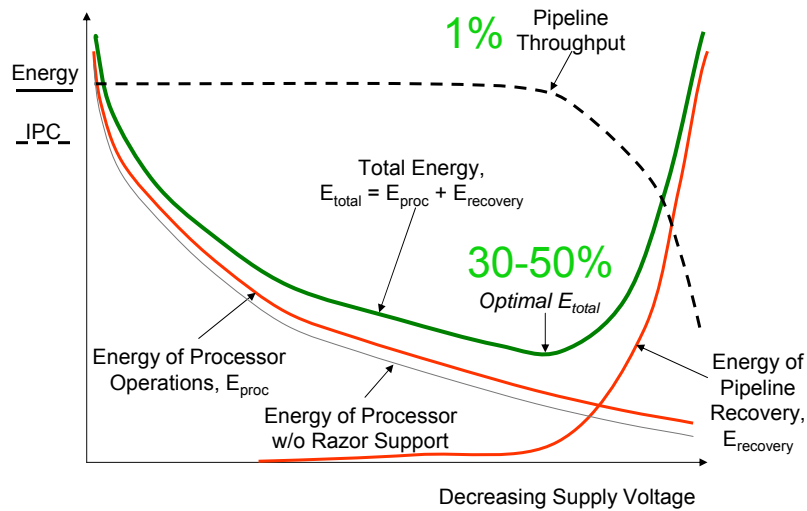
Configuration of the Razor Voltage Controller



Run-Time Response of Razor Voltage Controller



Energy/Performance Characteristics



Conclusions

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

References

1. C. Constantinescu 'Trend and Challenge in VLSI Circuit Reliability' intel
2. H. T. Nguyen 'A Systematic Approach to Processor SER Estimation and Solutions'
3. P. Shivakumar et. al, 'Modeling the effect of Technology trends on Soft Error Rate of Combinational Logic'
4. P. Shivakumar 'Fault-Tolerant Computing for Radiation Environment' Ph.D. Thesis Stanford University
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
10. J. von Neumann, "Probabilistic logic and the synthesis of reliable organisms from unreliable components,"
11. R. E. Lyons, et. al. 'The Use of Triple-Modular Redundancy to Improve Computer Reliability'
12. D. G. Mavis, et. al. 'Soft Error Rate Mitigation Techniques for Modern Microcircuits.' IEEE 40th Annual International Reliability Physics Symposium 2002.
13. C. Weaver, et. al. 'A Fault Tolerant Approach to Microprocessor Design' DSN'01
14. J. Ray, et. al. 'Dual Use of Superscalar Datapath for Transient-Fault Detection and Recovery', Proceedings of the 34th Annual Symposium on Microarchitecture (MICRO'01).
15. J. B. Nickel, et. al. 'REESE: A Method of Soft Error Detection in Microprocessors', Proceedings of the International Conference on Dependable Systems and Networks (DSN'01).
16. S. Reinhardt, et. al. 'Transient Fault Detection Simultaneous Multithreading'

References

1. D. Siewiorek 'Fault Tolerance in Commercial Computers' CMU
2. W. Bartlett, et. al. 'Commercial Fault Tolerance: A Tale of Two Systems' IEEE Dependable and Secure Computing 2004
3. T. Slegel et.al 'IBM's S/390 G5 Microprocessor Design'
4. L. Spainhower, et.al, 'IBM S/390 Parallel Enterprise Server G5 fault tolerance: A historical approach'
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.'
9. Milos Prvulovic, et. al. 'ReVive:Cost-Effective Architectural Support for Rollback Recovery in Shared-Memory Multiprocessors'
10. J. Smolens, et.al 'Fingerprinting: Bounding SoftError Detection Latency and Bandwidth'
11. D. Sorin, et.al 'Dynamic Verification of End-to-End Multiprocessor Invariants'

Questions?

?

?

?

?

?

?

?

?

?

?

?

?