Design and Runtime Techniques for Leakage Control and Minimization of CMOS VLSI Circuits in Active and Sleep Modes – PART I

Massoud Pedram
University of Southern California
Los Angeles, California
pedram@usc.edu

Farzan Fallah
Fujitsu Labs. of America
Sunnyvale, California
farzan@fla.fujitsu.com

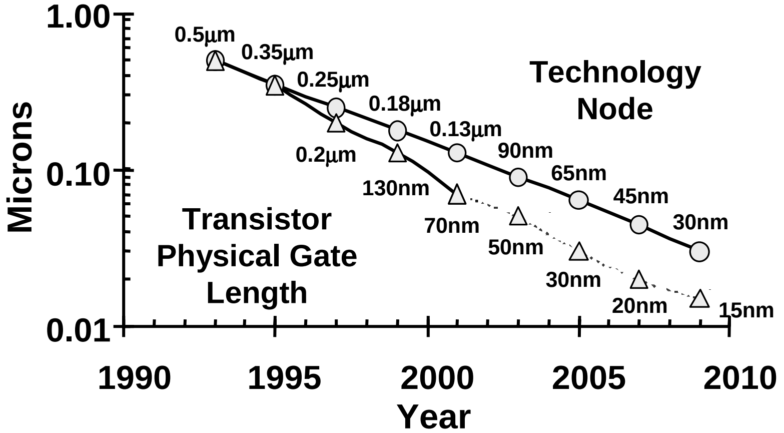
Global Outline

- ➔ PART I: Sources of Leakage Power and Trends
- PART II: Design Techniques for Leakage Minimization
- PART III: Leakage-aware Circuits and Memory

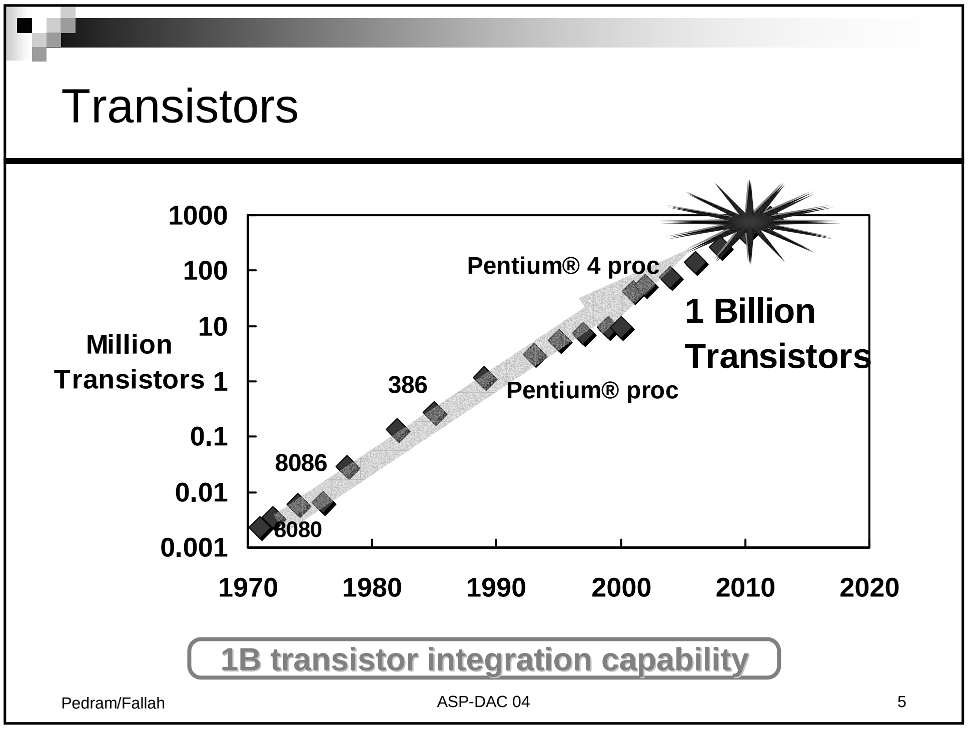
Lecture Outline

- Technology Trends
- Power Dissipation 101
- Leakage Currents
  - □ Subthreshold leakage
  - □ Gate leakage
  - □ Junction leakage
  - □ Gate-induced drain leakage
- Optimizing the Leakage Components
- Summary

Physical Gate Length Trend



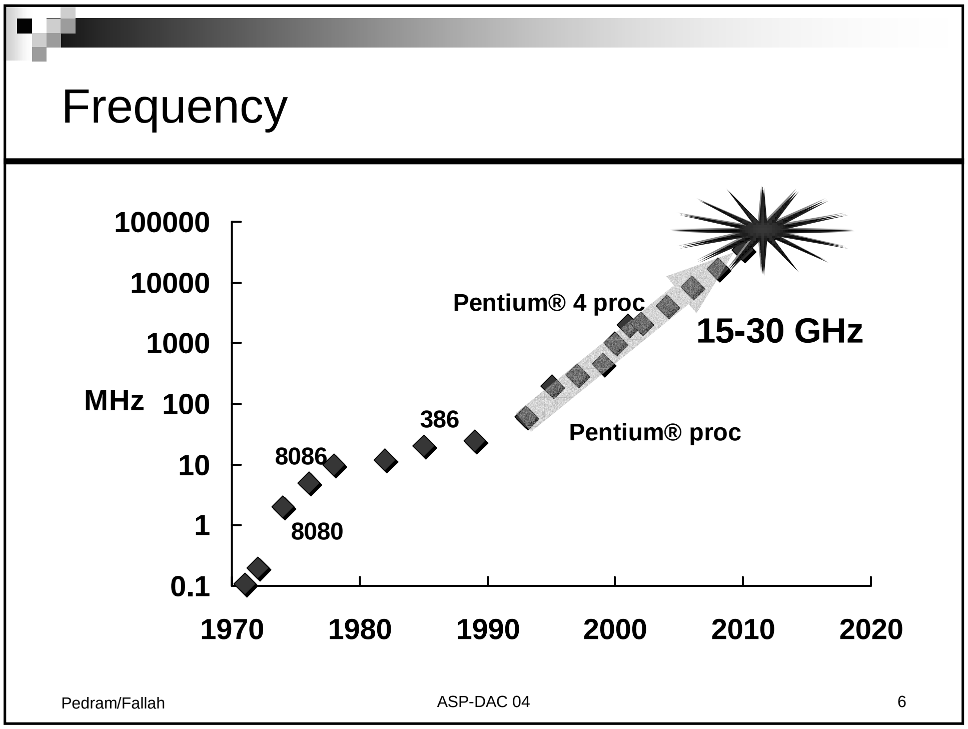
Facilitated by 248, 193, 157, EUV lithography evolution



Transistors

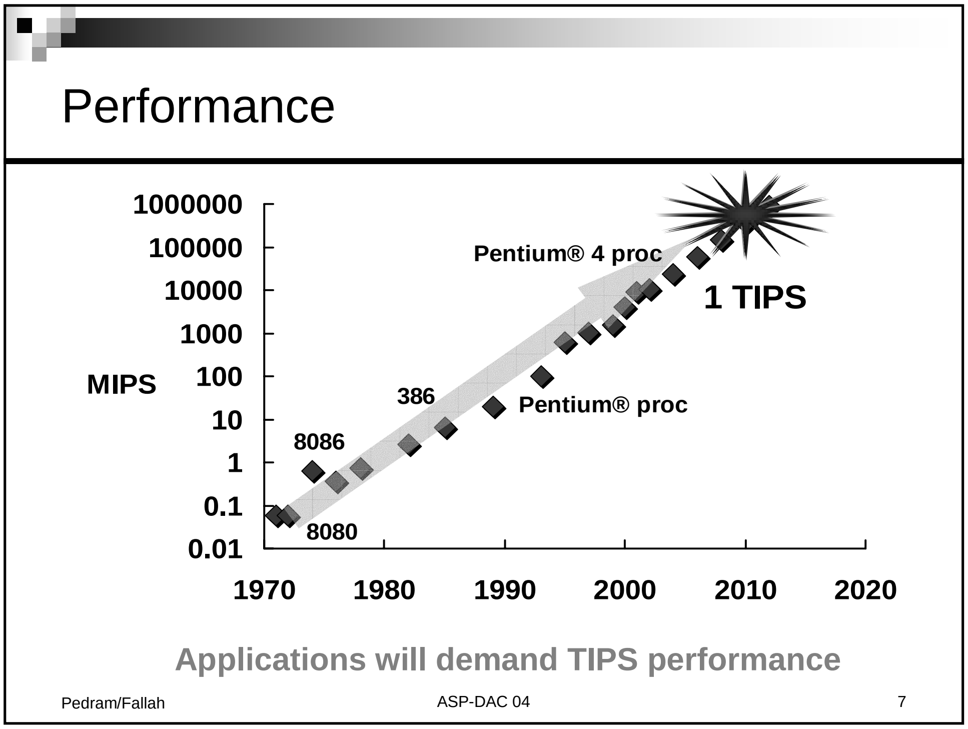
1 Billion Transistors

1B transistor integration capability



Frequency

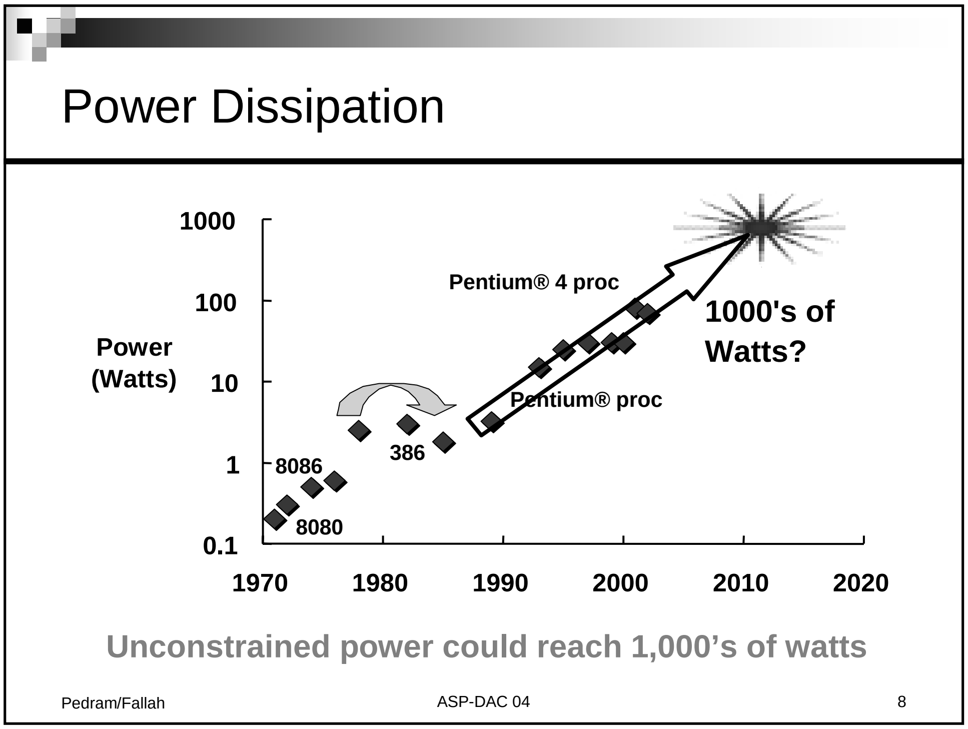
15-30 GHz



Performance

Applications will demand TIPS performance

Pedram/Fallah ASP-DAC 04 7

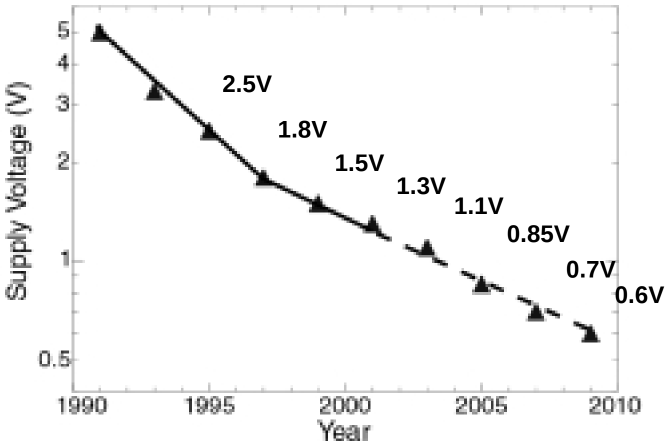


Power Dissipation

Unconstrained power could reach 1,000’s of watts

Pedram/Fallah ASP-DAC 04 8

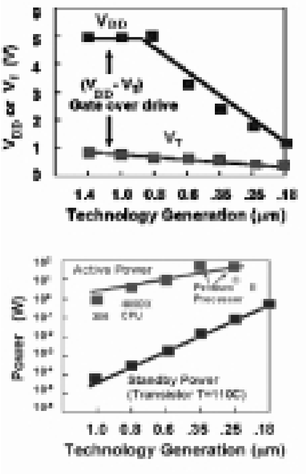
Supply Voltage Scaling



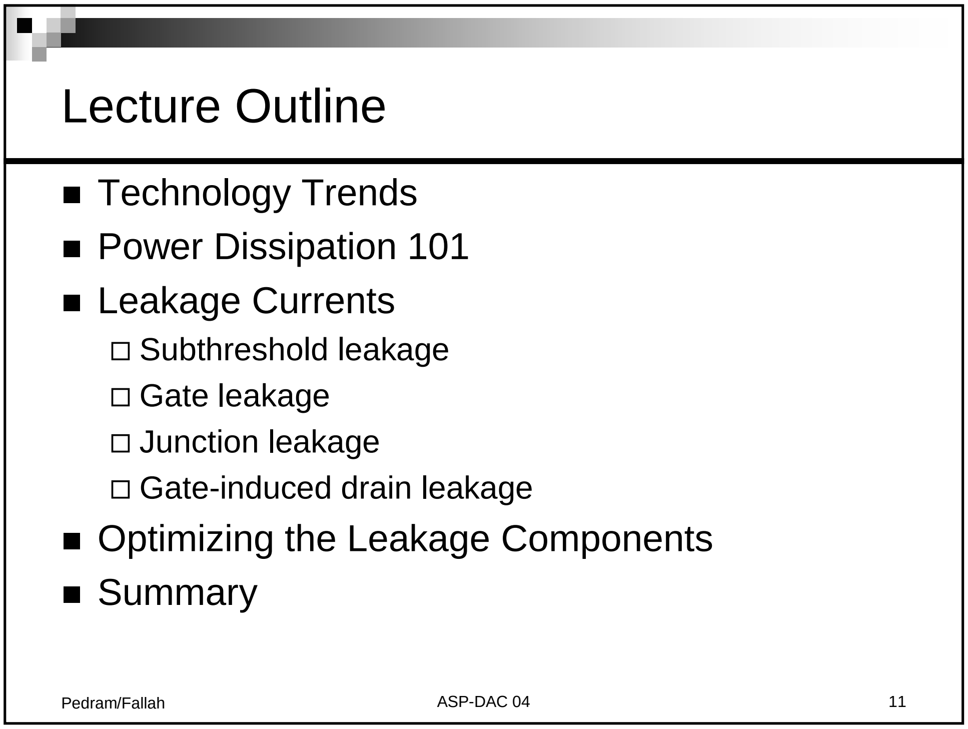
9

CMOS Scaling: An Overview

- Scaling improves:
  - □ Transistor Density & Functionality on a chip
  - □ Speed and frequency of operation ⇒ Higher performance
- Scaling and power dissipation
  - □ Active power ↑ - CV_{DD}^2f
    - Scale V_{DD}
    - Scale Vth ⇒ I_{leak}↑
  - □ Standby (or leakage) power ↑ $V_{DD}I_{leak}$
- Leakage power is catching up with the active power in VDSM CMOS circuits

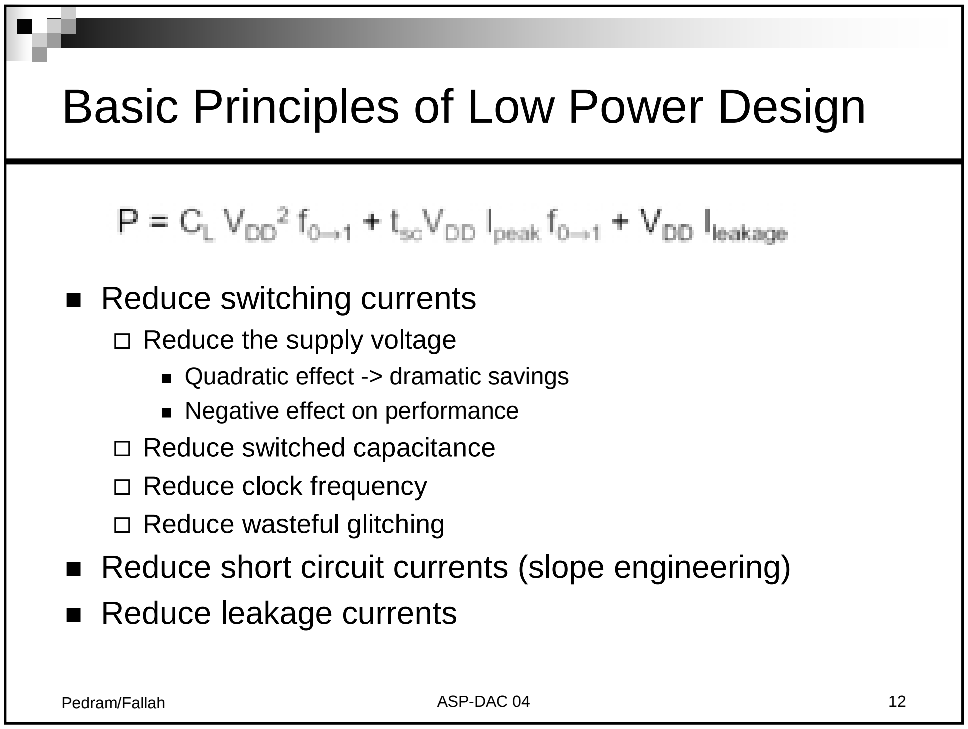


Source: Intel



Lecture Outline

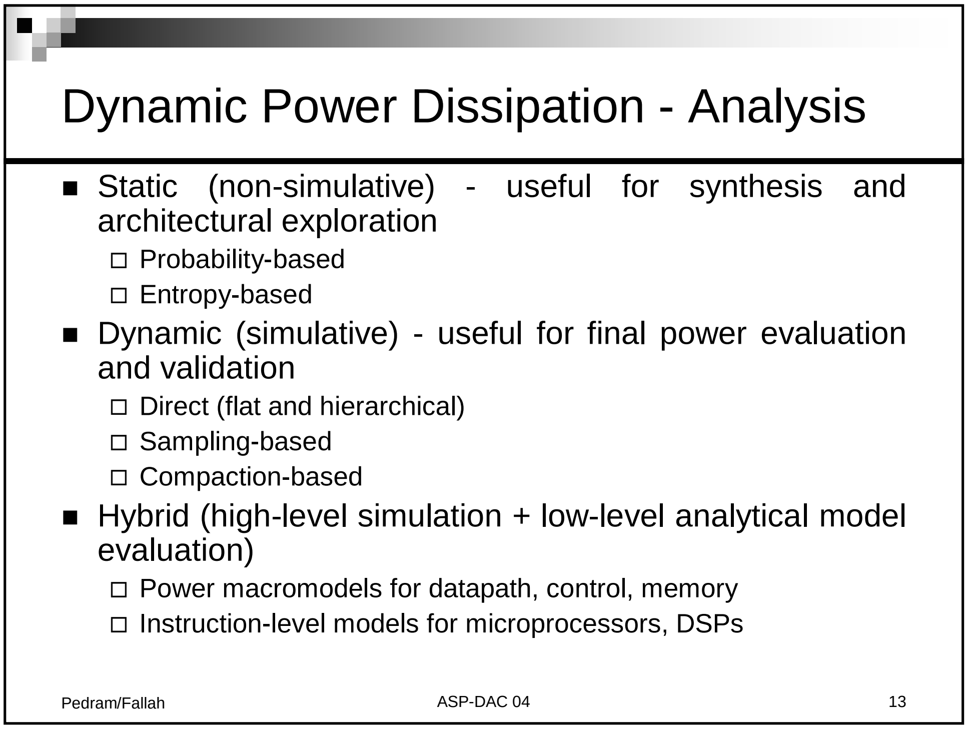
- Technology Trends
- Power Dissipation 101
- Leakage Currents
  - □ Subthreshold leakage
  - □ Gate leakage
  - □ Junction leakage
  - □ Gate-induced drain leakage
- Optimizing the Leakage Components
- Summary



Basic Principles of Low Power Design

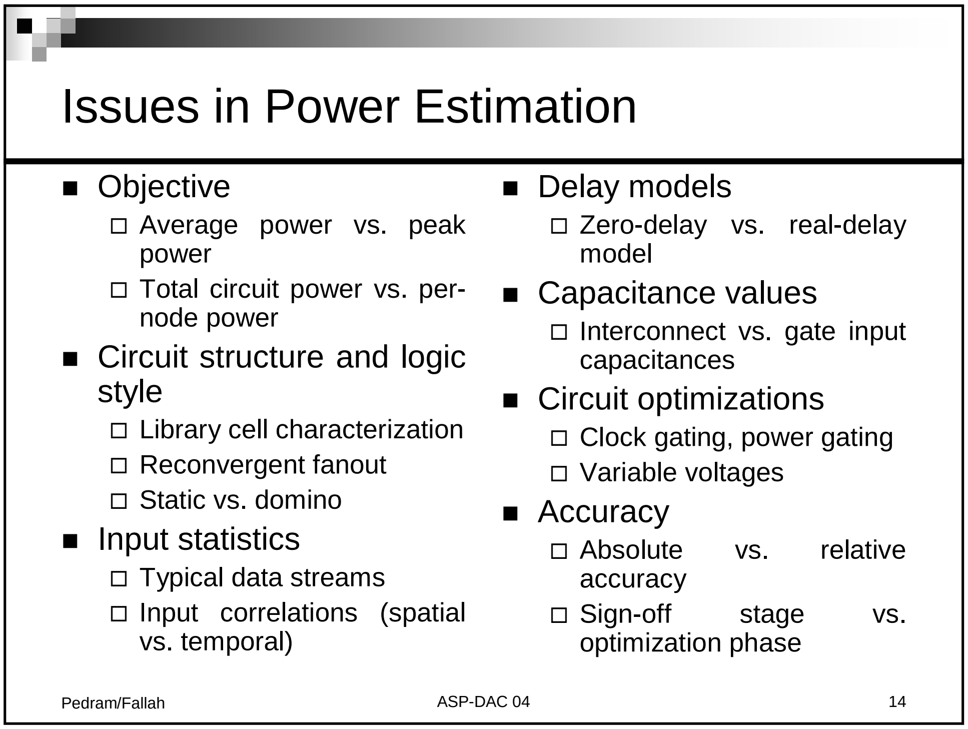
$$P = C_L V_{DD}^2 f_{0\rightarrow 1} + t_{sc} V_{DD} I_{peak} f_{0\rightarrow 1} + V_{DD} I_{leakage}$$

- Reduce switching currents
  - □ Reduce the supply voltage
    - Quadratic effect -> dramatic savings
    - Negative effect on performance
  - □ Reduce switched capacitance
  - □ Reduce clock frequency
  - □ Reduce wasteful glitching
- Reduce short circuit currents (slope engineering)
- Reduce leakage currents



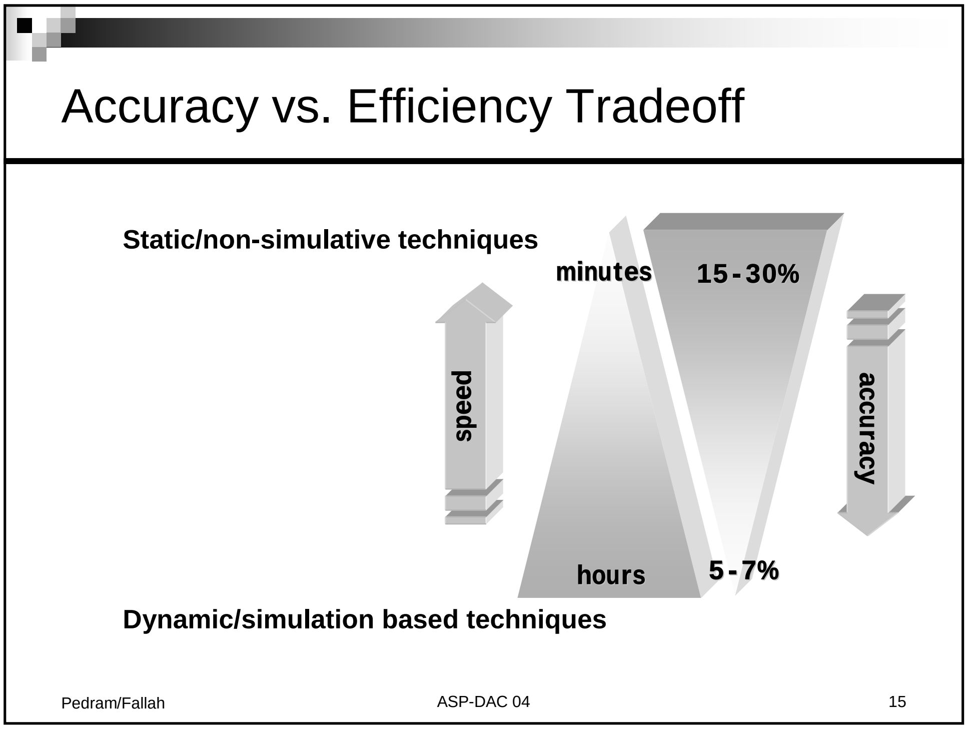
Dynamic Power Dissipation - Analysis

- Static (non-simulative) - useful for synthesis and architectural exploration
  - □ Probability-based
  - □ Entropy-based
- Dynamic (simulative) - useful for final power evaluation and validation
  - □ Direct (flat and hierarchical)
  - □ Sampling-based
  - □ Compaction-based
- Hybrid (high-level simulation + low-level analytical model evaluation)
  - □ Power macromodels for datapath, control, memory
  - □ Instruction-level models for microprocessors, DSPs



Issues in Power Estimation

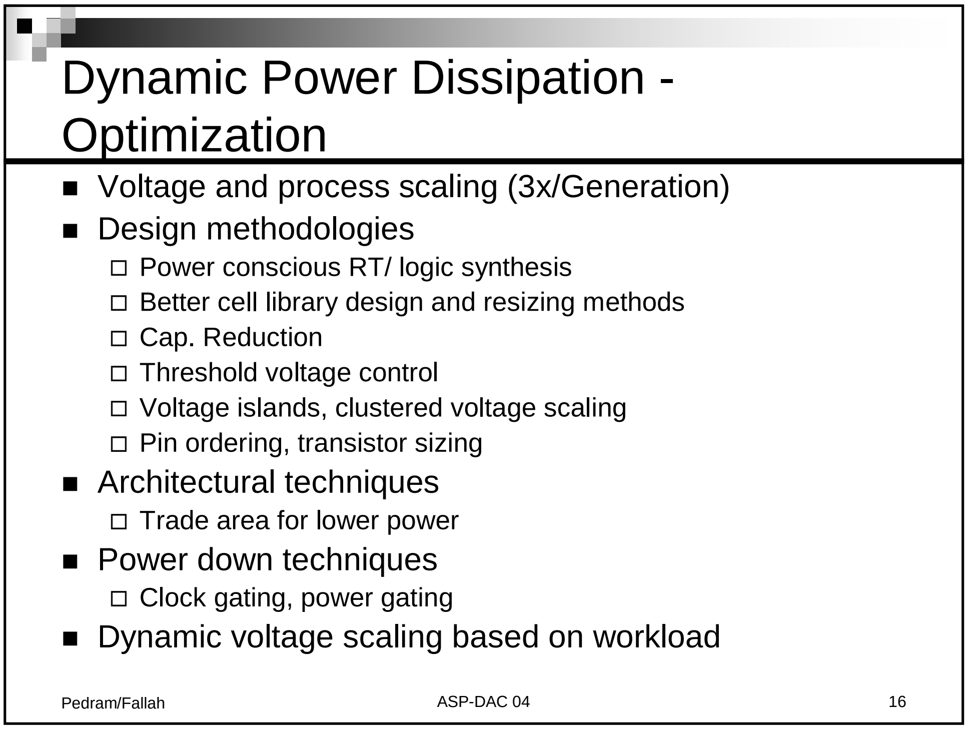
- Objective
  - □ Average power vs. peak power
  - □ Total circuit power vs. per-node power
- Circuit structure and logic style
  - □ Library cell characterization
  - □ Reconvergent fanout
  - □ Static vs. domino
- Input statistics
  - □ Typical data streams
  - □ Input correlations (spatial vs. temporal)
- Delay models
  - □ Zero-delay vs. real-delay model
- Capacitance values
  - □ Interconnect vs. gate input capacitances
- Circuit optimizations
  - □ Clock gating, power gating
  - □ Variable voltages
- Accuracy
  - □ Absolute vs. relative accuracy
  - □ Sign-off stage vs. optimization phase



Accuracy vs. Efficiency Tradeoff

Static/non-simulative techniques

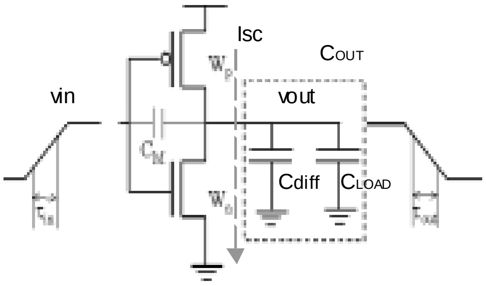
Dynamic/simulation based techniques

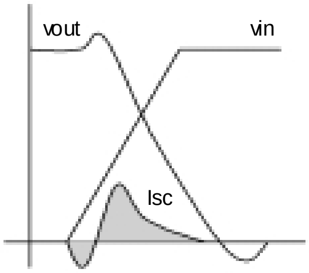


Dynamic Power Dissipation - Optimization

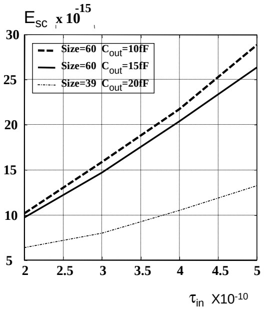
- Voltage and process scaling (3x/Generation)
- Design methodologies
  - □ Power conscious RT/ logic synthesis
  - □ Better cell library design and resizing methods
  - □ Cap. Reduction
  - □ Threshold voltage control
  - □ Voltage islands, clustered voltage scaling
  - □ Pin ordering, transistor sizing
- Architectural techniques
  - □ Trade area for lower power
- Power down techniques
  - □ Clock gating, power gating
- Dynamic voltage scaling based on workload

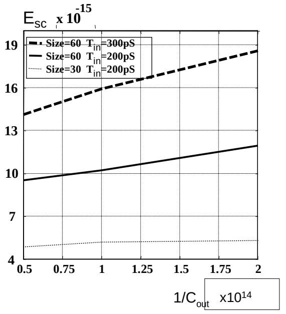
Short Circuit Power Dissipation

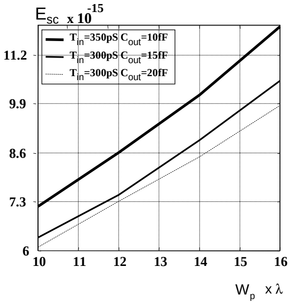




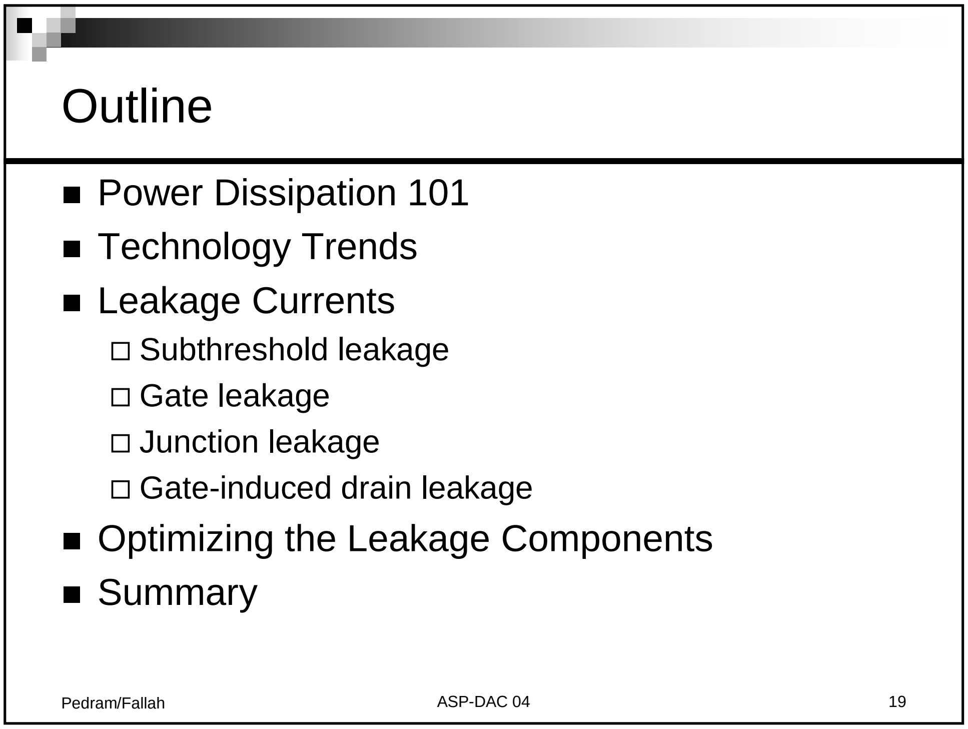
Short Circuit Power Dissipation (Cnt'd)





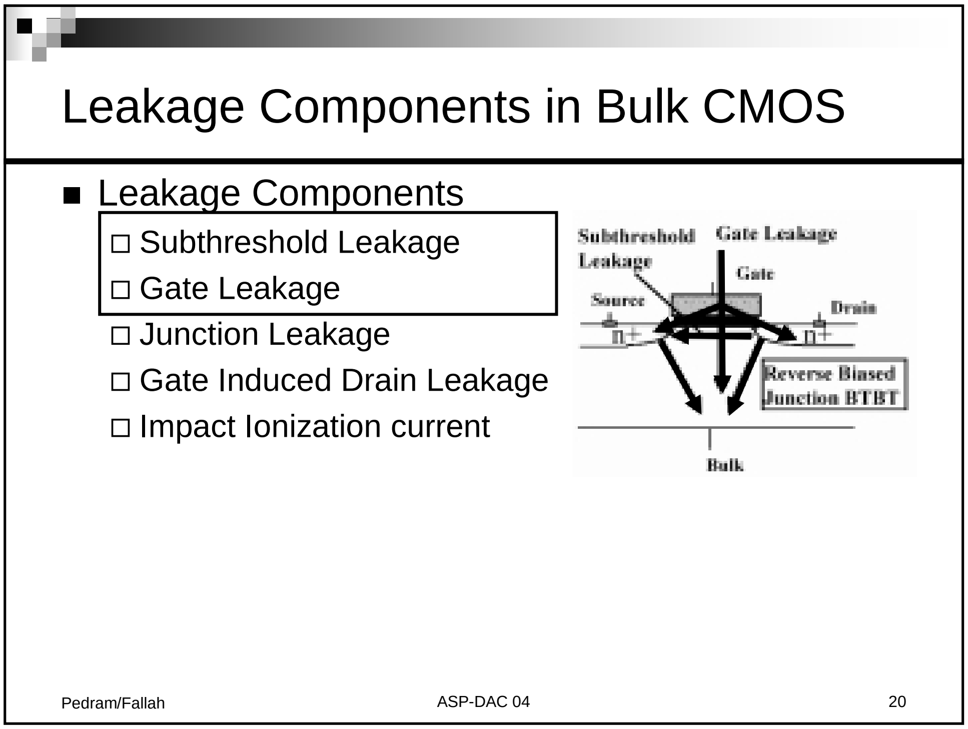


$$E_{sc}\left(\tau_{in}, W, C_{out}\right) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} m_{ijk} \frac{W^i \tau_{in}{}^j}{C_{out}{}^k} V_{DD}$$



Outline

- Power Dissipation 101
- Technology Trends
- Leakage Currents
  - □ Subthreshold leakage
  - □ Gate leakage
  - □ Junction leakage
  - □ Gate-induced drain leakage
- Optimizing the Leakage Components
- Summary



Leakage Components in Bulk CMOS

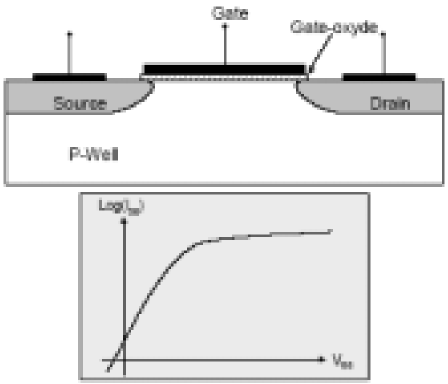
- Leakage Components
  - □ Subthreshold Leakage
  - □ Gate Leakage
  - □ Junction Leakage
  - □ Gate Induced Drain Leakage
  - □ Impact Ionization current

Subthreshold Leakage
Gate Leakage
Gate
Source
Drain
n+
n+
Reverse Biased Junction BTBT
Bulk

Scaling Effect

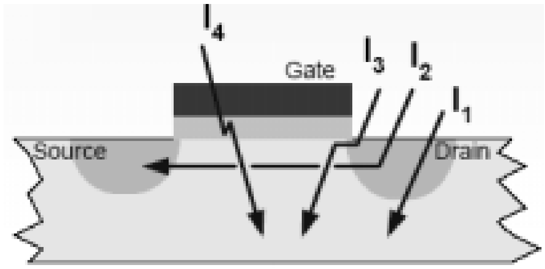
- Scaling increases all leakage components
- Leakage components are dependent on each other through the device geometry and doping profile – “Trade-off” is necessary
- Knowledge of each leakage component is necessary for process engineering and circuit/logic design

Standard CMOS n-channel Transistor Model



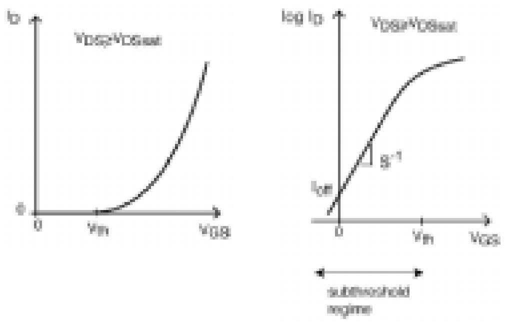
Subthreshold Leakage

- Subthreshold current (I_2)



Subthreshold Regime

Transfer characteristics of MOSFET for V_{GS} near V_t:



Experimental observation: $I_D \propto e^{\frac{q(V_{GS}-V_{th})}{nkT}}$

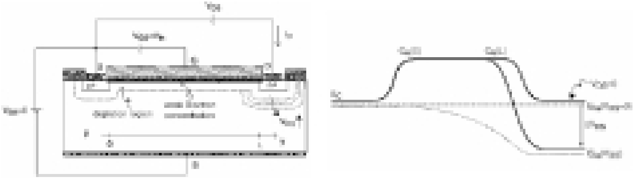
Two Key Figures of Subthreshold Regime

- The inverse subthreshold slope, S, is equal to the voltage required to increase I_D by 10X:

$$S = \frac{nkT}{q}\ln 10$$

  - □ If n = 1, S = 60 mV/dec at 300 K
  - □ We want S to be small to shut off the MOSFET quickly
  - □ In well designed devices, S is 70 - 90 mV/dec at 300 K
- Off current, I_{off} :
  - □ $I_{off} = I_D(V_{GS} = 0)$
  - □ For logic CMOS, we want I_{off} to be in the nA range
  - □ I_{off} set by S and V_{th}

Subthreshold Current



- In the subthreshold regime:
  - □ no longitudinal field in channel
  - □ energy band diagram looks like the base of bipolar transistor
  - □ electrons flow from source to drain by **diffusion**

$$I_{sub} = \frac{w}{L}\mu_e v_T^2 C_{sth} \exp\left(\frac{V_{GS} - V_{th}}{nv_T}\right)\left(1 - \exp\left(\frac{-V_{DS}}{nv_T}\right)\right)$$

Some Important Effects

- Drain-Induced Barrier Lowering
  - □ An effect called drain-induced barrier lowering (DIBL) takes place when a high-drain voltage is applied to a short-channel device. The source injects carriers into the channel surface (independent of the gate voltage)
- Short channel-length effect and V_t rolloff
  - □ Shorter channel length results in lower threshold voltages and increases subthreshold leakage
- Body effect
  - □ When the well-to-source junction of a MOSFET is reverse biased (i.e., V_{BS} is reduced) , there is a body effect that increases the threshold voltage and decreases subthreshold leakage
- Narrow-Width effect
  - □ Narrow width of the transistor can also modulate the threshold voltage and the subthreshold current
- Temperature effect
  - □ As temperature increases, subthreshold leakage is also increased

Modeling Subthreshold Current (I_{sub})

- Increases exponentially with reduction in V_{th}

$$I_{sub} = \frac{w}{L}\mu_e v_T^2 C_{sth} \exp\left(\frac{V_{GS}-V_{th}}{nv_T}\right)\left(1-\exp\left(\frac{-V_{DS}}{nv_T}\right)\right)$$

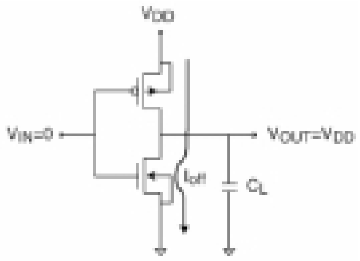
- Modulation of Vth in a Short Channel Transistor
  - □ L ↓ ⇒ V_{th} ↓: “V_{th} Rolloff”
  - □ V_{DS} ↑ ⇒ V_{th} ↓:”Drain Induced Barrier Lowering”
  - □ V_{SB} ↑ ⇒ V_{th} ↑: “Body Effect”

Modeling Subthreshold Current (continued)

- If $V_{DS} = 0 \Rightarrow I_{sub} = 0$
- If $V_{DS} > kT/q \Rightarrow I_{sub} \approx \frac{w}{L}\mu_e v_T^2 C_{sth} \exp\left(\frac{V_{GS} - V_{th}}{n v_T}\right)$
- With $n = 1 + \frac{\gamma}{2\sqrt{2\Phi_f}} = 1 + \frac{C_{sth}}{C_{ox}}$
- Key dependencies of the subthreshold slope:
  - □ $T_{ox} \downarrow \Rightarrow C_{ox} \uparrow \Rightarrow n \downarrow \Rightarrow$ sharper subthreshold
  - □ $N_A \uparrow \Rightarrow C_{sth} \uparrow \Rightarrow n \uparrow \Rightarrow$ softer subthreshold
  - □ $V_{SB} \uparrow \Rightarrow C_{sth} \downarrow \Rightarrow n \downarrow \Rightarrow$ sharper subthreshold
  - □ $T \uparrow \Rightarrow$ softer subthreshold
- *n* reflects electrostatic competition between the top gate and the body ("bottom gate")

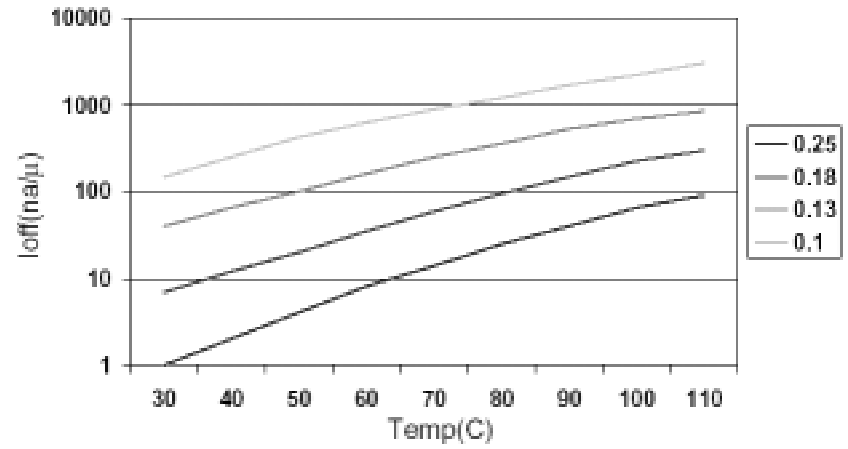
Importance of the subthreshold regime

- Determines the off current:

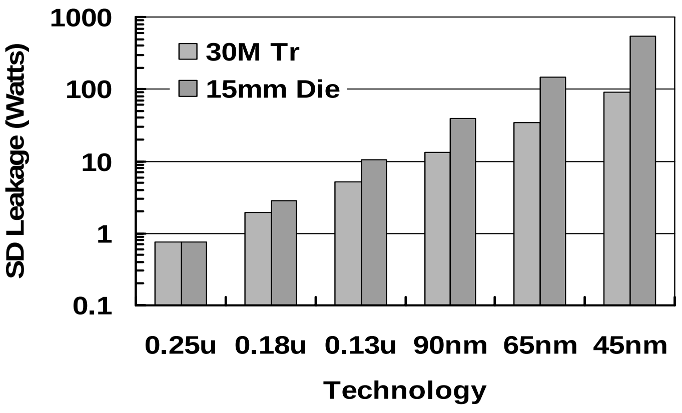


$$I_{off} = I_{sub}(V_{GS} = 0) = \frac{W}{L}\mu_e v_T^2 C_{sth} \exp\left(-\frac{V_{th}}{n v_T}\right)$$

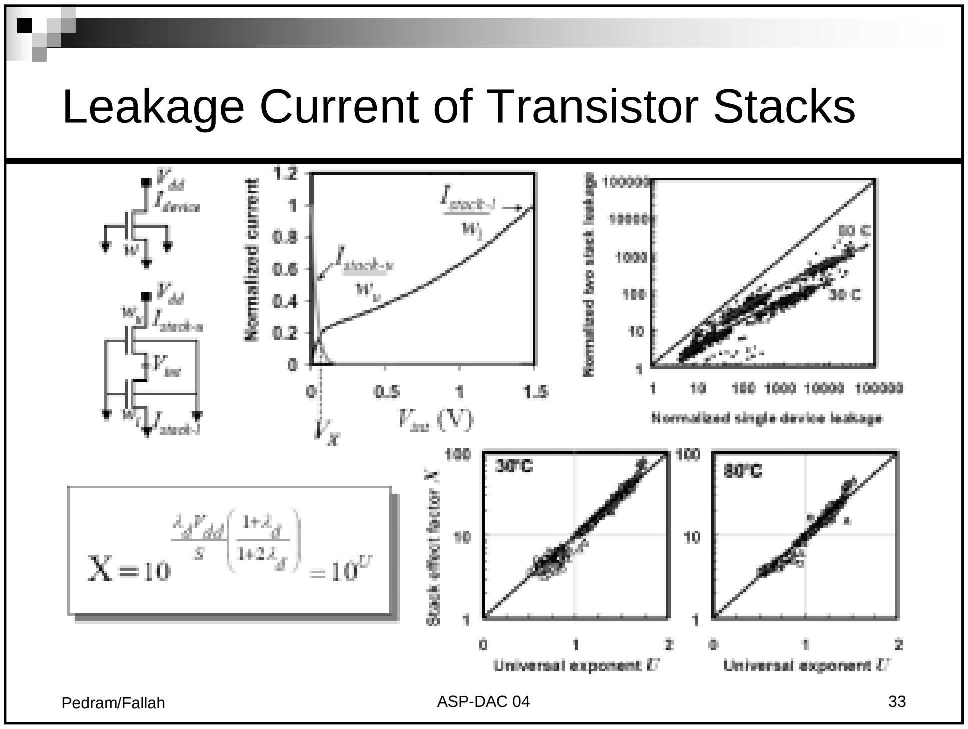
Projected Subthreshold Leakage Currents



Subthreshold Leakage Power



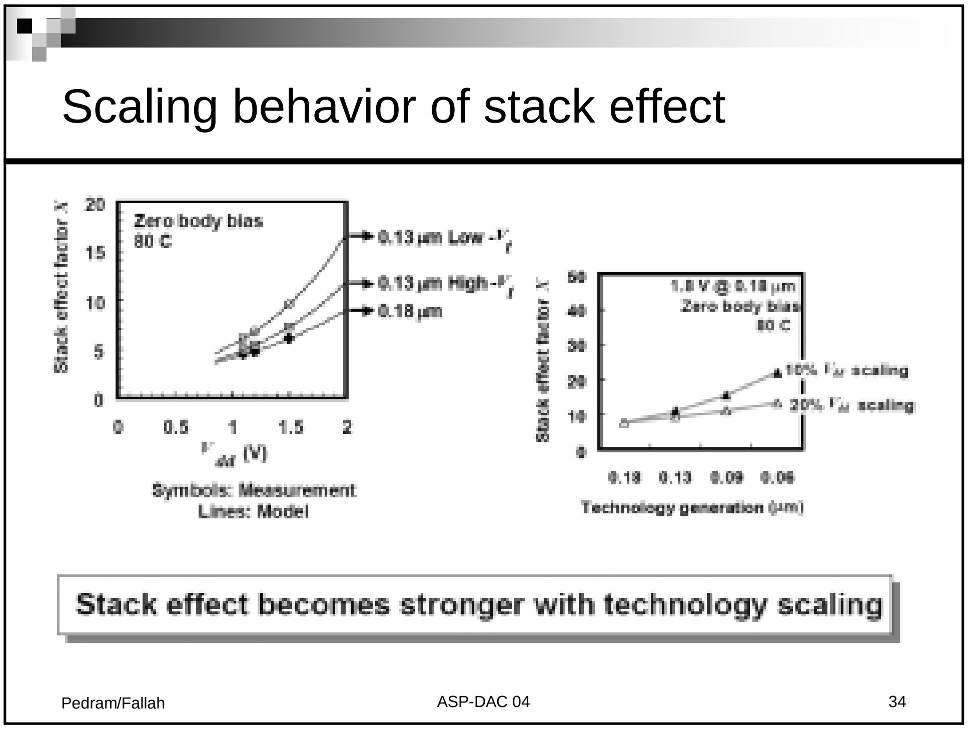
Excessive sub-threshold leakage power



Leakage Current of Transistor Stacks

$$X = 10^{\frac{\lambda_d V_{dd}}{S}\left(\frac{1+\lambda_d}{1+2\lambda_d}\right)} = 10^{U}$$

Pedram/Fallah ASP-DAC 04 33



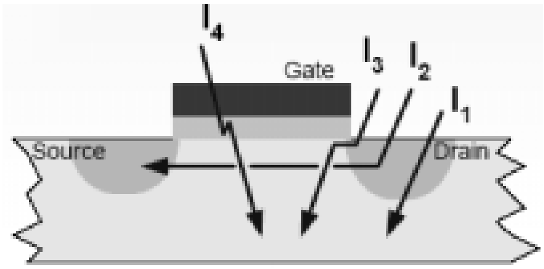
Scaling behavior of stack effect

Stack effect becomes stronger with technology scaling

Pedram/Fallah ASP-DAC 04 34

Gate Leakage

- Gate current (I_4)

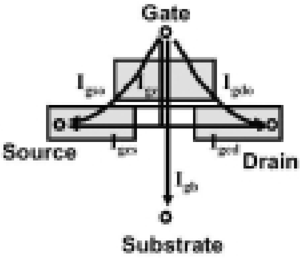


Gate Oxide Tunneling

- *Gate oxide tunneling* of electrons that can result in leakage when there is a high electric field across a thin gate oxide layer. Electrons may tunnel into the conduction band of the oxide layer; this is called Fowler-Nordheim tunneling
- In oxide layers less than 3–4 nm thick, there can also be direct tunneling through the silicon oxide layer. Mechanisms for direct tunneling include electron tunneling in the conduction band (ECB), electron tunneling in the valence band (EVB), and hole tunneling in the valence band (HVB)

Gate Current (I_{gate})

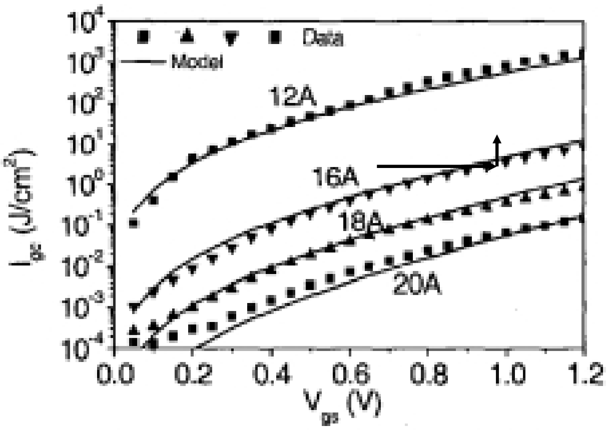
- Direct tunneling of electron through gate oxide is the dominant source
- Depends exponentially on the oxide thickness and the V_{dd} [BSIM 4]



$$J_{DT} = A_g \left(V_{ox}/T_{ox}\right)^2 \exp\left(\frac{-B_g\left(1-\left(1-V_{ox}/\Phi_{ox}\right)^{3/2}\right)}{V_{ox}/T_{ox}}\right)$$

- Gate Leakage Components [Cao et al 2000, BSIM 4]
  - □ Gate to S/D overlap region (I_{gso} & I_{gdo})
  - □ Gate to channel (I_{gc}) = to Source (I_{gcs}) + to Drain (I_{gcd})
  - □ Gate to substrate (I_{gb})

Gate Leakage



40% ΔVgs

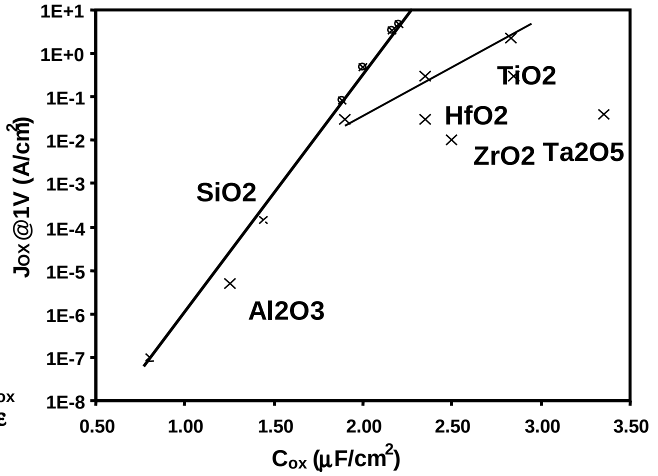
➔

~5X I_G

BSIM4 Gate Leakage Model Including Source-Drain Partition

K. M. Cao, W.-C. Lee*, W. Liu, X. Jin, P. Su, S. K. H. Fung#, J. X. An*, B. Yu*, and C. Hu
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA
Tel: 510-643-2639 Fax: 510-643-2636 Email: [illegible]
*Present addr: Intel Corp., Portland, OR, #IBM Corp., Fishkill, NY, ^AMD Corp., Sunnyvale, CA

High K Reduces Gate Leakage



$E_{ox}=V_{GS}/t_{ox}$
$E=qN_Ax/\varepsilon$

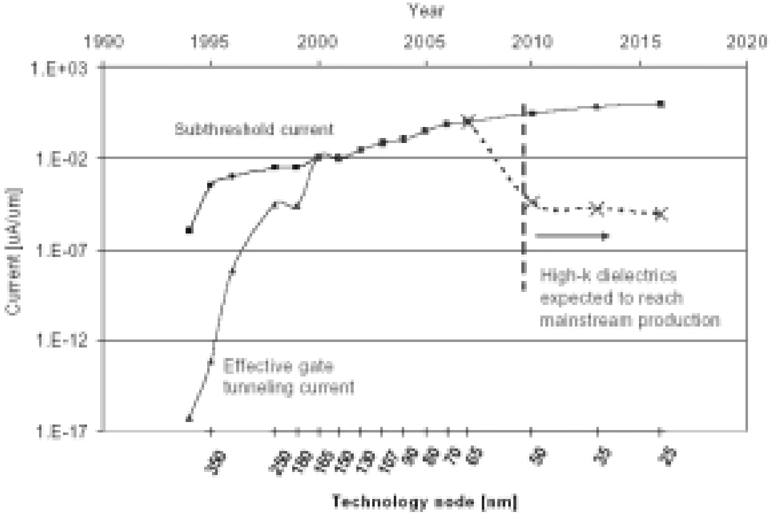
Gate-Oxide Leakage Current

- Aggressive scaling of the gate oxide layer thickness (T_{ox})
  - □ Necessary to maintain drive current with scaling
  - □ 90nm technology: 12~16 Å T_{ox}
  - □ Leads to significant gate tunneling leakage current (I_{gate})
- I_{gate}: A super exponential function of T_{ox}:
  - □ 30% reduction of T_{ox} (20 → 14 Å) ⇒ 1000x rise in I_{gate}

Scaling Trends

- Gate leakage is predicted to increase at a rate of more than 500X per technology generation
- Sub-threshold leakage increases by around 5X for each technology generation
- Gate leakage power, which was almost non-existent in previous technology generations, expected to contribute more than 15% to the total power consumption in a 2004 technology generation.

Gate-Oxide Leakage Current



Junction Leakage

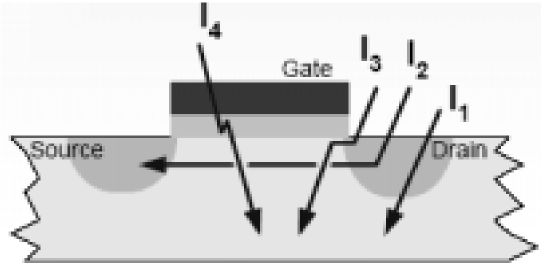
- *Junction leakage* that results from minority carrier diffusion and drift near the edge of depletion regions, and also from generation of electron hole pairs in the depletion regions of reverse-bias junctions. When both n regions and p regions are heavily doped, as is the case for some advanced MOSFETs, there is also junction leakage due to band-to-band tunneling (BTBT)

Diode Reverse Biased Leakage

- Diode reverse bias current (I_1)

$$I_1 = I_s\left(1 - e^{-\frac{V_{DB}}{V_{th}}}\right)$$

where V_{DB} is drain to bulk (substrate) voltage



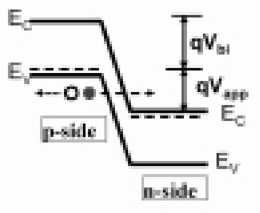
Modeling Source/Drain Junction BTBT

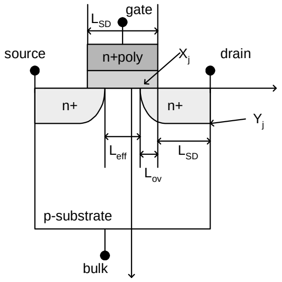
- Electron tunneling from Valence Band of the p-side to the Conduction Band of the n-side
- Jn. BTBT Current density
  - □ Junction field (ξ), junction voltage (V_{app}), band-gap (E_g).

$$J_{b-b} = A\frac{\xi V_{app}}{E_g^{1/2}}\exp\left(-B\frac{E_g^{3/2}}{\xi}\right)$$

$$A = \frac{\sqrt{2m^*}q^3}{4\pi^3\hbar^2}, \, and \ B = \frac{4\sqrt{2m^*}}{3q\hbar}$$

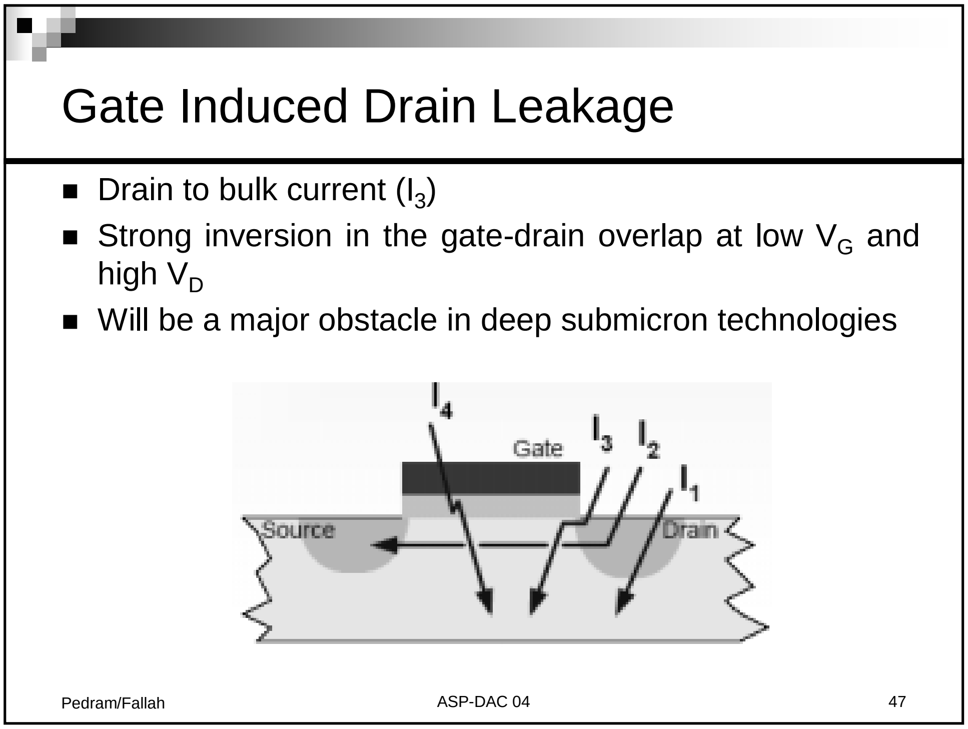
- For Jn. BTBT: (V_{bi} + V_{app}) > E_g
- Total Jn. BTBT = Source Jn. BTBT + Drain Jn. BTBT





Gate Induced Drain Leakage

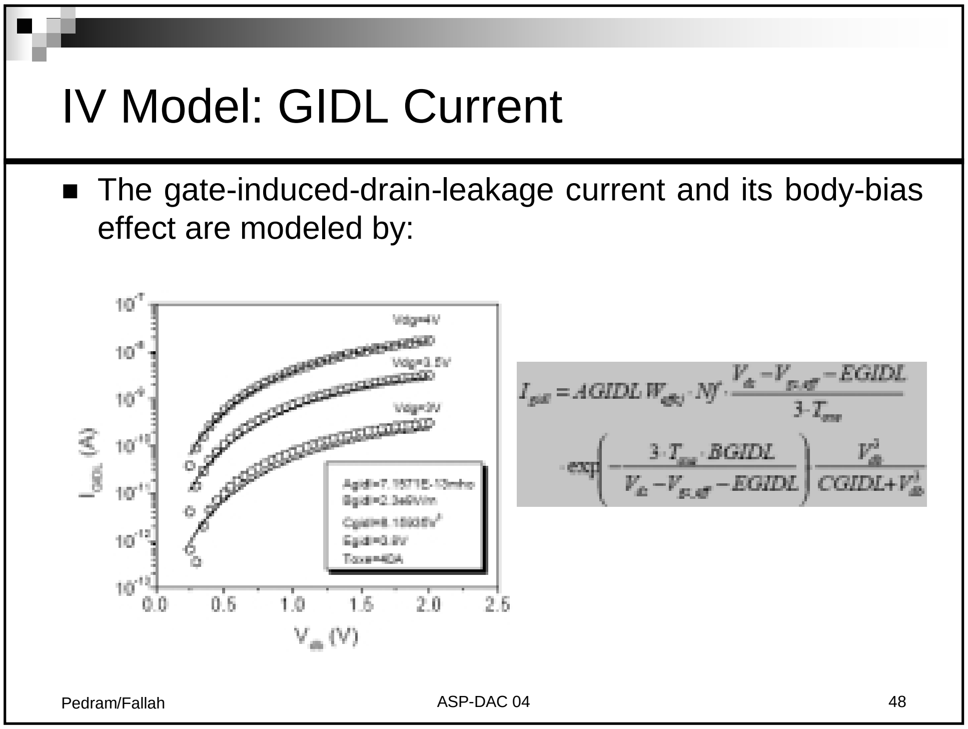
- Gate-induced drain leakage (GIDL) is caused by high field effect in the drain junction of MOS transistors.
  - □ In a negative-channel metal-oxide-semiconductor (NMOS) transistor, when the gate is biased to form accumulation layer in the silicon surface under the gate, the silicon surface has almost the same potential as the p-type substrate, and the surface acts like a p region more heavily doped than the substrate
  - □ When the gate is at zero or negative voltage and the drain is at the supply voltage level, there can be a dramatic increase of effects like avalanche multiplication and band-to-band tunneling. Minority carriers underneath the gate are swept to the substrate, completing the GIDL path
  - □ Thinner oxide and higher supply voltage increase GIDL



Gate Induced Drain Leakage

- Drain to bulk current (I_3)
- Strong inversion in the gate-drain overlap at low V_G and high V_D
- Will be a major obstacle in deep submicron technologies

I_4
Gate
I_3
I_2
I_1
Source
Drain

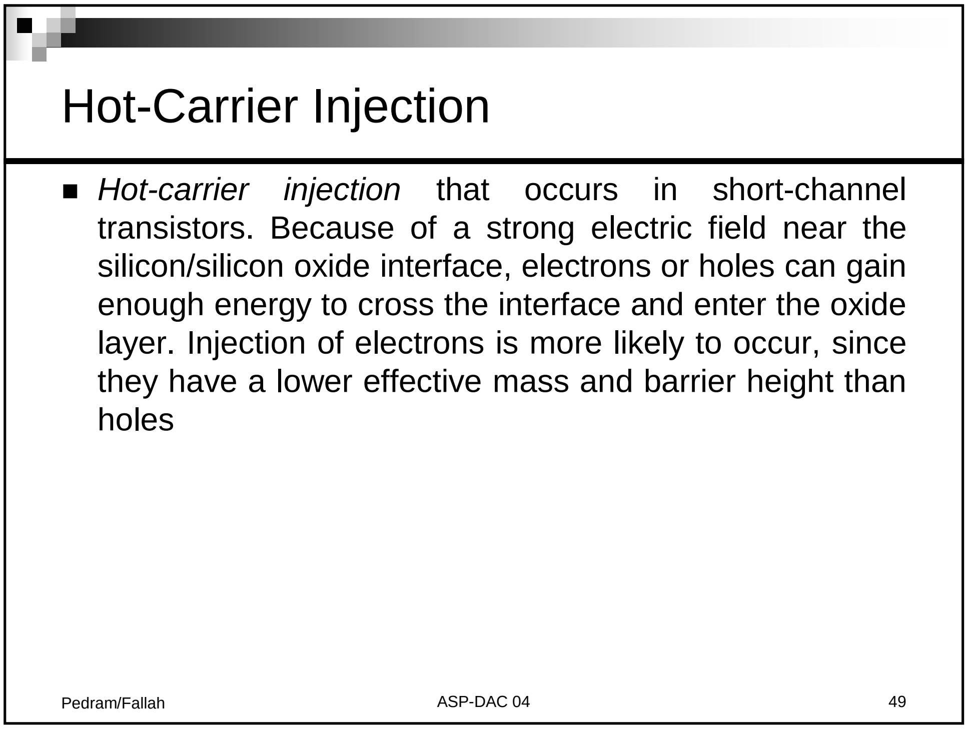


IV Model: GIDL Current

- The gate-induced-drain-leakage current and its body-bias effect are modeled by:

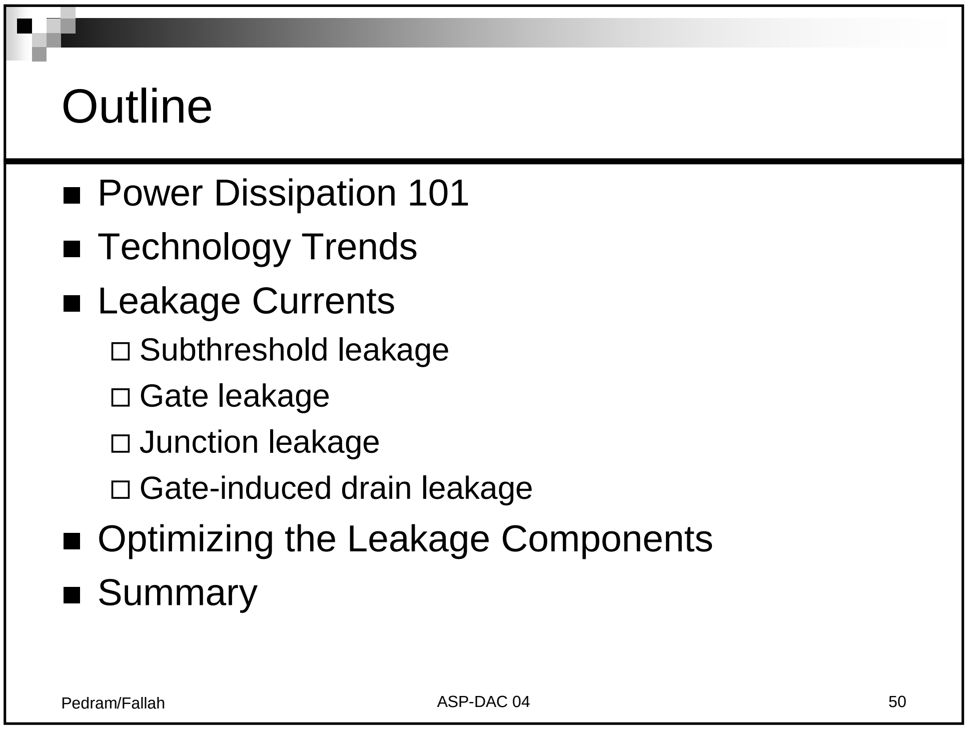
$$I_{gidl} = AGIDL \cdot W_{eff} \cdot Nf \cdot \frac{V_{ds} - V_{gs,eff} - EGIDL}{3 \cdot T_{oxe}} \cdot \exp\left(-\frac{3 \cdot T_{oxe} \cdot BGIDL}{V_{ds} - V_{gs,eff} - EGIDL}\right) \cdot \frac{V_{db}^3}{CGIDL + V_{db}^3}$$

Vdg=4V
Vdg=3.5V
Vdg=3V
Agidl=7.1571E-10mho
Bgidl=2.3e9V/m
Cgidl=8.1593V^3
Egidl=0.8V
Toxe=40A
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-13}
I_{GIDL} (A)
0.0
0.5
1.0
1.5
2.0
2.5
V_{db} (V)



Hot-Carrier Injection

- *Hot-carrier injection* that occurs in short-channel transistors. Because of a strong electric field near the silicon/silicon oxide interface, electrons or holes can gain enough energy to cross the interface and enter the oxide layer. Injection of electrons is more likely to occur, since they have a lower effective mass and barrier height than holes



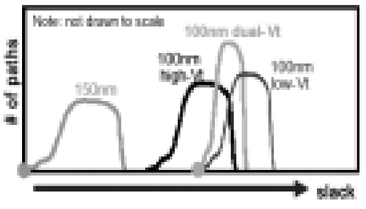
Outline

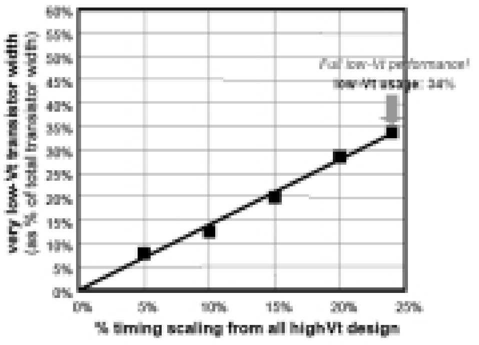
- Power Dissipation 101
- Technology Trends
- Leakage Currents
  - □ Subthreshold leakage
  - □ Gate leakage
  - □ Junction leakage
  - □ Gate-induced drain leakage
- Optimizing the Leakage Components
- Summary

Dual Threshold Voltages

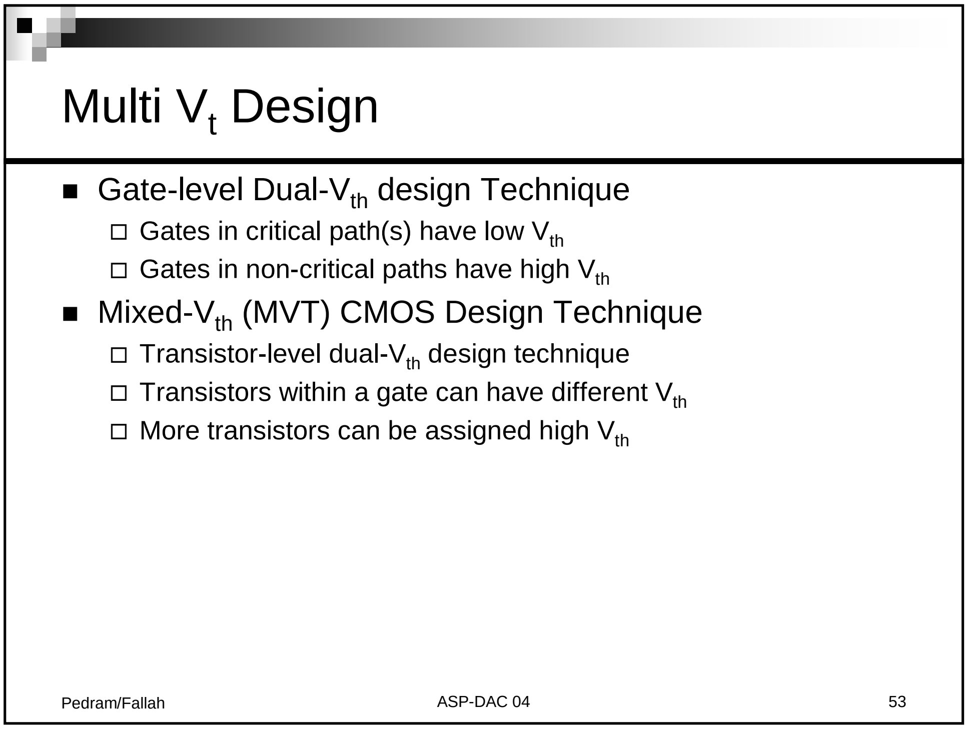
- Use two V_T's (e.g., 0.6V and 0.3V for V_{DD} = 2.5V)
  - □ Use the lower threshold for gates on critical path
  - □ Use the higher threshold for gates off the critical path
- Improves performance without an increase in power
- Cons
  - □ Increased fabrication complexity
  - □ Increased design time
  - □ Beware of increased leakage in low V_T portion of the circuit – could end up with increased power!

Dual-V_t design for leakage control



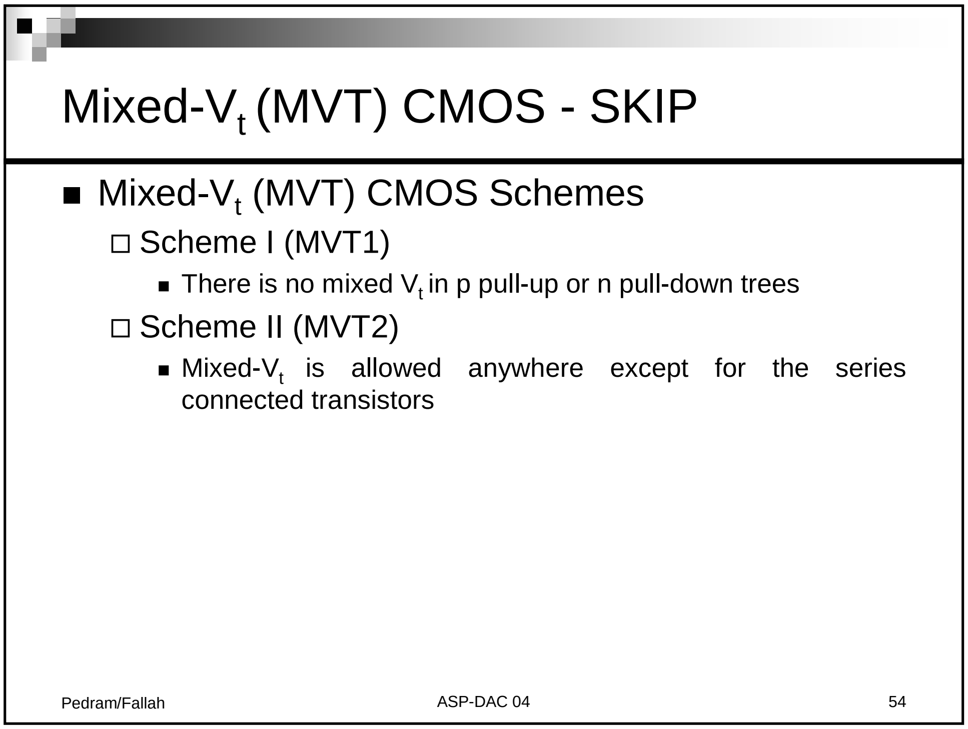


Active & standby leakage 3X smaller, no performance loss



Multi V_t Design

- Gate-level Dual-V_{th} design Technique
  - Gates in critical path(s) have low V_{th}
  - Gates in non-critical paths have high V_{th}
- Mixed-V_{th} (MVT) CMOS Design Technique
  - Transistor-level dual-V_{th} design technique
  - Transistors within a gate can have different V_{th}
  - More transistors can be assigned high V_{th}

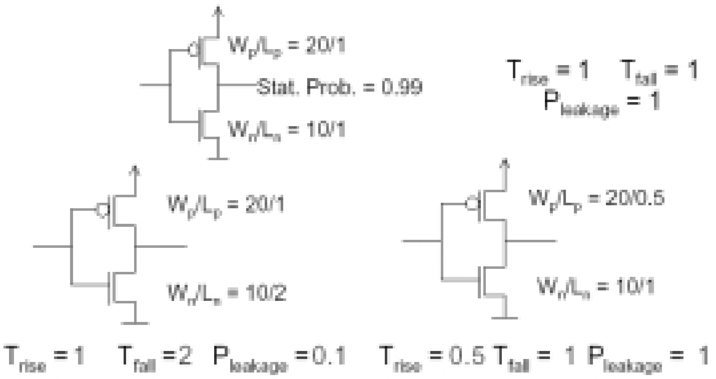


Mixed-V_t (MVT) CMOS - SKIP

- Mixed-V_t (MVT) CMOS Schemes
  - Scheme I (MVT1)
    - There is no mixed V_t in p pull-up or n pull-down trees
  - Scheme II (MVT2)
    - Mixed-V_t is allowed anywhere except for the series connected transistors

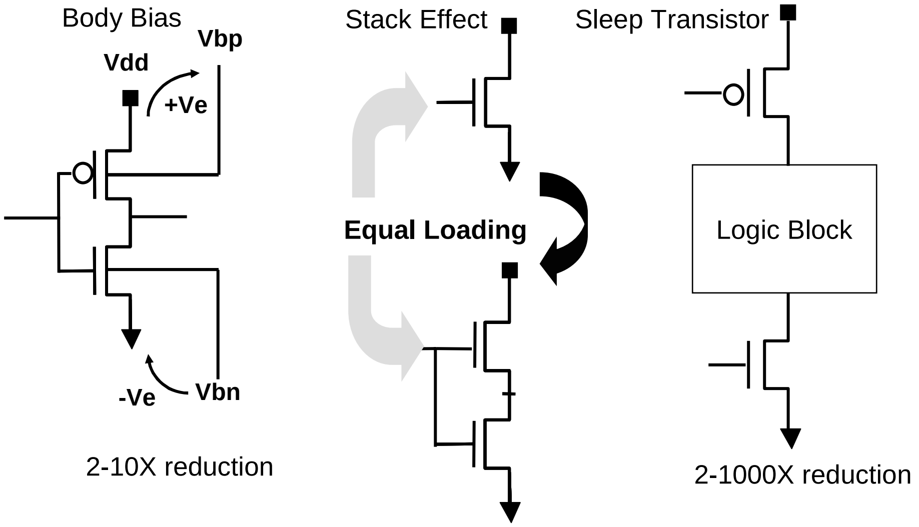
Transistor Sizing for Leakage

- Leakage power depends on logic state



$T_{rise} = 1$ $T_{fall} = 2$ $P_{leakage} = 0.1$ $T_{rise} = 0.5$ $T_{fall} = 1$ $P_{leakage} = 1$

Leakage Control



Body Bias

Vbp

Vdd

+Ve

-Ve

Vbn

2-10X reduction

Stack Effect

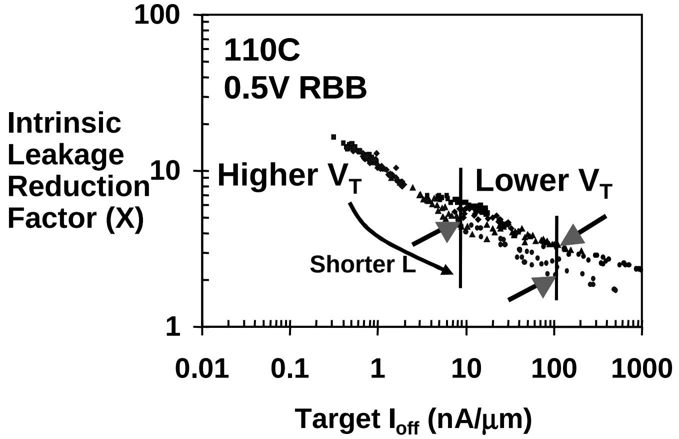
Equal Loading

Sleep Transistor

Logic Block

2-1000X reduction

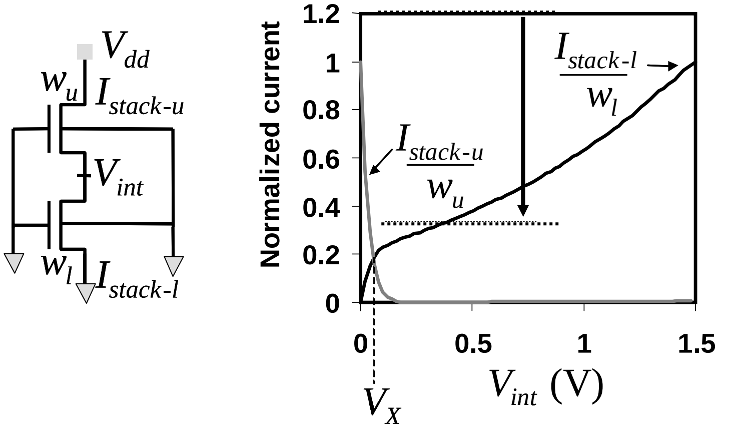
Effectiveness of RBB



RBB less effective at shorter L and lower V_T

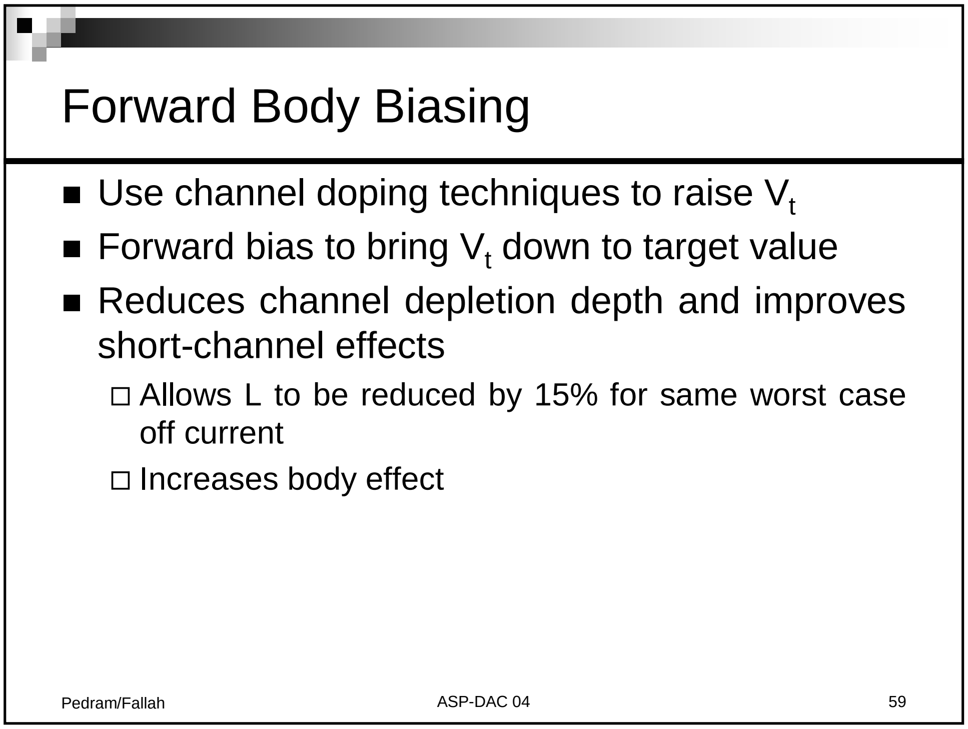
* A. Keshavarzi et. al., 1999 & 2001 International Symp. Low Power Electronics & Design (ISLPED)

SD Leakage of Stacks



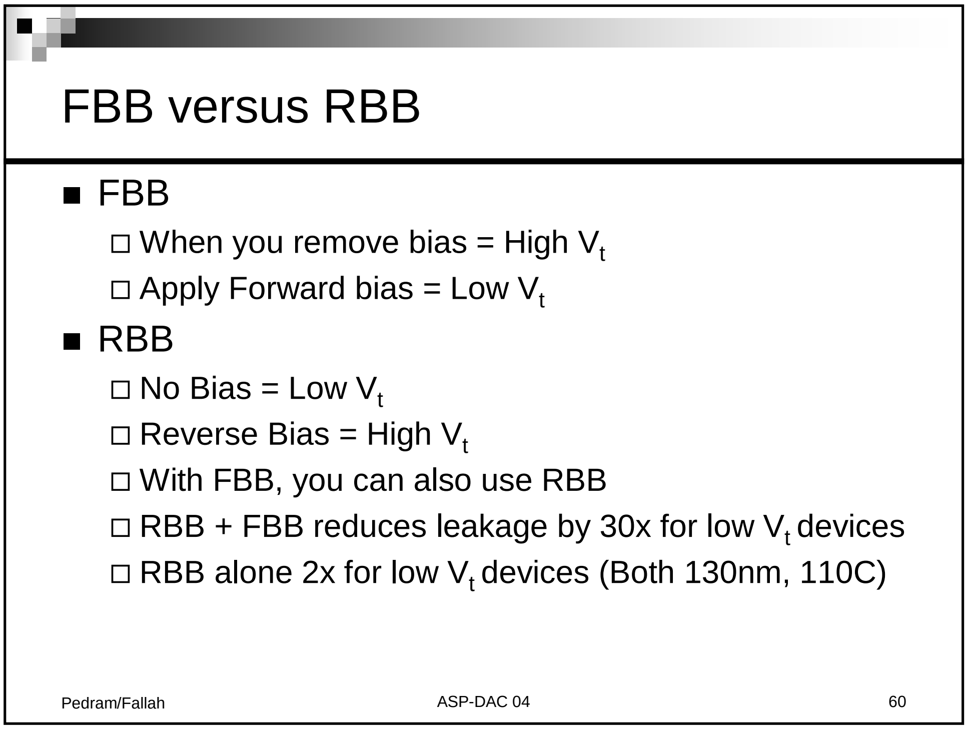
Stack leakage is ~5-10X smaller

* S. Narendra et. al., 2001 International Symp. Low Power Electronics & Design (ISLPED)



Forward Body Biasing

- Use channel doping techniques to raise V_t
- Forward bias to bring V_t down to target value
- Reduces channel depletion depth and improves short-channel effects
  - Allows L to be reduced by 15% for same worst case off current
  - Increases body effect

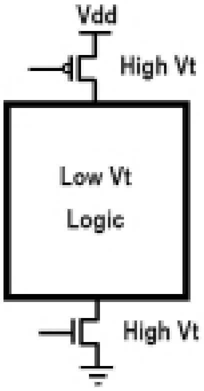


FBB versus RBB

- FBB
  - When you remove bias = High V_t
  - Apply Forward bias = Low V_t
- RBB
  - No Bias = Low V_t
  - Reverse Bias = High V_t
  - With FBB, you can also use RBB
  - RBB + FBB reduces leakage by 30x for low V_t devices
  - RBB alone 2x for low V_t devices (Both 130nm, 110C)

Supply Gating Techniques

MTCMOS

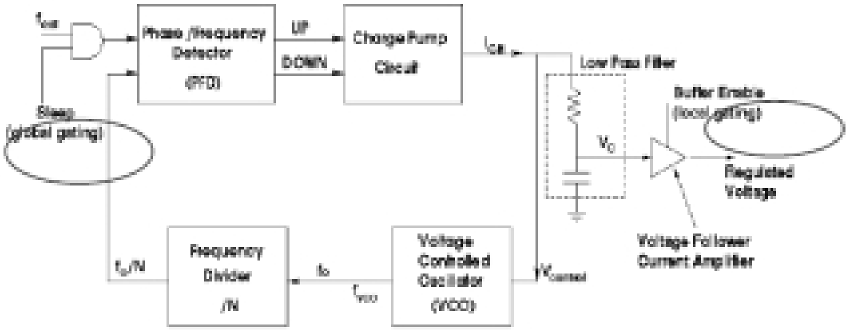


Sizing of the Sleep Transistor

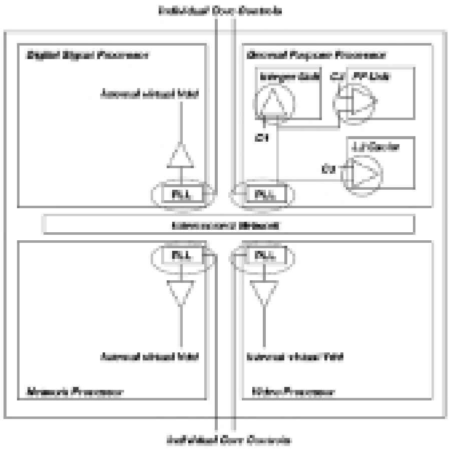
- Peak current that the sleep transistor can take influences performance
  - □ Peak current design needs more silicon area
  - □ Peak current design increases the off current
- Smaller Sleep transistor can cause reverse current, reduce noise margins, and improve performance

Power Supply Gating

Phase-Locked Loop (PLL) as a voltage regulator, intended to support DVS at run-time and leakage reduction during idle times

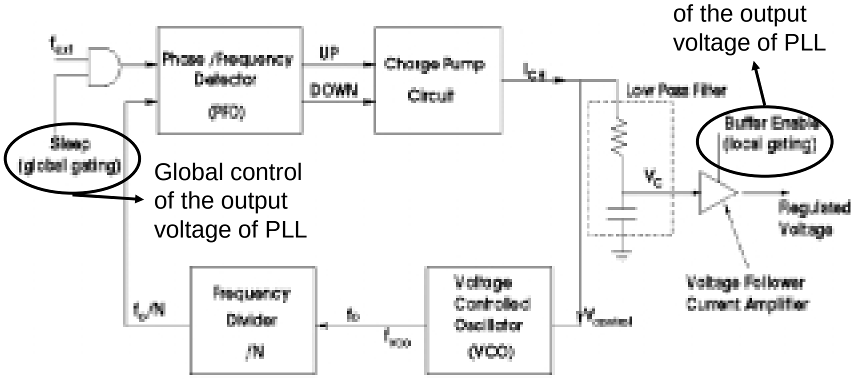


PSG Implementation: Global + Local



Implementation – Power Supply Gating (PSG)

- Datapath logics
  - □ Set the output of PLL to 0V during sleep mode



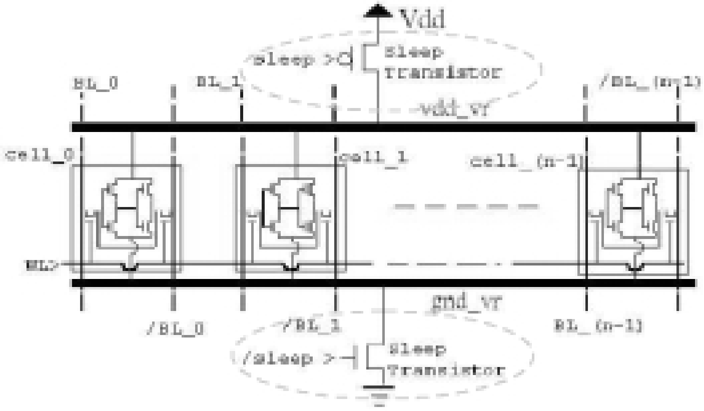
Local control of the output voltage of PLL

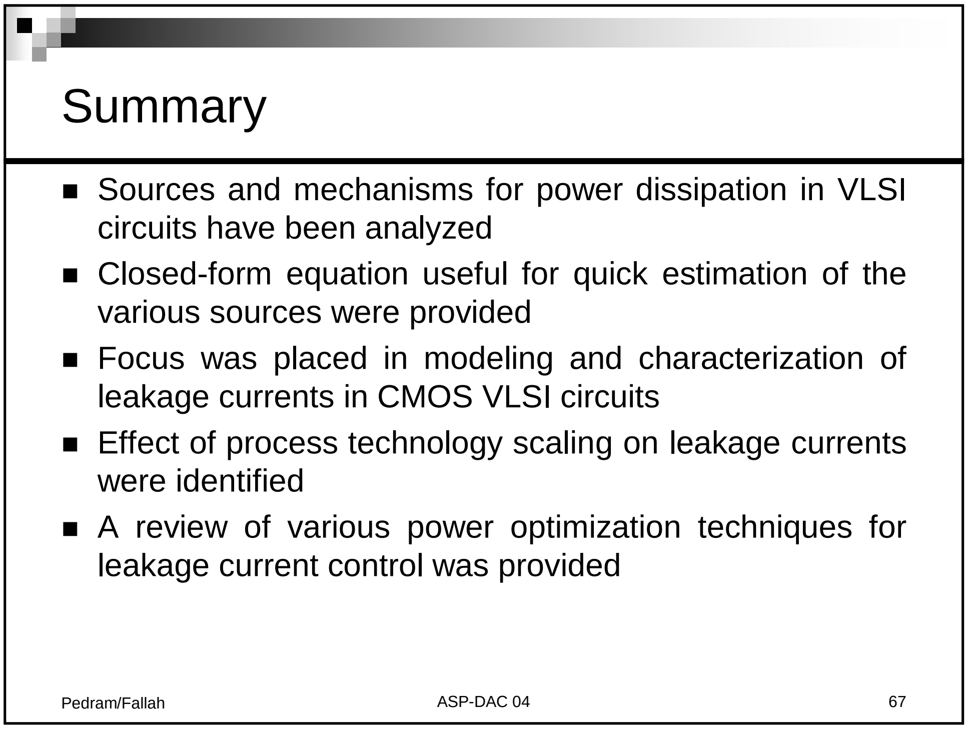
Global control of the output voltage of PLL

Implementation- Power Supply Gating (PSG)

- Memory Structures

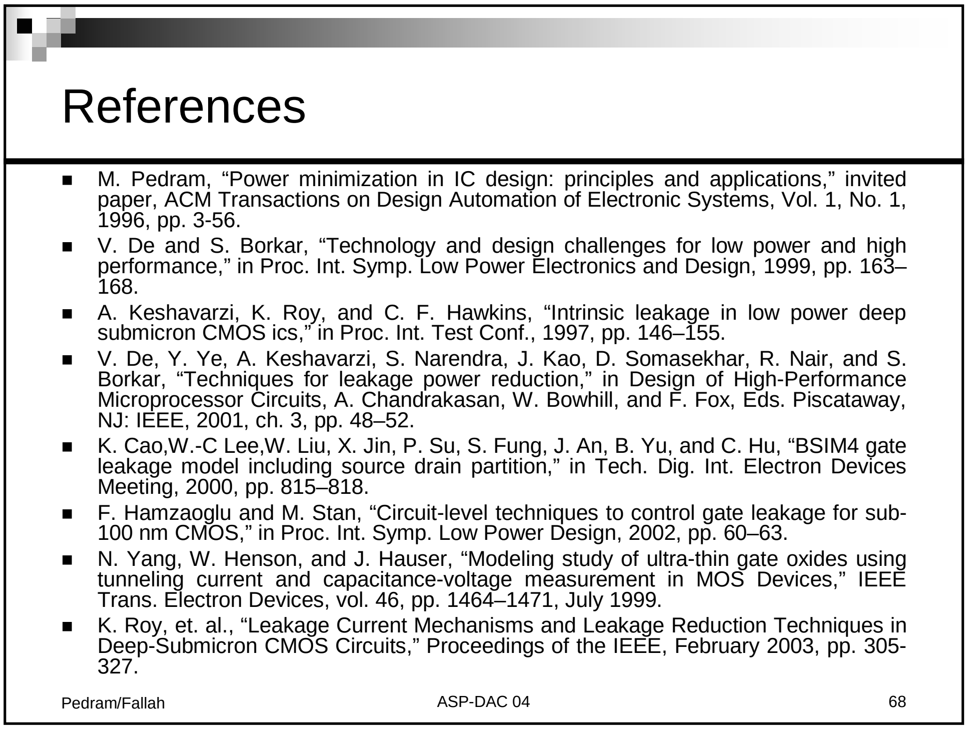
CMOS/NMOS/PMOS sleep transistor Gated-Vdd





Summary

- Sources and mechanisms for power dissipation in VLSI circuits have been analyzed
- Closed-form equation useful for quick estimation of the various sources were provided
- Focus was placed in modeling and characterization of leakage currents in CMOS VLSI circuits
- Effect of process technology scaling on leakage currents were identified
- A review of various power optimization techniques for leakage current control was provided



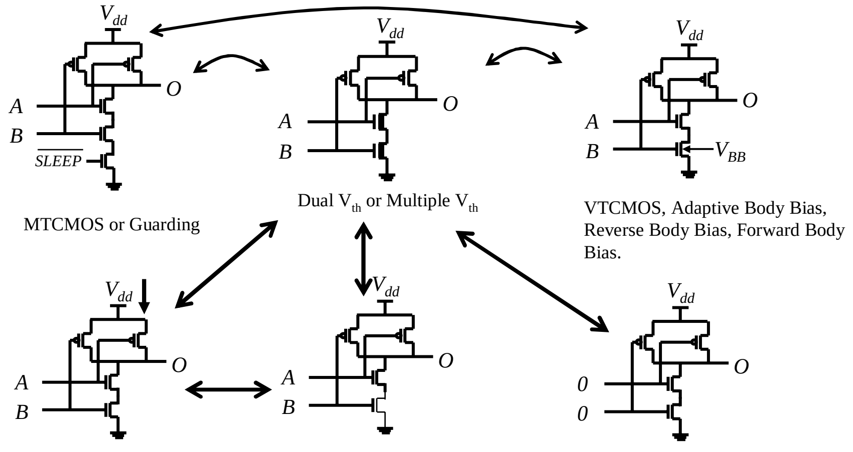
References

- M. Pedram, “Power minimization in IC design: principles and applications,” invited paper, ACM Transactions on Design Automation of Electronic Systems, Vol. 1, No. 1, 1996, pp. 3-56.
- V. De and S. Borkar, “Technology and design challenges for low power and high performance,” in Proc. Int. Symp. Low Power Electronics and Design, 1999, pp. 163–168.
- A. Keshavarzi, K. Roy, and C. F. Hawkins, “Intrinsic leakage in low power deep submicron CMOS ics,” in Proc. Int. Test Conf., 1997, pp. 146–155.
- V. De, Y. Ye, A. Keshavarzi, S. Narendra, J. Kao, D. Somasekhar, R. Nair, and S. Borkar, “Techniques for leakage power reduction,” in Design of High-Performance Microprocessor Circuits, A. Chandrakasan, W. Bowhill, and F. Fox, Eds. Piscataway, NJ: IEEE, 2001, ch. 3, pp. 48–52.
- K. Cao,W.-C Lee,W. Liu, X. Jin, P. Su, S. Fung, J. An, B. Yu, and C. Hu, “BSIM4 gate leakage model including source drain partition,” in Tech. Dig. Int. Electron Devices Meeting, 2000, pp. 815–818.
- F. Hamzaoglu and M. Stan, “Circuit-level techniques to control gate leakage for sub-100 nm CMOS,” in Proc. Int. Symp. Low Power Design, 2002, pp. 60–63.
- N. Yang, W. Henson, and J. Hauser, “Modeling study of ultra-thin gate oxides using tunneling current and capacitance-voltage measurement in MOS Devices,” IEEE Trans. Electron Devices, vol. 46, pp. 1464–1471, July 1999.
- K. Roy, et. al., “Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicron CMOS Circuits,” Proceedings of the IEEE, February 2003, pp. 305-327.

Global Outline

- PART I: Sources of Leakage Power and Trends
- ➔ PART II: Design Techniques for Leakage Minimization
- PART III: Leakage-aware Circuits and Memory

Leakage Reduction Techniques



MTCMOS or Guarding

Dual V_{th} or Multiple V_{th}

VTCMOS, Adaptive Body Bias, Reverse Body Bias, Forward Body Bias.

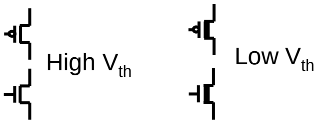
V_{dd} reduction

Transistor sizing

Stack Effect

Introduction

- Uses two different threshold voltages, low V_{th} and high V_{th}
  - □ Total four different types of transistors (2 NMOS and 2 PMOS)
- Low V_{th} transistors for gates on critical path and high V_{th} transistors for other gates.
- It is called Multiple V_{th} technology as well.

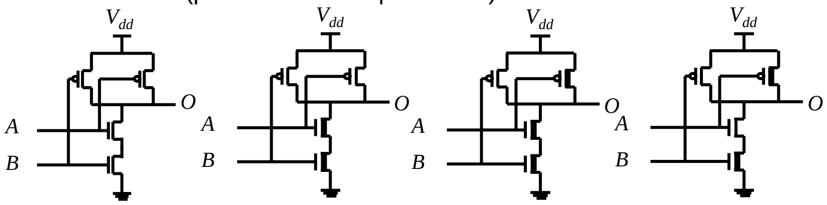


Important Questions

- How can we assign threshold voltages to transistors?
  - □ Not all non-critical gates can be made high V_{th}.
- How many different threshold voltages do we need?
  - □ Two or more?
- What are their optimal values?

Different Types

- Use only a single type of transistor for each gate.
- Use multiple types of transistor for each gate (Mixed V_{th} CMOS, MVT CMOS)
  - □ MVT1: Same threshold voltage for all transistors in N or P networks.
  - □ MVT2: Same threshold voltage for all transistors of a stack (due to proximity).
  - □ No limitation (possible in some processes).



High V_{th} Gate (DVT)　MVT1　MVT2　No Limitation

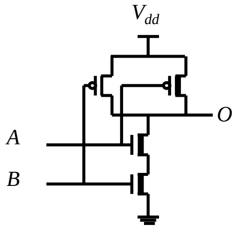
[Roy-DAC99-A]

Differences between Algorithms

Granularity	Gate	Pull-up/Pull-down Network	Stack	Transistor
Methodology	Cell-based	Custom		
Leakage Estimation	Known Input	Probabilistic	Average	Dominant States
Threshold Voltages	Pre-determined	Optimized		
Number of Threshold Voltages	Two	More Than Two		

Mixed-V_{th} (MVT) CMOS

- Use several different threshold voltages for transistors of each gate.
  - □ Use the same threshold for all transistors of pull-up or pull-down network.
  - □ Use the same threshold for all transistors of a stack.
    - Manufacturing limitation due to proximity.



- Higher leakage saving
- More complex threshold assignment algorithm

MVT CMOS- Algorithm

- Assume all low-V_{th} transistors.
- For each transistor of each gate,
  - □ Find the increase in the gate **delay** if high-V_{th} is used (Δt_d).
  - □ Find the decrease in the gate **leakage** if high-V_{th} is used ($\Delta leak_i = K \times W_{eff_i} \frac{\mu_i}{\mu_n}$).
  - □ Calculate $priority(i) = \frac{\Delta leak_i}{\Delta t_{d_i}}$
    - Higher value means more leakage can be saved using one unit of slack.

[Roy-DAC99-A]

MVT CMOS- Algorithm (2)

- Needs transistor-level static timing analysis.
- Propagation delay of a transistor,

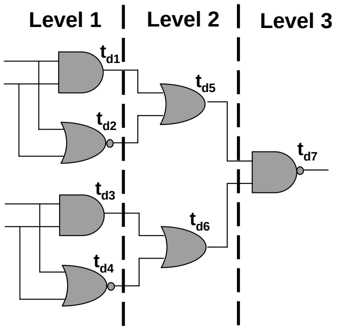
$$t_d = t_{intrinsic} + t_{output} C_L$$

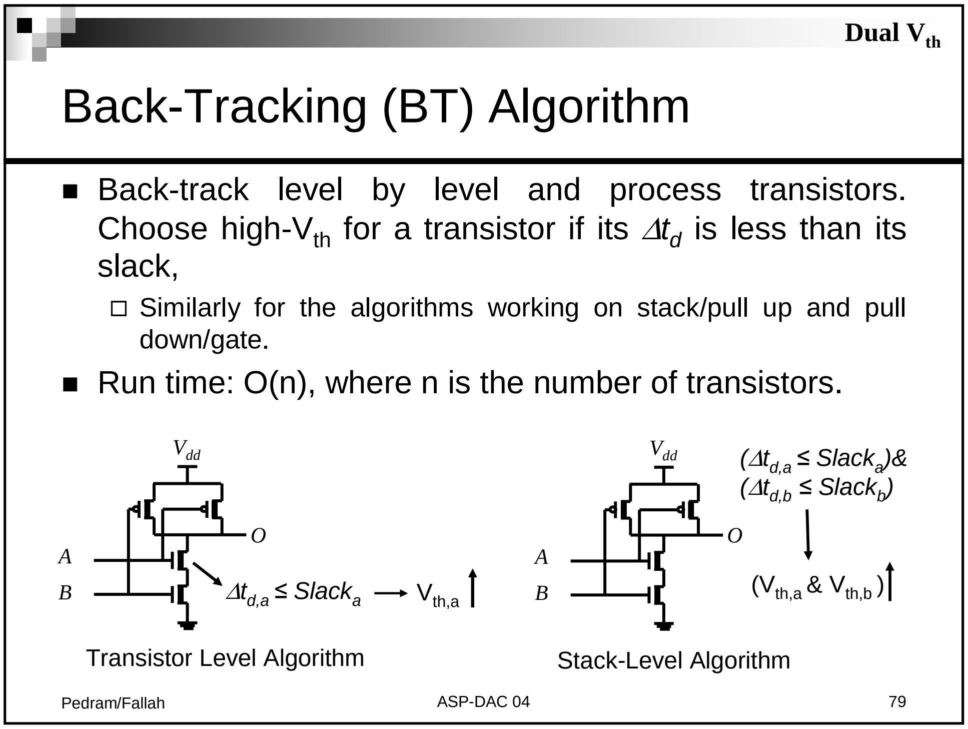
- For each gate G, calculate its departure times, $T_{lr}(G)$ and $T_{lf}(G)$,

$$T_{lr}(G) = \max_i \{T_{lf}(GI_i) + t_d(p_i)\}$$
$$T_{lf}(G) = \max_i \{T_{lr}(GI_i) + t_d(n_i)\}$$

MVT CMOS- Algorithm (3)

- Start from inputs and check gates level by level to calculate departure time of each gate.
- Back-track level by level to calculate the slack of transistors.





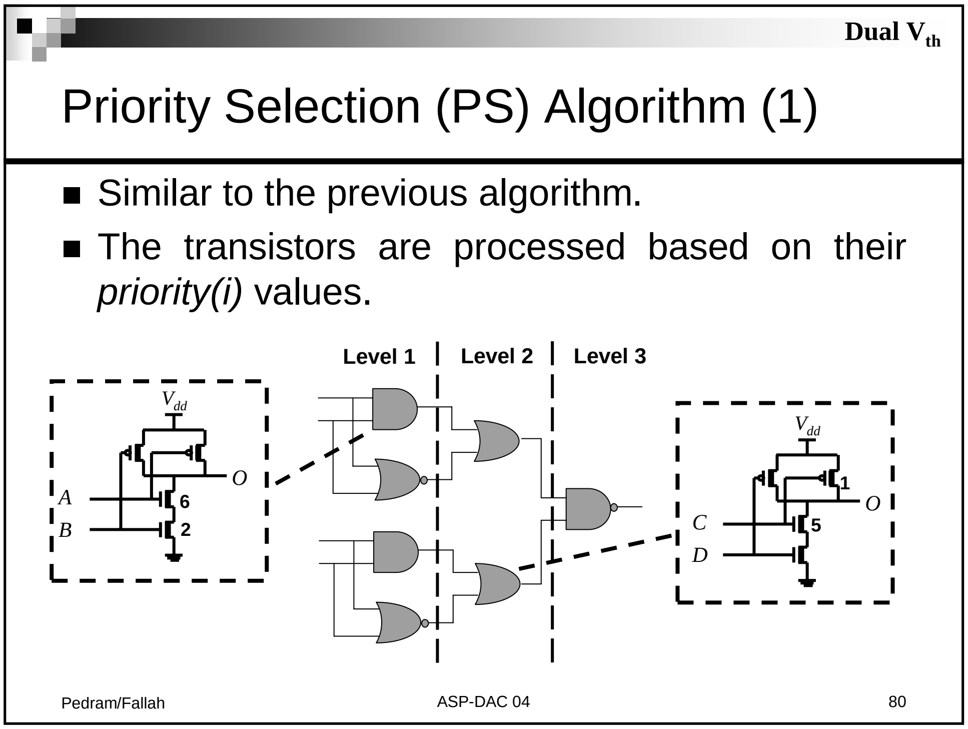
Back-Tracking (BT) Algorithm

- Back-track level by level and process transistors. Choose high-V_{th} for a transistor if its Δt_d is less than its slack,
  - □ Similarly for the algorithms working on stack/pull up and pull down/gate.
- Run time: O(n), where n is the number of transistors.

V_{dd}
O
A
B
$\Delta t_{d,a} \le Slack_a \longrightarrow V_{th,a} \uparrow$
V_{dd}
O
A
B
$(\Delta t_{d,a} \le Slack_a)\&$
$(\Delta t_{d,b} \le Slack_b)$
$(V_{th,a} \& V_{th,b})\uparrow$

Transistor Level Algorithm

Stack-Level Algorithm

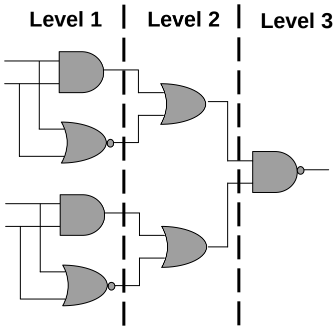


Priority Selection (PS) Algorithm (1)

- Similar to the previous algorithm.
- The transistors are processed based on their *priority(i)* values.

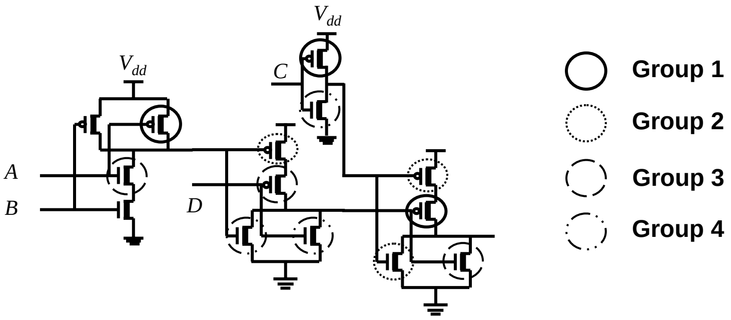
Priority Selection (PS) Algorithm (2)

- After modifying each transistor, the slack values have to be recalculated.
- Run time: $O(n^2)$, where n is the number of transistors.



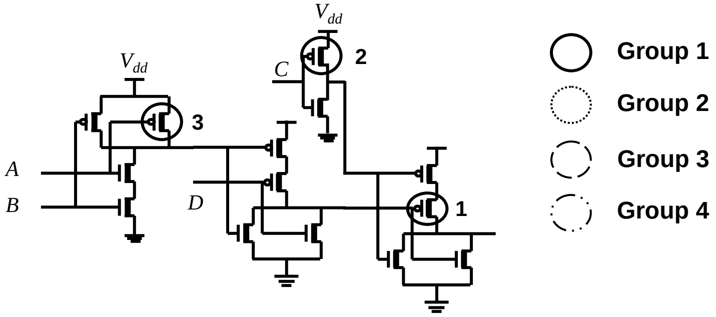
Priority-Based Backtracking (PB) Algorithm (1)

- A combination of two previous algorithms.
- Transistors are put in *m* different groups according to their priority values.



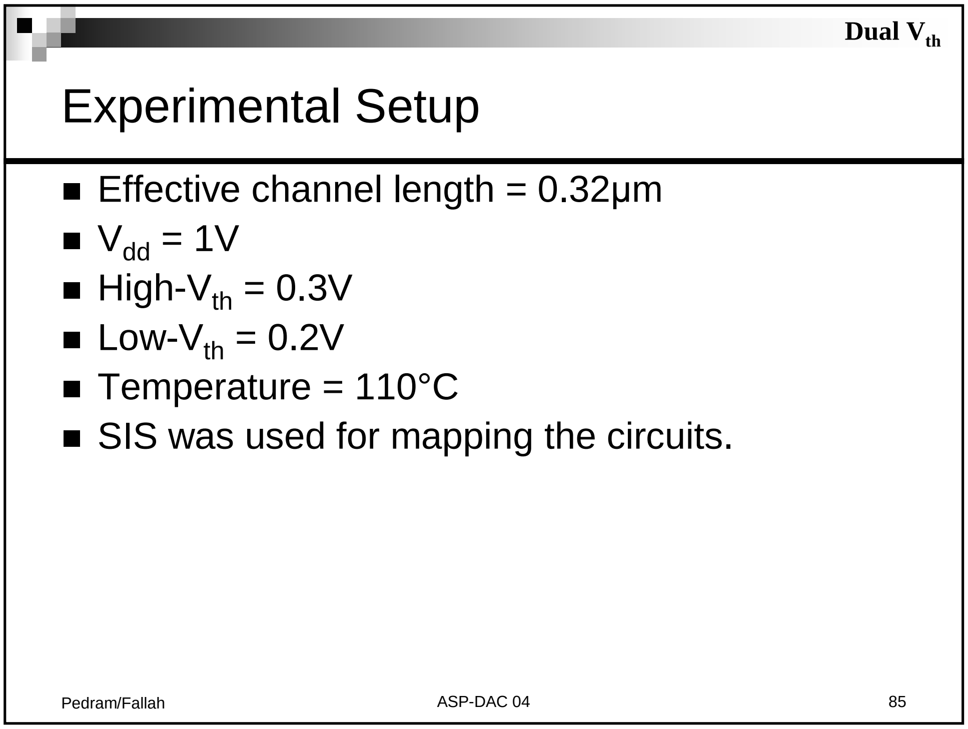
Priority-Based Backtracking (PB) Algorithm (2)

- For each group backtracking is done once during which only the transistors in that group are processed.
- It starts with the group with highest priority values.



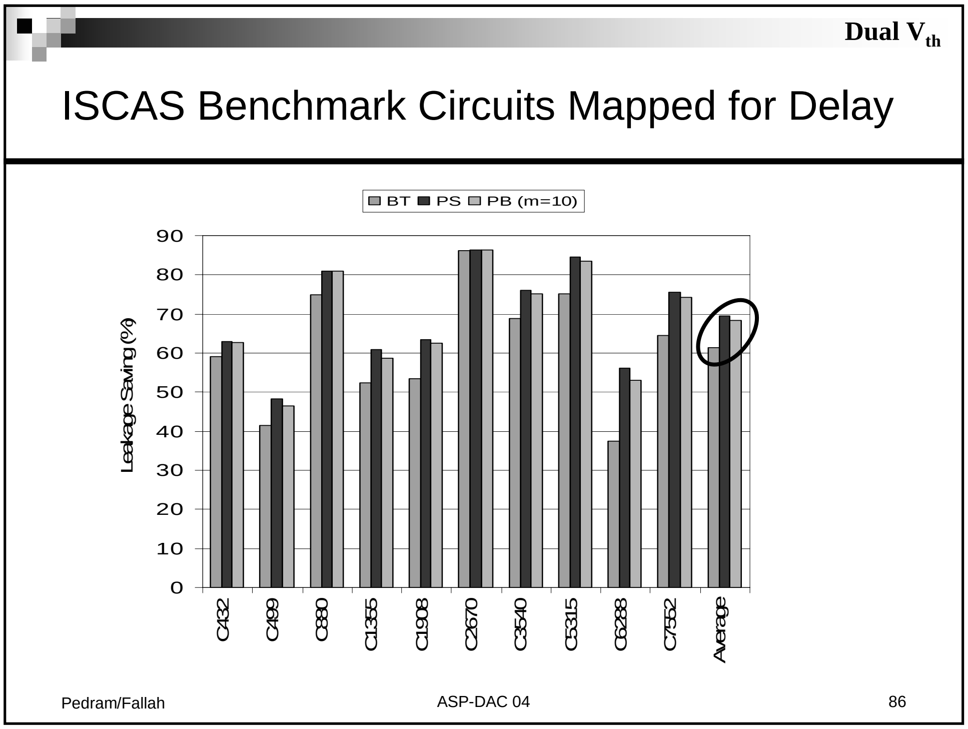
Priority-Based Backtracking (PB) Algorithm (3)

- m=1 → the algorithm is equivalent to the backtracking algorithm.
- m=n → the algorithm is equivalent to the priority selection algorithm.
- After each group is processed, the transistor slacks have to be recalculated.
- Run time: O(mn)



Experimental Setup

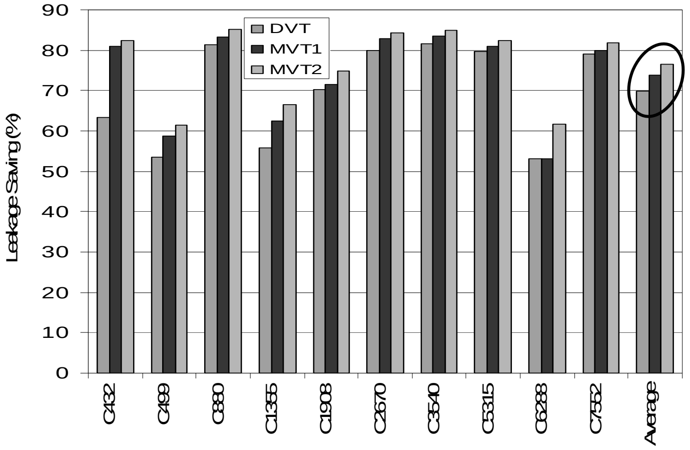
- Effective channel length = 0.32μm
- V_{dd} = 1V
- High-V_{th} = 0.3V
- Low-V_{th} = 0.2V
- Temperature = 110°C
- SIS was used for mapping the circuits.



ISCAS Benchmark Circuits Mapped for Delay

Dual V_{th}

ISCAS Benchmark Circuits Mapped for Area

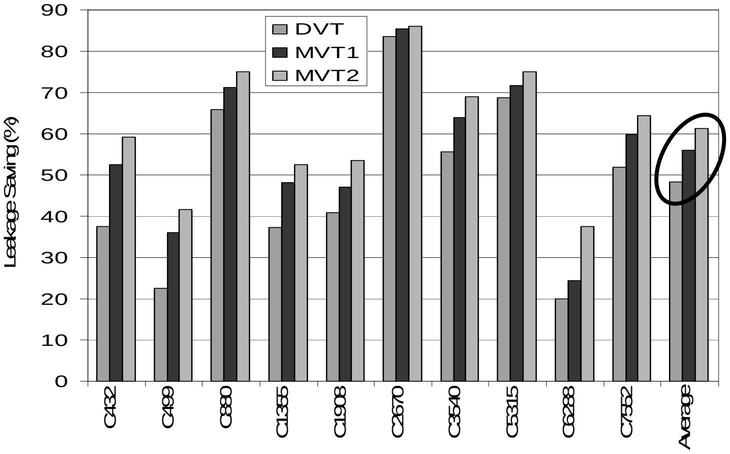


Results for the Backtracking Algorithm

Pedram/Fallah ASP-DAC 04 87

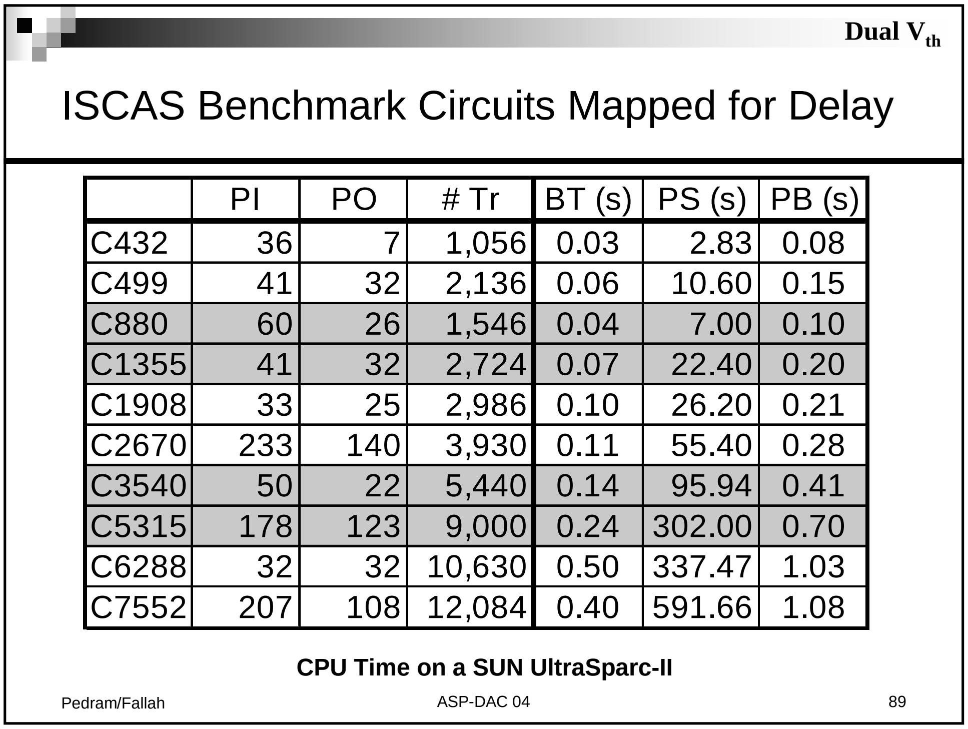
Dual V_{th}

ISCAS Benchmark Circuits Mapped for Delay



Results for the Backtracking Algorithm

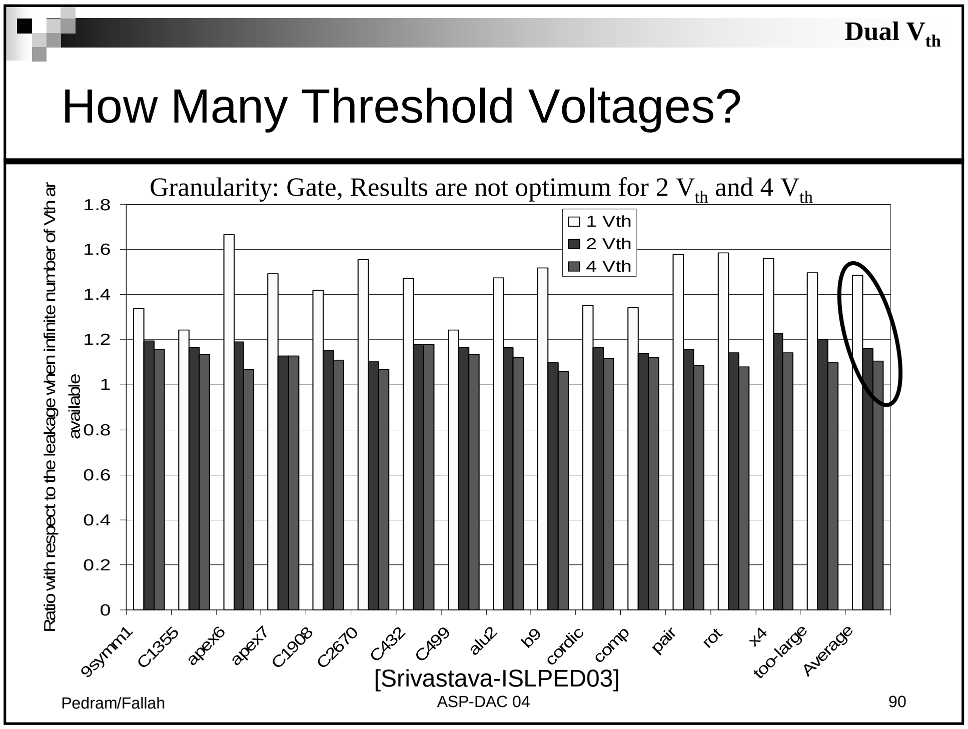
Pedram/Fallah ASP-DAC 04 88



ISCAS Benchmark Circuits Mapped for Delay

	PI	PO	# Tr	BT (s)	PS (s)	PB (s)
C432	36	7	1,056	0.03	2.83	0.08
C499	41	32	2,136	0.06	10.60	0.15
C880	60	26	1,546	0.04	7.00	0.10
C1355	41	32	2,724	0.07	22.40	0.20
C1908	33	25	2,986	0.10	26.20	0.21
C2670	233	140	3,930	0.11	55.40	0.28
C3540	50	22	5,440	0.14	95.94	0.41
C5315	178	123	9,000	0.24	302.00	0.70
C6288	32	32	10,630	0.50	337.47	1.03
C7552	207	108	12,084	0.40	591.66	1.08

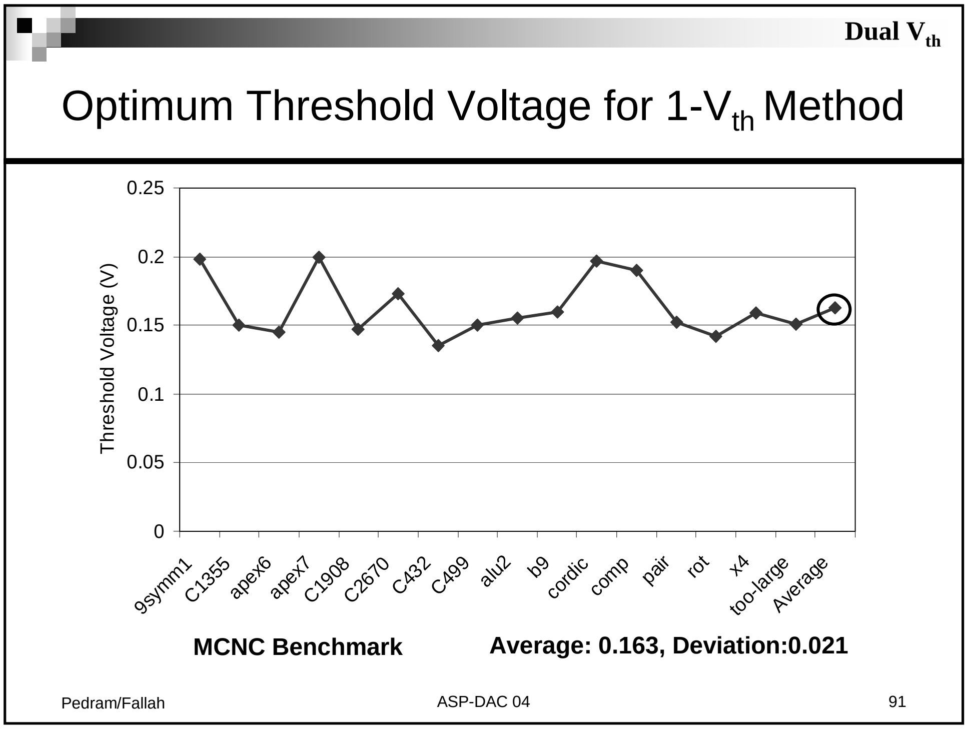
CPU Time on a SUN UltraSparc-II



How Many Threshold Voltages?

Granularity: Gate, Results are not optimum for 2 V_{th} and 4 V_{th}

[Srivastava-ISLPED03]



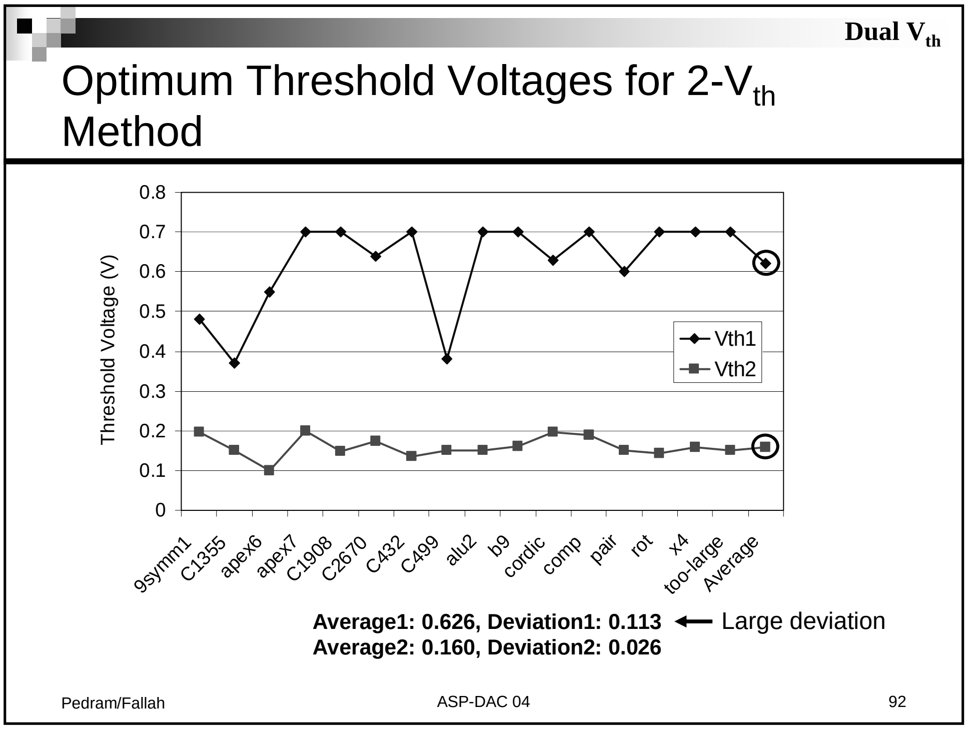
Dual V_{th}

Optimum Threshold Voltage for 1-V_{th} Method

MCNC Benchmark

Average: 0.163, Deviation:0.021

Pedram/Fallah ASP-DAC 04 91



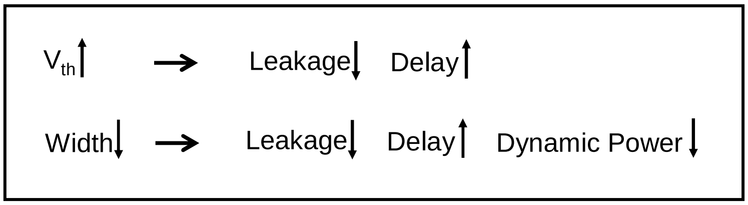
Dual V_{th}

Optimum Threshold Voltages for 2-V_{th} Method

Average1: 0.626, Deviation1: 0.113 ← Large deviation
Average2: 0.160, Deviation2: 0.026

Pedram/Fallah ASP-DAC 04 92

Minimizing Total Power Consumption

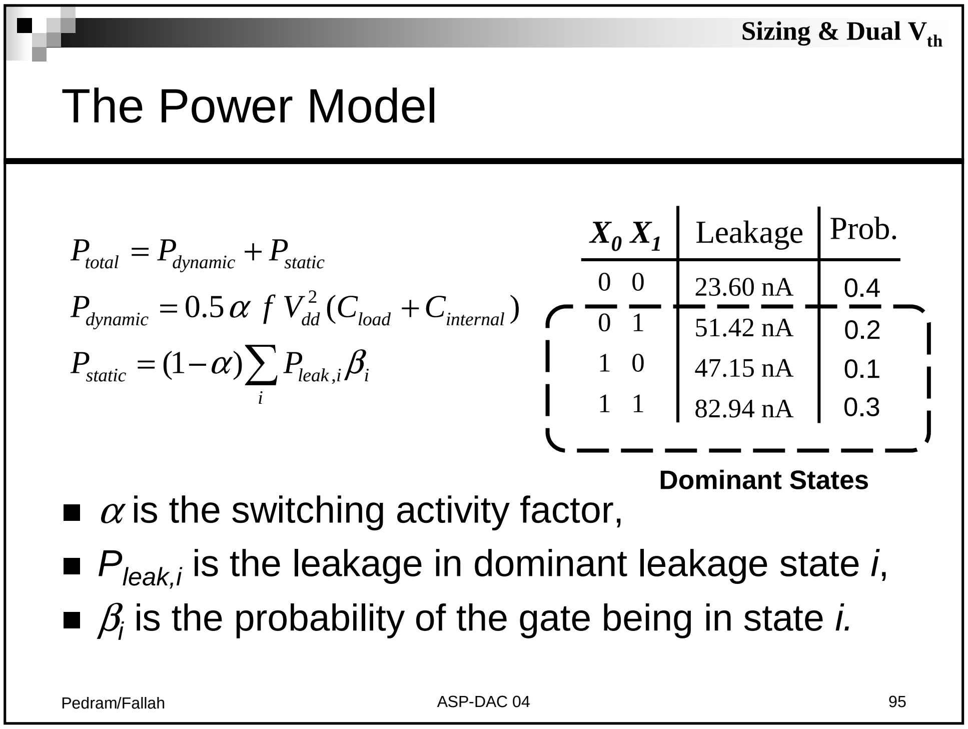

V_{th}↑ → Leakage↓ Delay↑
Width↓ → Leakage↓ Delay↑ Dynamic Power↓

- Combine the threshold assignment and sizing to achieve better results.

Minimizing Total Power Consumption

- Complex optimization problem
  - □ Non-linear delay and power models
  - □ Leakage is a function of a gate's input values
  - □ Delay of a gate depends on its fanout
- Need to simplify the problem
  - □ Use a cell-based approach
    - Six different sizes for each cell
    - V_{th} allocation is done at gate level
  - □ Use the dominant leakage states when calculating the total power consumption
  - □ Use a delay model which takes into account the load capacitance

[Keutzer-ISLPED03]



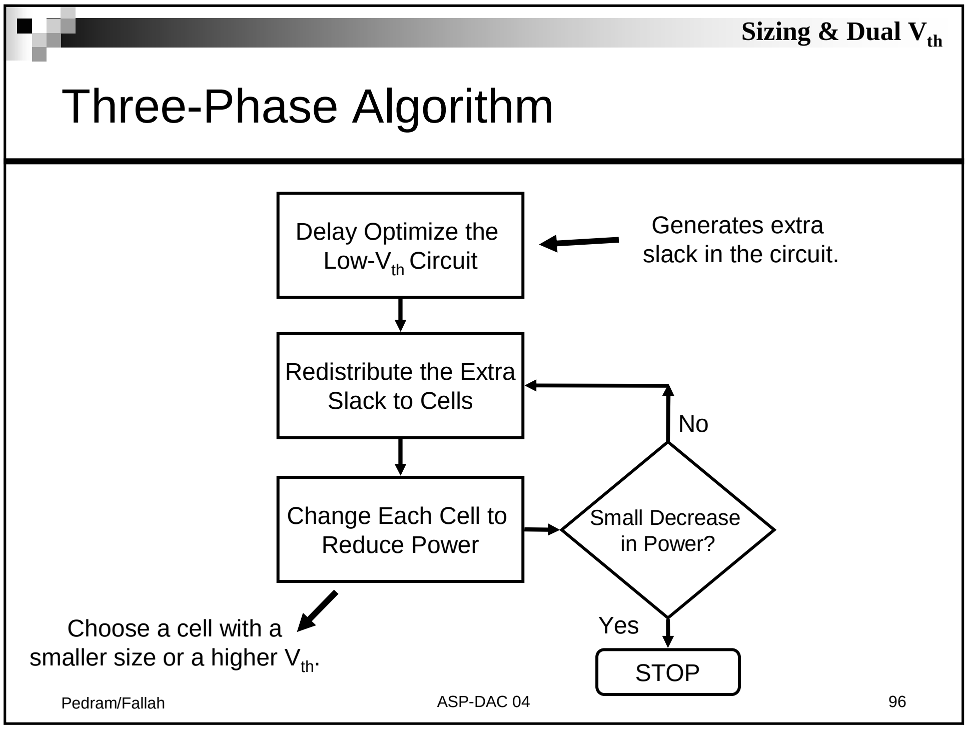
The Power Model

$$P_{total} = P_{dynamic} + P_{static}$$
$$P_{dynamic} = 0.5\alpha\ f\ V_{dd}^2 (C_{load} + C_{internal})$$
$$P_{static} = (1-\alpha)\sum_i P_{leak,i}\beta_i$$

X_0 X_1	Leakage	Prob.
0 0	23.60 nA	0.4
0 1	51.42 nA	0.2
1 0	47.15 nA	0.1
1 1	82.94 nA	0.3

Dominant States

- α is the switching activity factor,
- $P_{leak,i}$ is the leakage in dominant leakage state *i*,
- β_i is the probability of the gate being in state *i.*



Three-Phase Algorithm

Delay Optimize the Low-V_{th} Circuit

Generates extra slack in the circuit.

Redistribute the Extra Slack to Cells

Change Each Cell to Reduce Power

Small Decrease in Power?

No

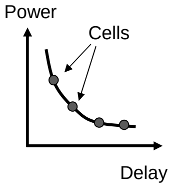
Yes

STOP

Choose a cell with a smaller size or a higher V_{th}.

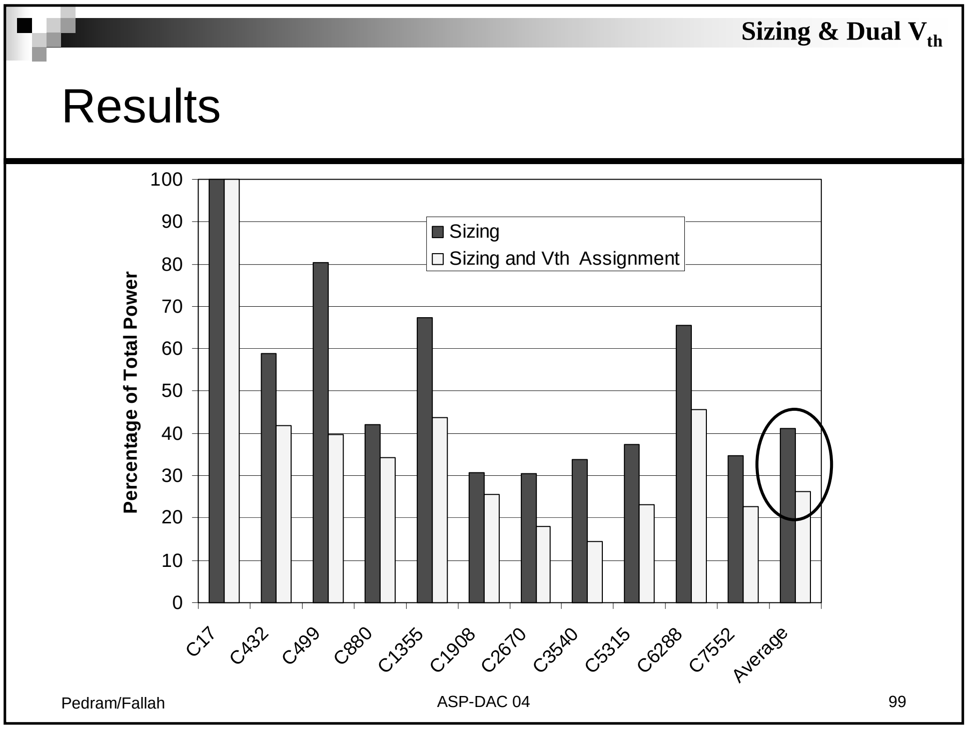
Slack Redistribution

- Evenly distributing the extra slack is not good,
  - □ Some gates are better in trading off delay for power
- Calculate $\frac{\Delta P(i)}{\Delta D(i)}$ for every gate.
  - □ The higher the number, the higher the assigned slack.
- Power is not a linear function of delay.
  - □ Recalculate $\frac{\Delta P(i)}{\Delta D(i)}$ at every iteration.
- Discrete optimization: extra slack assigned to a gate may not be used,
  - □ Assign it to another gate at next iteration.



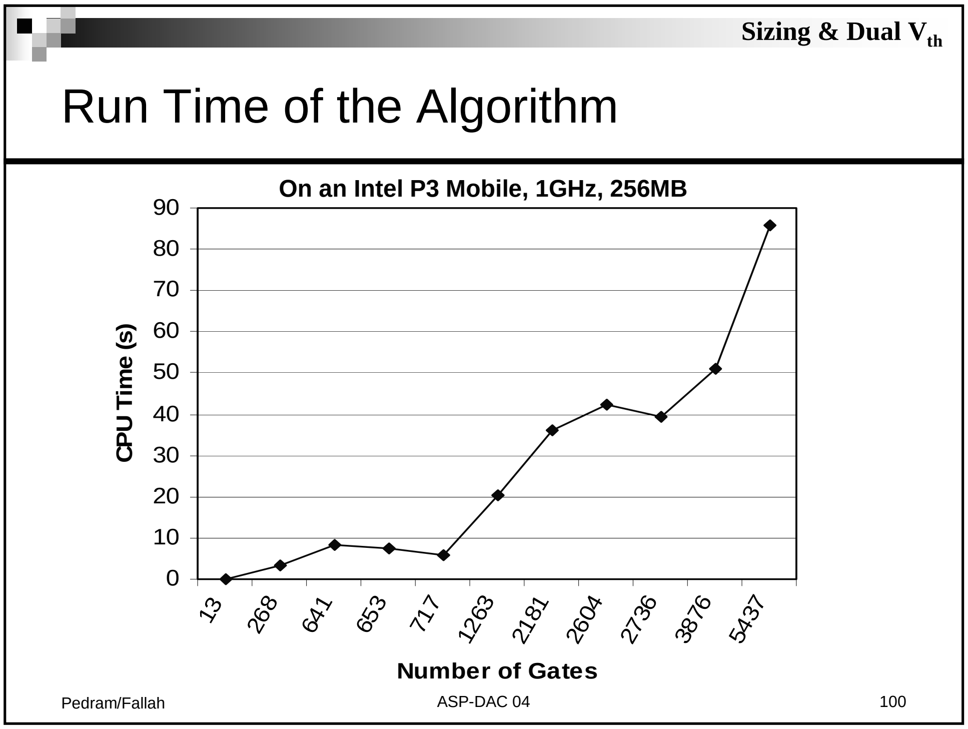
Experimental Setup

- ISCAS85 Circuits
- 0.18µm process
  - □ V_{dd} = 1.8V
  - □ high-V_{th}= ±0.45V, low-V_{th}= ±0.30V
- Developed gates with 6 different sizes ranging from 0.18µm to 1.8µm
- Threshold assignment was done at gate level.



Sizing & Dual V_{th}

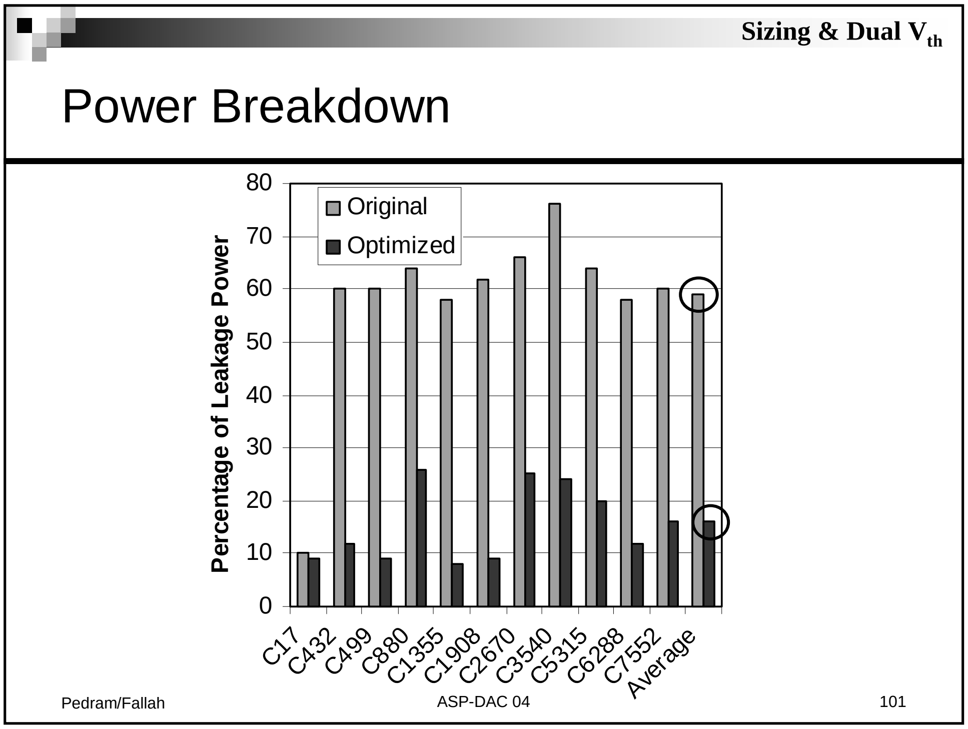
Results



Sizing & Dual V_{th}

Run Time of the Algorithm

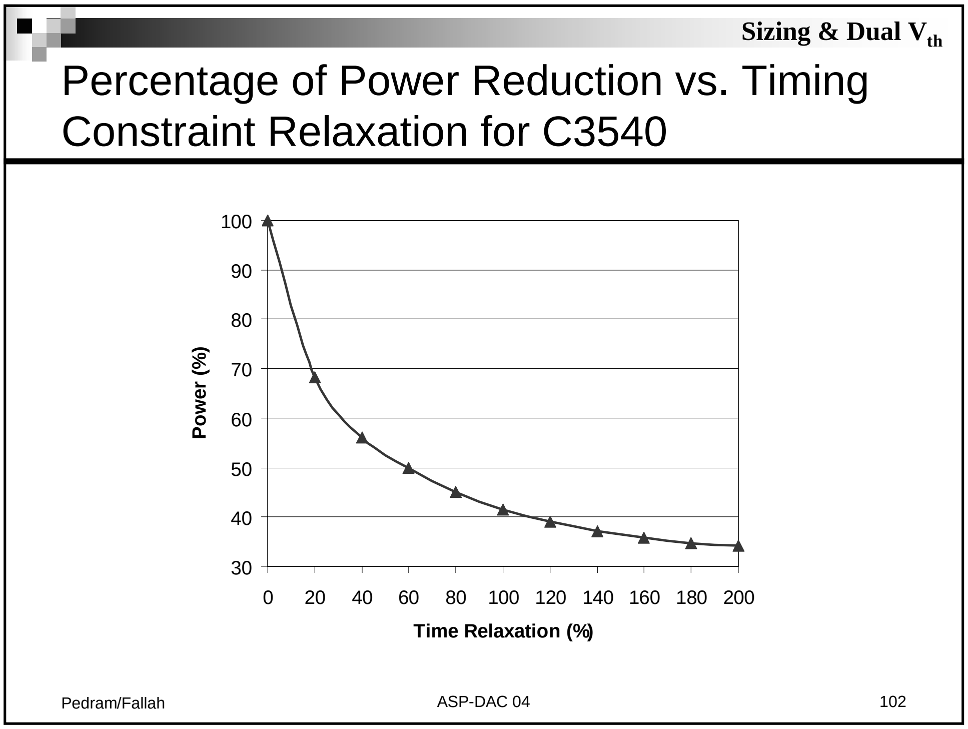
On an Intel P3 Mobile, 1GHz, 256MB



Sizing & Dual V_{th}

Power Breakdown

Pedram/Fallah ASP-DAC 04 101



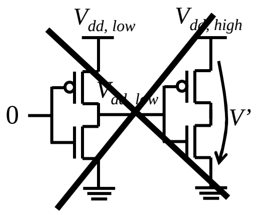
Sizing & Dual V_{th}

Percentage of Power Reduction vs. Timing Constraint Relaxation for C3540

Pedram/Fallah ASP-DAC 04 102

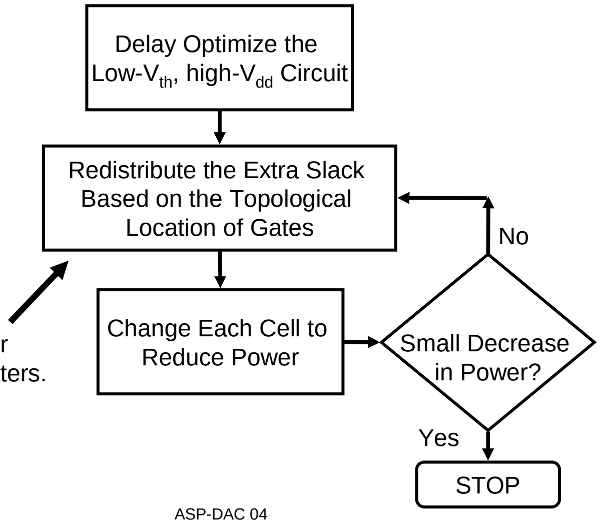
Extending the Algorithm – The Idea

- It is possible to modify the algorithm to handle dual supply technologies.
- Topological limitation:
  - □ A low-V_{dd} gate cannot drive a high-V_{dd} gate directly.
    - Level converters can be used, but the overhead is large.
  - □ A high-V_{dd} gate can drive any gates.
- Due to topological constraints and the fact that low-V_{dd} gates have significantly higher delay, the original slack distribution method is not good.



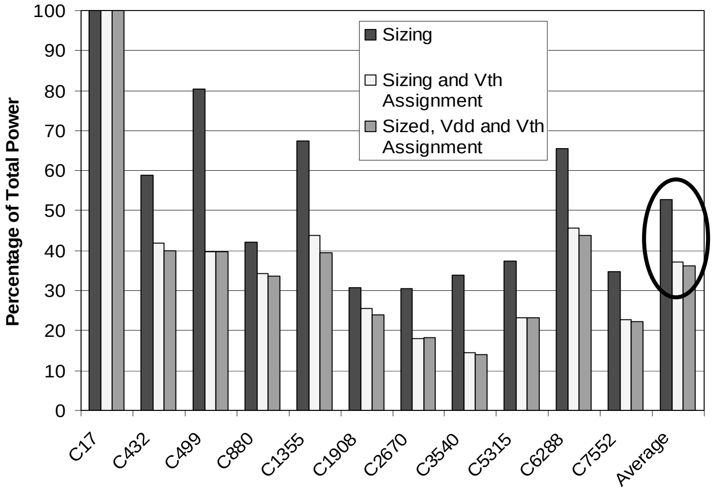
$0 < V' < V_{dd, high}$

Extending the Algorithm


Delay Optimize the Low-V_{th}, high-V_{dd} Circuit
Redistribute the Extra Slack Based on the Topological Location of Gates
Change Each Cell to Reduce Power
Small Decrease in Power?
No
Yes
STOP

To minimize the number of required level converters.

Results



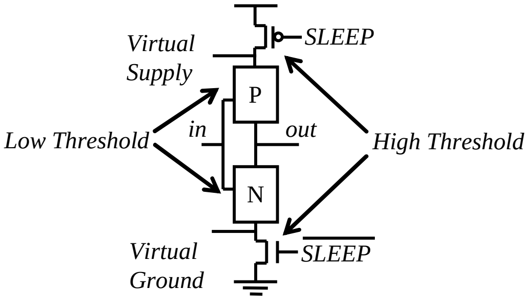
New Unpublished Result: 18% additional reduction when using Dual-Vdd method.

Guidelines

- Careful sizing (by using signals' probabilities) can reduce the total power on average by 60% for a low-V_{th} circuit.
  - □ No need to use multiple threshold voltages!
- If the threshold voltage value can be optimized, a single threshold voltage may be enough,
  - □ In any case, more than two threshold voltages is not necessary.
- Use gate-level threshold assignment (simpler library)
  - □ The transistor-level method is on average only 10% better.

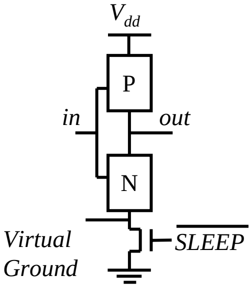
Multi-Threshold CMOS (MTCMOS)

- It is also called guarding, power gating, ground gating, using sleep transistor, etc.
- A high-V_{th} is used to disconnect low-V_{th} transistors from the ground (V_{dd}).



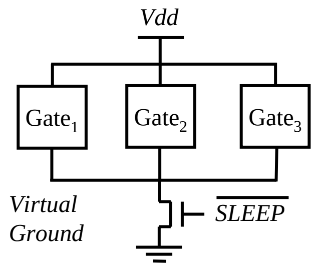
Simplification

- Instead of two sleep transistors, one can be used.
- Usually NMOS:
  - □ $\mu_n > \mu_p$ → smaller size
  - □ But, PMOS usually has a lower leakage.

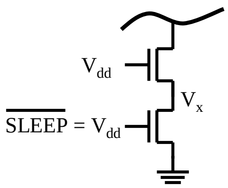


Further Simplification

- One sleep transistor can be shared between several gates.
  - □ Reduction in the number of sleep transistors, area overhead, dynamic and leakage power.
  - □ Increase in the complexity.



Sleep Transistor Sizing



$$V_{gs} < V_{dd}$$

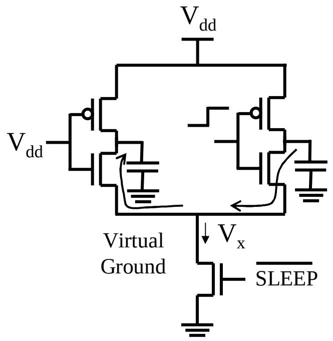
$$V_{th} = V_{th_0} + \gamma' V_{app} - \eta V_{ds}$$

- Reduction in the high to low transition due to,
  - □ Reduction in the gate drive from V_{dd} to $V_{dd} – V_x$.
  - □ Increase in the threshold voltage of NMOS due to the body effect.
- Increase the sleep transistor width to solve the problem
  - □ Increase in the area overhead, dynamic power and leakage.
- Technology scale down →have to enlarge the sleep transistor

[Kao-DAC97]

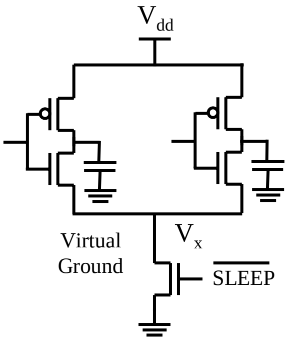
Reverse Conduction Path

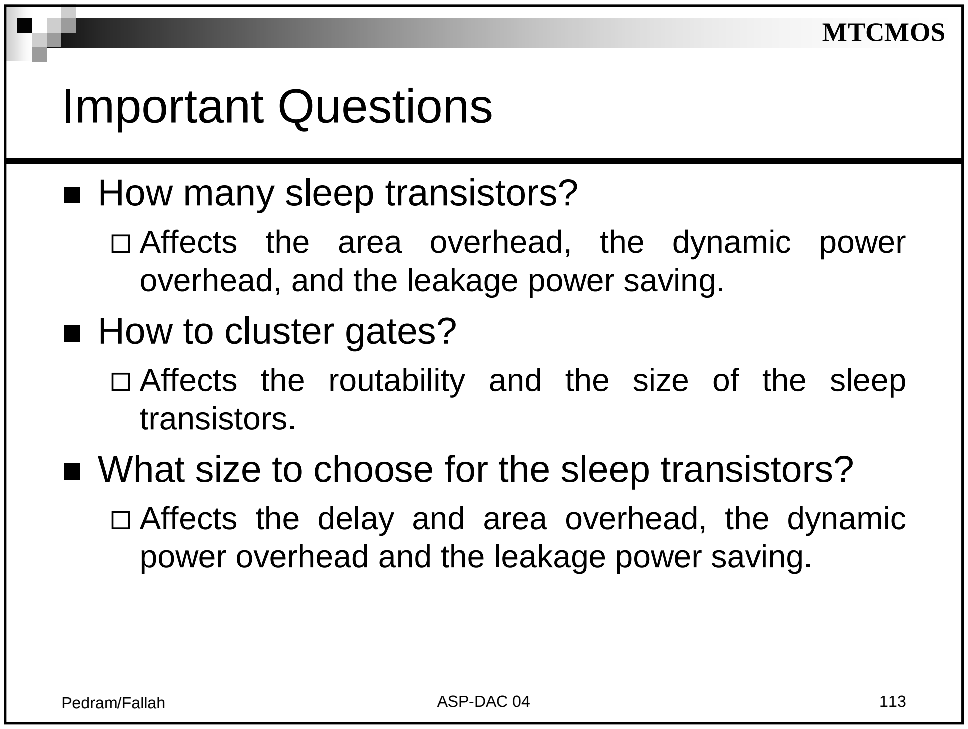
- Current may flow from one output to another.
- Some nodes have a voltage between V_{dd} and Ground.
- V_x is smaller than expected.
- Low to high transition is faster (V_x → V_{dd}).



Worst Case Vector

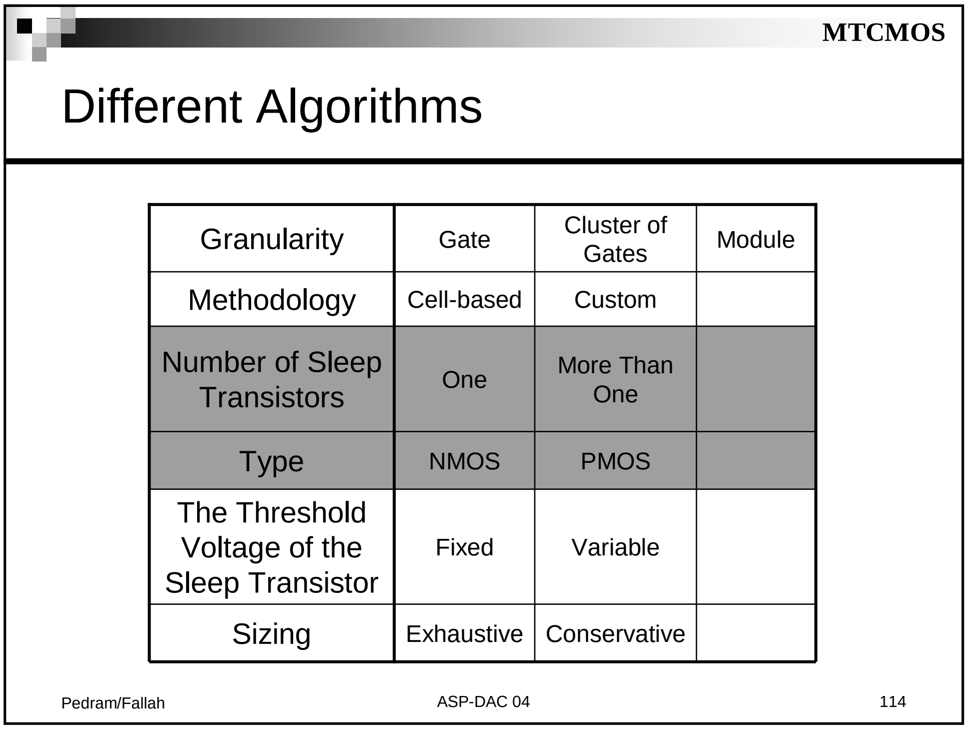
- Usually changes after adding the sleep transistor
  - □ V_x effect
  - □ It depends on the critical path and the profile of current flowing to the virtual ground.
- May even change by resizing the sleep transistor.





Important Questions

- How many sleep transistors?
  - □ Affects the area overhead, the dynamic power overhead, and the leakage power saving.
- How to cluster gates?
  - □ Affects the routability and the size of the sleep transistors.
- What size to choose for the sleep transistors?
  - □ Affects the delay and area overhead, the dynamic power overhead and the leakage power saving.



Different Algorithms

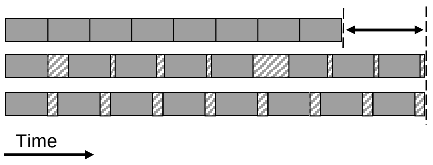
Granularity	Gate	Cluster of Gates	Module
Methodology	Cell-based	Custom	
Number of Sleep Transistors	One	More Than One	
Type	NMOS	PMOS	
The Threshold Voltage of the Sleep Transistor	Fixed	Variable	
Sizing	Exhaustive	Conservative	

Sizing: The Exhaustive Approach

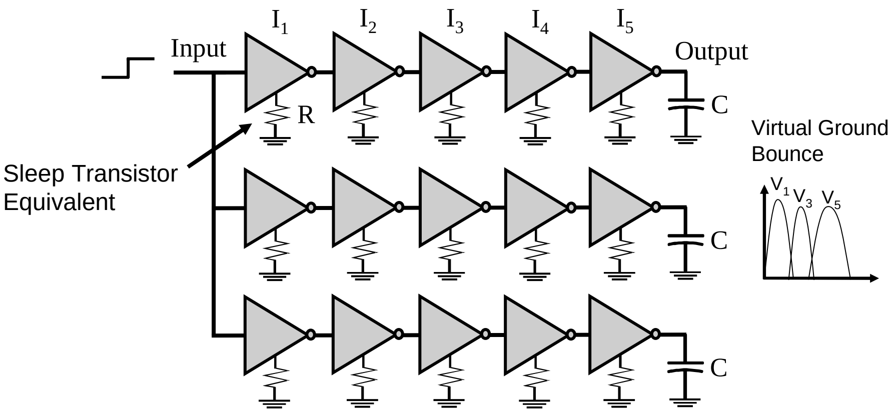
- Objective: Finding the size of the sleep transistor for a given *overall* delay degradation ($\Delta d/d$).
- Exhaustively simulate a circuit with a sleep transistor under all test vectors,
  - □ Will find the optimum size
  - □ Works perfectly for library cells
  - □ Impractical for larger circuits
    - 16-bit adder → $2^{16} \times 2^{16} \cong 4.2$ billion vectors, need to simulate the circuit under each vector for different size of the sleep transistor.
    - In practice, the delay depends on the vectors of the previous cycle as well, i.e., $2^{16} \times 2^{16} \times 2^{16} \times 2^{16} = 2^{64} \cong 18.4 \times 10^{18}$ vectors!!!

Sizing: The Conservative Approach

- Limit the delay degradation of each gate to $\Delta d/d.$
- Find the optimum size of the sleep transistor for each gate,
  - □ Much more demanding, but much easier to achieve.
  - □ Assumes both low-to-high and high-to-low transitions degrade.
- Combine the sleep transistors of different gates.

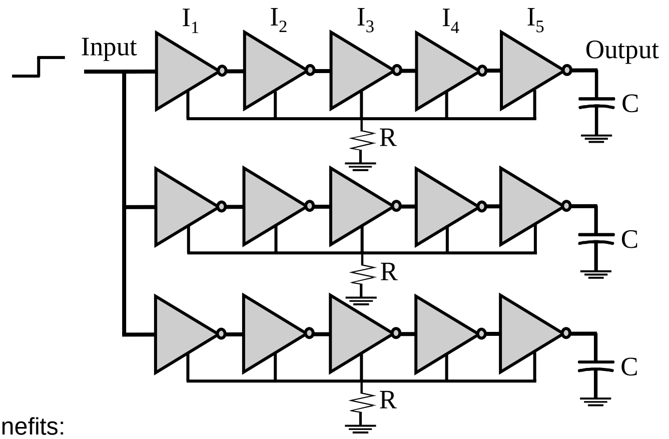


Mutual Exclusion-Circuit A



[Kao-DAC98]

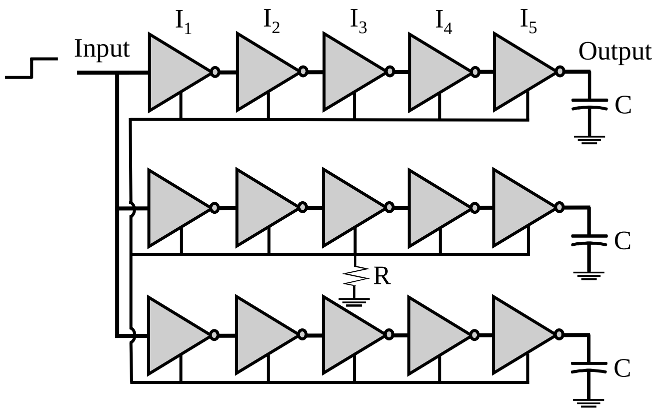
Mutual Exclusion-Circuit B



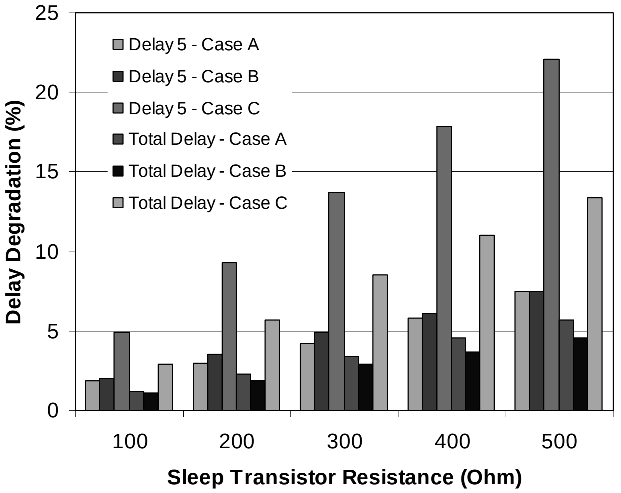
Benefits:

- Reduction in the area overhead, leakage and dynamic power.
- Decrease in the virtual ground bounce due to increase in the parasitic capacitance.

Mutual Exclusion-Circuit C



Percentage of Delay Degradation



Merging Parallel Transistors

$$V(t) = \min(V_1(t), V_2(t))$$

$$\downarrow$$

$$R_{eq} = \frac{\min(V_1(t), V_2(t))}{I(t)}$$

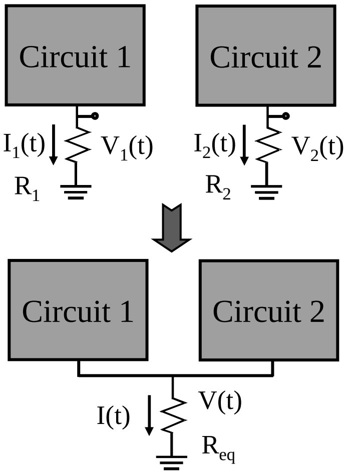
$$\downarrow$$

$$R_{eq} = \frac{\min(V_1(t), V_2(t))}{I_1(t) + I_2(t)}$$

$$V_1(t) = V_2(t)$$

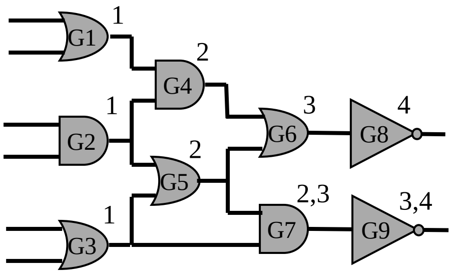
$$\downarrow$$

$$R_{eq} = \frac{\min(V_1(t), V_2(t))}{\frac{V_1(t)}{R_1} + \frac{V_2(t)}{R_2}} = \frac{R_1 R_2}{R_1 + R_2}$$



The Algorithm

Find all possible transition times of gates assuming single cycle delay for each gate.



$$Group1 = \{G1, G4, G6, G8\}$$
$$Group2 = \{G2, G5, G9\}$$
$$Group3 = \{G3, G7\}$$

Comparison

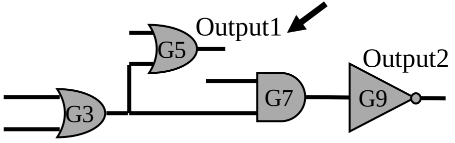
Circuit	Method	Sleep Transistor Resistance
Inverter	Sizing	340Ω
3 Chains	Mutual Exclusion	113Ω
3 Chains	Sizing	180Ω

- 60% overestimation!!!
  - Because in practice only half of gates switch from high to low.

How to Improve the Results

- The method gives an upper bound on the size of a sleep transistor.
- To improve the estimation,
  - Use logical information instead of structural information to find out mutual exclusion.
  - Limit the delay degradation of the entire circuit, not every module.

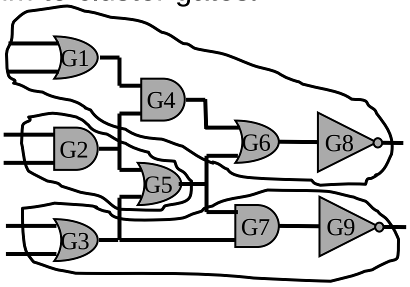
Not on the critical path; no need to limit its delay degradation.



Gate Clustering through Bin Packing and Set Partitioning

- Using one sleep transistor per module → complex interconnect and parasitic resistance
- Solution: cluster gates and use one sleep transistor per cluster

1. Calculate the maximum allowed current for each cluster.
2. Use an algorithm to cluster gates.



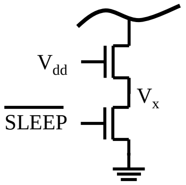
[Anis-DAC02]

Finding the Maximum Current and the Optimum Size of the Sleep Transistor

$$\tau_d = \frac{C_L V_{dd}}{(V_{dd} - V_{th_L})^{\alpha}} \qquad \alpha \approx 1.3 \text{ for } 0.18\mu m$$

$$\tau_{d_{SLEEP}} = \frac{C_L V_{dd}}{(V_{dd} - V_X - V_{th_L})^{\alpha}}$$

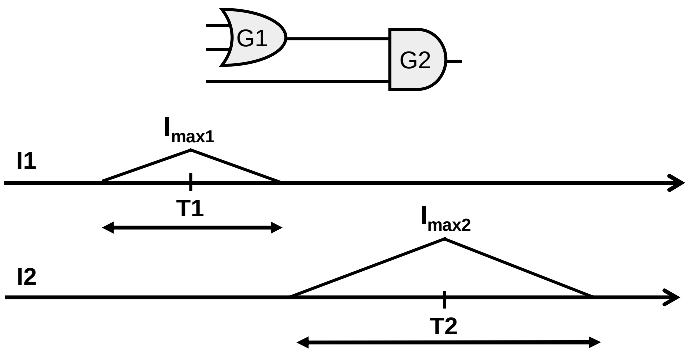
$$\text{Performance Degradation} = 1 - \frac{\tau_d}{\tau_{d_{SLEEP}}}$$



$$I_{sleep} = \mu_n C_{ox} (W/L)_{sleep} [(V_{dd} - V_{th_H}) V_X - V_X^2 / 2]$$

$(W/L)_{sleep}$ can be calculated for a given performance degradation and I_{sleep} value.

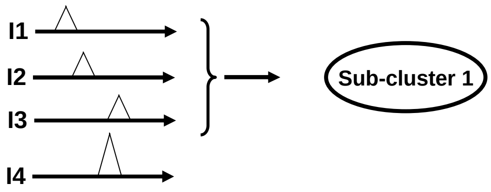
Find Current Forms for Each Gate



I1: 0,0,4,8,12,16,12,8,4,0, 0, 0, 0, 0, 0,0,0,0,0,0
I2: 0,0,0,0, 0, 0, 0,3,6,9,12,15,18,15,12,9,6,3,0,0

Preprocessing

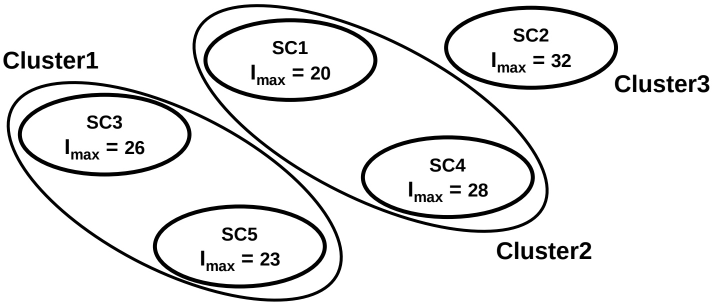
- Make sub-clusters: choose gates whose leakage combined does not exceed the maximum leakage of any of them.



Bin-Packing

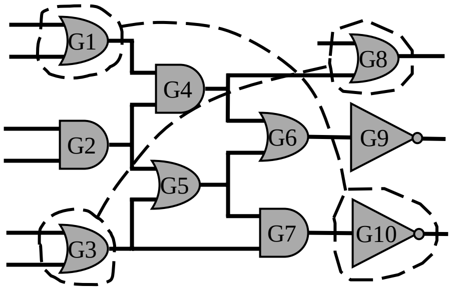
- Combine sub-clusters to form clusters whose max currents are less than a threshold selected before (i.e., I_{sleep}).
- Objective: to minimize the number of clusters (sleep transistors).

Threshold: $I_{max} \leq 50$



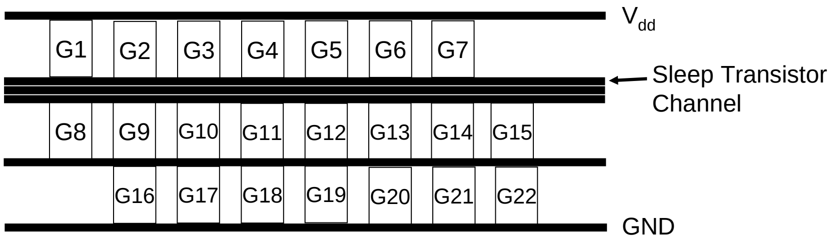
Problems

- Uses an Integer Linear Programming package to solve the problem
  - □ For a circuit with 220 gates: BP about 200s and SP about 1000s.
- No physical location information is used
  - □ Two gates located far from each other may be put in the same cluster
    - Complex routing and high interconnect resistance.

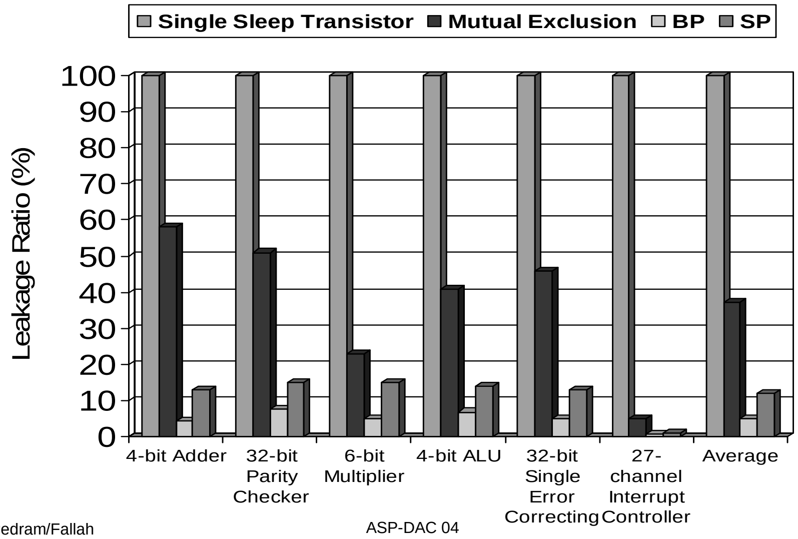


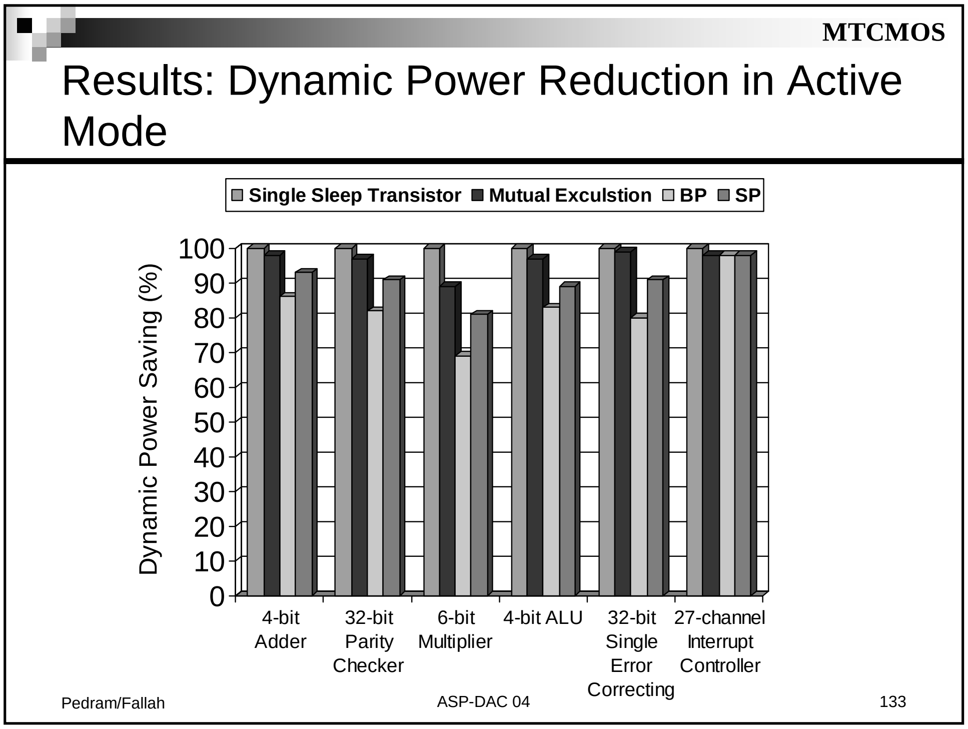
Set-Partitioning Method

- Perform placement and route.
- Modify the cost function to take into account the distance between cells when choosing clusters.



Results: Leakage Reduction

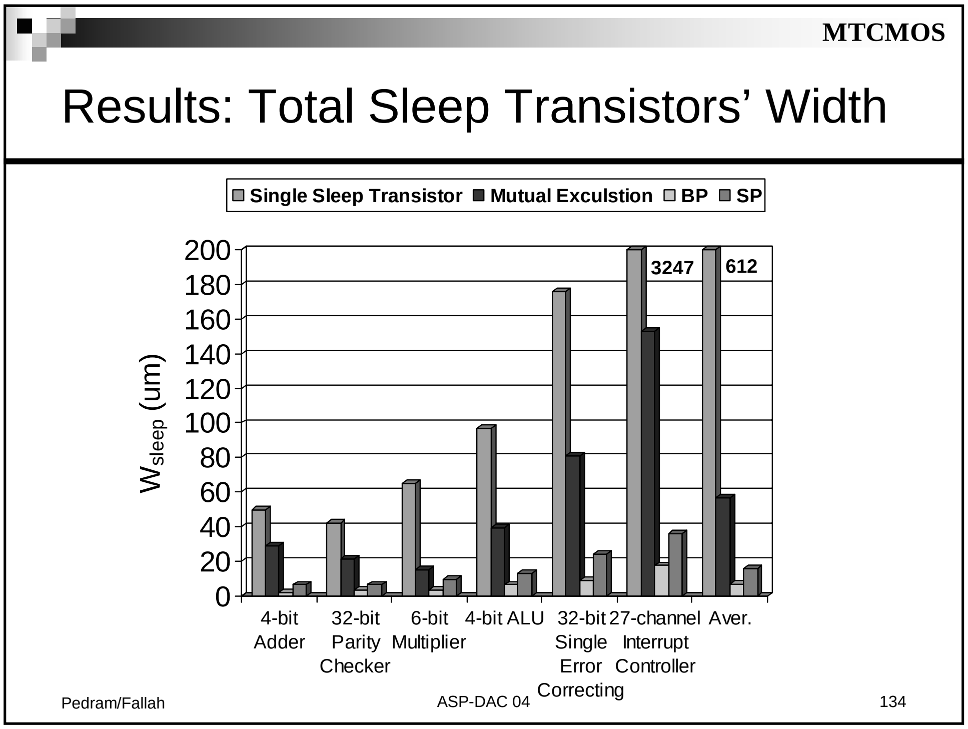




MTCMOS

Results: Dynamic Power Reduction in Active Mode

Pedram/Fallah ASP-DAC 04 133

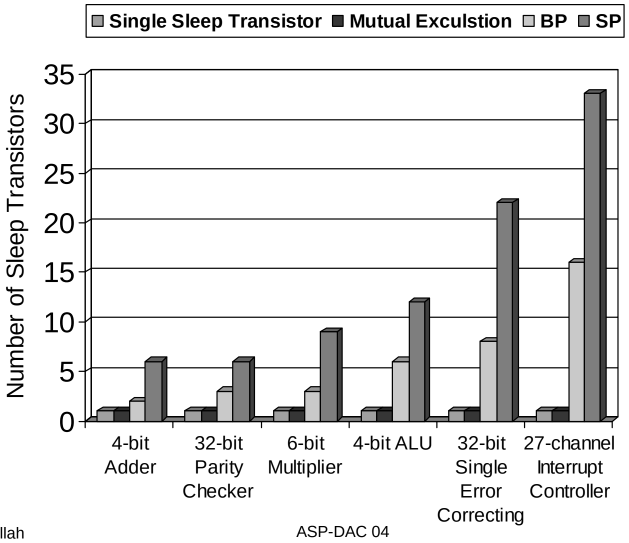


MTCMOS

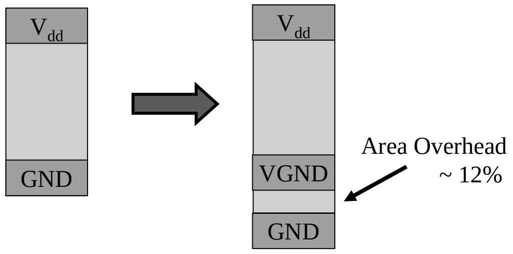
Results: Total Sleep Transistors’ Width

Pedram/Fallah ASP-DAC 04 134

Results: Number of Sleep Transistors



Samsung’s MTCMOS Design Methodology



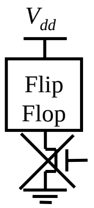
Area Overhead ~ 12%

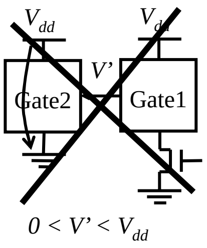
- Design new cells that have sleep transistors.
- Use conventional P&R methodology.

[Won-ISLPED03]

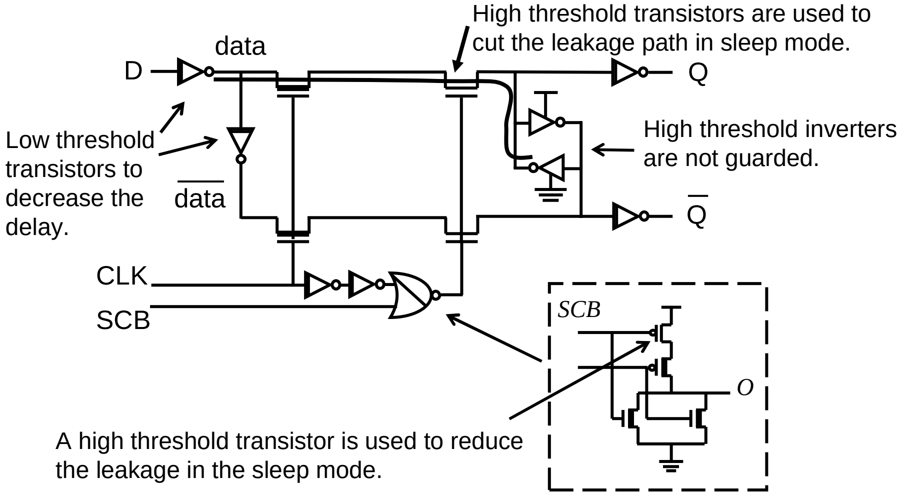
Problems

- MTCMOS cannot be applied to Flip Flops
  - □ Data loss
  - □ Can copy the data to an external memory
    - Delay and dynamic power overhead
    - Energy overhead of external memory
- In an SoC, not all IP blocks are guarded,
  - □ Short circuit current if a guarded output drives a regular input.





Complementary Pass-Transistor Flip Flop (CPFF)



High threshold transistors are used to cut the leakage path in sleep mode.

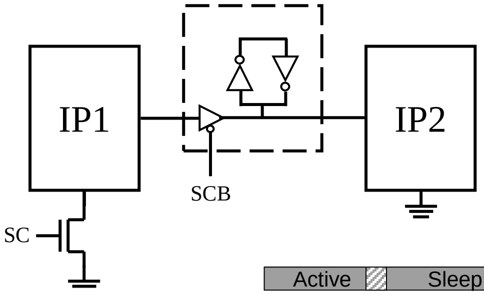
Low threshold transistors to decrease the delay.

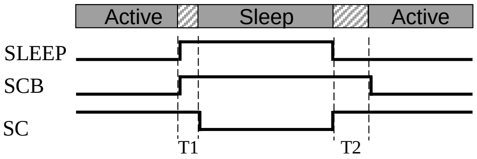
High threshold inverters are not guarded.

A high threshold transistor is used to reduce the leakage in the sleep mode.

Preventing Short Circuit Current

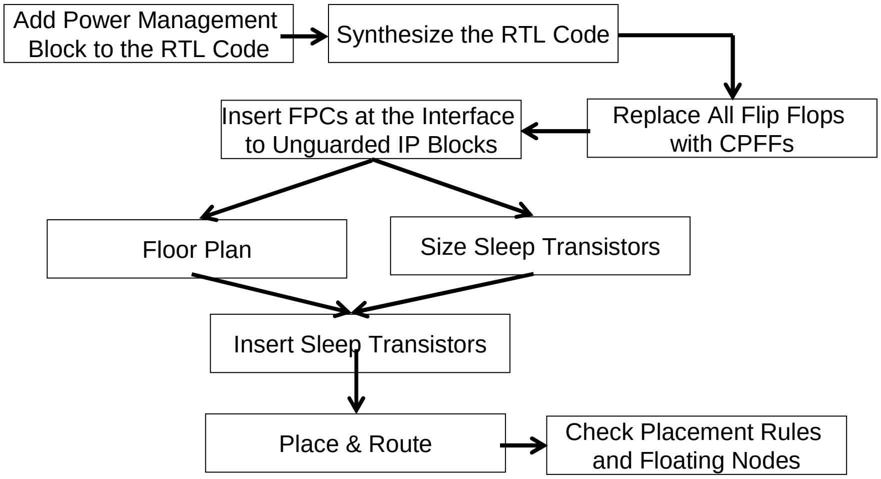
Floating Prevention Circuit





- Store the data in a latch before disconnecting the module from the ground.

Design Flow



Add Power Management Block to the RTL Code

Synthesize the RTL Code

Replace All Flip Flops with CPFFs

Insert FPCs at the Interface to Unguarded IP Blocks

Floor Plan

Size Sleep Transistors

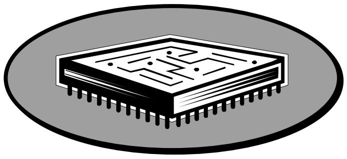
Insert Sleep Transistors

Place & Route

Check Placement Rules and Floating Nodes

DSP Core

- The method was applied to a 16-bit DSP chip
- 0.18μm, V_{dd} = 1.8V
- Inserted 324 sleep transistors with the size of 5μm
- Ground bounce: average=9mV, max=49mV
- Performance degradation = 2%



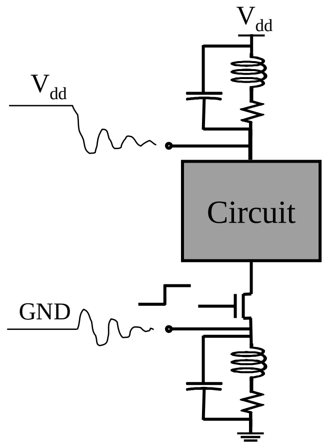
A 32-bit RISC Processor used in a PDA

Chip Size	Process	# Gates	Clock	Total Sleep Transistors Width	Power Dissipation
5.7mm × 5.7mm	0.18μm 5-metal	1,914K	333MHz	18mm	270mW

Leakage Power	Reduction
2μW	6000x

Minimizing Ground Bounce

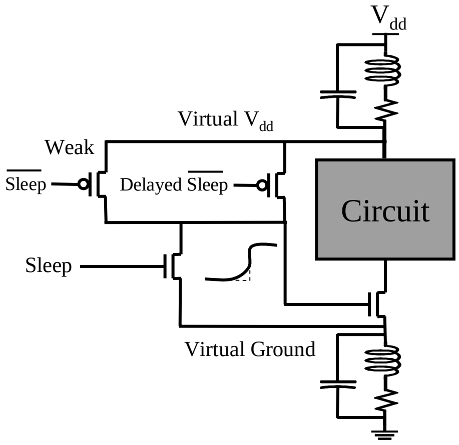
- During sleep period, internal nodes are charged up.
- When the sleep transistor is turned on, there is a current spike flowing to the ground (due to the large V_{ds} of the sleep transistor).
- This creates large V_{dd} and ground noise.



[Kim-ISLPED03]

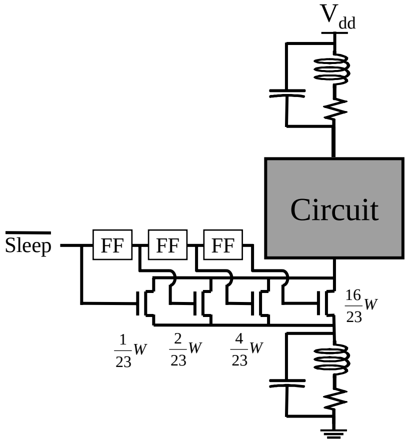
IBM's Solution 1

- Turn-on the sleep transistor at two steps,
  1. Using a weak PMOS: $V_{gs} < V_{dd}$ for the sleep transistor (linear region). Originally, V_{ds} is high.
  2. Using a strong PMOS: $V_{gs} = V_{dd}$ for the sleep transistor (saturation region). V_{ds} is low. Therefore, the peak current reduces.



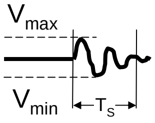
IBM's Solution 2

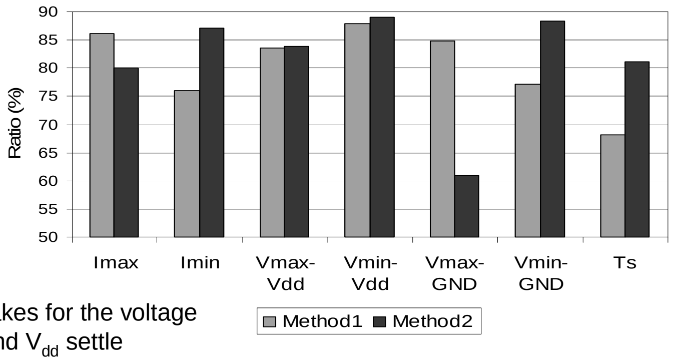
- Use several sleep transistors.
- Turn them on with some delay.
- The resistance between the virtual ground and the ground is reduced when the V_{ds} of the sleep transistor is low. This reduces the peak current.



Results

- Applied to a 16-bit ALU (with a multiplier)
- Designed at 0.13µm, 1.5V, operating at 500MHz.

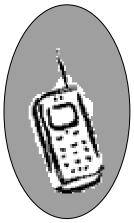




T_s is the time it takes for the voltage of both ground and V_{dd} settle within ±5% of their final values.

Toshiba's Mixed MTCMOS and Dual V_{th} Method

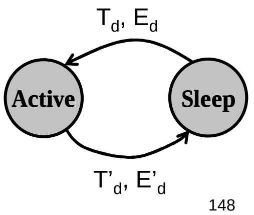
- Used to reduce the leakage power in a DSP core for W-CDMA cell phones
- Cell phones spend a significant amount of time in the standby mode.
  - □ High leakage power
- But they have to exchange some information with base stations every 100ms.



[Usami-ISLPED02]

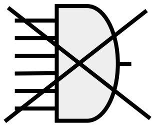
Combining MTCMOS and Dual V_{th}

- Using MTCMOS for the entire circuit means Flip Flop values have to be saved and restored every 100ms.
  - □ Significant delay and power overhead.
- Solution: use MTCMOS for selected cells only (critical path)
  - □ Use Dual V_{th} for other cells including Flip Flops.
- Use one sleep transistor per cell
  - □ Simpler analysis



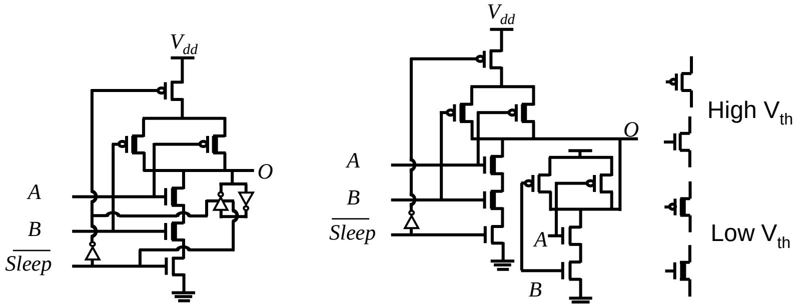
Cell Generation

- Exclude Flip Flops and Latches
- Exclude cells with small drive
  - □ Unlikely to be used on the critical path
- Exclude high fanin gates
  - □ Can be implemented using 2-input gates.
- Develop complex gates to speed up the critical path
- Overall 56 MTCMOS cells were developed using low-V_{th} transistors for logic.



Floating Node Problem

- An MTCMOS gate should not drive a regular gate (static current).
- Use a latch-type or bypass-type gate.
  - □ Note that two sleep transistors are used.

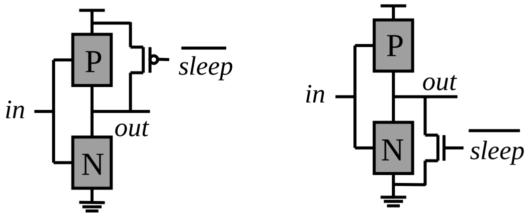


Latch Type

Bypass Type

Another Possible Solution

- Use transistors to pull-up or pull-down outputs of MTCMOS gates
  - □ Smaller number of transistors, but almost the same area overhead.
    - The area overhead is dominated by the sleep transistor size
  - □ Extra switching activity in the circuit every time the circuit goes to the standby mode.

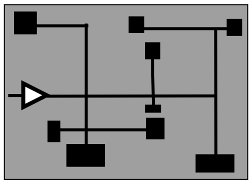


Applying the Technique

- It is not possible to limit conventional tools to use MTCMOS cells for critical paths and high-V_{th} cells for non-critical ones.
- A high-V_{th} circuit was developed first.
- Critical paths were identified.
- Cells on the critical paths were replaced by MTCMOS cells,
  - □ Started from output and continued backward until the timing constraint was met.

Driving Sleep Transistors

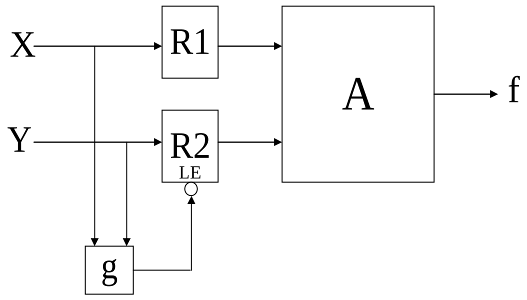
- Many sleep transistors and long wires.
  - □ Electromigration problem, etc.
- A clock-tree-synthesis tool was used to generate a buffer-tree.
  - □ Only tree-construction and buffer placement, no skew issue.



Experimental Setup and Result

- Applied to a 34K-cell module.
- High-V_{th}=0.55V, Low-V_{th}=0.35V, 0.18µm, V_{dd}=1.5V @ 100MHz.
- 30 out of 53 levels of gates on the critical path were replaced by MTCMOS cells to meet the timing constraint.
  - □ Reduction from 10.27ns to 8.85ns (a 14% improvement).
- 12% of total cells were replaced with MTCMOS cells.
  - □ Area overhead=10%.
- Leakage at 85°C,
  - □ Active mode: 86µA
  - □ Standby mode: 28µA ≅ leakage of a high-V_{th} design

Precomputation



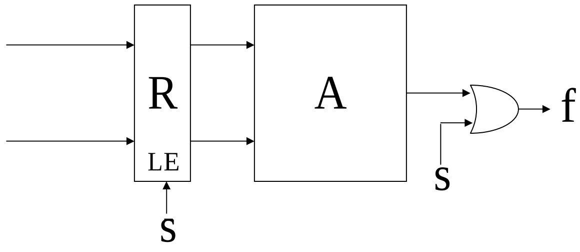
Basic idea:

Freeze some of the inputs when a specific condition on input values holds.

If f is independent of Y, then freeze Y.

Goals: Minimize the size of g, maximize |Y| and the likelihood of the condition happening.

Guarded Evaluation

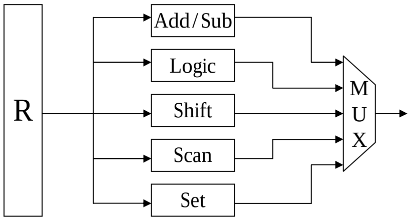


$$(s=1) \Rightarrow (f=1)$$

- Disabling gates that perform redundant computation.
- Select an existing signal s instead of generating a new signal.

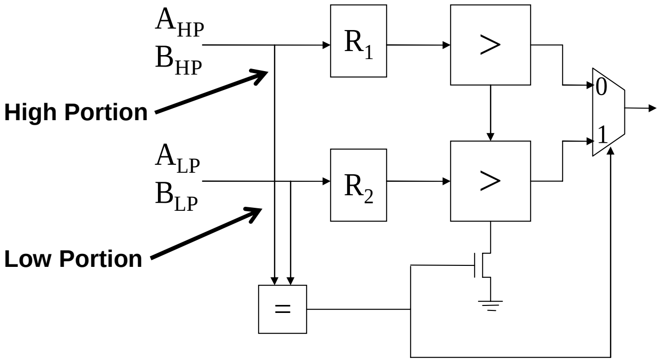
Precomputation-based Guarding

- A combination of precomputation and MTCMOS (ground gating).
- Reduces both switching and leakage power
- Can be used to disable part of a register as well.
- Solves the input sharing problem
  - □ There is switching activity in all modules, but at each cycle the output of only one module is used.



[Abdollahi-ICCD03]

Comparators



If A_{HP}>B_{HP} , then A>B

If A_{HP}<B_{HP} , then A<B

A = 1001,1010 0101,0100

B = 0001,1010 1101,0100

Adders

- Partition the adder into HP and LP.
- The goal is to disable the HP when there is no need to it.
- If the sign extension part of operands exceed the HP range, it is disabled.

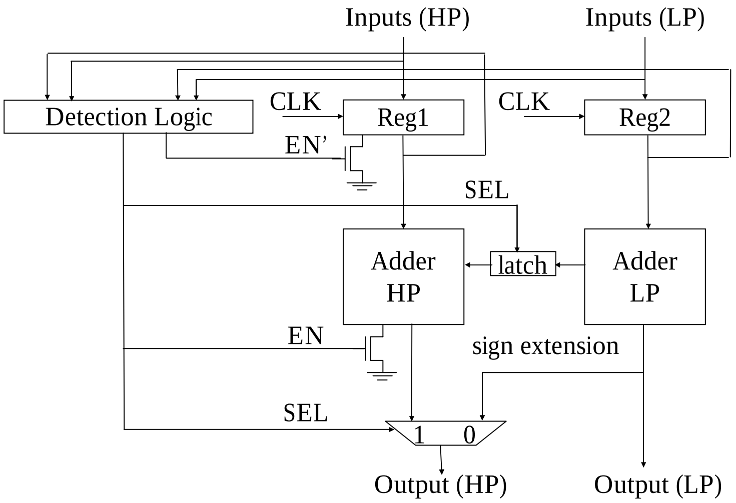
	HP	LP
A	= 1111,1111	1101,0011
B	= 0000,0000	0011,0110
Sum	= 0000,0000	0000,1001

Choosing the HP and the LP

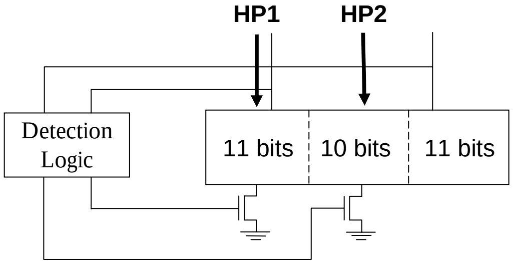
- Increasing the size of the HP (i.e., the number of bits),
  - □ decreases the number of times it can be disabled.
  - □ Increases the amount of power that can be saved each time the portion is disabled.
- There is a tradeoff.

	HP	LP	HP	LP	HP	LP
	1111,1111	**0001,0011**	**1111,111100**	**01,0011**	**1111,111**	**10001,0011**
	0000,0000	**0111,0110**	**0000,000001**	**11,0110**	**0000,000**	**00111,0110**
# bits disabled:	8		10		7	
# times disabled:	1		0		1	

Adders

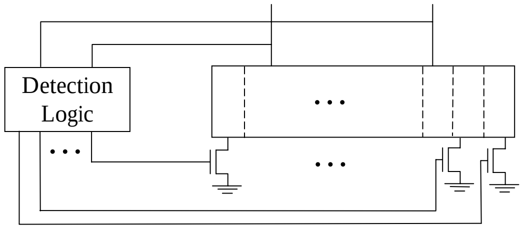


Hybrid Guarding



- Partition the HP to two or more segments.
- Disable one or more segments based on the input data.
- More saving at the cost of more complexity.

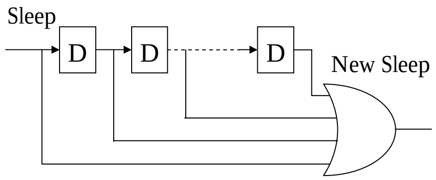
Dynamic Guarding



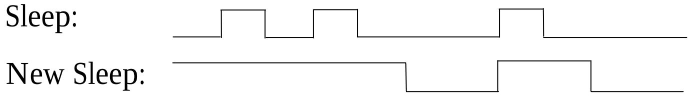
- Number of partitions = number of bits
- Dynamically detect the sign extension length and disable bits.
- At each cycle the maximum possible number of bits are disabled.

Reducing the Switching Activity

- Turning the sleep transistor on and off consumes dynamic power.
  - □ Turn-off the sleep transistor only if it is going to be off for a long period.
  - □ Predict the future behavior based on previous cycles behavior.



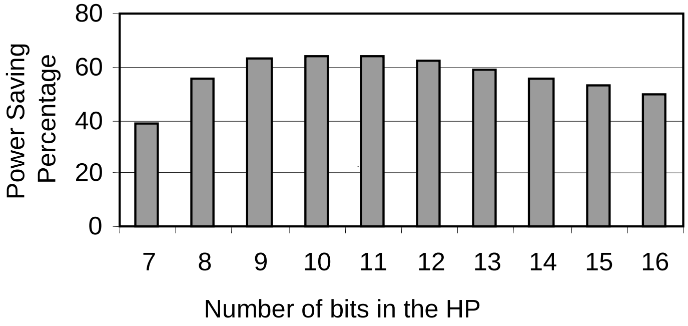
Low Pass Filter



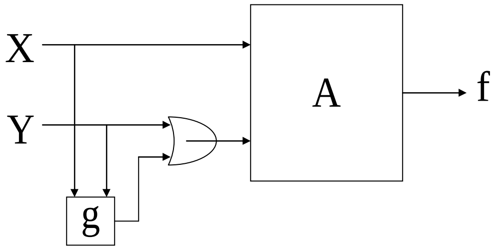
Experimental Setup

- 32-bit functional units
- Process Technology
  - □ BSIM 70nm
  - □ V_{dd} = 0.9V
  - □ NMOS V_{th} = 0.2V
  - □ PMOS V_{th} = -0.22V
  - □ Sleep transistors' V_{th} = 0.5V
- Test Bench
  - □ Thousand vectors corresponding to ALU unit in the data-path of a processor executing a JPEG decoder program

Results for a Comparator



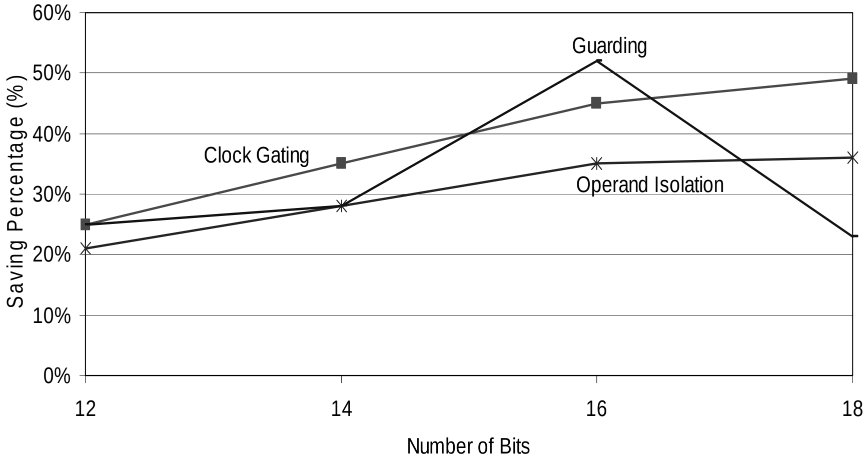
Operand Isolation

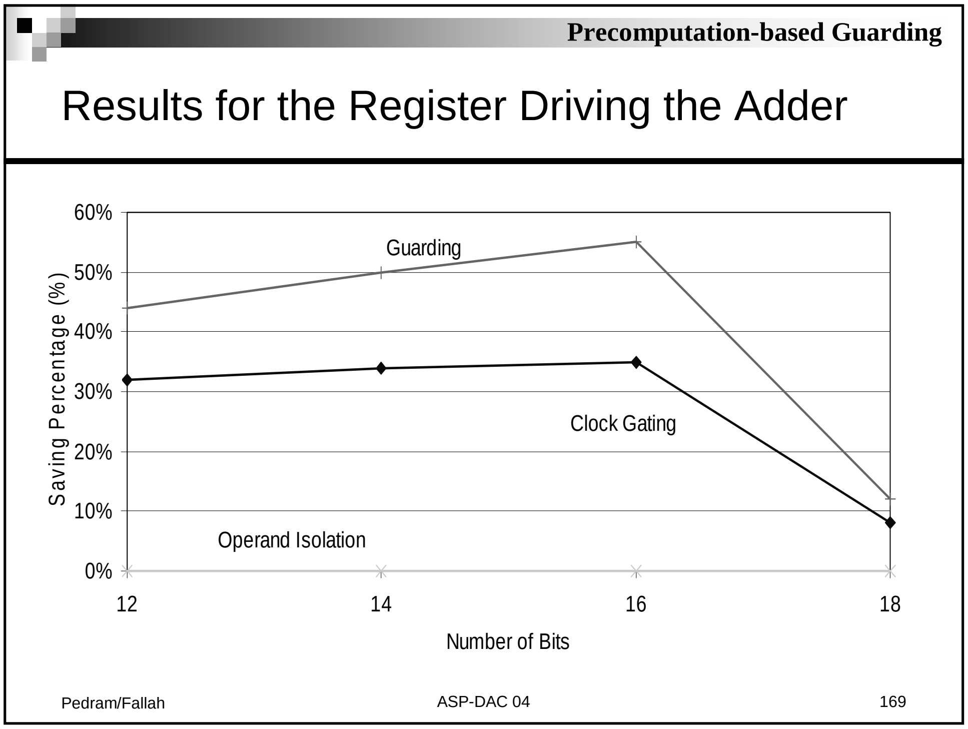


- If f is independent of Y, then set g to 1.
- May freeze Y for several consecutive cycles.
- Alternatively, an AND gate or a Multiplexer can be used.

[When-DATE00]

Results for an Adder

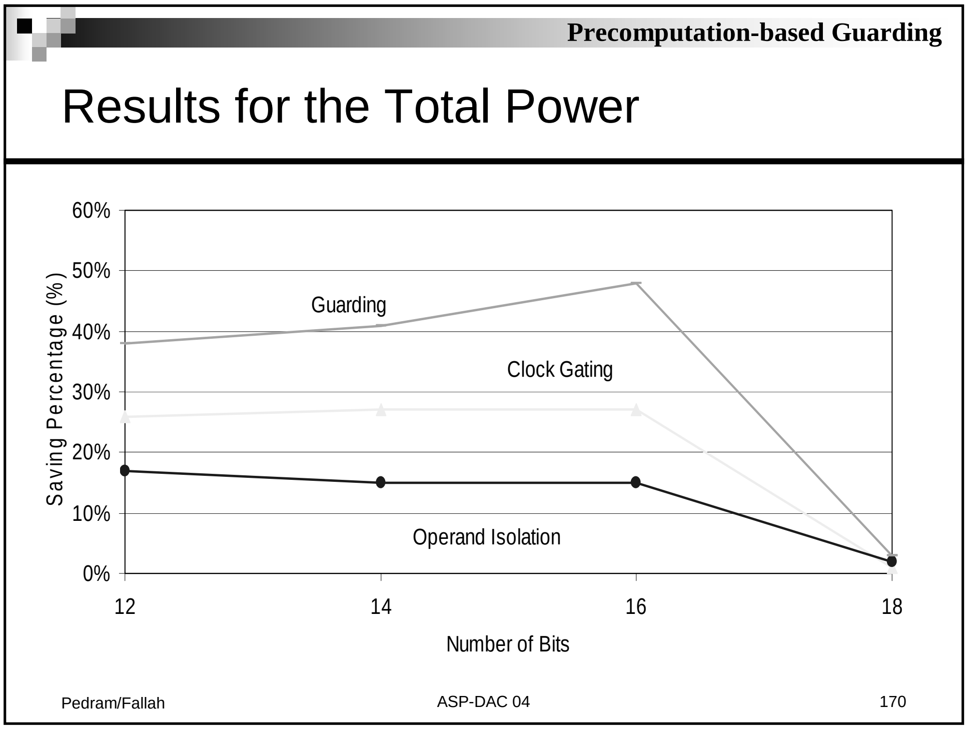




Precomputation-based Guarding

Results for the Register Driving the Adder

Pedram/Fallah ASP-DAC 04 169



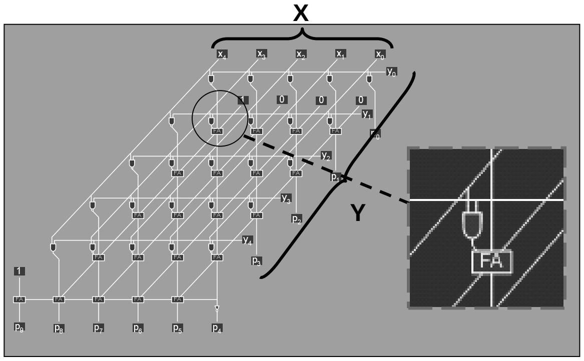
Precomputation-based Guarding

Results for the Total Power

Pedram/Fallah ASP-DAC 04 170

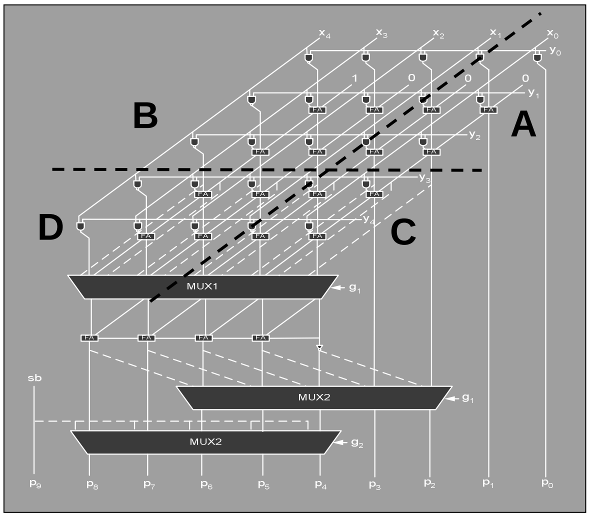
Multipliers

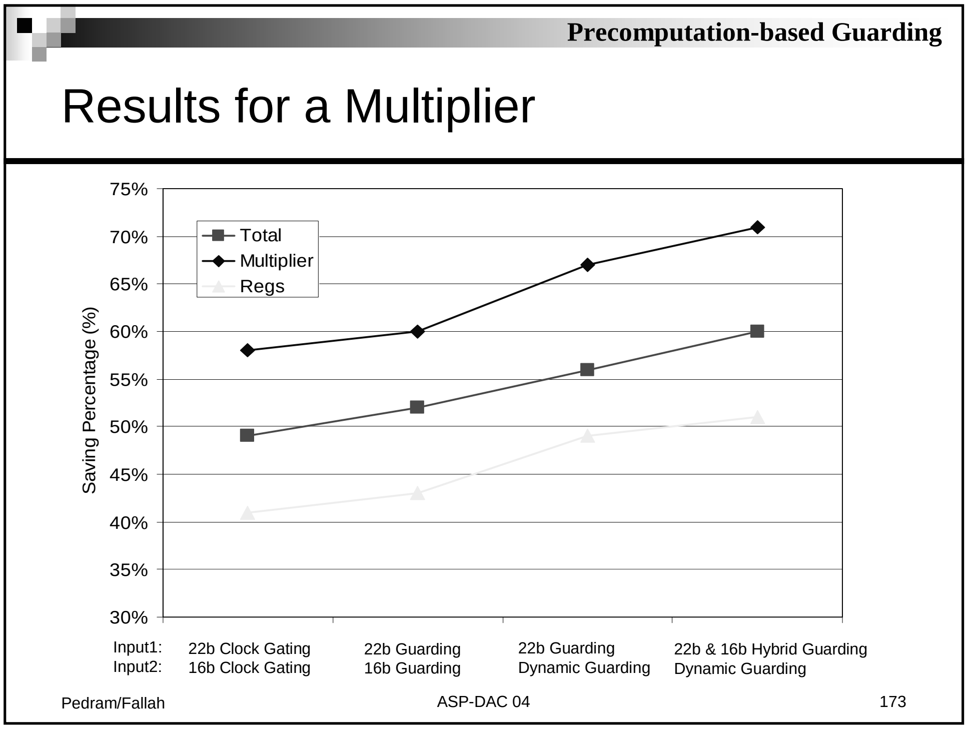
- A two-dimension array of adders.
- Sign extension of X can be used to disable left part of the array.
- Sign extension of Y can be used to disable bottom part of the array.



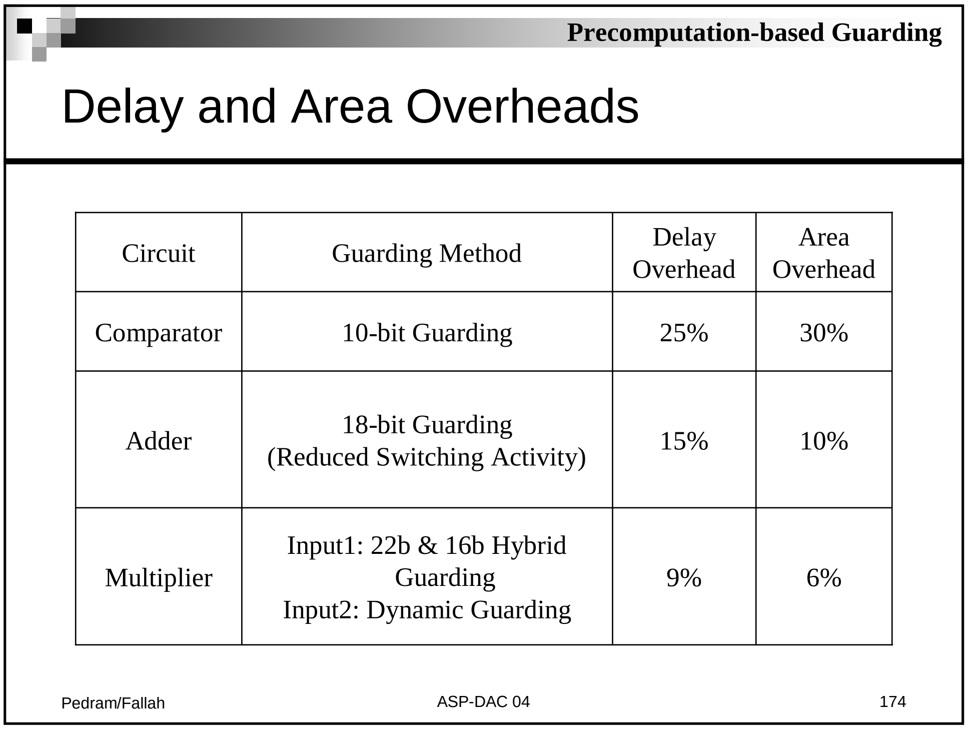
Two Dimensional Guarding

- The multiplier is partitioned to four segments.
- Segment A is always active.
- Segment D is active if both B and C are active.



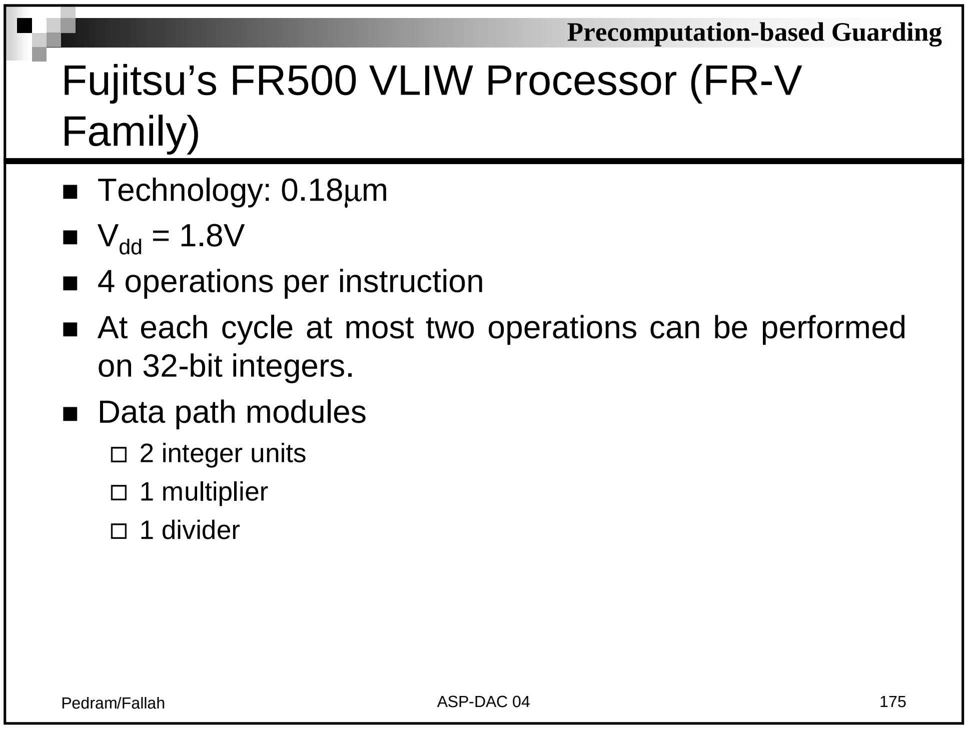


Results for a Multiplier



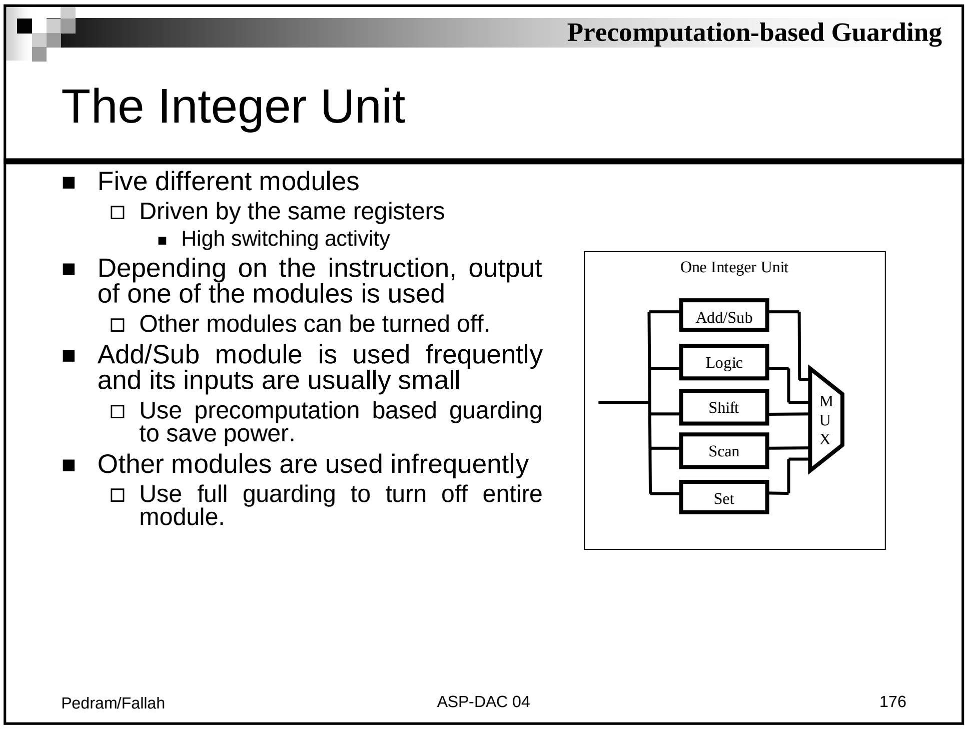
Delay and Area Overheads

Circuit	Guarding Method	Delay Overhead	Area Overhead
Comparator	10-bit Guarding	25%	30%
Adder	18-bit Guarding (Reduced Switching Activity)	15%	10%
Multiplier	Input1: 22b & 16b Hybrid Guarding Input2: Dynamic Guarding	9%	6%



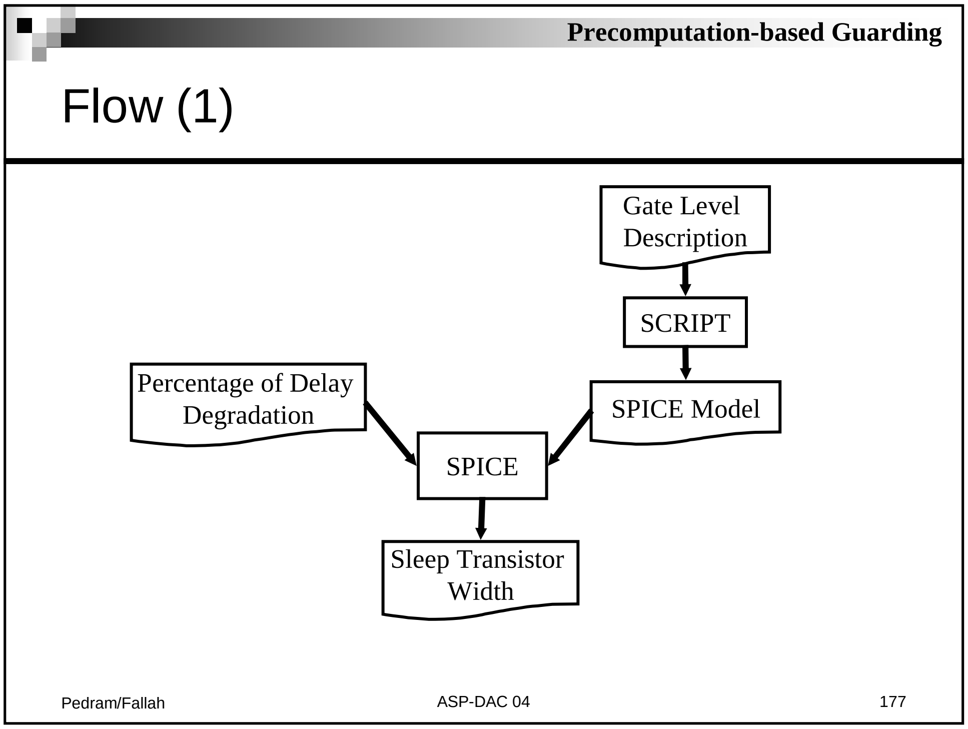
Fujitsu’s FR500 VLIW Processor (FR-V Family)

- Technology: 0.18µm
- V_{dd} = 1.8V
- 4 operations per instruction
- At each cycle at most two operations can be performed on 32-bit integers.
- Data path modules
  - □ 2 integer units
  - □ 1 multiplier
  - □ 1 divider



The Integer Unit

- Five different modules
  - □ Driven by the same registers
    - High switching activity
- Depending on the instruction, output of one of the modules is used
  - □ Other modules can be turned off.
- Add/Sub module is used frequently and its inputs are usually small
  - □ Use precomputation based guarding to save power.
- Other modules are used infrequently
  - □ Use full guarding to turn off entire module.



Precomputation-based Guarding

Flow (1)

Gate Level Description

SCRIPT

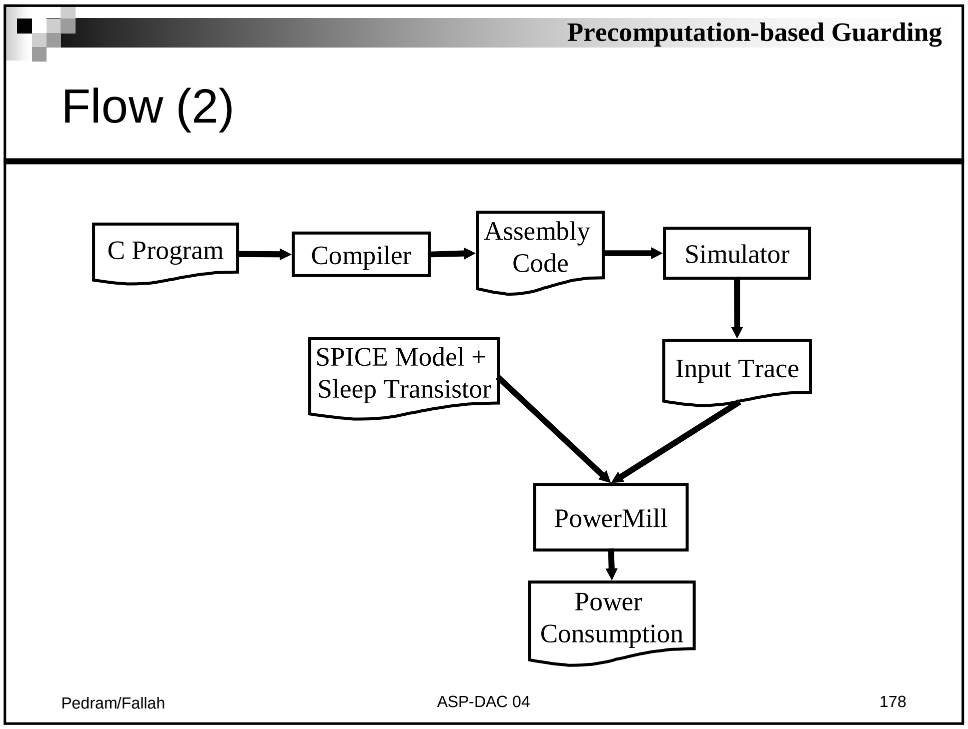
SPICE Model

Percentage of Delay Degradation

SPICE

Sleep Transistor Width

Pedram/Fallah ASP-DAC 04 177



Precomputation-based Guarding

Flow (2)

C Program

Compiler

Assembly Code

Simulator

Input Trace

SPICE Model + Sleep Transistor

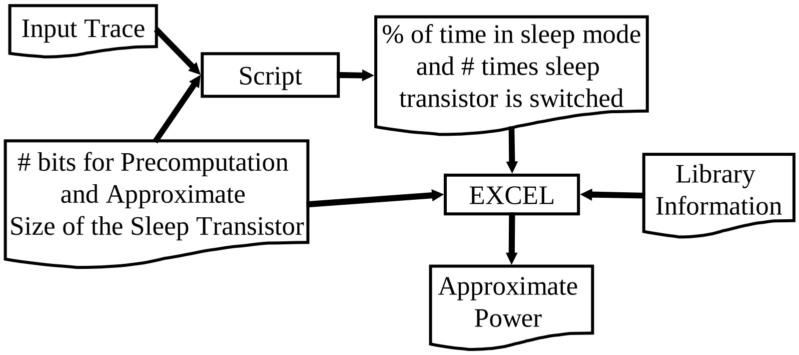
PowerMill

Power Consumption

Pedram/Fallah ASP-DAC 04 178

Fast Power Estimation for Add/Sub

- Can be used instead of PowerMill to speed up the search for optimum number of precomputation bits.


Input Trace
Script
% of time in sleep mode and # times sleep transistor is switched
# bits for Precomputation and Approximate Size of the Sleep Transistor
EXCEL
Library Information
Approximate Power

Results for the Integer Unit

Method	Power Saving	Area Overhead
Precomputation-based Guarding	81%	9%
Operand Isolation	58%	11%
Clock Gating + Operand Isolation	61%	14%

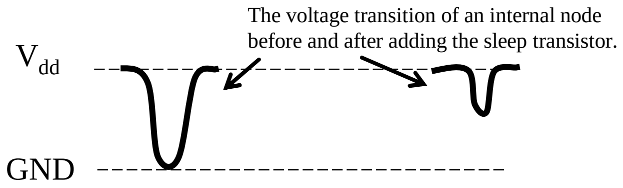
- The large slack time of some modules was used to decrease the size of their sleep transistors.
- Delay overhead = 12%
  - May not be important as the critical path usually corresponds to memory read stage.
- Integer units are the hot spots of processors. Decreasing their temperature helps to simplify packaging and cooling system.

Advantages of the Method

- Decreases both dynamic and leakage power.
- Can be applied to circuits that other techniques cannot handle easily.
- Higher power saving.
- Lower area overhead.

Additional Benefit

- Sleep transistors can decrease the dynamic power even when they are on,
  - □ Because of decreasing the glitches on internal nodes.
  - □ About 9% in Add/Sub module.
- Sleep transistors can increase the speed of low-to-high transition of some nodes of the circuit.



The voltage transition of an internal node before and after adding the sleep transistor.

Guidelines (1)

- To achieve a very low leakage use MTCMOS method.
- If the size of the circuit or the number of gates that will be guarded is small, use one sleep transistor per cell/gate.
  - □ Higher area overhead, but less complexity.
- To achieve the best result, carefully analyze the current profile of gates,
  - □ 20X improvement in leakage saving
  - □ 38X improvement in area overhead

Guidelines (2)

- To reduce potential problems (e.g., routing, area overhead, and design complexity) use MTCMOS for critical-path gates and Dual-V_{th} method for other gates.
  - □ Note: guarded gates should not drive non-guarded ones.
- If reducing both leakage and dynamic power are important, use precomputation-based guarding.

Impact of Well Bias on the Leakage

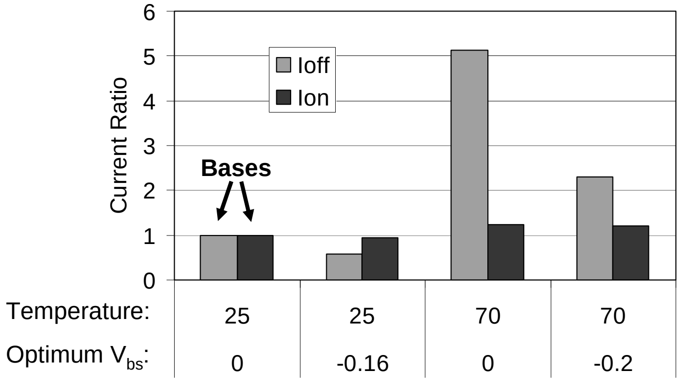
$$I_{sub} = A \times e^{\frac{1}{mv_T}(V_G - V_S - V_{th0} - \gamma' V_{bs} + \eta V_{ds})} \times (1 - e^{-\frac{V_{ds}}{V_T}}) \ \ where,$$

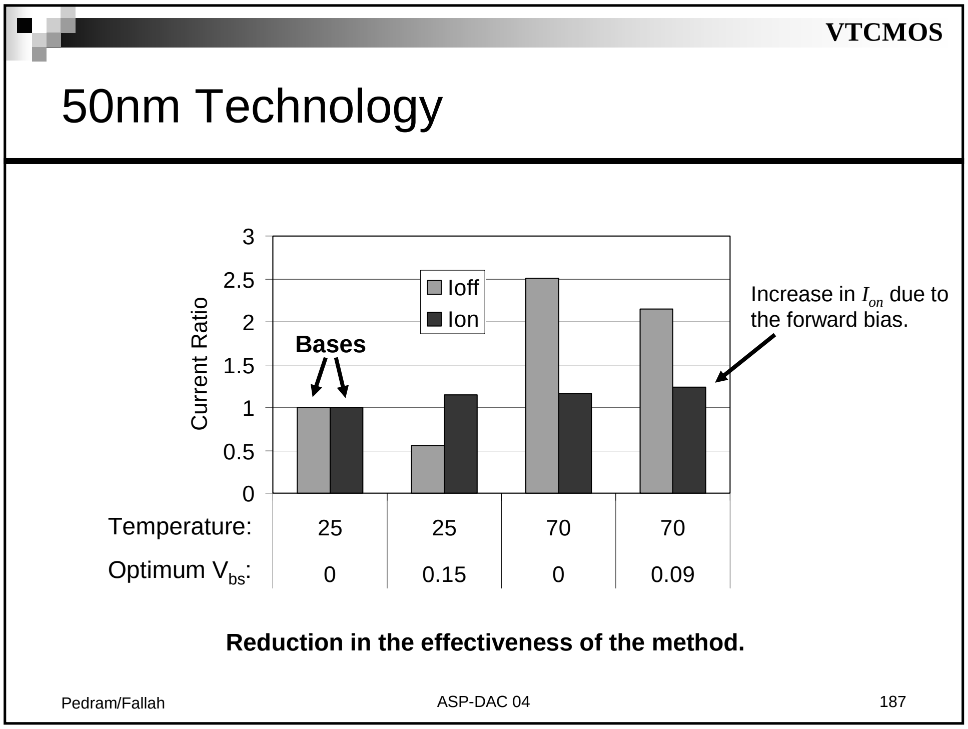
$$A = \mu_0 C'_{ox} \frac{W}{L_{eff}} (v_T)^2 e^{1.8} e^{-\frac{\Delta V_{th}}{\eta V_T}}$$

V_T is the thermal voltage,
V_{app} is the applied reverse body bias,
η is the DIBL coefficient,
C_{ox} is the gate oxide capacitance,
μ_0 is the zero bias mobility,
m is the sub-threshold swing coefficient of the transistor,
ΔV_{th} is a term introduced to account for transistor-to-transistor leakage variations.

[Roy-ISLPED03]

70nm Technology

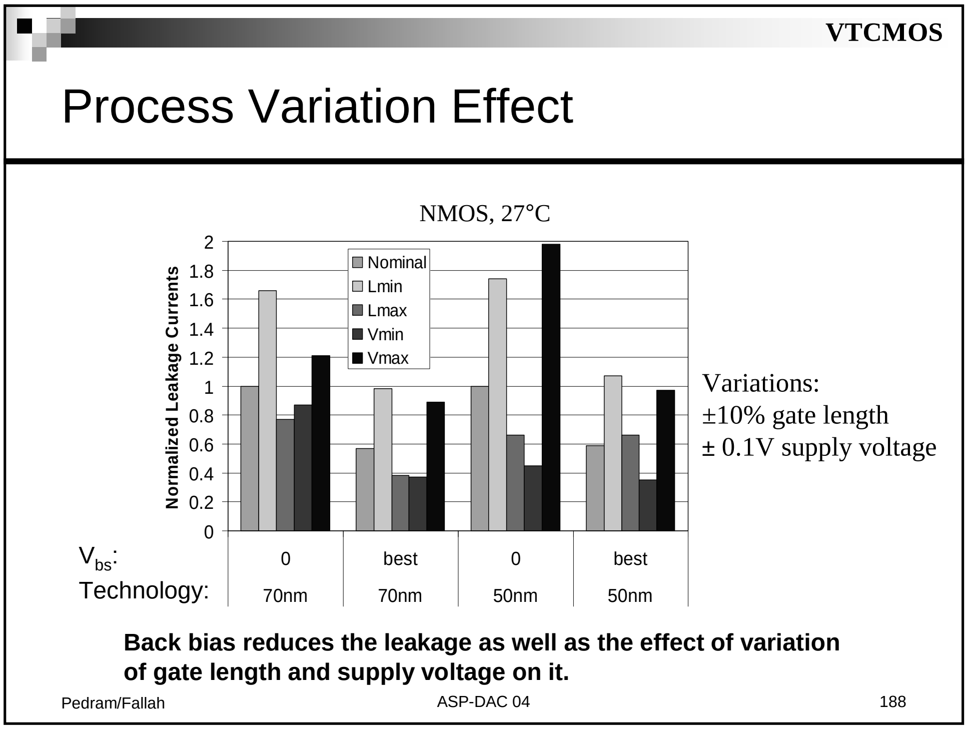




50nm Technology

Increase in I_{on} due to the forward bias.

Reduction in the effectiveness of the method.



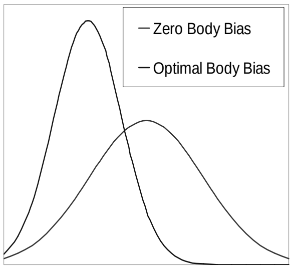
Process Variation Effect

NMOS, 27°C

Variations:
±10% gate length
± 0.1V supply voltage

Back bias reduces the leakage as well as the effect of variation of gate length and supply voltage on it.

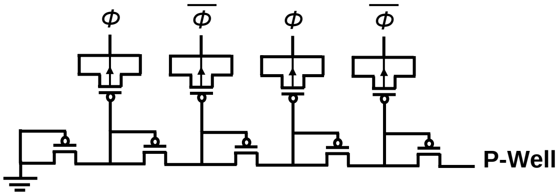
Leakage Distribution Improvement



- Channel length: Gaussian distribution μ=50nm, σ=2.5nm
- Both the mean and the standard deviation of leakage values reduced by 41%,
  - □ Similar results when changing doping profile and supply voltage.
  - □ Good when testing chips

How to Change the Substrate Voltage

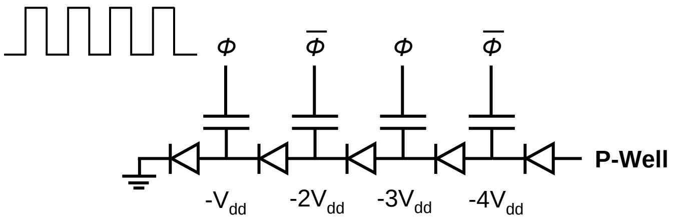
- Use a charge pump to generate a variable voltage.



[Kim-DATE02]

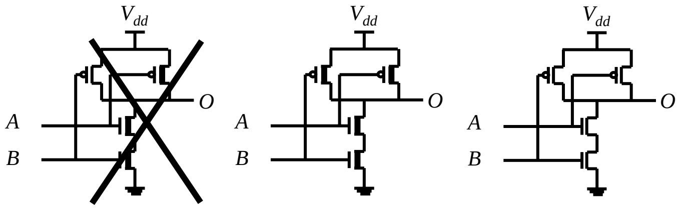
Charge Pump Equivalent Circuit

- Shift the charge from the P-well to the ground.
- Change the frequency of *Φ*, to change the voltage of the well.

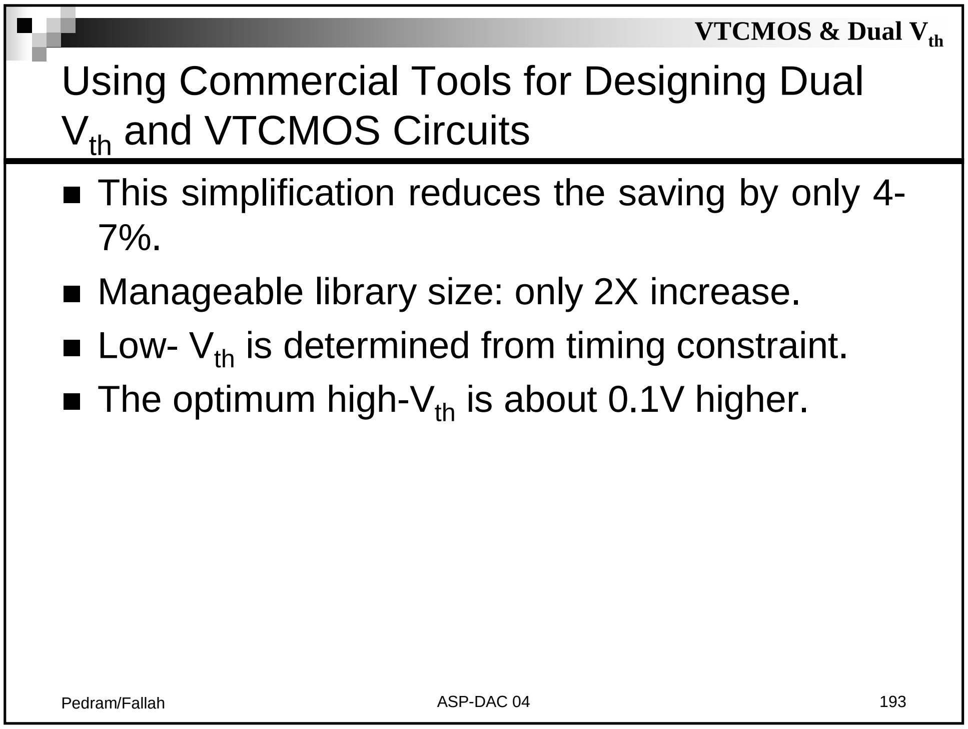


Using Commercial Tools for Designing Dual V_{th} and VTCMOS Circuits

- No need to use library cells which have all combinations of V_{th}'s.
- Significant leakage saving can be achieved by using all low-V_{th} and all high-V_{th} gates only.

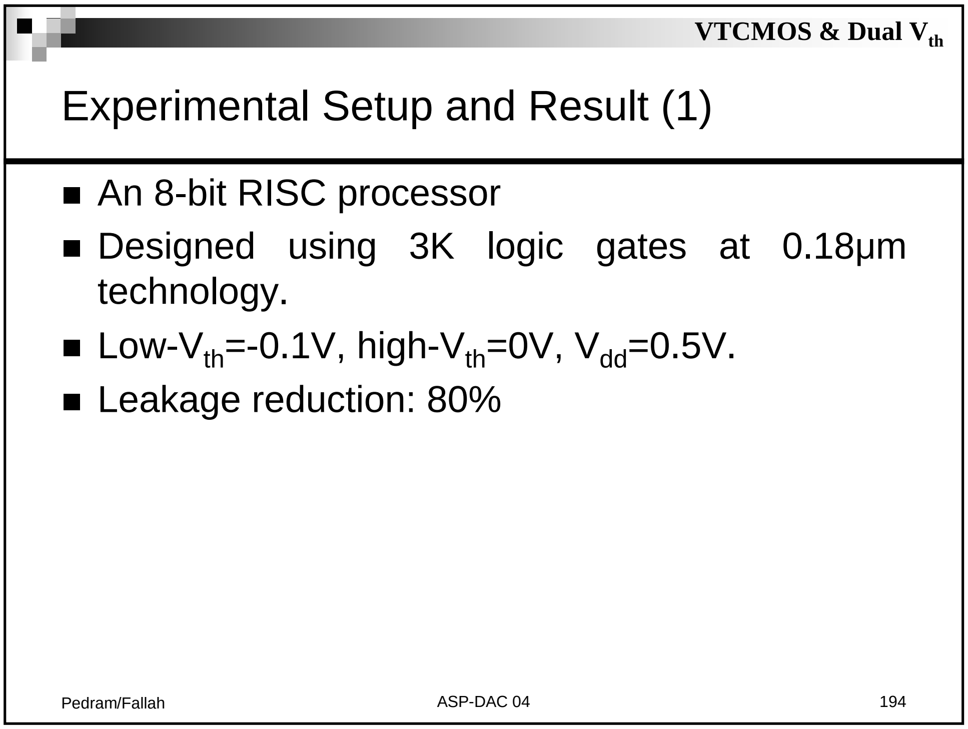


[Sakurai-ISLPED01]



Using Commercial Tools for Designing Dual V_{th} and VTCMOS Circuits

- This simplification reduces the saving by only 4-7%.
- Manageable library size: only 2X increase.
- Low- V_{th} is determined from timing constraint.
- The optimum high-V_{th} is about 0.1V higher.



Experimental Setup and Result (1)

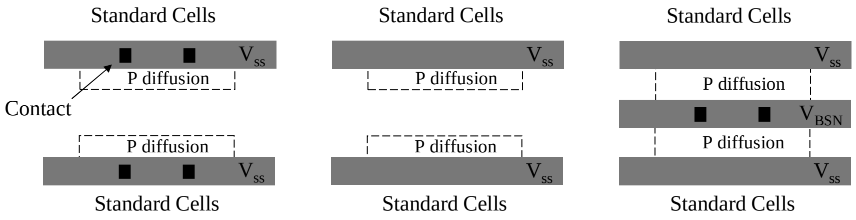
- An 8-bit RISC processor
- Designed using 3K logic gates at 0.18μm technology.
- Low-V_{th}=-0.1V, high-V_{th}=0V, V_{dd}=0.5V.
- Leakage reduction: 80%

Experimental Setup (2)

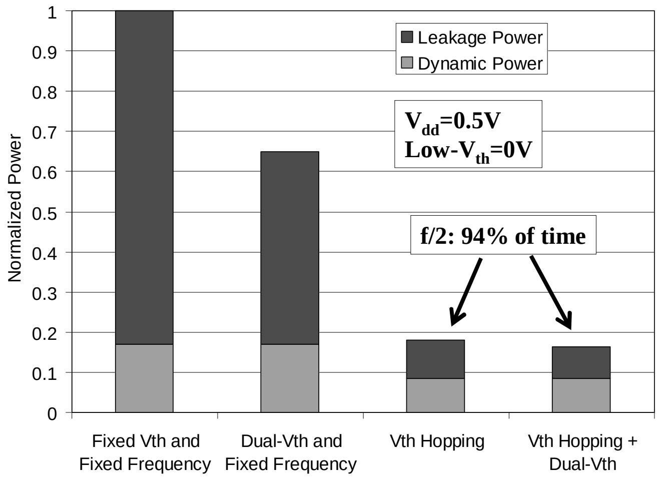
- In order to improve saving, VTCMOS method can be combined with Dual V_{th} technique.
  - □ Threshold voltage of transistors can be changed while in sleep mode or in active mode (clock frequency has to be decreased).
- The technique was applied to an MPEG-4 encoder.
- Place & route (P&R) was done using a commercial tool,
  - □ In order to add metal lines to control back-bias, the cells were placed with extra space between them.
- After that, substrate/well contacts were modified.

Experimental Setup (3)

- Next, well contacts on the V_{dd} line and substrate contacts on the ground line were removed by using a script.
- Finally, the n-well and p-well patterns were added between the cells.
- Area overhead: 9%



Power Comparison



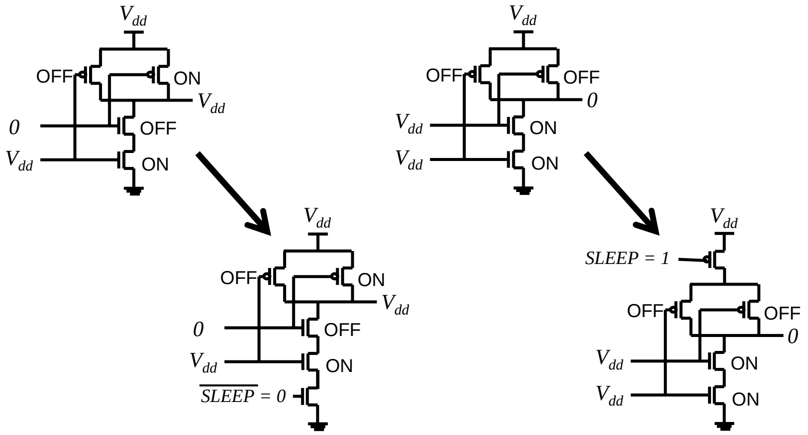
V_{dd}=0.5V
Low-V_{th}=0V

f/2: 94% of time

Guidelines

- The optimum value for high-V_{th} is about 0.1V more than low-V_{th}.
- If clock frequency can be reduced, VTCMOS is better than dual-V_{th} technique because it reduces the leakage of gates in critical path as well.
- VTCMOS is good for improving yield.
- Using VTCMOS and dual-V_{th} techniques together is not a good idea.

Transistor Stacks in Single Threshold CMOS- The Idea

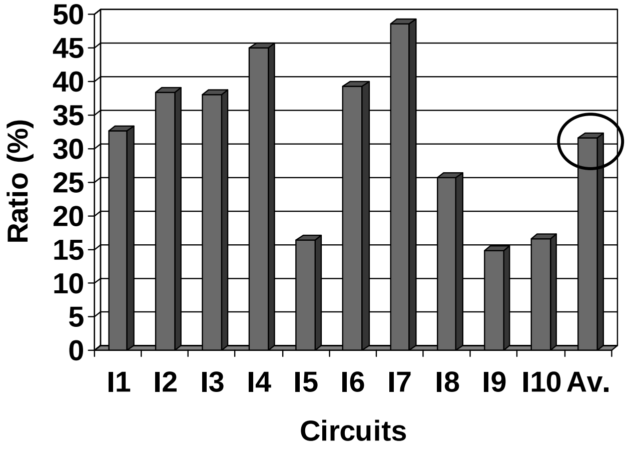


[Roy-DAC99-B]

Transistor Stacks in Single Threshold CMOS- The Algorithm

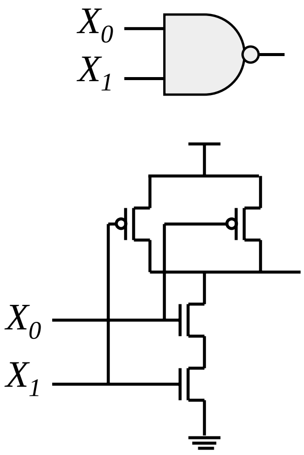
1. Find the minimum leakage vector using a heuristic.
2. Perform critical path and slack analysis.
3. Choose a gate which is in high leakage state (and not on the critical path).
4. Output = 1 → add an NMOS sleep transistor
5. Output = 0 → add a PMOS sleep transistor
6. Repeat steps 3-5.

Leakage of New Technique vs. Applying Minimum Leakage Vector Only



MCNC Benchmark

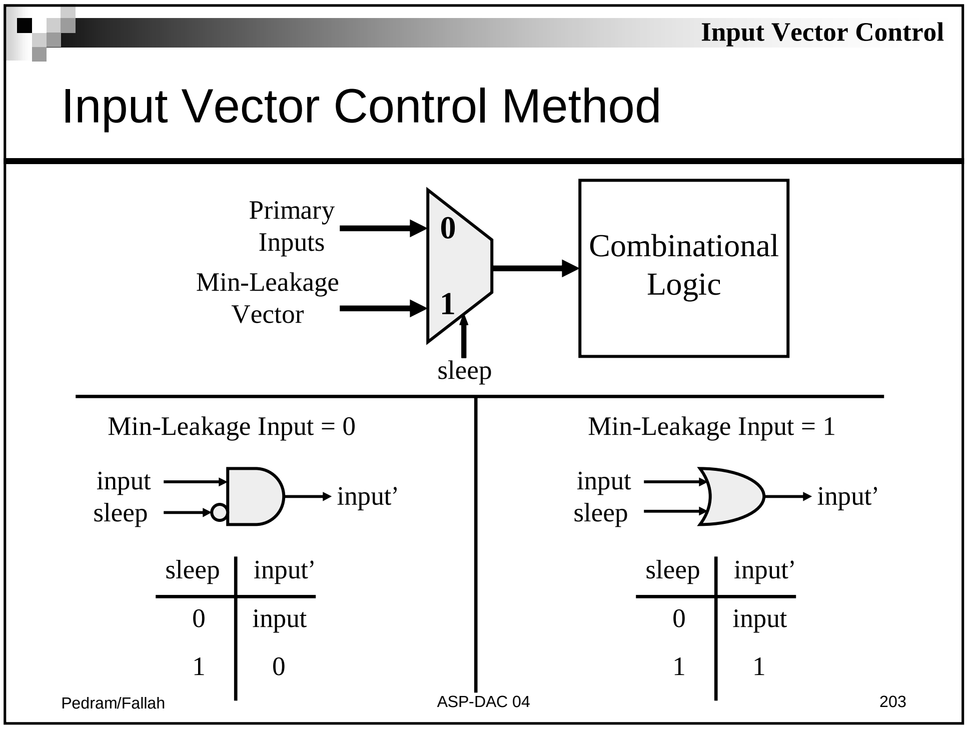
Input Dependence of the Leakage Current



Technology: 0.18 µm
Supply Voltage = 1.5V
Threshold Voltage = 0.2V

X_0 X_1	Leakage
0 0	23.60 nA
0 1	51.42 nA
1 0	47.15 nA
1 1	82.94 nA

[Abdollahi-ISLPED02]



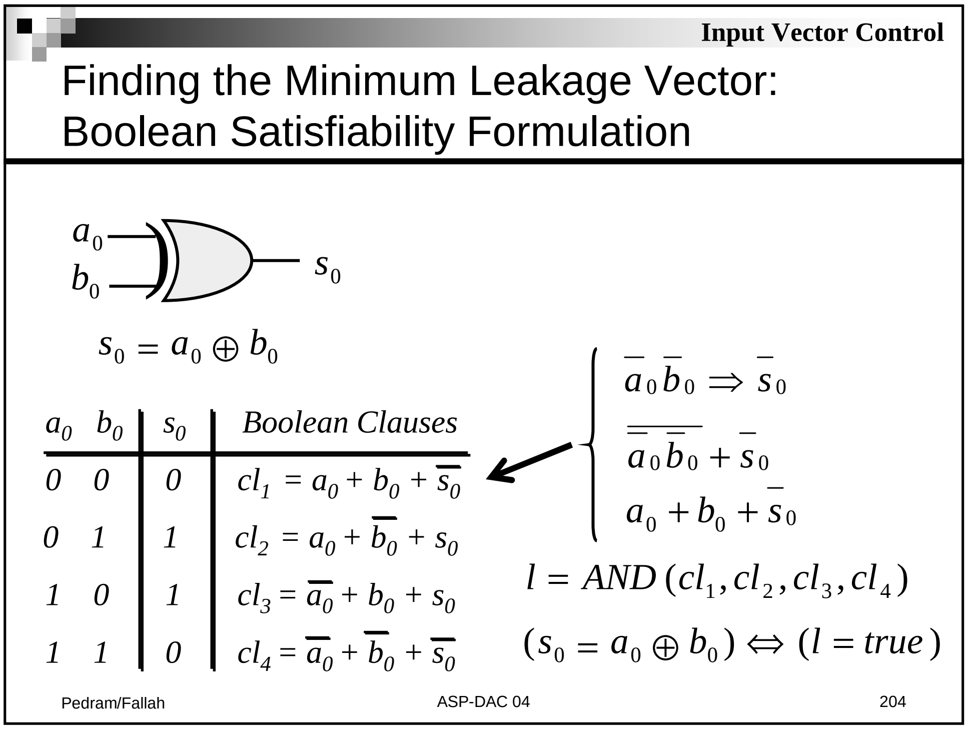
Input Vector Control Method

Min-Leakage Input = 0

sleep	input'
0	input
1	0

Min-Leakage Input = 1

sleep	input'
0	input
1	1



Finding the Minimum Leakage Vector: Boolean Satisfiability Formulation

$$s_0 = a_0 \oplus b_0$$

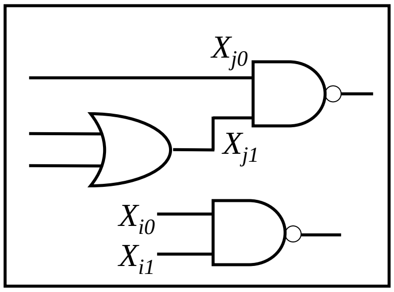
a_0	b_0	s_0	*Boolean Clauses*
0	0	0	$cl_1 = a_0 + b_0 + \overline{s_0}$
0	1	1	$cl_2 = a_0 + \overline{b_0} + s_0$
1	0	1	$cl_3 = \overline{a_0} + b_0 + s_0$
1	1	0	$cl_4 = \overline{a_0} + \overline{b_0} + \overline{s_0}$

$$\left\{ \begin{array}{l} \bar{a}_0 \bar{b}_0 \Rightarrow \bar{s}_0 \\ \overline{\bar{a}_0 \bar{b}_0} + \bar{s}_0 \\ a_0 + b_0 + \bar{s}_0 \end{array} \right.$$

$$l = AND\,(cl_1, cl_2, cl_3, cl_4)$$

$$(s_0 = a_0 \oplus b_0) \Leftrightarrow (l = true)$$

Computing Leakage



0.18 µm

V_{DD} = 1.5V

V_T = 0.2V

X_0 X_1	Leakage	
0 0	23.60 nA	L_{00}
0 1	51.42 nA	L_{01}
1 0	47.15 nA	L_{10}
1 1	82.94 nA	L_{11}

-Quantize the leakage values to *k* levels.

$$D_{00}^{j} = \overline{X}_{j1}\overline{X}_{j0} \quad D_{01}^{j} = \overline{X}_{j1}X_{j0} \quad D_{10}^{j} = X_{j1}\overline{X}_{j0} \quad D_{11}^{j} = X_{j1}X_{j0}$$

$$Leakage\ (X_j) = D_{00}^{j}L_{00} + D_{01}^{j}L_{01} + D_{10}^{j}L_{10} + D_{11}^{j}L_{11}$$

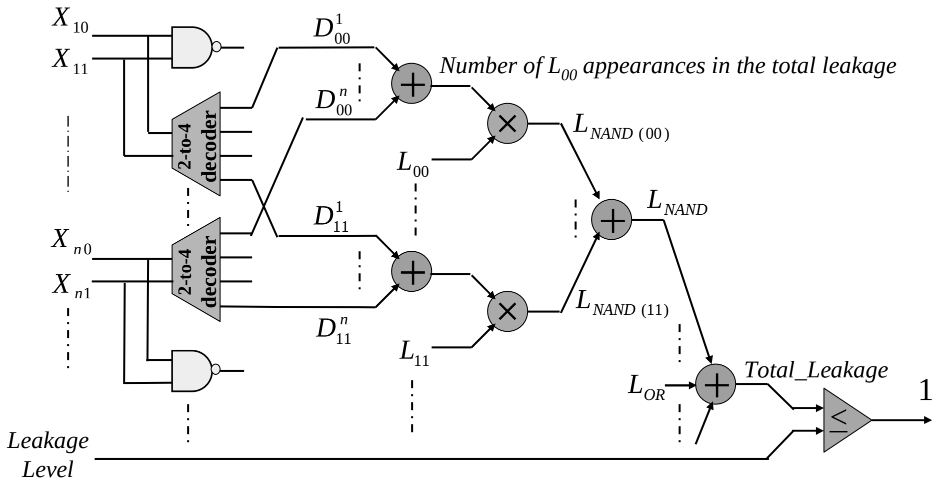
Decreasing the Number of Additions

$$\left[\begin{array}{c}\text{Contribution of all NAND}\\ \text{gates to the total leakage}\end{array}\right] = L_{NAND} = \sum_{j=1}^{n} Leakage_{NAND}(X_j)$$

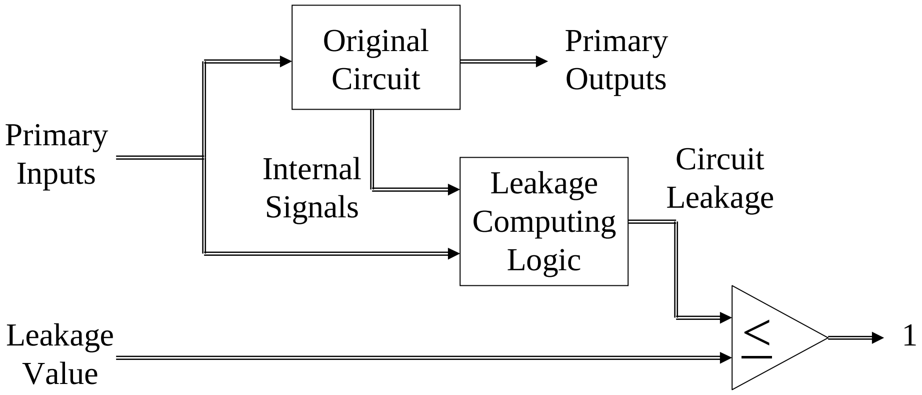
$$L_{NAND} = \underbrace{\sum_{j=1}^{n}(\overset{\text{one bit}}{D_{00}^{j}}\overbrace{L_{00}}^{m\ bits})}_{(n\text{-}1)m\ single\text{-}bit\ additions} + \sum_{j=1}^{n}(D_{01}^{j}L_{01}) + \sum_{j=1}^{n}(D_{10}^{j}L_{!0}) + \sum_{j=1}^{n}(D_{11}^{j}L_{11})$$

$$L_{NAND} = \underbrace{(\overbrace{\sum_{j=1}^{n} D_{00}^{j}}^{log\ n\ \ bits})\overbrace{L_{00}}^{m\ bits}}_{(n\ \text{-}1+\ m\ log\ n)\ \ single\text{-}bit\ additions} + (\sum_{j=1}^{n} D_{00}^{j})L_{00} + (\sum_{j=1}^{n} D_{00}^{j})L_{00} + (\sum_{j=1}^{n} D_{11}^{j})L_{11}$$

Leakage Computing Circuit

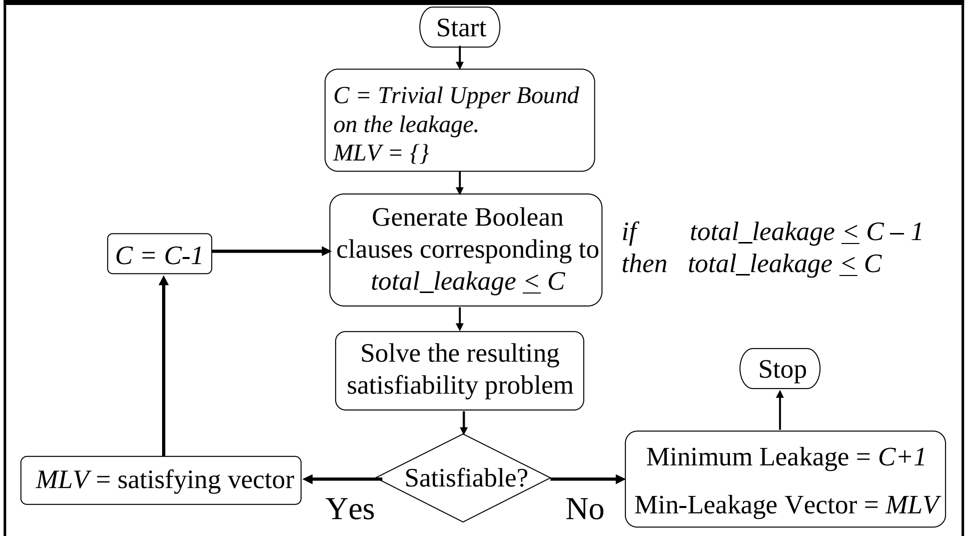


Minimum Leakage Vector Identification



Search for the minimum leakage value for which the above Boolean network is satisfiable.

Linear Search Algorithm for Minimum Leakage



Start

C = Trivial Upper Bound on the leakage.
MLV = {}

Generate Boolean clauses corresponding to $total\_leakage \leq C$

if $total\_leakage \leq C - 1$
then $total\_leakage \leq C$

Solve the resulting satisfiability problem

Satisfiable?

Yes

MLV = satisfying vector

C = C-1

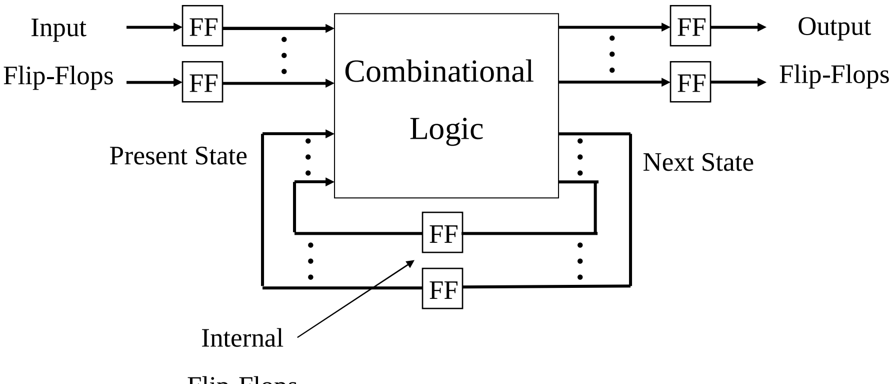
No

Minimum Leakage = *C+1*
Min-Leakage Vector = *MLV*

Stop

Applying the Minimum Leakage Vector

Sequential Circuit



Input Flip-Flops

FF

FF

Combinational Logic

FF

FF

Output Flip-Flops

Present State

Next State

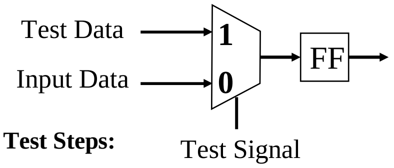
FF

FF

Internal Flip-Flops

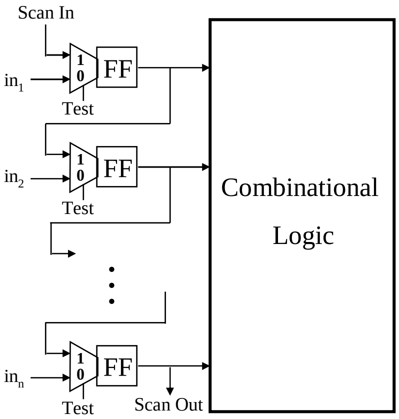
[Abdollahi-ISQED02]

Scan Based Testing



Test Steps:

1. Test = 1
   - Apply n clocks and shift in the test vector.
2. Test = 0
   - Apply one clock and capture the circuit response.
3. Test = 1
   - Apply n clocks and shift out the response.

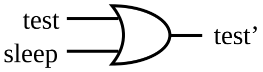


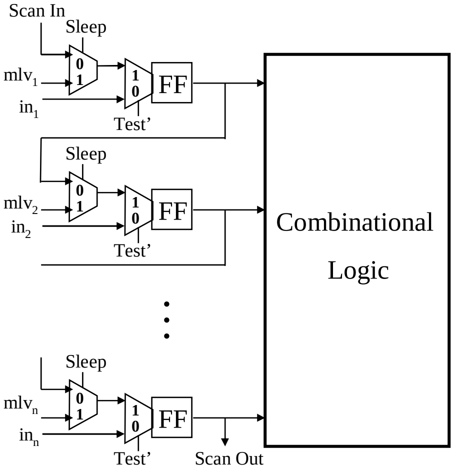
Modifying the Scan Chain

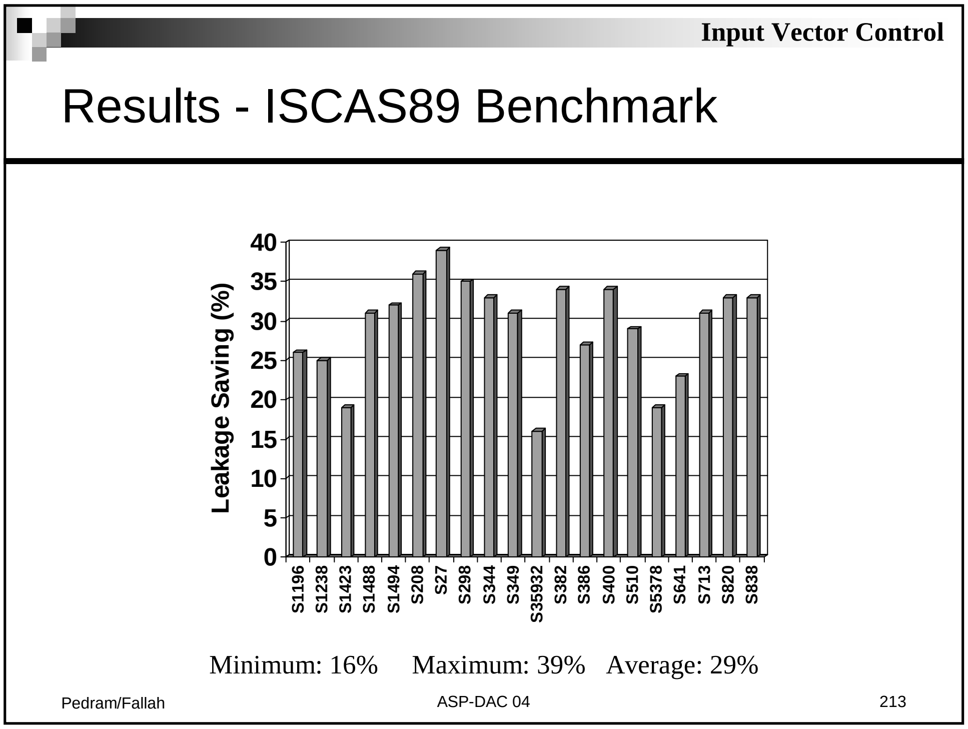
Sleep mode:
Sleep = 1
Test’ = 1
The minimum Leakage Vector is applied to inputs of the combinational logic

Operational mode:
Test’ = 0
Inputs are directly applied to the combinational logic.

Extra multiplexers are not on critical paths.

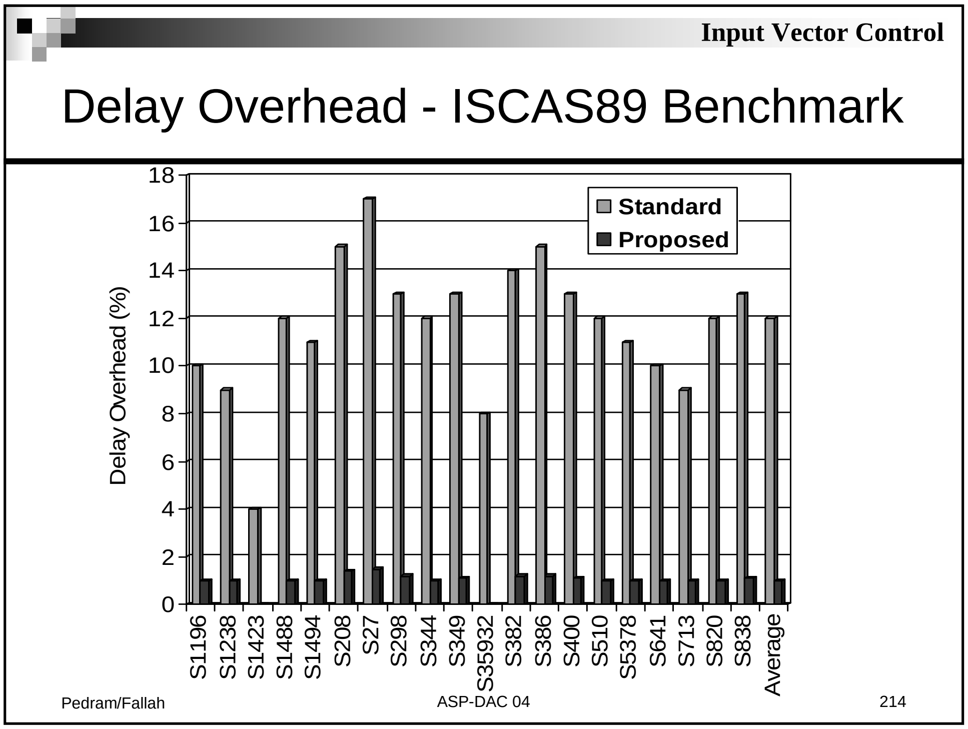






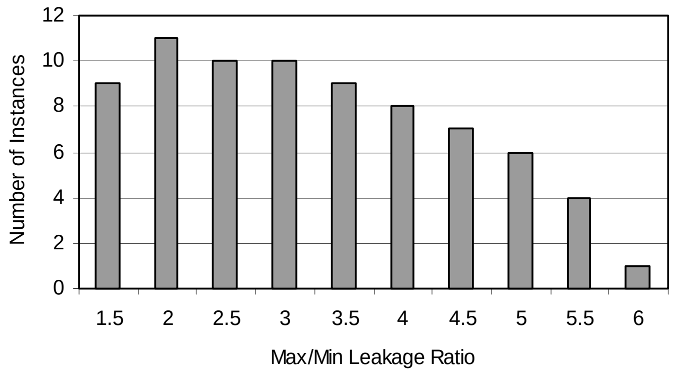
Results - ISCAS89 Benchmark

Minimum: 16%   Maximum: 39%   Average: 29%



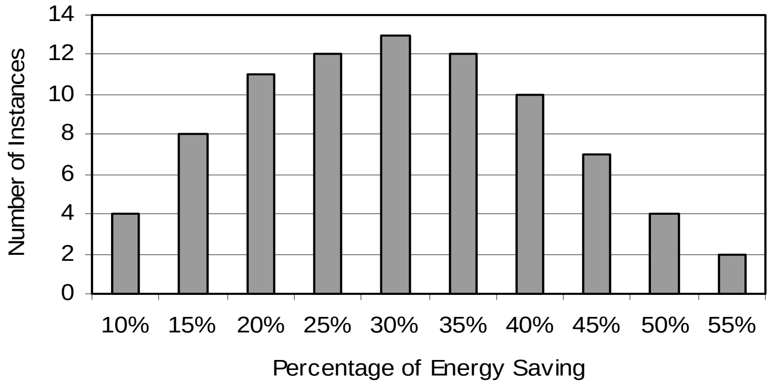
Delay Overhead - ISCAS89 Benchmark

Comparing the Maximum and Minimum Leakage Values

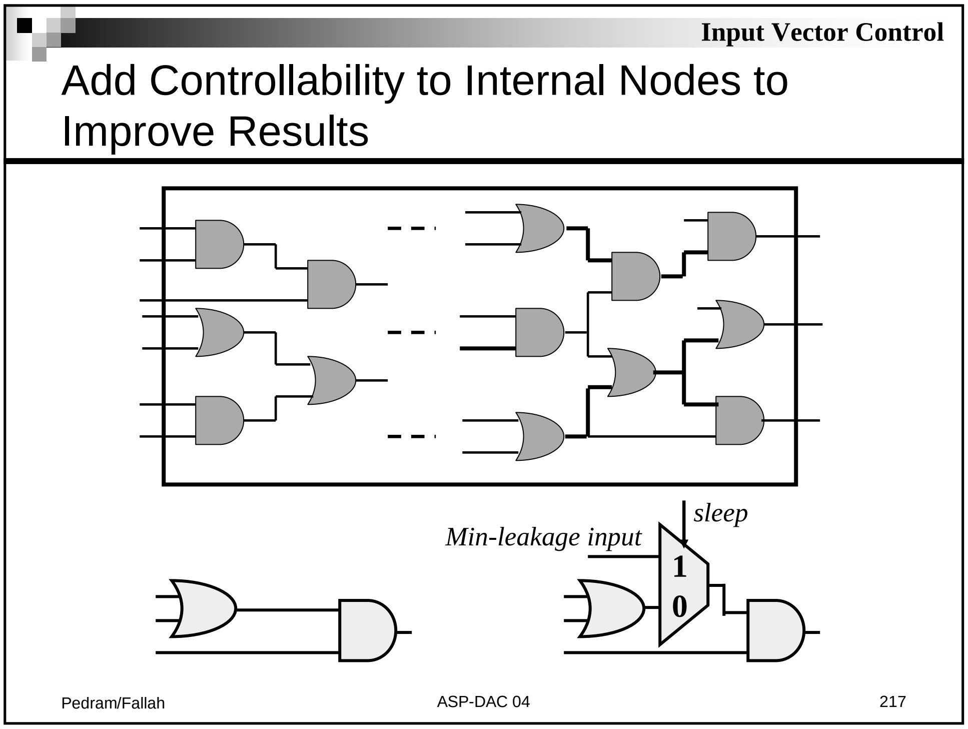


Upper bound on the leakage saving that can be achieved.

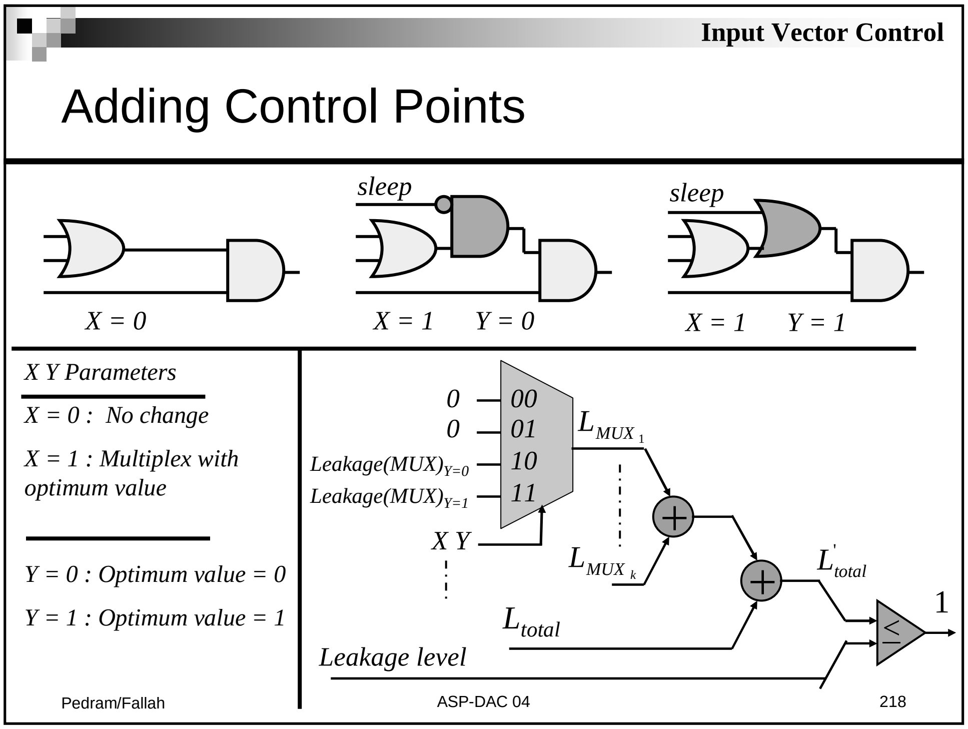
Minimum Leakage vs. Average Leakage in the Sleep mode



- **Average leakage values were found using random vectors.**
- **The figure shows the actual leakage saving that may be achieved.**
- **Note that in a circuit, many gates are not in their minimum leakage states.**



Add Controllability to Internal Nodes to Improve Results



Adding Control Points

X Y Parameters

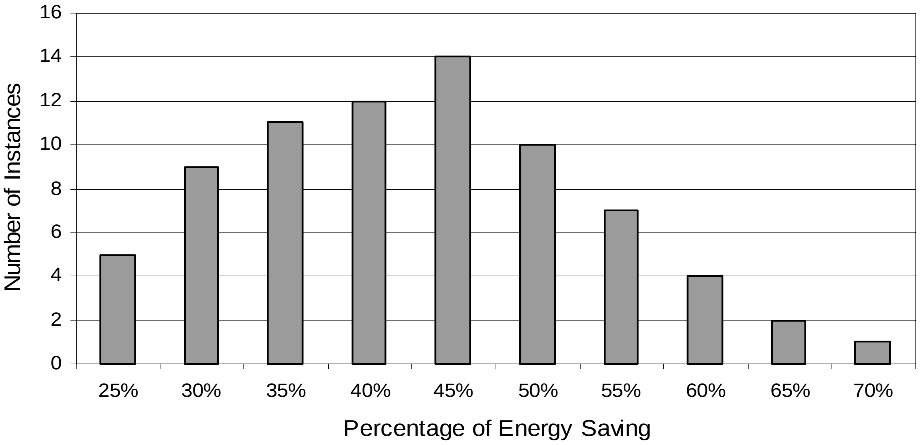
X = 0 : No change

X = 1 : Multiplex with optimum value

Y = 0 : Optimum value = 0

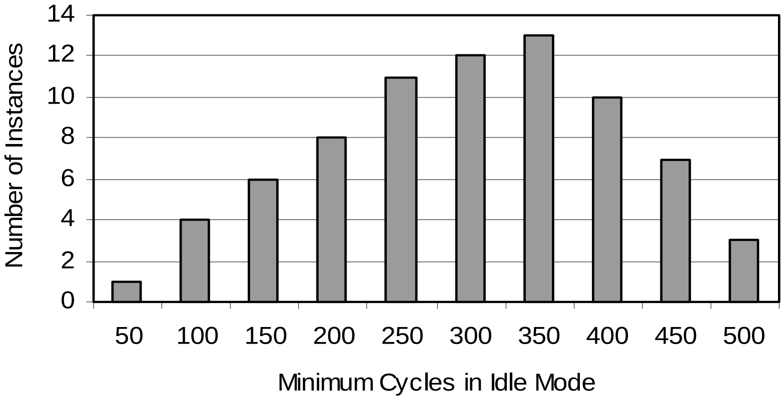
Y = 1 : Optimum value = 1

Power Reduction by Adding Control Points

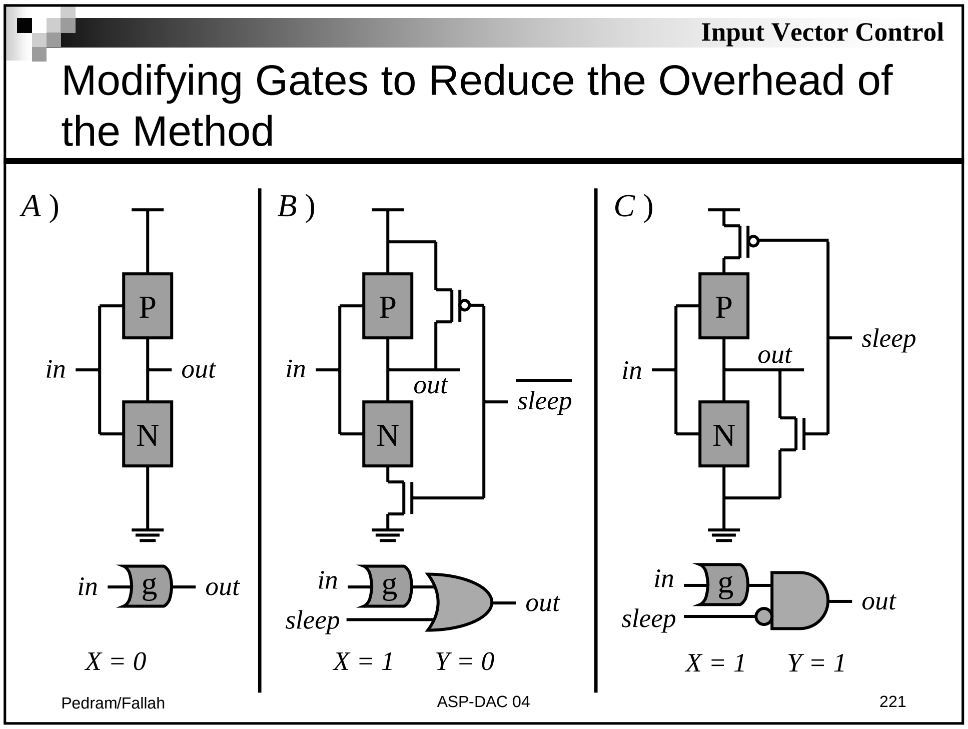


About 15% improvement by adding control points.

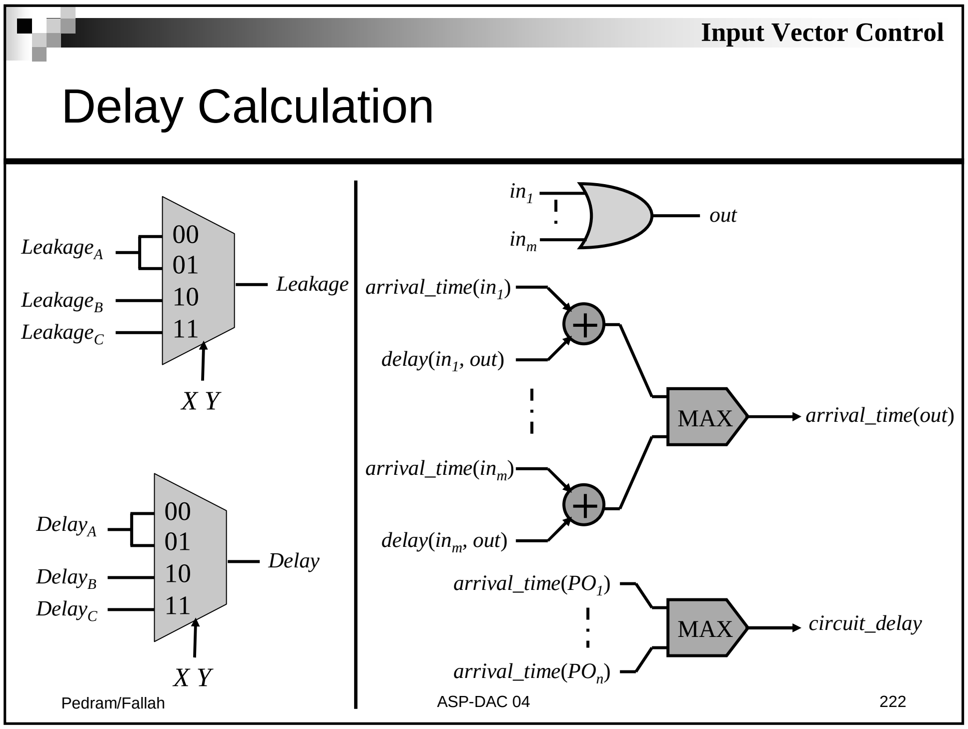
Break-Even Time



The sleep period has to be larger than the break-even time in order to save power.

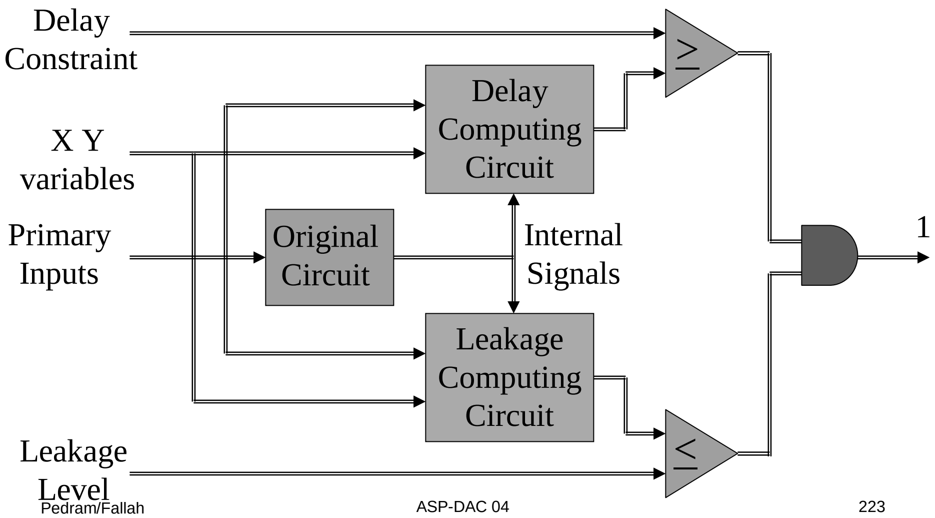


Modifying Gates to Reduce the Overhead of the Method

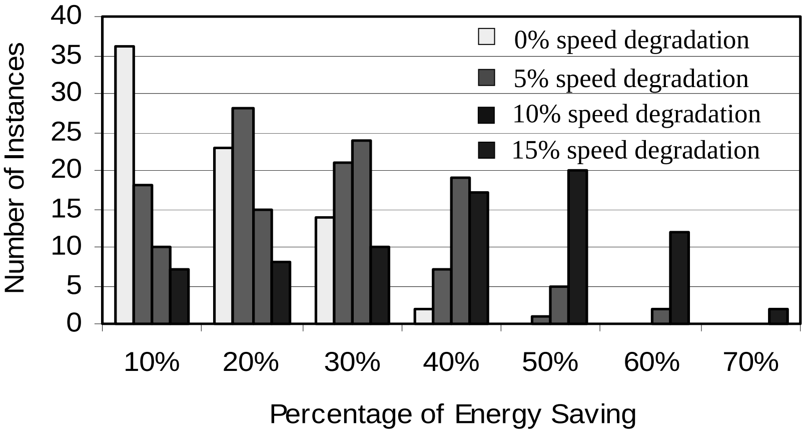


Delay Calculation

Equivalent Boolean Network

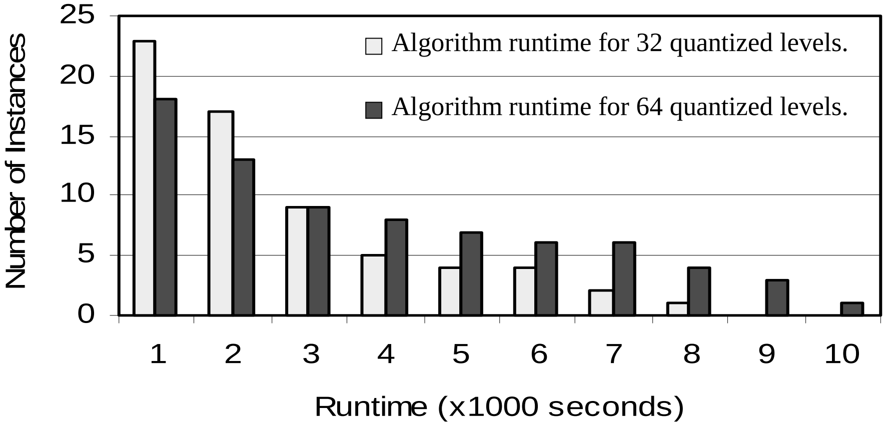


Energy Saving for Different Speed Degradations



Input Vector Control

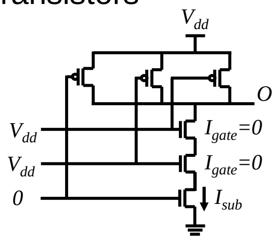
Runtime of the Algorithm



Pin Reordering

State Dependence of Sub-Threshold and Gate Leakages

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



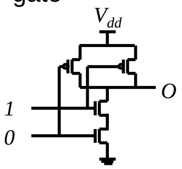
[Lee-DAC03]

State	I_{sub} (nA)	I_{gate} (nA)	I_{total} (nA)
000	0.382	0.000	0.382
001	0.709	6.339	7.048
010	0.709	1.275	1.984
011	5.626	12.677	18.303 ←
100	0.676	0.000	0.676
101	3.804	6.339	10.143
110	3.804	0.000	3.804 ←
111	28.273	19.015	47.288

5x reduction

Combining It with Input Vector Control

- Input Vector Control can be used to reduce I_{sub}.
- Pin reordering can be used to reduce I_{gate}
  - □ Place off-transistor at bottom of stack
    - Affects performance
- Inter-dependent problems
  - □ Use simultaneous optimization



- Minimum leakage vector depends on the relative magnitude of I_{sub}, I_{gate}, and I_{BTBT}.
  - □ For a 2-input NAND
    - I_{sub} is at minimum → 00
    - I_{gate} is at minimum → 10

Technology (nm)	90	50	25
Minimum Leakage Vector	00	10	10

Result

- Sleep mode savings
  - □ Avg. 18% using state assignment alone
  - □ Avg. 27% by using pin reordering along with state assignment
  - □ I_{gate} reduced by 45% up to 82%
- The impact of state assignment and pin re-ordering: C6288
  - □ State assignment works equally well for I_{sub} & I_{gate}
  - □ The addition of pre –reordering provides substantial benefits for both I_{gate} & I_{leak} with slight improvement for I_{sub}
    - Effectiveness will increase for technologies with higher components of I_{gate}

Comparing Effectiveness of Several Techniques

$$I_{sub} \approx A \times e^{\frac{1}{mv_T}(V_G - V_s - V_{th_0} - \gamma V_{app} + \eta V_{ds})} \qquad for \quad \frac{V_{ds}}{v_T} >> 1$$

$$Changing \;\; V_{ds} \;\rightarrow\; \left.\frac{\Delta I_{sub}}{I_{sub}}\right|_{V_{ds}} = 1 - e^{-\frac{\eta \Delta V_{ds}}{mv_T}(V_G - V_s - V_{th_0} - \gamma V_{SB} + \eta V_{ds})}$$

$$Changing \;\; L_{eff} \;\rightarrow\; \left.\frac{\Delta I_{sub}}{I_{sub}}\right|_{L_{eff}} = 1 - \frac{1}{1 + \frac{\Delta L_{eff}}{L_{eff}}} e^{-\frac{\Delta V_{th_0}}{mv_T}}$$

$$Using\ stack \rightarrow \left.\frac{\Delta I_{sub}}{I_{sub}}\right|_{stack} = 1 - e^{-\frac{I_{sub} R_{off}\,(1+\gamma+\eta)}{mv_T}}$$

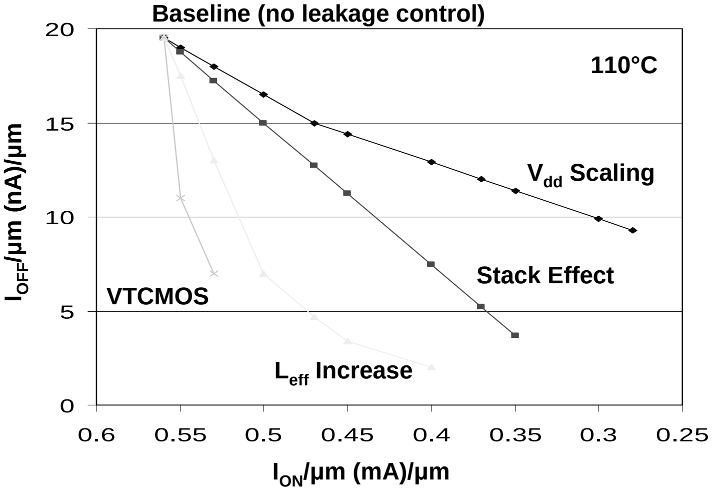
$$Using\ VTCMOS \rightarrow \left.\frac{\Delta I_{sub}}{I_{sub}}\right|_{VTCMOS} = 1 - e^{-\frac{\gamma V_{SB}}{mv_T}}$$

[Borkar-ISLPED03]

Leakage Reduction for a 130nm Technology

Technique	Simulation Results	Theoretical Model
Reduction in V_{dd} by 30%	2.2X	1.9X
Increase in L_{eff} by 30%	9.3X	8.7X
Stack Effect	12.0X	11.5X
Reverse Bias by 30% of V_{dd}	2.3X	2.1X

I_{off} – I_{on} Curve



Normalized I_{off}/I_{on} Degradation: Scaling Trends

$\zeta = \frac{\partial I_{OFF}}{\partial I_{ON}} \Big/ \frac{I_{OFF}}{I_{ON}}$	Changing V_{dd}	Changing L_e	Stack Effect	VTCMOS
130nm	1.1	3.1	2.2	20
100nm	1	3.1	2.1	9
70nm	0.8	2.8	1.9	7.5

Higher values are better.

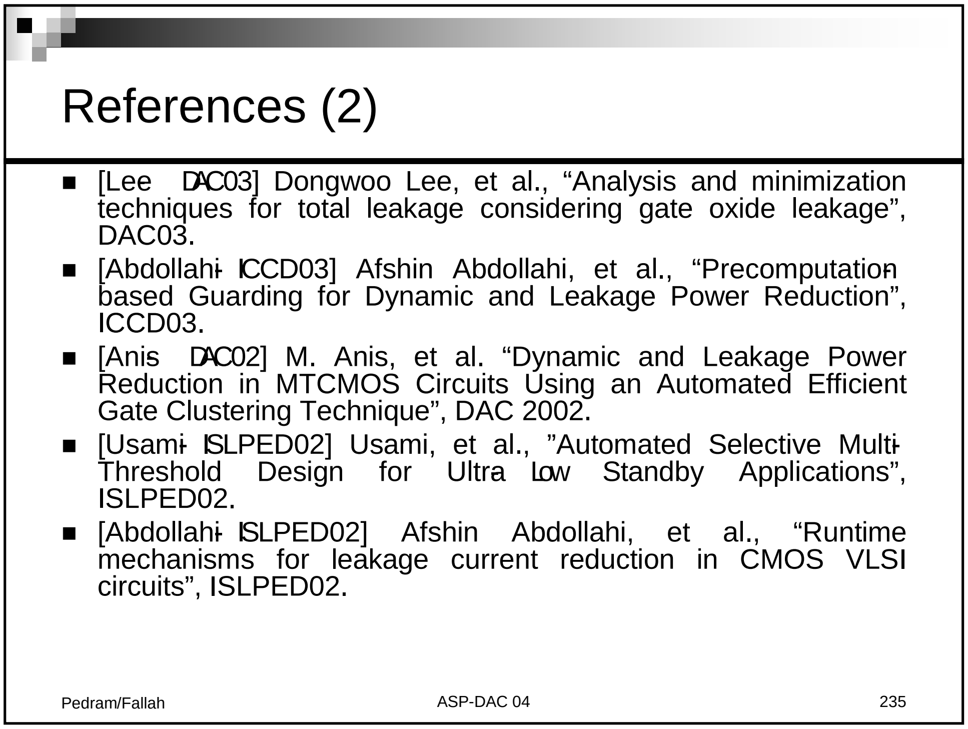
Comparing Different Techniques

	Leakage Saving		Flip Flop	Proc. Mod.	Design Compl.	Scalab.	Overheads			
	Sleep	Active					Delay in Active Mode	Wakeup Delay	Area	Dyn. Power
Dual-V_{th}	H	Y	Y	L	L	Y	N-Y	N	N	N
MTCMOS	Very H	N	N	L	H	N	Y	Y	H	H
VTCMOS	H	N	Y	H	H	N	N	Y	M	H
Stack Effect	L-H	Y	N	N	L	Y	N-Y	N	M	M

L: Low, M: Medium, H: High, N: No, Y: Yes

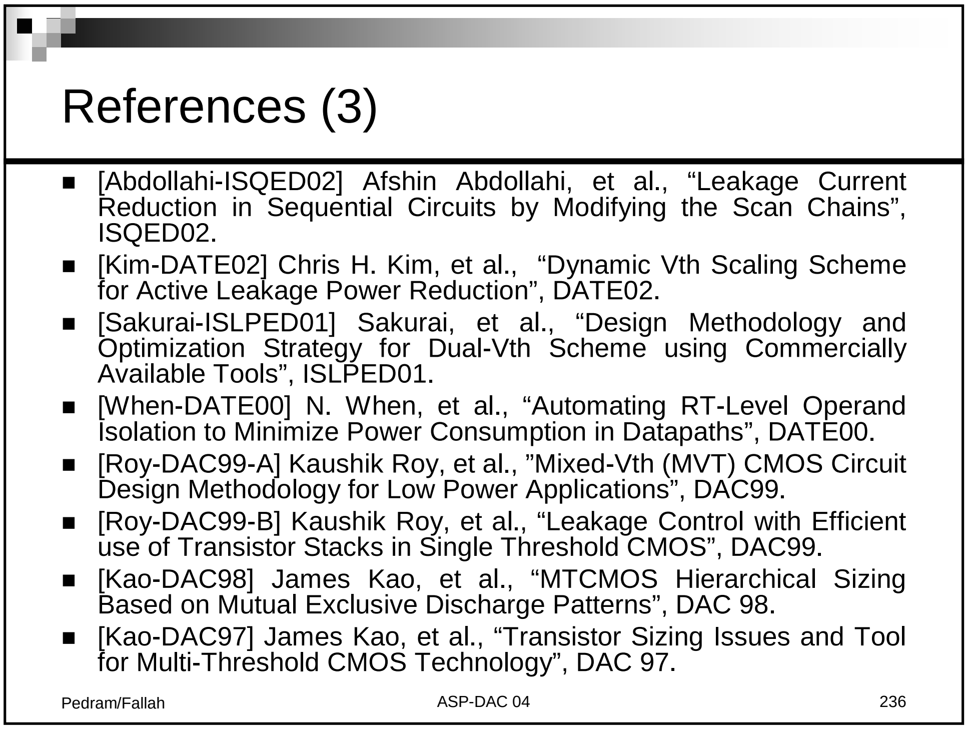
References (1)

- [Roy-ISLPED03] Kaushik Roy, et al., “Optimal Body Bias Selection for Leakage Improvement and Process Compensation Over Different Technology Generations”, ISLPED03.
- [Borkar-ISLPED03] Shkhar Borkar, et al., “Effectiveness and Scaling Trends of Leakage Control Techniques for Sub-130nm CMOS Technologies”, ISLPED03.
- [Kim-ISLPED03] Suhwan Kim, et al., “Understanding and Minimizing Ground Bounce During Mode Transition of Power Gating Structures”, ISLPED03.
- [Won-ISLPED03] Hyo-Sig Won, et al., “An MTCMOS Design Methodology and Its Application to Mobile Computing”, ISLPED03.
- [Srivastava-ISLPED03] Ankur Srivastava, “Simultaneous Vt Selection and Assignment for Leakage Optimization”, ISLPED03.
- [Keutzer-ISLPED03] Kurt Keutzer, et al., “Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization”, ISLPED03.



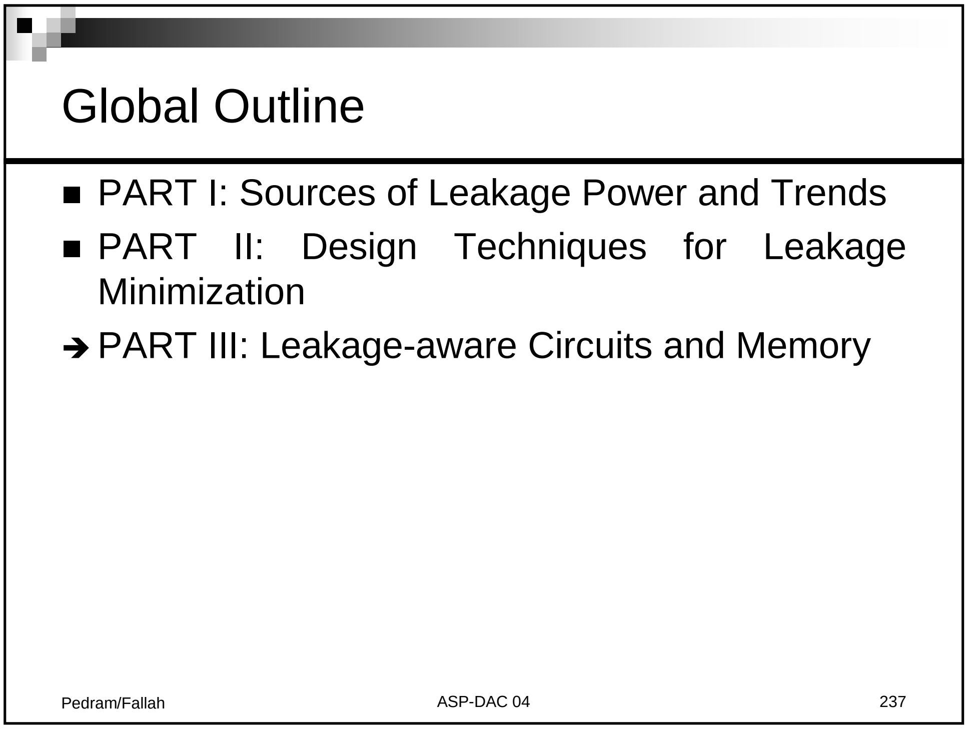
References (2)

- [Lee DAC03] Dongwoo Lee, et al., “Analysis and minimization techniques for total leakage considering gate oxide leakage”, DAC03.
- [Abdollahi-ICCD03] Afshin Abdollahi, et al., “Precomputation-based Guarding for Dynamic and Leakage Power Reduction”, ICCD03.
- [Anis DAC02] M. Anis, et al. “Dynamic and Leakage Power Reduction in MTCMOS Circuits Using an Automated Efficient Gate Clustering Technique”, DAC 2002.
- [Usami-ISLPED02] Usami, et al., ”Automated Selective Multi-Threshold Design for Ultra Low Standby Applications”, ISLPED02.
- [Abdollahi-ISLPED02] Afshin Abdollahi, et al., “Runtime mechanisms for leakage current reduction in CMOS VLSI circuits”, ISLPED02.



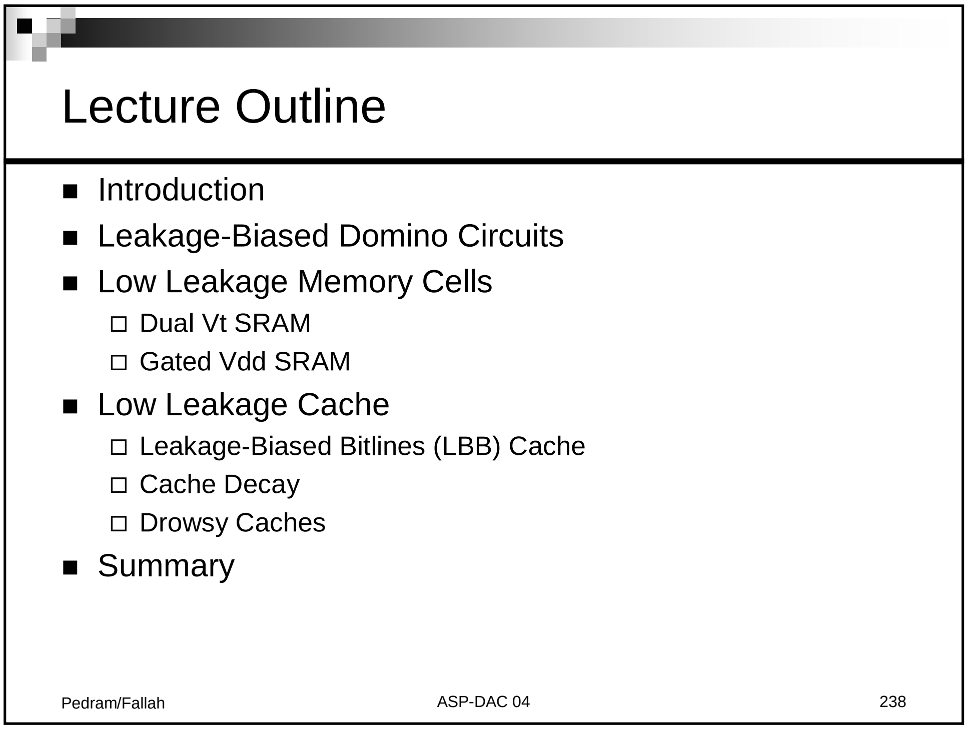
References (3)

- [Abdollahi-ISQED02] Afshin Abdollahi, et al., “Leakage Current Reduction in Sequential Circuits by Modifying the Scan Chains”, ISQED02.
- [Kim-DATE02] Chris H. Kim, et al., “Dynamic Vth Scaling Scheme for Active Leakage Power Reduction”, DATE02.
- [Sakurai-ISLPED01] Sakurai, et al., “Design Methodology and Optimization Strategy for Dual-Vth Scheme using Commercially Available Tools”, ISLPED01.
- [When-DATE00] N. When, et al., “Automating RT-Level Operand Isolation to Minimize Power Consumption in Datapaths”, DATE00.
- [Roy-DAC99-A] Kaushik Roy, et al., ”Mixed-Vth (MVT) CMOS Circuit Design Methodology for Low Power Applications”, DAC99.
- [Roy-DAC99-B] Kaushik Roy, et al., “Leakage Control with Efficient use of Transistor Stacks in Single Threshold CMOS”, DAC99.
- [Kao-DAC98] James Kao, et al., “MTCMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Patterns”, DAC 98.
- [Kao-DAC97] James Kao, et al., “Transistor Sizing Issues and Tool for Multi-Threshold CMOS Technology”, DAC 97.



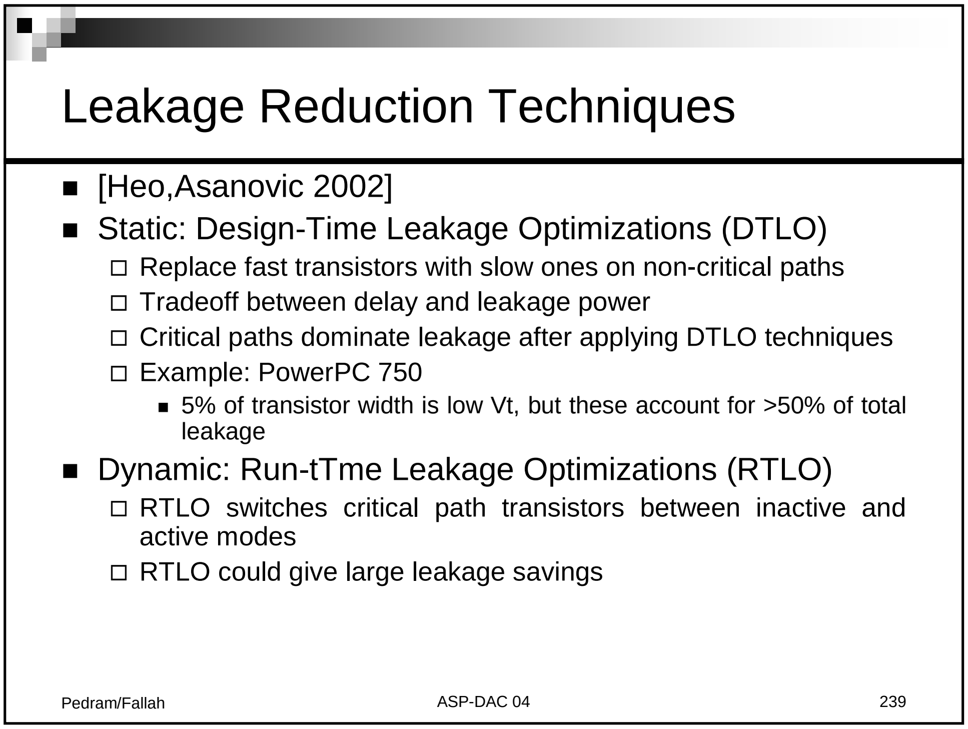
Global Outline

- PART I: Sources of Leakage Power and Trends
- PART II: Design Techniques for Leakage Minimization
- ➔ PART III: Leakage-aware Circuits and Memory



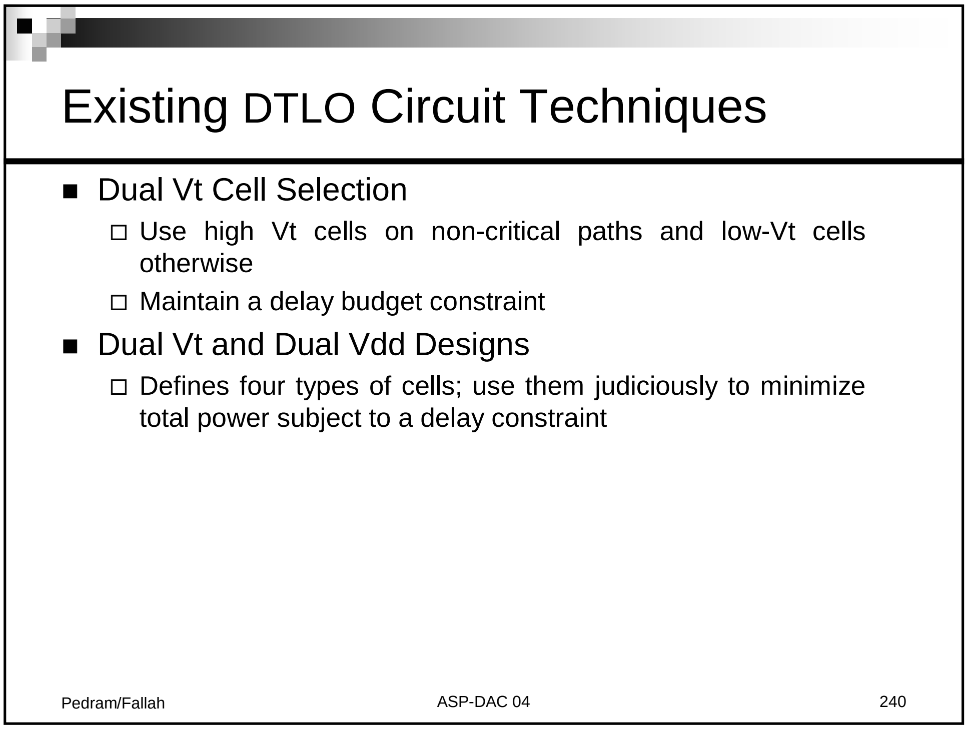
Lecture Outline

- Introduction
- Leakage-Biased Domino Circuits
- Low Leakage Memory Cells
  - □ Dual Vt SRAM
  - □ Gated Vdd SRAM
- Low Leakage Cache
  - □ Leakage-Biased Bitlines (LBB) Cache
  - □ Cache Decay
  - □ Drowsy Caches
- Summary



Leakage Reduction Techniques

- [Heo,Asanovic 2002]
- Static: Design-Time Leakage Optimizations (DTLO)
  - □ Replace fast transistors with slow ones on non-critical paths
  - □ Tradeoff between delay and leakage power
  - □ Critical paths dominate leakage after applying DTLO techniques
  - □ Example: PowerPC 750
    - 5% of transistor width is low Vt, but these account for >50% of total leakage
- Dynamic: Run-tTme Leakage Optimizations (RTLO)
  - □ RTLO switches critical path transistors between inactive and active modes
  - □ RTLO could give large leakage savings

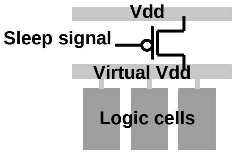


Existing DTLO Circuit Techniques

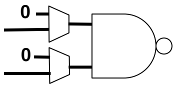
- Dual Vt Cell Selection
  - □ Use high Vt cells on non-critical paths and low-Vt cells otherwise
  - □ Maintain a delay budget constraint
- Dual Vt and Dual Vdd Designs
  - □ Defines four types of cells; use them judiciously to minimize total power subject to a delay constraint

Existing RTLO Circuit Techniques

- Power Gating
  - □ Sleep transistor between supply and virtual supply lines
  - □ Increased delay due to sleep transistor



- Sleep Vector
  - □ Input vector which minimizes leakage
  - □ Increased delay due to mux and active energy due to spurious toggles after applying sleep vector



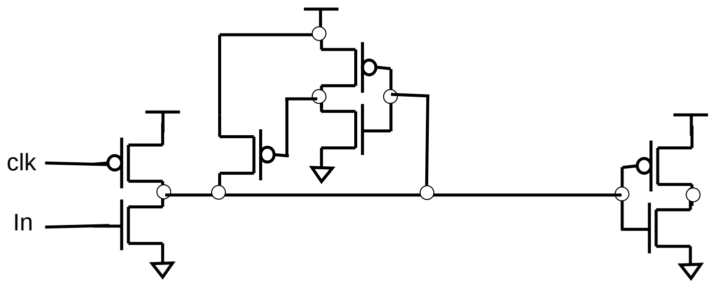
Fine-Grain RTLO Techniques

- Have to turn off small pieces of an active processor for short periods of time
  - □ Difficult to turn off large pieces for long periods → Fine-grain RTLO techniques
- Requirements of Fine-grain RTLO techniques
  - □ Circuits with low active delay penalty, low energy moving in and out of sleep, and fast wakeup time
  - □ Micro-architectural scheduling to keep the sleep time as long and often as possible
- Compare to coarse-grain RTLO techniques
  - □ O.S. puts whole processor to sleep for a long time ⇒ doesn't save power when running code
  - □ Low steady-state leakage only concern

Lecture Outline

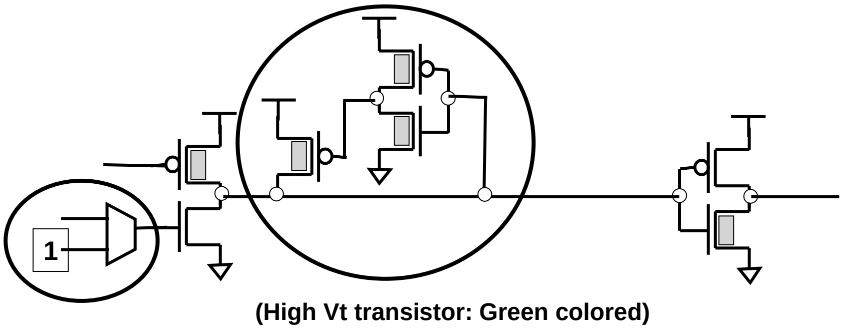
- Introduction
- Leakage-Biased Domino Circuits
- Low Leakage Memory Cells
  - Dual Vt SRAM
  - Gated Vdd SRAM
- Low Leakage Cache
  - Leakage-Biased Bitlines (LBB) Cache
  - Cache Decay
  - Drowsy Caches
- Summary

Conventional Domino



Dual-Vt Domino

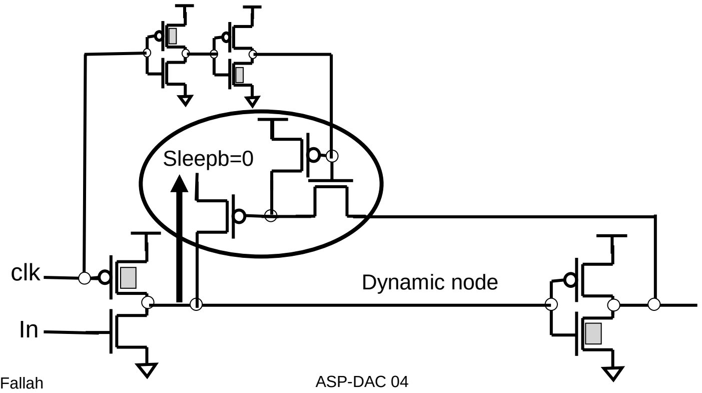
- [Kao and Chandrakasan, 2000]
  - □ High Vt for precharge phase
  - □ Input gating → increased delay and active energy
  - □ High Vt keeper → increased noise margin



(High Vt transistor: Green colored)

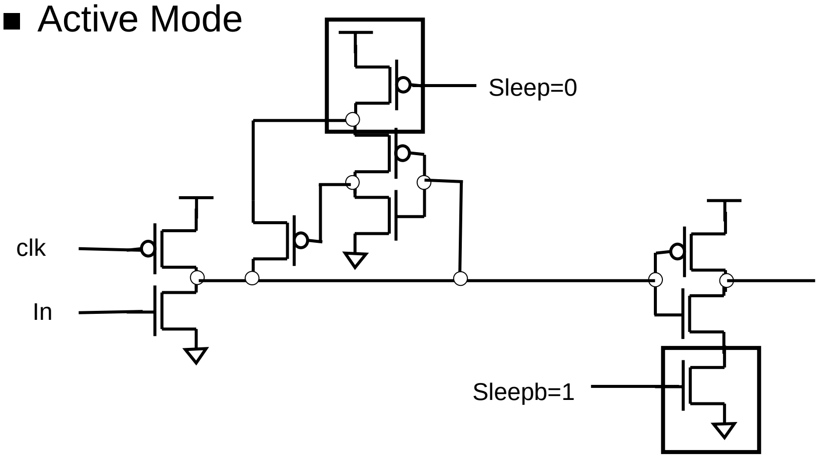
MHS-Domino

- [Allam, Anis, Elmasry, 2000]
  - □ Clock-delayed keeper
  - □ Pull-down through PMOS → short circuit-current in static inverter



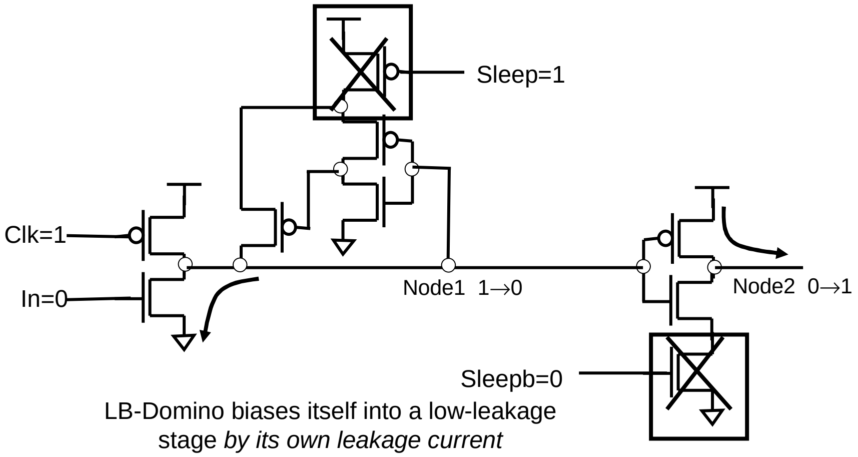
Leakage-Biased (LB) Domino

- [Heo and Asanovic, 2002]
- Active Mode



Leakage-Biased (LB) Domino

- Sleep Mode

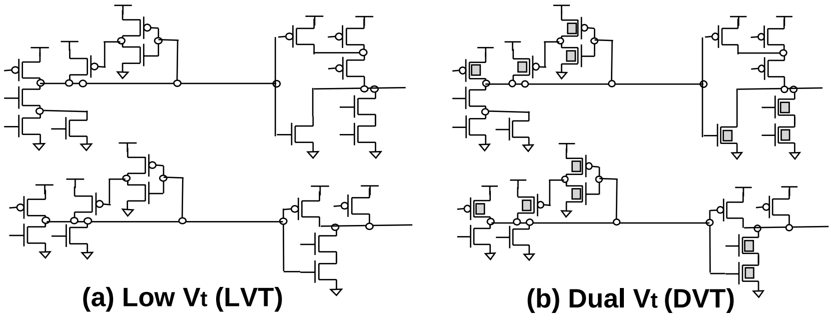


LB-Domino biases itself into a low-leakage stage *by its own leakage current*

Han-Carlson Adder

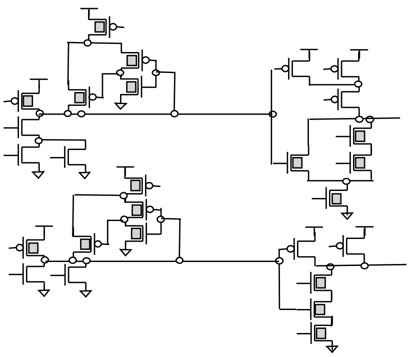
- Evaluation with carry generation circuit of a 32-bit Han-Carlson adder
  - □ 6 levels of alternating dynamic and static logic
  - □ 4 circuits: LVT, DVT, LB, and LB2
- Constraints
  - □ Input/Output noise margin kept to 10% of V_{dd}
  - □ Precharge/Evaluation delay equalized to within 1% error

PG Cells of Han-Carlson Adder

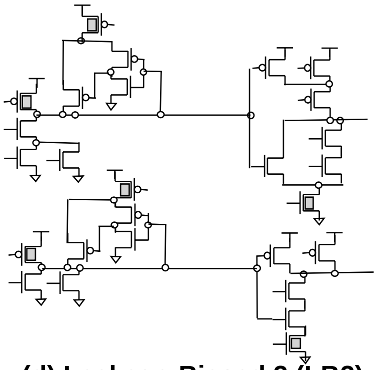


(a) Low V_t (LVT) (b) Dual V_t (DVT)

PG Cells of Han-Carlson Adder



(c) Leakage-Biased 1 (LB)



(d) Leakage-Biased 2 (LB2)

Processes

- 180nm: TSMC 180nm Processes
- 70nm: BPTM 70nm Processes

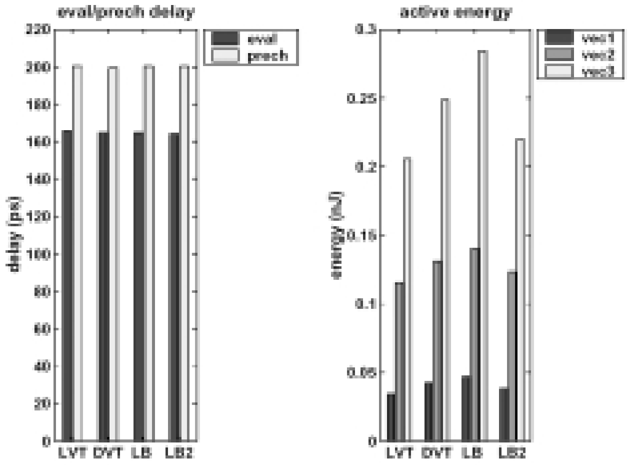
Process	180nm	70nm
High V_t (NMOS/PMOS)	0.46V/-0.45V	0.39V/-0.40V
Low V_t (NMOS/PMOS)	0.27V/-0.23V	0.15V/-0.18V
V_{dd}	1.8V	0.9V
Temperature	100C	100C

Input Vectors

- 3 different input vectors
  - □ Active energy and leakage power dependent upon inputs
  - □ Vec1 discharges no dynamic nodes
  - □ Vec2 discharge half of dynamic nodes
  - □ Vec3 discharge all dynamic nodes

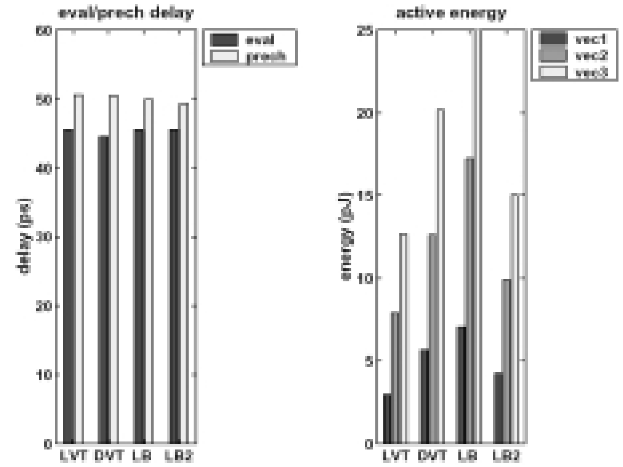
	A	B	Ci
Vector 1	0x00000000	0x00000000	0
Vector 2	0xffffffff	0x00000000	0
Vector 3	0xffffffff	0xffffffff	1

Delay and Active energy consumption – 180 nm Process



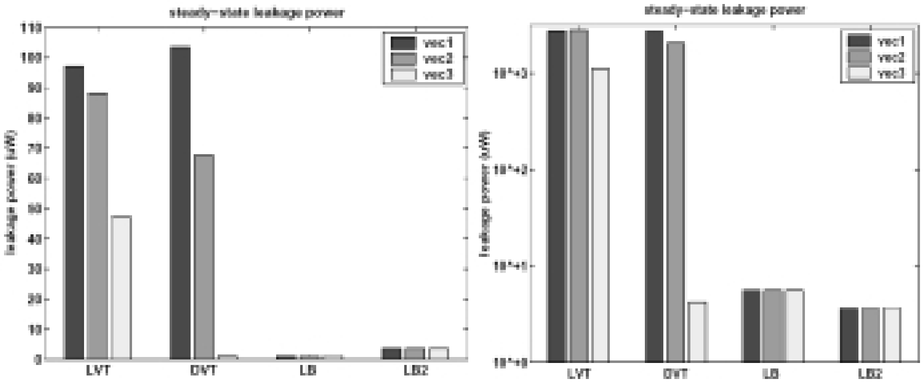
Delay and Active energy consumption : 180 nm process

Delay and Active energy consumption – 70 nm Process

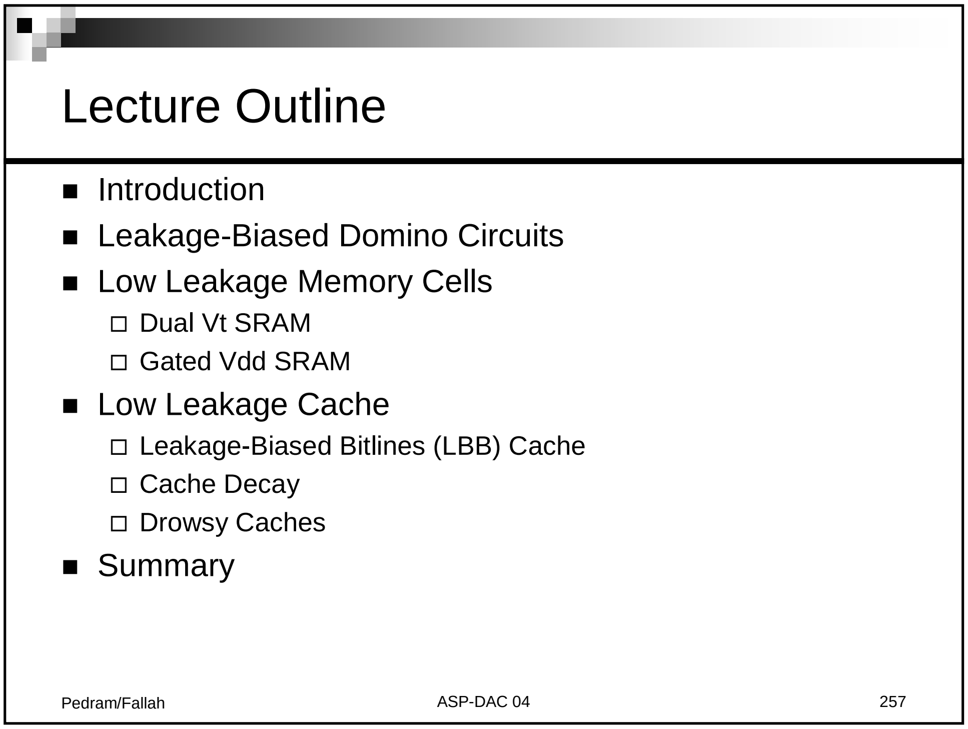


Delay and Active energy consumption : 70 nm process

Steady-State Leakage Power

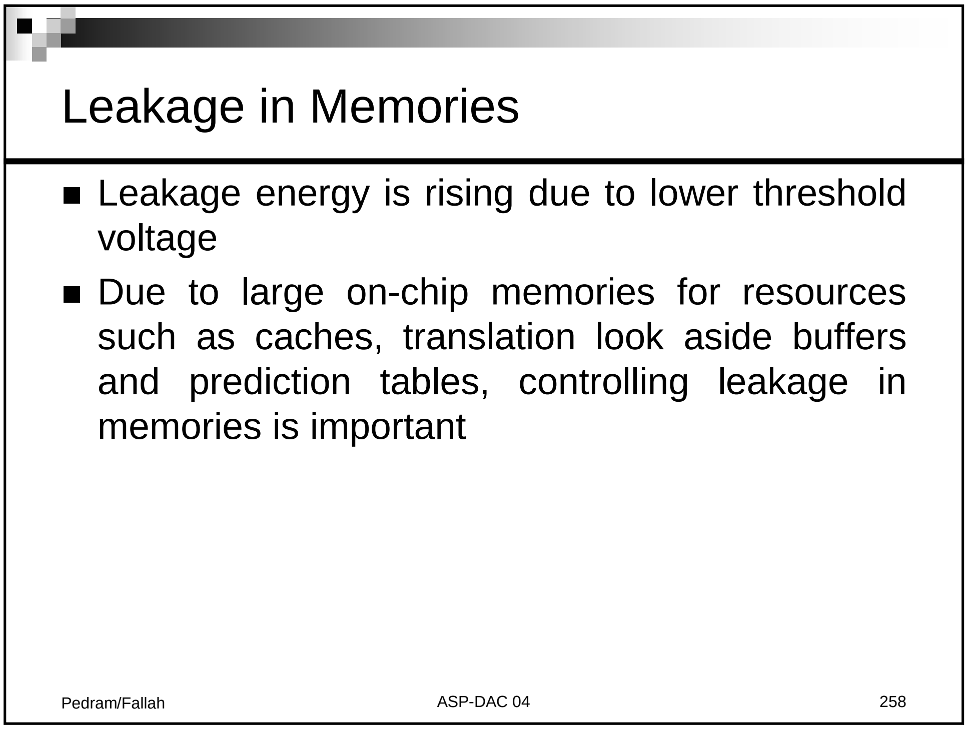


Steady-state leakage power:180 nm process for the left one and 70 nm for the right one. Clk is high for all and sleep is asserted for LB and LB2. Note that y-axis for the left one is log-scale.



Lecture Outline

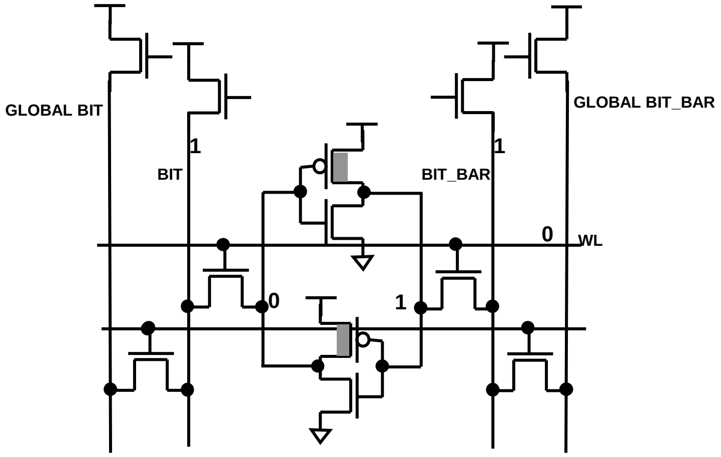
- Introduction
- Leakage-Biased Domino Circuits
- Low Leakage Memory Cells
  - □ Dual Vt SRAM
  - □ Gated Vdd SRAM
- Low Leakage Cache
  - □ Leakage-Biased Bitlines (LBB) Cache
  - □ Cache Decay
  - □ Drowsy Caches
- Summary



Leakage in Memories

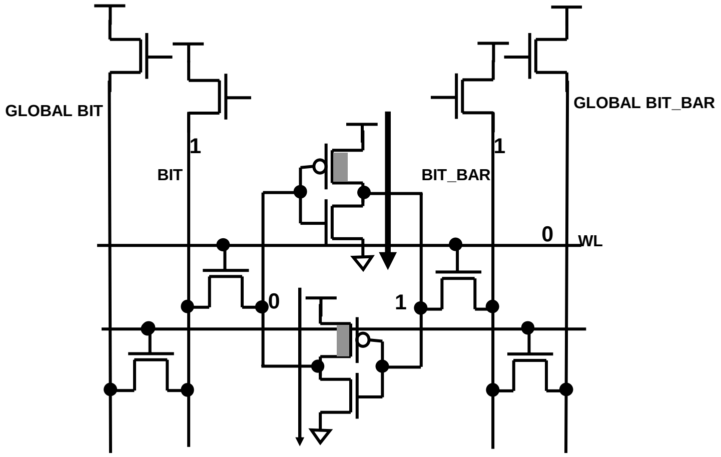
- Leakage energy is rising due to lower threshold voltage
- Due to large on-chip memories for resources such as caches, translation look aside buffers and prediction tables, controlling leakage in memories is important

Dual Vt SRAM Cell

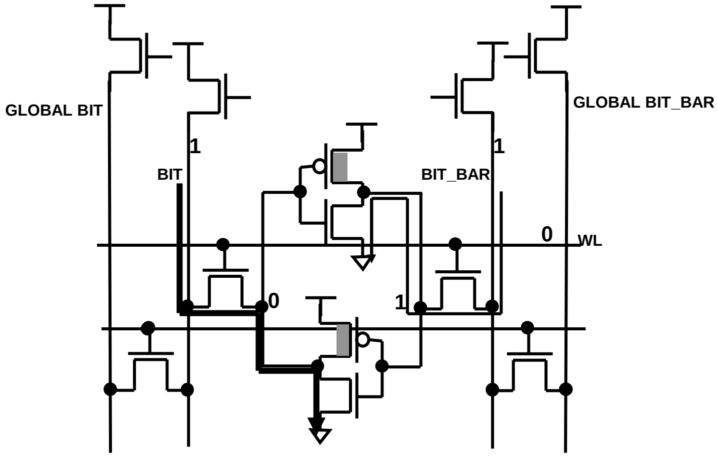


HVT transistors: green-colored

Leakage Paths in Dual Vt SRAM Cell



Leakage Paths in Dual Vt SRAM Cell (Cnt'd)

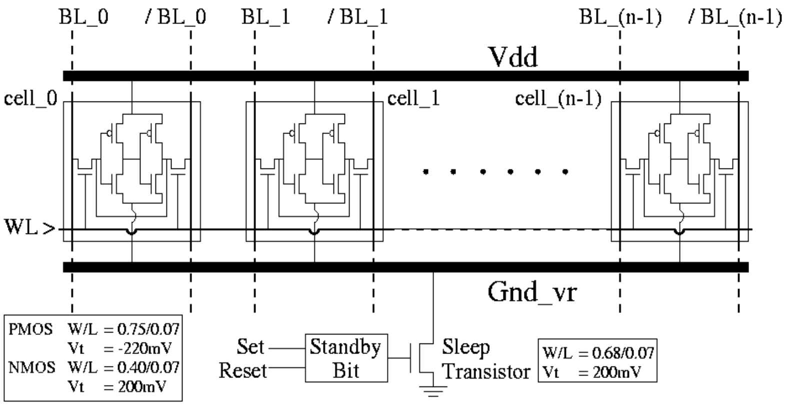


Bitline leakage depends on the stored value

Gated-Vdd SRAM Cell

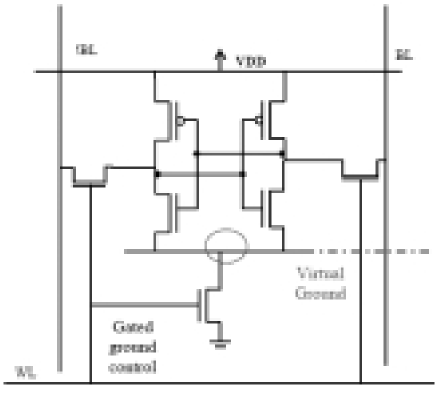
- State-destroying mechanism (Gated-Vdd)
  - □ Introduces a power-switch between the ground and the circuit to reduce leakage
  - □ Does sizing to maximize the static power saving but loses data in SRAM cells
- State-preserving mechanism (Modified Gated-Vdd)
  - □ Appropriately sizes the NMOS power-switch to provide the required minimum supply voltage to maintain the state of a static memory cell

Modified Gated-Vdd SRAM Cell



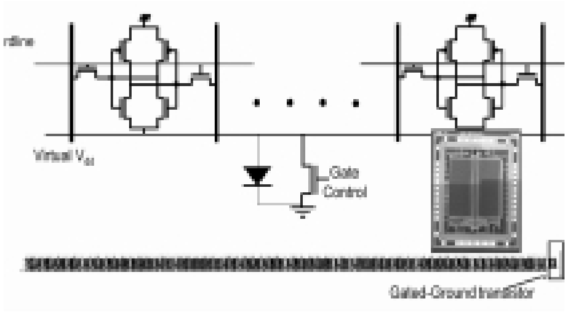
Gated GND (or Source-Biased) SRAM Cell

- [Kim and Roy 2002]
- Add a gated ground control transistor in between the actual ground and the sources of NMOS transistors in the SRAM cell
- The creation of a virtual ground (floating near 0.4V) during the low leakage mode operation can potentially make this circuit more vulnerable to soft errors (for 0 to 1 bit flip)



Gated-Ground Transistor Sharing

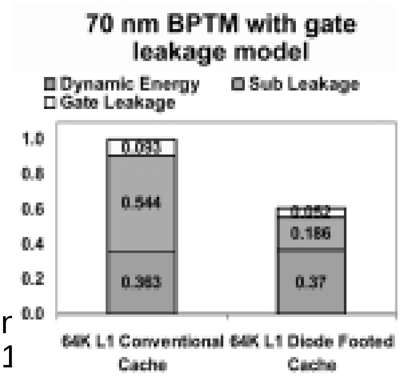
- Gated-ground transistor is shared by a BANK of SRAM cells
- The gated-ground transistor should be turned on before the word line goes high
- Bank decoder turns on the gated-ground transistor before word-line reaches the SRAM cell pass transistor
- No extra control logic is required



Energy Savings in 64KB L1 Cache

- Leakage energy:
  - □ 54% Subthreshold
  - □ 9.3% Gate leakage
- Leakage savings:
  - □ 65.8% Subthreshold
  - □ 44.1% Gate leakage
- Energy overhead:
  - □ 2% @ 70nm

Overall energy reductior achieved by diode footed L1 cache is 39.2% in 70nm process



Lecture Outline

- Introduction
- Leakage-Biased Domino Circuits
- Low Leakage Memory Cells
  - □ Dual Vt SRAM
  - □ Gated Vdd SRAM
- Low Leakage Cache
  - □ Leakage-Biased Bitlines (LBB) Cache
  - □ Cache Decay
  - □ Drowsy Caches
- Summary

Introduction

- On chip caches represent a sizable fraction of the total power consumption of Processor
- As feature sizes shrink, the dominant component of this power loss will be leakage
- In a time interval the activity in cache is centered only on a small set of lines
- Hence leakage power can be reduced by putting the cold cache lines in state-preserving low power “Drowsy Mode”

Review of Some Techniques

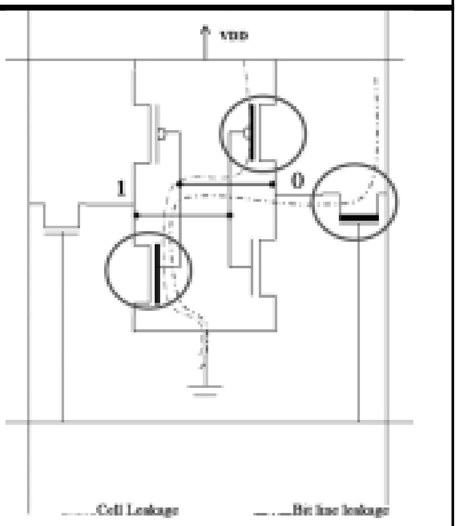
- Turn-Off circuits by creating a high-impedance path to ground; Trade-Off increased execution time for reduced static power consumption
- Gated-V_{dd} Technique turns off cache lines that are not likely to be used
- Drawbacks
  - □ State loss once power is turned off
  - □ Reloading from L2 has potential to negate the energy savings
  - □ Performance is affected due to reloading
  - □ Complex algorithms are needed to reduce these effects
- Adaptive body-biasing with Multi-Threshold CMOS (ABB-MTCMOS) -> threshold voltage of cache line is varied (increased) dynamically to yield reduction in leakage energy

Review: Preferred State Caches

Asymmetric SRAM

(Optimized for Leakage of “0”)

- The lower current drive of the high threshold voltage transistors make this design vulnerable to a stored value of 1 (its non favorable state for leakage reduction)
- Similarly for the cell optimized for 1

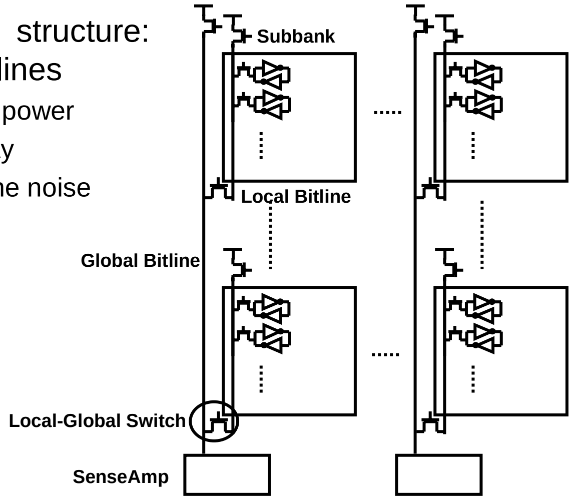


Dynamic Fine-Grain Leakage Reduction Using Leakage-Biased Bitlines

- Metrics for comparing fine-grain dynamic deactivation techniques
  - □ Steady-stage leakage, Transition time, Fixed transition energy
- Presents a new circuit-level leakage reduction technique, Leakage-Biased Bitlines (LBB)
  - □ Low deactivation energy and fast wakeup
- Save leakage power of I-Cache and Multiported regfile by LBB
  - □ I-cache: Idle subbank deactivation
  - □ Multiported regfile: Idle read ports and dead register deactivation

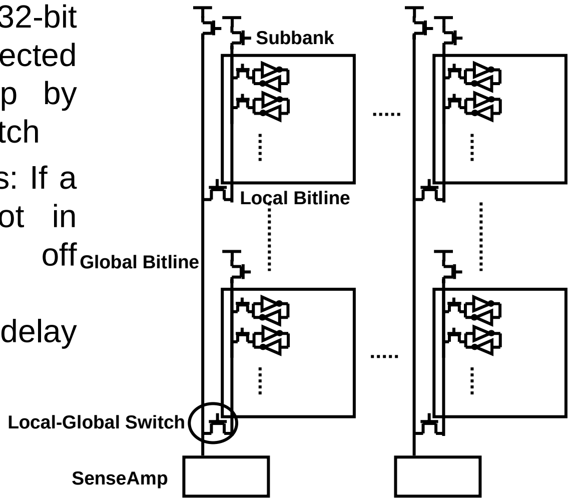
LBB for Caches

- Modern cache structure: Hierarchical Bitlines
  - □ To save active power
  - □ To reduce delay
  - □ To reduce bitline noise

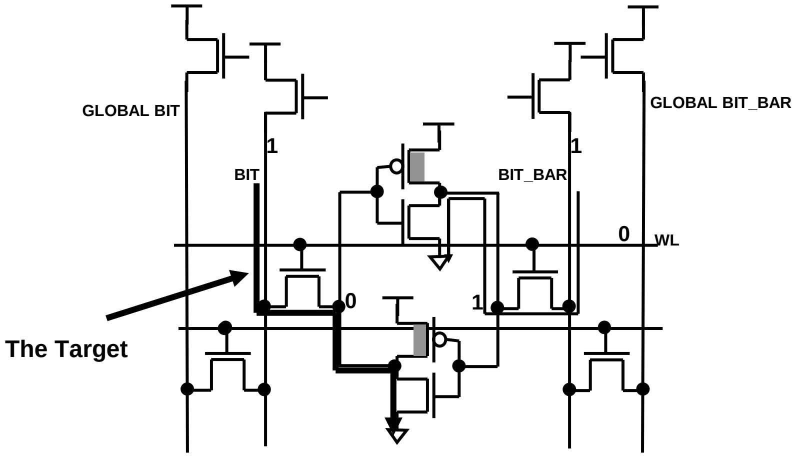


LBB for Caches (Cont’d)

- Local bitlines (32-bit cells) disconnected from senseamp by local-global switch
- LBB for Caches: If a subbank is not in use, turn off precharge transistors and delay precharging

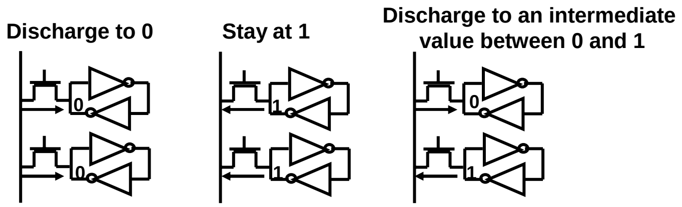


Dual Vt SRAM cell



Bitline leakage depends on the stored value

Leakage-Biased Bitlines (LBB)

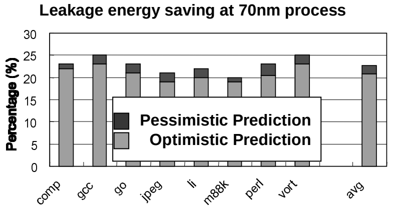


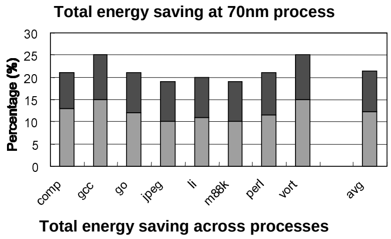
- LBB lets bitlines float by turning off the local HVT NMOS precharge transistors
  - □ No static current draw because local bitline isolated
  - □ LBB uses leakage itself to bias bitlines to the voltage which minimizes leakage!
- A good fine-grain dynamic technique
  - □ Minimal transition energy:
    - Same number of precharges (delayed precharge)
  - □ Minimal transition time:
    - Wakeup latency is only that of precharge phase

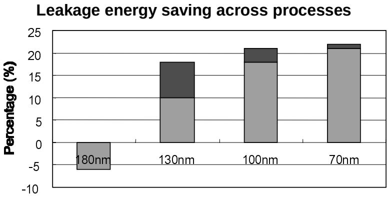
Performance Issues for LBB Caches

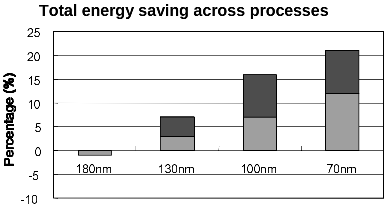
- Subbank must be precharged before use
  - □ Case 1 (best): subbank decode and precharge happen before more complex word-line decode, therefore no penalty.
  - □ Case 2 (worst): add additional pipeline stage for precharge
    - One cycle increase in branch misprediction penalty
  - □ Focus on I-Cache because any latency increase can be partly hidden by branch prediction

I-Cache Subbank Deactivation





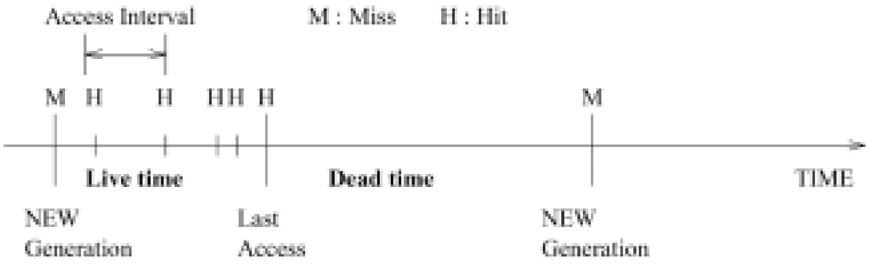




Case 2 (worst) assumption (adding additional pipeline stage) → 2.5% IPC decrease on average

Cache Decay - Preliminary

- [Kaxiras, Hu, and Martonosi, 2001]
- Consider a data cache

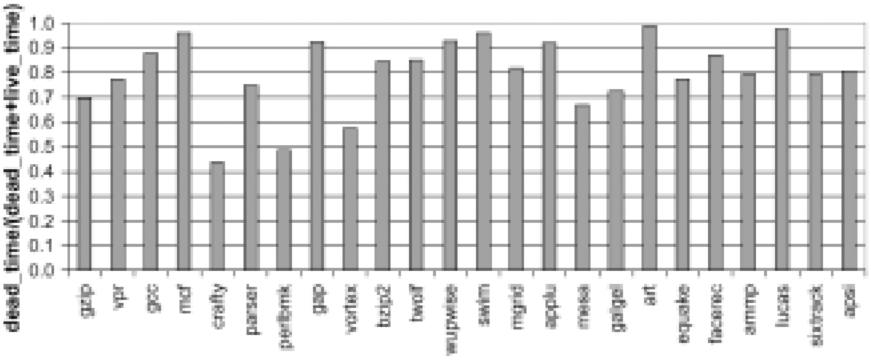


Main Idea

- During the dead time of a cache line
  - □ Discard items from the cache
  - □ Mark the lines invalid
  - □ Put the cache lines to sleep based on generational aspect of cache line usage to reduce the leakage current of cache
- The basic premise is that cache lines are storing items that will not be used again
  - □ Any static power dissipated on behalf of these cache items is wasted
- Use a transistor structure limiting the static leakage power by
  - □ Banking cache
  - □ Providing sleep transistor (Gating off the V_{dd})
- Reduce the power wasted on dead items in the cache
  - □ Without significantly worsening either program performance or dynamic power dissipation by exploiting sleep transistor at a finer granularity

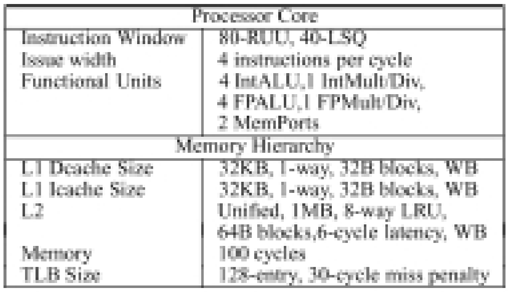
Potential Benefits

- Fraction of time that the cached data are dead
  - □ 65 % for integer benchmarks
  - □ 80 % for FP benchmarks



Methodology and Modeling

- Simulation based on
  - SimpleScalar framework
- Benchmarks from
  - SPEC CPU2000
  - MediaBench suites
- Static power
  - Saved by turning off portions of the cache
- The extra dynamic power
  - Additional hardware
    - Counter to support decay policy
  - Extra cache misses
    - Extra L2 cache reads and writebacks
      - early writebacks



Processor Core	
Instruction Window	80-RUU, 40-LSQ
Issue width	4 instructions per cycle
Functional Units	4 IntALU,1 IntMult/Div, 4 FPALU,1 FPMult/Div, 2 MemPorts
Memory Hierarchy	
L1 Dcache Size	32KB, 1-way, 32B blocks, WB
L1 Icache Size	32KB, 1-way, 32B blocks, WB
L2	Unified, 1MB, 8-way LRU, 64B blocks,6-cycle latency, WB
Memory	100 cycles
TLB Size	128-entry, 30-cycle miss penalty

$$\begin{aligned} EnergyMetric &= ActiveRatio \\ &+ (Ovhd : leak)(OvhdActivity) \\ &+ (L2Access : leak)(extraL2Accesses) \end{aligned}$$

Relating Dynamic and Static Energy Costs

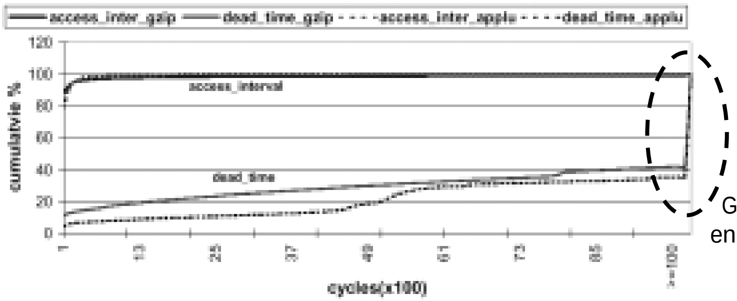
- Implications of increasing the miss rate of L1 cache
  - Dynamic power dissipation ** predominant
    - Due to an access to L2 cache, and possible additional accesses down the memory hierarchy
  - Instruction stall ** marginal
    - Interfering with smooth pipeline operation and dissipating extra power
  - Lengthened program execution cycles ** marginal
    - Lead to extra power being dissipated
- In some cases, the stalls and execution cycles decrease due to the early writebacks

Cache Decay

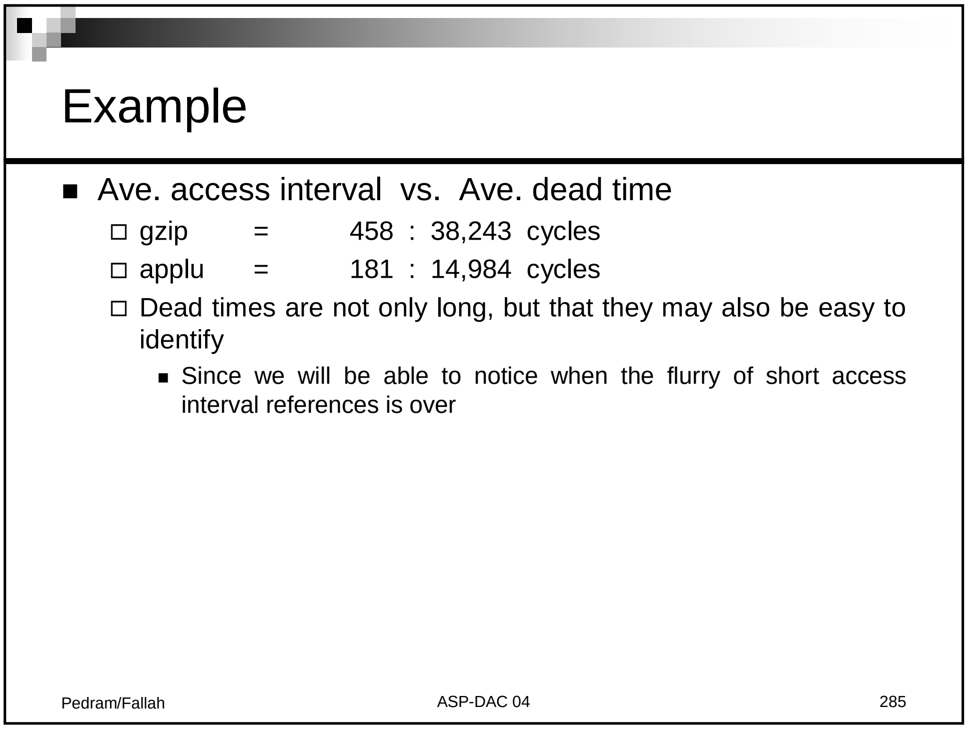
- Time-based leakage control
  - □ Balancing potential for
    - Saving leakage energy
    - Incurring extra L2 cache accesses
  - □ Based on competitive algorithm
  - □ Long wait: leakage energy increases
  - □ Immediate off: # of extra misses increase
- When to turn a cache line off
  - □ Until the static energy dissipation since its last access is equal to the overhead of an extra miss

Example

- To be effective, the wait times before turning a cache line off must be short enough to be seen in real-life
  - □ E of an L2 access = 9 * (E of L1 in a cycle)
    - Decay interval = 10,000 cycles, where 1024 lines are assumed

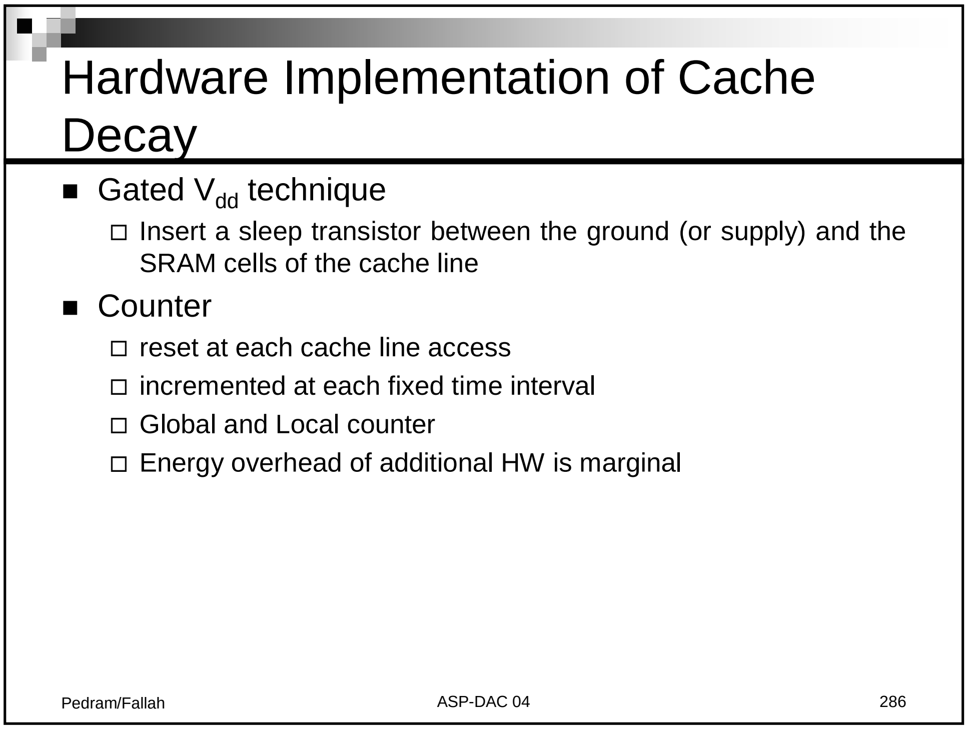


Good enough



Example

- Ave. access interval vs. Ave. dead time
  - □ gzip = 458 : 38,243 cycles
  - □ applu = 181 : 14,984 cycles
  - □ Dead times are not only long, but that they may also be easy to identify
    - Since we will be able to notice when the flurry of short access interval references is over



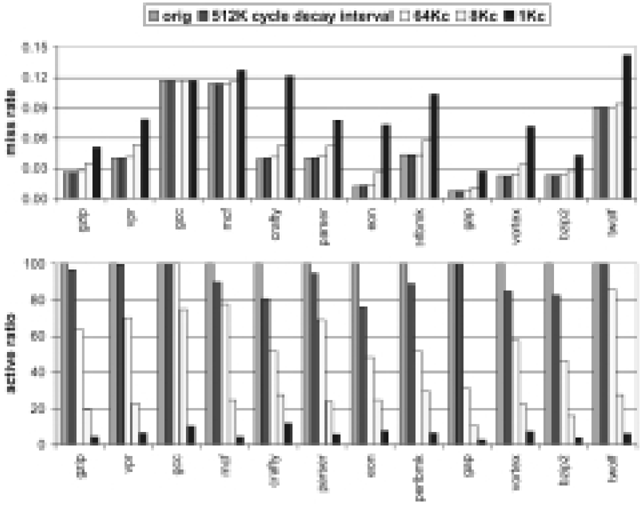
Hardware Implementation of Cache Decay

- Gated V_{dd} technique
  - □ Insert a sleep transistor between the ground (or supply) and the SRAM cells of the cache line
- Counter
  - □ reset at each cache line access
  - □ incremented at each fixed time interval
  - □ Global and Local counter
  - □ Energy overhead of additional HW is marginal

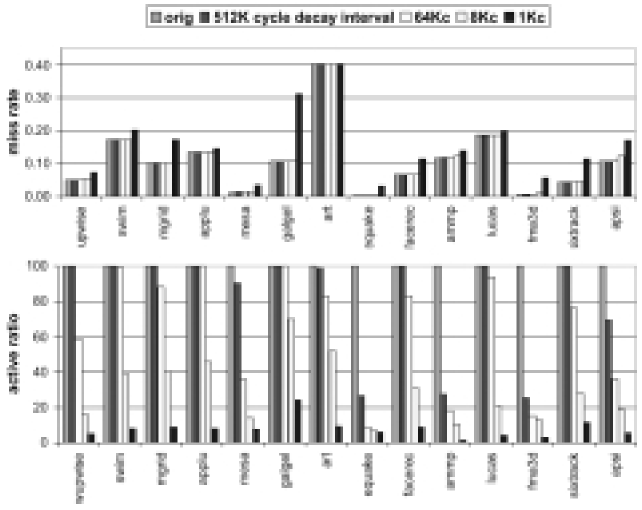
Implications of Cache Line Power Off

- The first access to a powered-off cache line
  - □ Cache miss
  - □ Resetting counter and power the cache line on
  - □ Delayed until the cache line is stabilized
  - □ So,use the valid bit during such delay

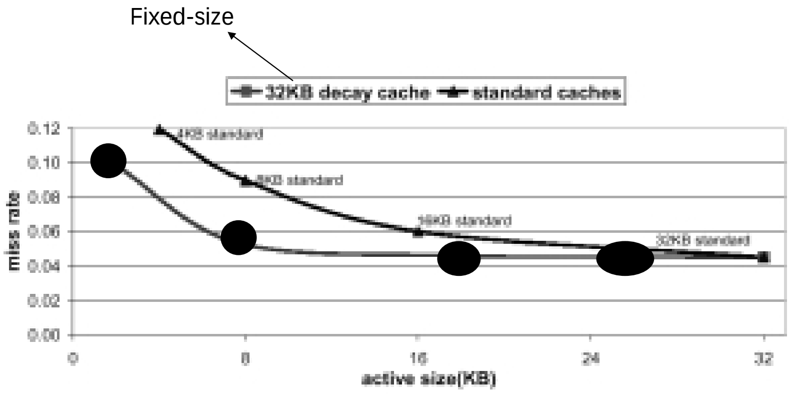
Results (SPECint 2000)



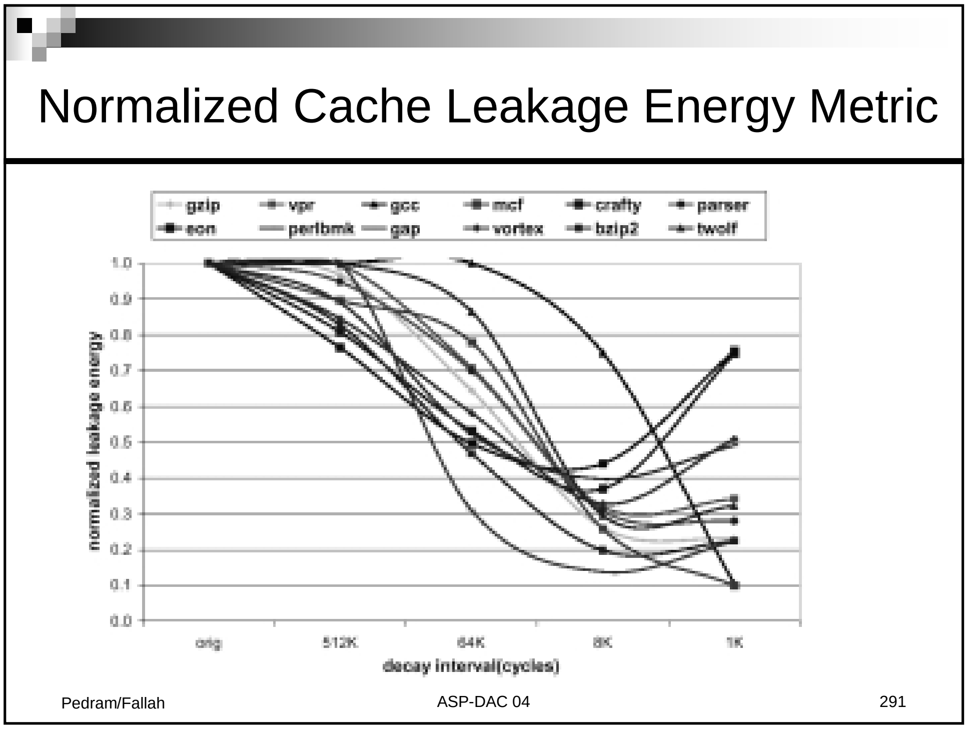
(SPEC fp2000) - SKIP



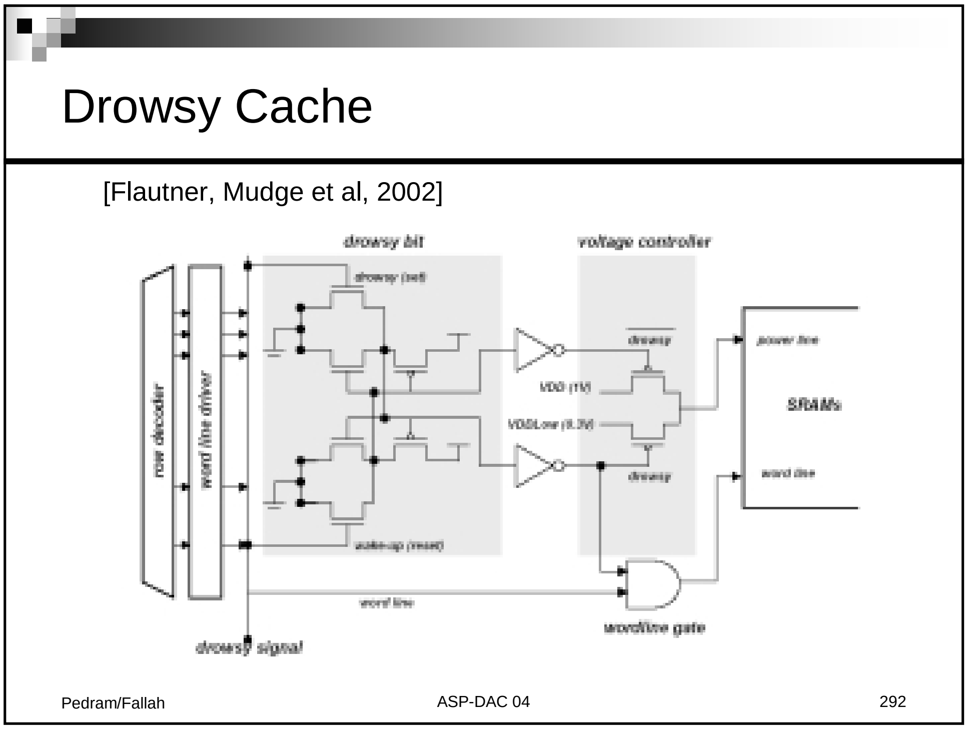
The Influence of the Decay Interval



(average over all the benchmarks)

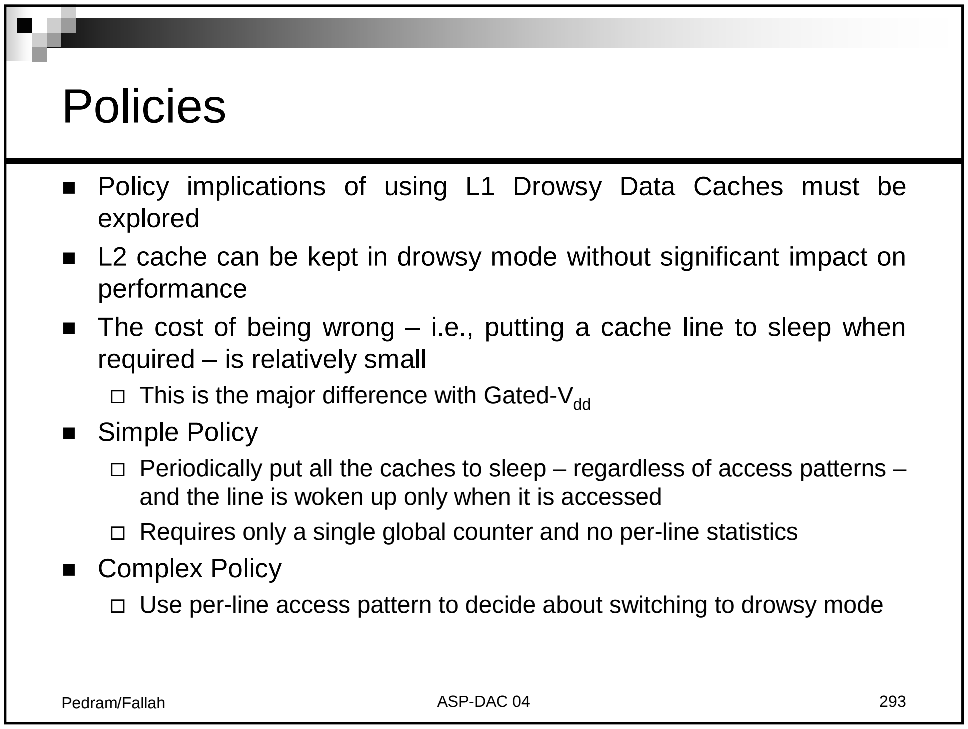


Normalized Cache Leakage Energy Metric



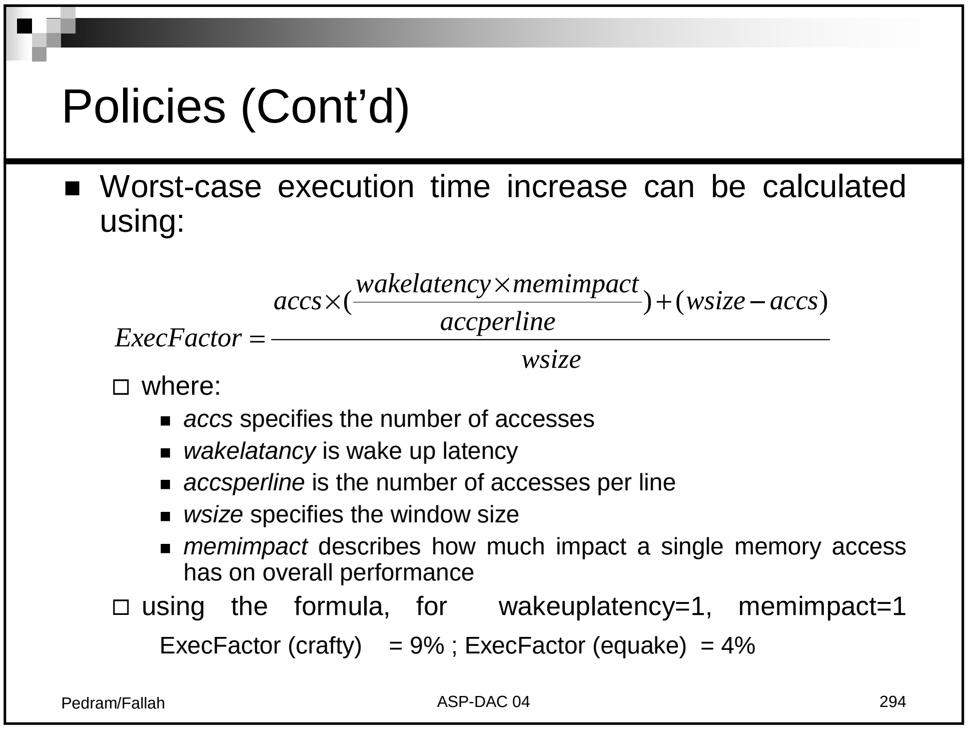
Drowsy Cache

[Flautner, Mudge et al, 2002]



Policies

- Policy implications of using L1 Drowsy Data Caches must be explored
- L2 cache can be kept in drowsy mode without significant impact on performance
- The cost of being wrong – i.e., putting a cache line to sleep when required – is relatively small
  - □ This is the major difference with Gated-V_{dd}
- Simple Policy
  - □ Periodically put all the caches to sleep – regardless of access patterns – and the line is woken up only when it is accessed
  - □ Requires only a single global counter and no per-line statistics
- Complex Policy
  - □ Use per-line access pattern to decide about switching to drowsy mode



Policies (Cont’d)

- Worst-case execution time increase can be calculated using:

$$ExecFactor = \frac{accs \times (\frac{wakelatency \times memimpact}{accperline}) + (wsize - accs)}{wsize}$$

- □ where:
  - *accs* specifies the number of accesses
  - *wakelatancy* is wake up latency
  - *accsperline* is the number of accesses per line
  - *wsize* specifies the window size
  - *memimpact* describes how much impact a single memory access has on overall performance
- □ using the formula, for wakeuplatency=1, memimpact=1

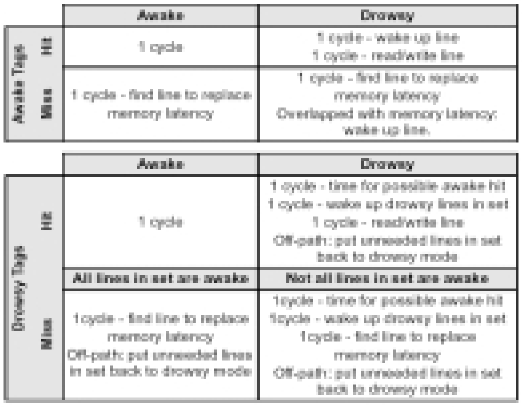
  ExecFactor (crafty) = 9% ; ExecFactor (equake) = 4%

Policies (Cont'd)

- Memory Impact is a function of both micro-architecture and the workload:
  - The workload determines the ratio of the number of memory accesses to instructions
  - The micro-architecture determines what fraction of wake up transitions can be hidden, i.e., not translated into global performance degradation
  - The micro-architecture also has a significant bearing on IPC which in turn determines the number of memory accesses per cycle
- Assuming that half of the wake-up transition latencies can be hidden by the micro-architecture, based on the ratio of 0.63 of memory accesses per cycle, ExecFactor for crafty benchmark reduces to 2.8%
- The actual impact of the technique is likely to be significantly lower than the results from analytical model

Drowsy Tags

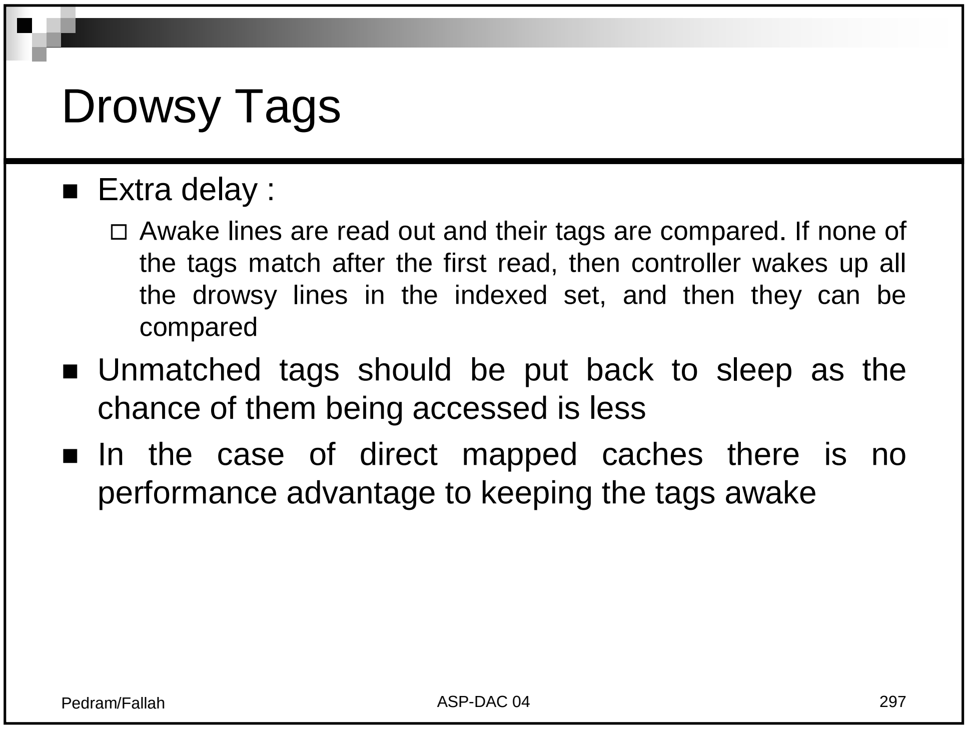
- The question is whether the tags are put into drowsy mode along with the data or whether they are always on!



		Awake	Drowsy
Awake Tags	Hit	1 cycle	1 cycle - wake up line 1 cycle - read/write line
	Miss	1 cycle - find line to replace memory latency	1 cycle - find line to replace memory latency Overlapped with memory latency: wake up line.

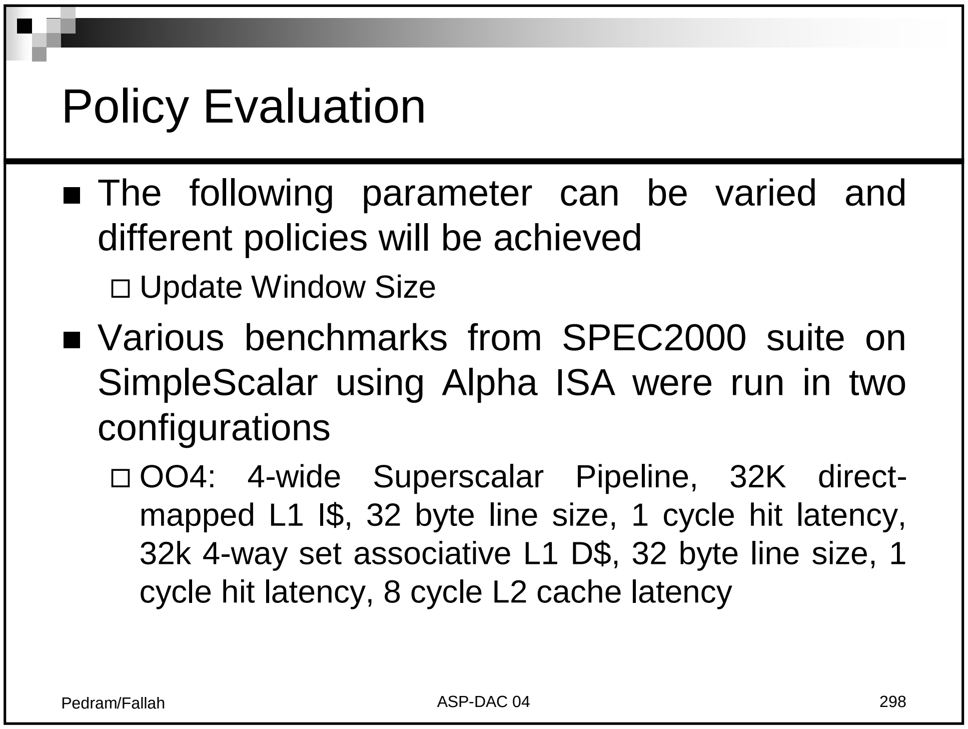
		Awake	Drowsy
Drowsy Tags	Hit	1 cycle	1 cycle - time for possible awake hit 1 cycle - wake up drowsy lines in set 1 cycle - read/write line Off-path: put unneeded lines in set back to drowsy mode
		All lines in set are awake	**Not all lines in set are awake**
	Miss	1cycle - find line to replace memory latency Off-path: put unneeded lines in set back to drowsy mode	1cycle - time for possible awake hit 1cycle - wake up drowsy lines in set 1cycle - find line to replace memory latency Off-path: put unneeded lines in set back to drowsy mode

Latencies of accessing lines in the drowsy cache



Drowsy Tags

- Extra delay :
  - □ Awake lines are read out and their tags are compared. If none of the tags match after the first read, then controller wakes up all the drowsy lines in the indexed set, and then they can be compared
- Unmatched tags should be put back to sleep as the chance of them being accessed is less
- In the case of direct mapped caches there is no performance advantage to keeping the tags awake

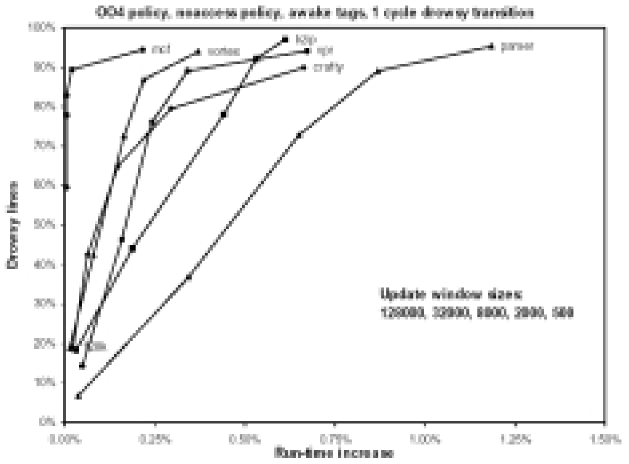


Policy Evaluation

- The following parameter can be varied and different policies will be achieved
  - □ Update Window Size
- Various benchmarks from SPEC2000 suite on SimpleScalar using Alpha ISA were run in two configurations
  - □ OO4: 4-wide Superscalar Pipeline, 32K direct-mapped L1 I$, 32 byte line size, 1 cycle hit latency, 32k 4-way set associative L1 D$, 32 byte line size, 1 cycle hit latency, 8 cycle L2 cache latency

Policy Evaluation (Cont’d)

- Impact of Window size on performance and on the fraction of drowsy lines



Summary

- A number of circuit and architecture-level optimization techniques targeting leakage current control and minimization were reviewed
- Special emphasis was placed on leakage reduction in memory cells, on-chip caches and the cache hierarchy
- Results demonstrate that a significant leakage power saving is possible depending on the logic style, circuit design, and architecture being used

References

- M. Anis, S. Areibi, M. Mahmoud, M. Elmasry, “Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique,” *Proc of 39th Design Automation Conference*, 2002, pp. 480 –485
- C. H. Kim, K. Roy, ” Dynamic V/sub t/ SRAM: a leakage tolerant cache memory for low voltage microprocessors,” *Proc. of International Conference on Low Power Electronics and Design*, 2002, pp. 251 –254
- C. H. Kim, J. J. Kim, S. Mukhopadhyay, K. Roy, “A forward body-biased low-leakage SRAM cache: device and architecture considerations,” *Proc. of International Conference on Low Power Electronics and Design*, 2003, pp. 6 –9
- S. Heo, K. Barr, M. Hampton, K. Asanovic, “Dynamic fine-grain leakage reduction using leakage-biased bitlines,” *Proc. of Annual International Symposium on Computer Architecture*, 2002, pp. 137 -147
- S. Heo, K. Asanovic, “Leakage-biased domino circuits for dynamic fine-grain leakage reduction,” *Proc of Symposium on VLSI Circuits*, 2002, pp. 316 –319
- K. Flautner, N. S. Kim, S. Martin, D. Blaauw, T. Mudge, “Drowsy caches: simple techniques for reducing leakage power,” *Proc. of Annual International Symposium on Computer Architecture*, 2002, pp.148 -157

References (Cnt’d)

- S. Kaxiras, Z. Hu; M. Martonosi, “Cache decay: exploiting generational behavior to reduce cache leakage power,” *Proc. of 28th Annual International Symposium on Computer Architecture*, 2001, pp. 240 –251
- L. Li, I. Kadayif, Y. -F. Tsai, N. Vijaykrishnan, M. Kandemir,M. J. Irwin, A. Sivasubramaniam, “Leakage energy management in cache hierarchies,” *Proc. of International Conference on Parallel Architectures and Compilation Techniques*, 2002, pp.131 -140
- R. S. Guindi, F. N. Najm, “Design techniques for gate-leakage reduction in CMOS circuits,” *Proc. of Fourth International Symposium on Quality Electronic Design*, 2003, pp. 61 -65
- Y. Li, D. Parikh, Y. Zhang, K. Sankaranarayanan, M. R. Stan, and K. Skadron. “State-Preserving vs. Non-State-Preserving Leakage Control in Caches,” *to appear in Proc. of Design, Automation and Test in Europe Conference*, 2004
- M. Powell, S. -H. Yang; B. Falsafi, K. Roy, T. N. Vijaykumar, “Gated-Vdd: a circuit technique to reduce leakage in deep-submicron cache memories”, *Proc. of International Symposium on Low Power Electronics and Design*, 2000, pp. 90 -95
- J. Kao, A. Chandrakasan, "MTCMOS Sequential Circuits," *Proc. of 27th European Solid State Circuits Conference*, 2001, pp. 332-335