

# Leakage Power Modelling and Minimization

**Massoud Pedram**

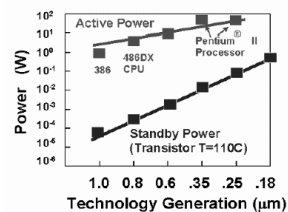
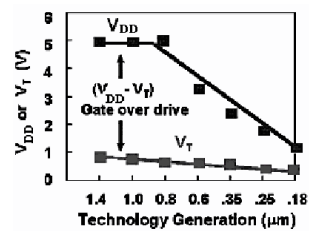
University of Southern California  
 Dept. of EE-Systems  
 Los Angeles, CA 90089

pedram@ceng.usc.edu

## CMOS Scaling

### CMOS Scaling: An Overview

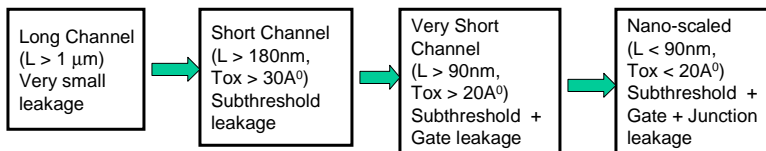
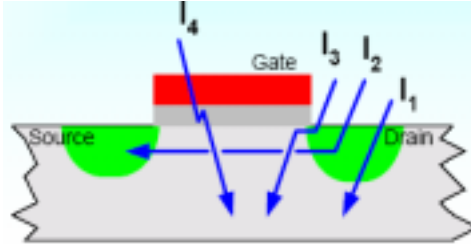
- Scaling improves:
  - Transistor Density & Functionality on a chip
  - Speed and frequency of operation  $\Rightarrow$  Higher performance
- Scaling and power dissipation
  - Active power  $\uparrow - CV_{DD}^2f$ 
    - Scale  $V_{DD}$
    - Scale  $V_{th} \Rightarrow I_{leak} \uparrow$
  - Standby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



Source: Intel

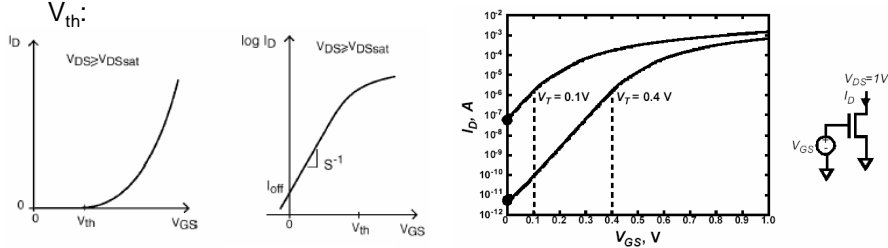
## Leakage Components in Bulk CMOS

- $I_1$  Diode reverse bias current
- $I_2$  Subthreshold current
- $I_3$  Gate induced drain leakage
- $I_4$  Gate oxide tunneling



## Subthreshold Leakage Current

$I_2$ : Transfer characteristics of MOSFET for  $V_{GS}$  near  $V_{th}$ :



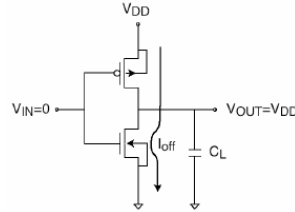
$$I_{sub} = \frac{W}{L} \mu_e v_T^2 C_{sth} e^{\frac{V_{GS} - V_{th} + \eta V_{DS}}{n v_T}} \left( 1 - e^{\frac{-V_{DS}}{n v_T}} \right) \propto e^{\frac{V_{GS} - V_{th} + \eta V_{DS}}{n v_T}} = 10^{\frac{V_{GS} - V_{th} + \eta V_{DS}}{S}}$$

- The inverse subthreshold slope,  $S$ , is equal to the voltage required to increase  $I_D$  by 10X, i.e.,
  - 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

# Modeling Subthreshold ( $I_{sub}$ ) and off ( $I_{off}$ ) Currents

## Subthreshold Leakage

- Increases exponentially with reduction in  $V_{th}$
- Modulation of  $V_{th}$  in a short channel transistor
  - $L \downarrow \Rightarrow V_{th} \downarrow$ : "V<sub>th</sub> Rolloff"
  - $V_{DS} \uparrow \Rightarrow V_{th} \downarrow$ : "Drain Induced Barrier Lowering"
  - $V_{SB} \uparrow \Rightarrow V_{th} \uparrow$ : "Body Effect"
- If  $V_{DS} = 0 \Rightarrow I_{sub} = 0$



If  $V_{DS} \gg nv_T \Rightarrow I_{sub} = \frac{W}{L} \mu_e v_T^2 C_{sth} e^{\frac{V_{GS}-V_{th}}{nv_T}}$

$W_{off} = I_{off} V_{DD}$

With  $n = 1 + \frac{\gamma}{2\sqrt{2}\Phi_f} = 1 + \frac{C_{sth}}{C_{ox}}$

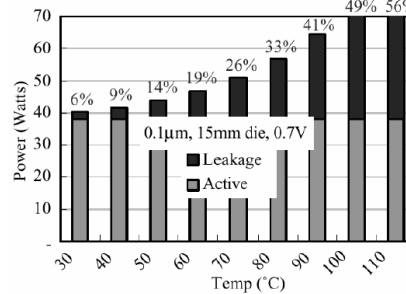
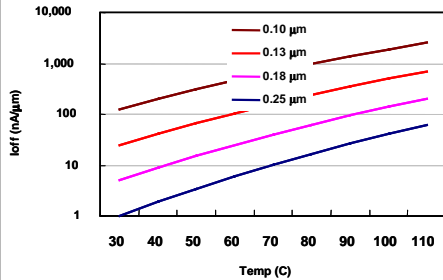
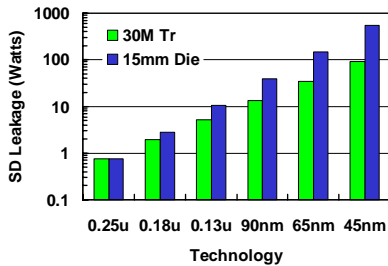
$I_{off} = I_{sub}(V_{GS} = 0) = \frac{W}{L} \mu_e v_T^2 C_{sth} e^{-\frac{V_{th}}{nv_T}}$

- 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

- Occurs when transistor is "off"

# Projected Subthreshold Leakage and Temperature Effect

## Subthreshold Leakage



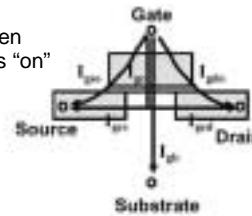
Power consumption of a 15mm die fabricated in a 0.1 μm technology with a supply voltage of 0.7V

## Gate Oxide Tunneling

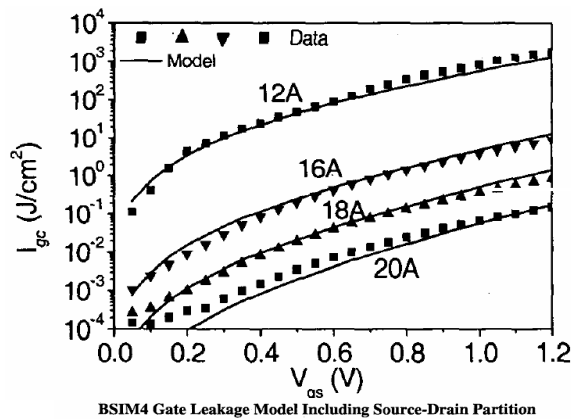
- $I_4$ : Gate oxide tunneling of electrons that can result in leakage when there is a high electric field across a thin gate oxide layer. Electrons may tunnel into the conduction band of the oxide layer; this is called Fowler-Nordheim tunneling
- In oxide layers less than 3–4 nm thick, there can also 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)
- Direct tunneling of electrons through gate oxide is the dominant source. This current depends exponentially on the oxide thickness and the  $V_{DD}$  [BSIM 4]

$$J_{DT} = A_s \left( \frac{V_{gs}}{T_{ox}} \right)^2 e^{-B_s \left( 1 - \left( 1 - \frac{V_{gs}}{\Phi_{ox}} \right)^{1.5} \right) \frac{V_{gs}}{T_{ox}}}$$

- Occurs when transistor is "on"



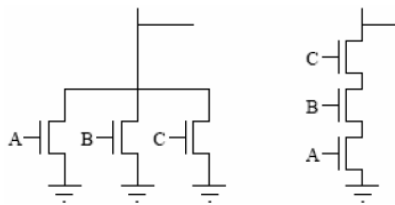
## BSIM4 Gate Leakage Model



40%  $\Delta V_{gs} \rightarrow \sim 5X I_G$

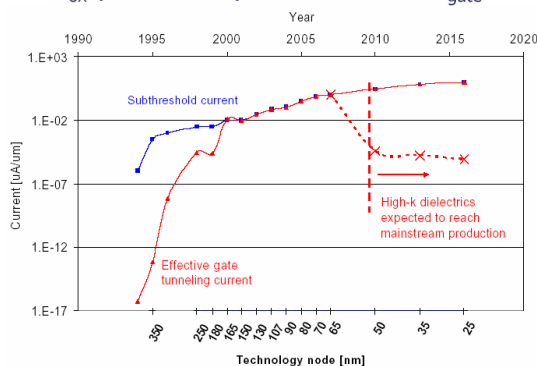
## $I_{gate}$ : NOR versus NAND Gates

- $I_{gate}$  for a PMOS device is typically one order of magnitude smaller than that of an NMOS device with identical  $T_{ox}$  and  $V_{DD}$  when using SiO<sub>2</sub> as the gate dielectric
- In a NOR-gate, each NMOS transistor will leak when turned on, independently from others. In that sense, the NOR structure is a worst-case structure
- In the NAND structure, however, we find that transistor B is at the threshold if A is off and C is on. So it is not leaking
- Transistor C leaks in only one case, when all three transistors are on



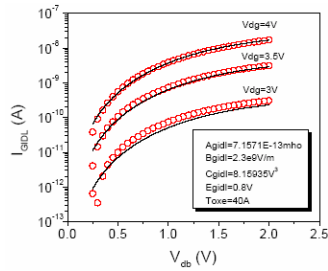
## Scaling Trends and Impact of High-K Dielectrics

- 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  $I_{gate}$

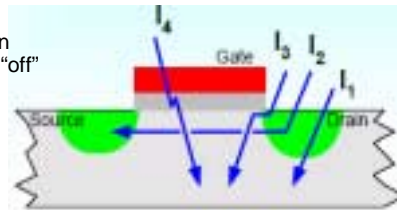


## Gate Induced Drain Leakage

- $I_3$ : Gate-induced drain leakage (GIDL) is caused by high field effect in the drain junction of MOS transistors
  - In an 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



- Occurs when transistor is "off"



E. Macii, M. Pedram

11

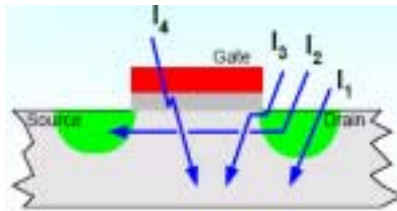
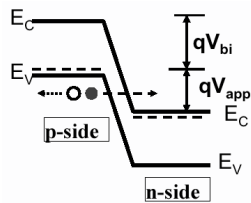
## Junction Leakage

- $I_1$ : Junction leakage that results from minority carrier diffusion & drift near the edge of depletion regions, and also from generation of electron hole pairs in the depletion regions of reverse-bias junctions
- Diode reverse bias current

$$I_{junc} = I_s \left( 1 - e^{-\frac{V_{DB}}{V_{bi}}} \right)$$

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

- When both n regions and p regions are heavily doped, as is the case for some advanced MOSFETs, there will also be junction leakage due to band-to-band tunneling (BTBT), i.e., electron tunneling from Valence Band of the p-side to the Conduction Band of the n-side



ICCAD'04 Tutorial

E. Macii, M. Pedram

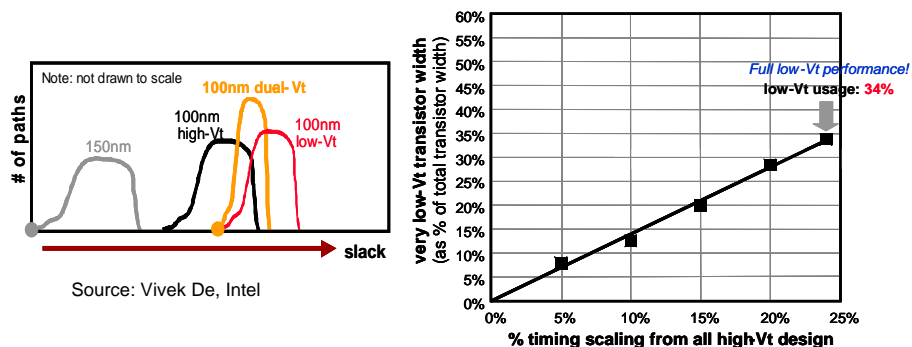
12

## Leakage Reduction Techniques

- Lowering  $V_{dd}$  (voltage islands, dynamic voltage scaling)
- SOI technology
- Cooling and/or refrigeration
- Gate-level dual- $V_{th}$  design
- Mixed- $V_{th}$  (MVT) CMOS design
- Transistor sizing (shorter  $W$ , longer  $L$ )
- Transistor stacking
- Body bias control (static and/or adaptive)
- Input vector control during sleep mode
- MTCMOS (sleep transistors, power gating)

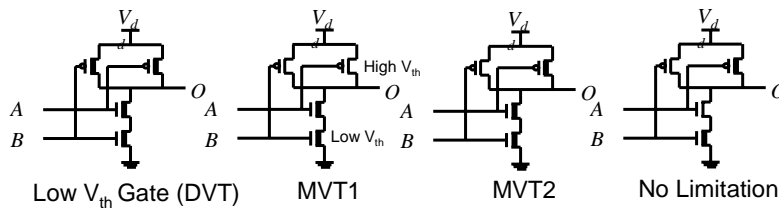
## Gate-level Dual $V_{th}$ Design

- 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



## Mixed- $V_{th}$ CMOS Design

- 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  $V_{th}$
- Use multiple types of transistors within each gate
  - MVT1: Same threshold voltage for all transistors in N or P networks
  - MVT2: Same threshold voltage only for all transistors of a series stack
  - No limitation (possible in some processes)

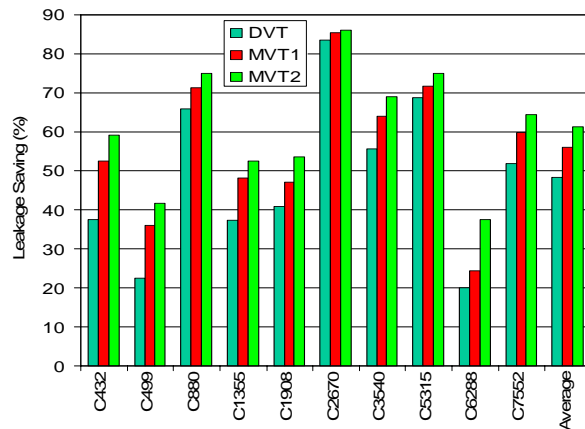


## MVT CMOS Design Algorithm

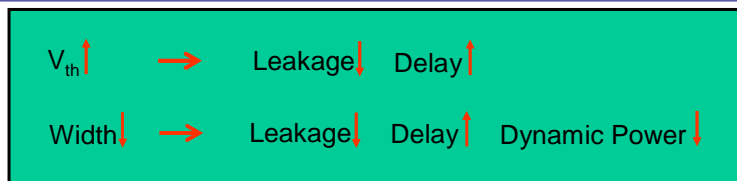
- Assume all low- $V_{th}$  transistors
- For each transistor of each gate,
  - Find the increase in the gate **delay** if high- $V_{th}$  is used ( $\Delta t_d$ )
  - Find the decrease in the gate **leakage** if high- $V_{th}$  is used ( $\Delta leak$ )
  - Calculate  $priority(i) = \frac{\Delta leak_i}{\Delta t_{d_i}}$
  - Higher value means more leakage can be saved using one unit of slack
  - The transistors are processed based on their  $priority(i)$  values
  - After modifying each transistor, the slack values have to be recalculated



## MVT Design Results

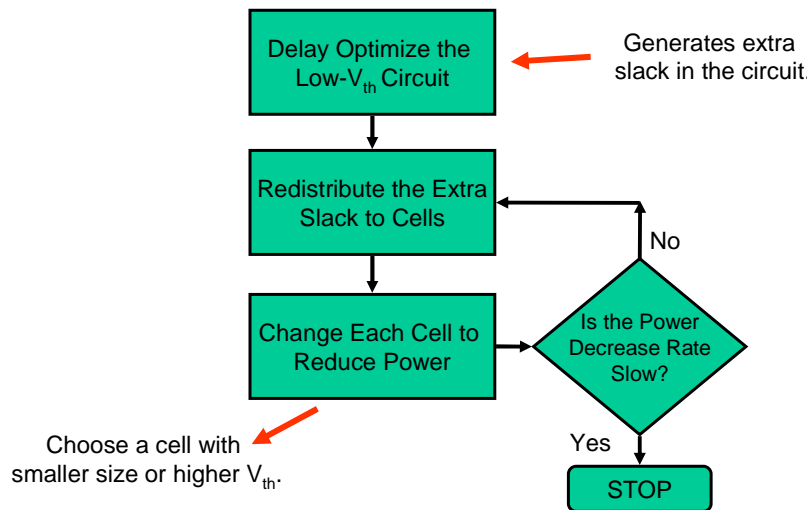


## Simultaneous Sizing and Dual- $V_{th}$ Assignment



- Complex optimization problem
  - Non-linear delay and power models
  - Leakage is a function of input values of logic gates
  - 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 the gate level
  - Use the dominant leakage states when calculating the total power consumption
  - Use an accurate delay model which takes the effect of sizing into account

## A Three-Step Algorithm

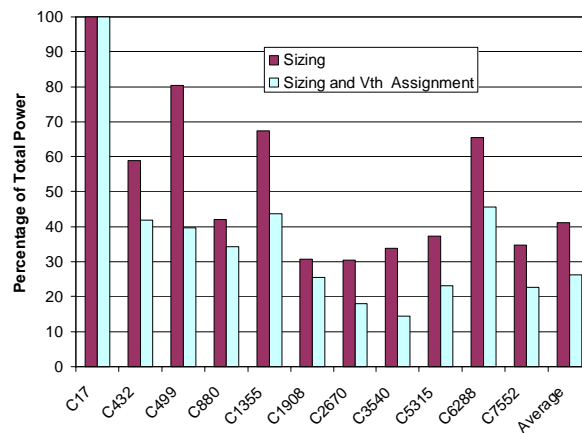


## Slack Redistribution

- 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

## Results of Sizing and Dual- $V_{th}$ Assignment

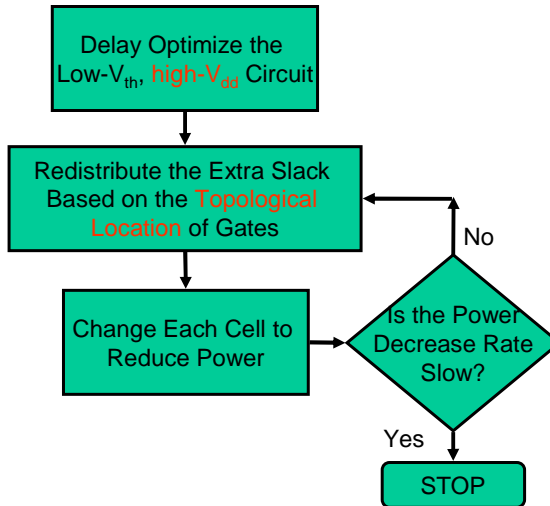
- 0.18 $\mu$ m process
  - high- $V_{th} = \pm 0.45V$ , low- $V_{th} = \pm 0.30V$ ;  $V_{dd} = 1.8V$
- Developed gates with 6 different sizes ranging from 0.18 $\mu$ m to 1.8 $\mu$ m



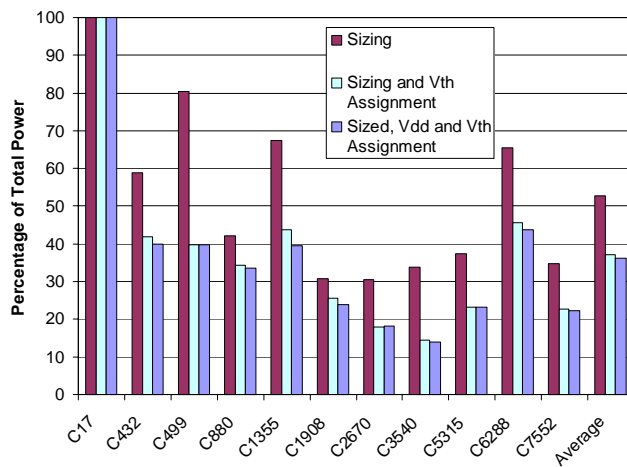
## Extending the Algorithm

- 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 kind of gate
- Due to topological constraints and the fact that low- $V_{dd}$  gates have significantly higher delay, the original slack distribution method is not good

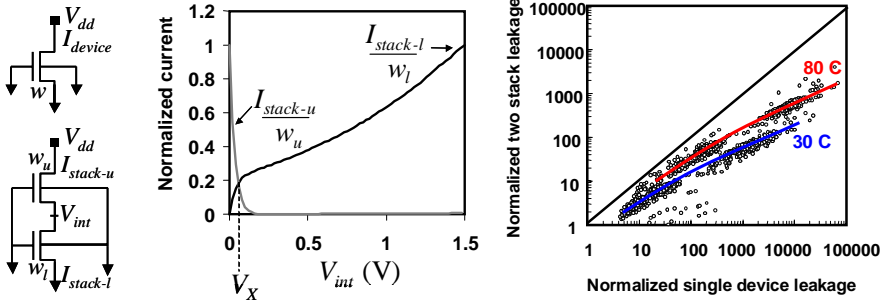
## Extending the Algorithm



## Results of Sizing, Dual $V_{th}$ and Dual $V_{dd}$

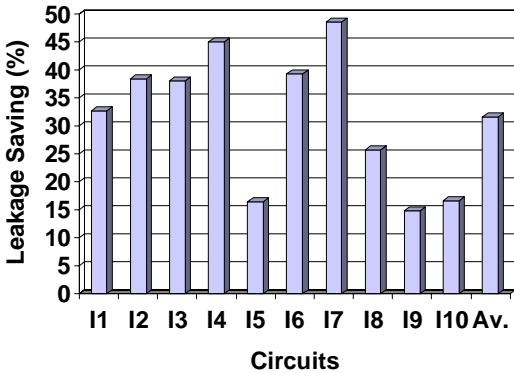


# Leakage Current of Transistor Stacks



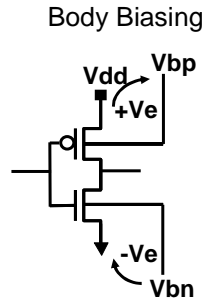
Source: Vivek De, Intel

# Results of Transistor Stacking



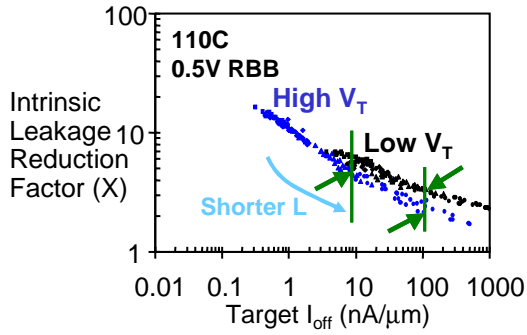
## Reverse and Forward Body Biasing

- Reverse Body Biasing (RBB)
  - No Bias = Low  $V_{th}$
  - Apply Reverse Bias = High  $V_{th}$
- Forward Body Biasing (FBB)
  - No bias = High  $V_{th}$
  - Apply Forward Bias = Low  $V_{th}$

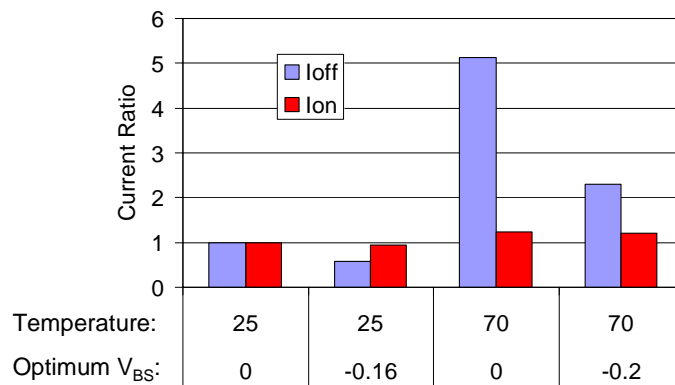


RBB becomes less effective at shorter L and lower  $V_{th}$

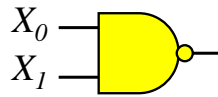
\* A. Keshavarzi et. al., 1999 & 2001  
International Symp. Low Power Electronics & Design (ISLPED)



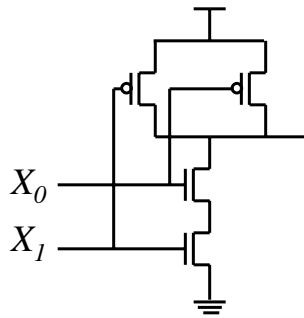
## Results of Reverse Body Biasing



## Input Dependence of the Leakage Current

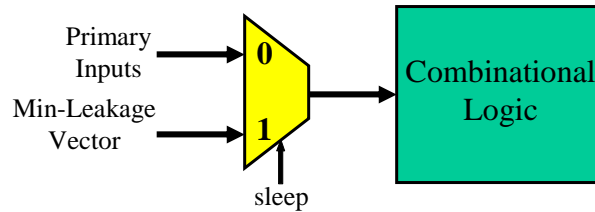


Technology: 0.18  $\mu\text{m}$   
 Supply Voltage = 1.5V  
 Threshold Voltage = 0.2V

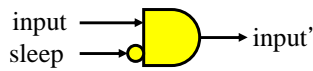


$X_0$	$X_1$	Leakage
0	0	23.60 nA
0	1	47.15 nA
1	0	51.42 nA
1	1	82.94 nA

## Input Vector Control During Sleep Mode

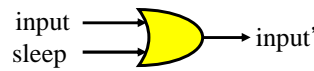


Min-Leakage Input = 0



sleep	input'
0	input
1	0

Min-Leakage Input = 1

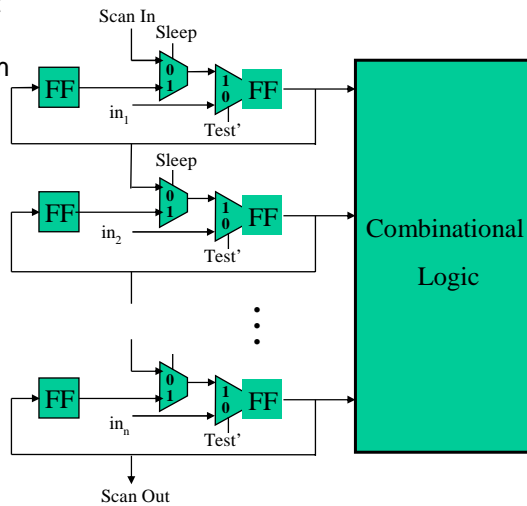


sleep	input'
0	input
1	1

## Modifying Scan Chain Registers to Preserve Circuit State

Input Vector Control

- Originally MLV is stored in left Flip-Flops.
- While changing the mode from Sleep to operation and vice versa the content of flip-flops are swapped.
- . Sleep mode:
  - Sleep = 1
  - Test' = 1
  - MLV is applied (right FF's); State is stored in the left FF's
- . Operational mode:
  - Sleep = 0
  - Test' = 0
  - Inputs are directly applied to combinational logic; MLV is stored in the left FF's



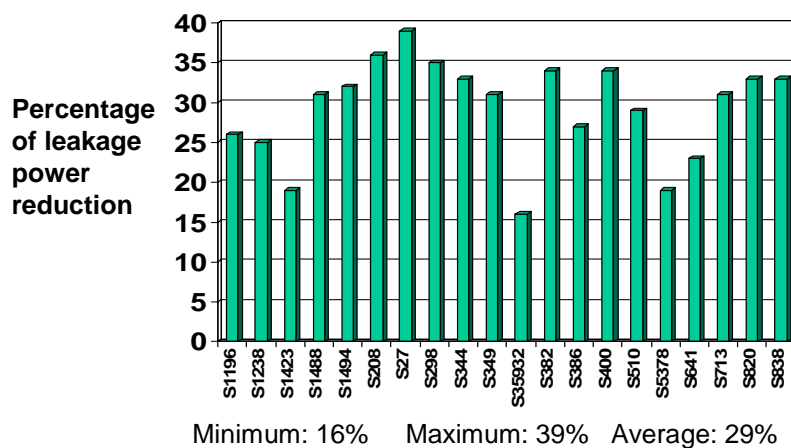
ICCAD'04 Tutorial

E. Macii, M. Pedram

31

## Results of Input Vector Control

Input Vector Control



ICCAD'04 Tutorial

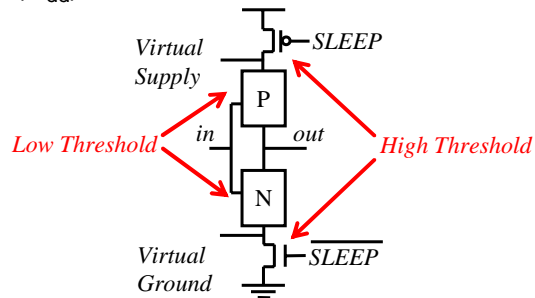
E. Macii, M. Pedram

32



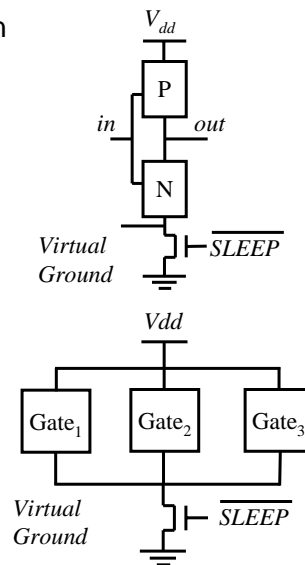
## Multi-Threshold CMOS (MTCMOS)

- 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 ( $V_{dd}$ )

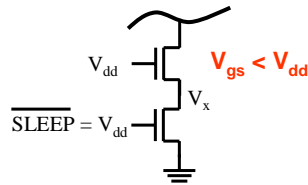


## Simplifications

- Instead of two sleep transistors, one can be used
- Usually NMOS:
  - $\mu_n > \mu_p \rightarrow$  smaller size
  - However, PMOS usually has a lower leakage
- One sleep transistor can be shared between several gates
  - Reduction in the number of sleep transistors, area overhead, dynamic and leakage power dissipations
  - Increase in the complexity of design optimization process



## Sleep Transistor Sizing



- Reduction in the high to low transition due to:
  - Reduction in the gate overdrive from  $V_{dd}$  to  $V_{dd} - V_x$
  - Increase in the threshold voltage of top NMOS transistor due to the body effect
- Increase the sleep transistor width to solve the problem
  - Increase in the area overhead, dynamic power and leakage
- As technology scales down, we have to enlarge the sleep transistor

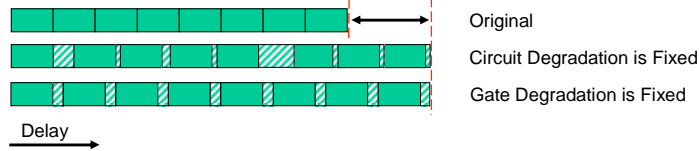
## Important Questions

- How many sleep transistors?
  - Affects the area overhead, the dynamic power overhead, and the leakage power saving
- How to cluster gates?
  - Affects 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

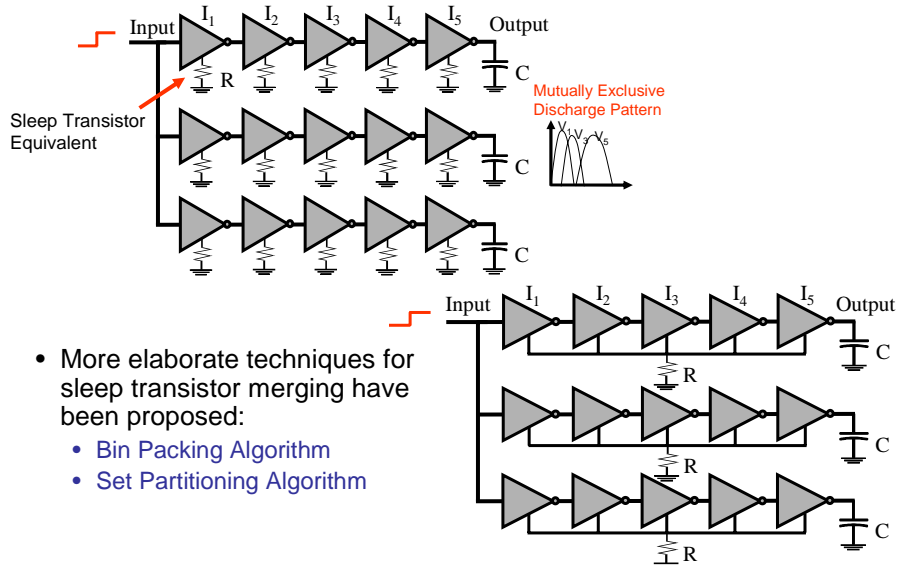
Granularity	Gate	Cluster of Gates
Methodology	Cell-based	Custom
Number of Sleep Transistors	One	More Than One
Type	NMOS	PMOS
Threshold Voltage of the Sleep Transistor	Fixed	Variable
Sizing	Exhaustive	Conservative

## Sizing: Exhaustive Method

- Objective: Finding the size of the sleep transistor for a given *overall* circuit delay degradation ( $\Delta delay$ )
- Exhaustively simulate a circuit with sleep transistors under all possible input vectors
  - Will find the optimum size
  - Works well for simple circuits e.g., cells in an ASIC library; Impractical for large circuits
- Alternative: Limit the delay degradation of each logic gate to  $\Delta d$
- Find the optimum size of the sleep transistor for each logic gate
  - More restrictive, but much easier to achieve
  - Assumes high-to-low transitions of the gate output on every critical path
- Combine the sleep transistors for different gates



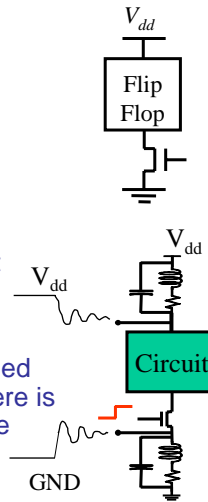
## Mutual Exclusion and Sleep Transistor Merging



- More elaborate techniques for sleep transistor merging have been proposed:
  - Bin Packing Algorithm
  - Set Partitioning Algorithm

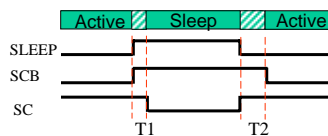
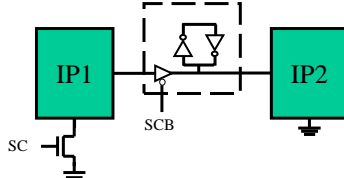
## Problems with MTCMOS

- The electro-migration effect on vias and wires
  - The number of vias is determined based on the average and the maximum allowable currents for each via
- MTCMOS cannot be easily applied to Flip Flops
  - May use balloon latches, etc.
- In an SoC, not all IP blocks are guarded
  - Short circuit current will result if a guarded output drives a regular input
- Ground bounce problem
  - During the sleep period, internal nodes are charged to  $V_{dd}$ ; When the sleep transistor is turned on, there is a current spike flowing to the ground (due to large  $V_{DS}$ ); This creates large  $V_{dd}$  and ground noise

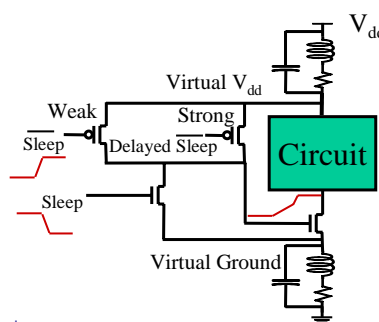
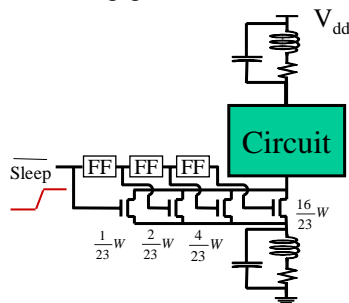


## Solutions to DC Current and Ground Bounce

- Preventing short circuit current in unguarded blocks



- Preventing ground bounce



## Results of MTCMOS

- The MTCMOS techniques was applied to a 32-bit RISC processor used in a PDA
- Ground bounce: average=9mV, max=49mV
- Performance degradation = 2%

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 w/ MTCMOS	Reduction
2μW	6000x

- Can Combine MTCMOS and dual  $V_{th}$ 
  - All logic cells in the circuits are made of high  $V_{th}$  transistors
  - Cells on the critical timing paths are replaced by MTCMOS cells

## Summary

- Leakage currents are rising fast and must be controlled by circuit design and optimization tools
- Gate leakage is rising at the fastest rate, but is expected to be controlled by the introduction of high-K dielectric material; thus, subthreshold leakage remains the most worrisome component of standby power dissipation
- Voltage islands, Dual- $V_{th}$  designs, and MTCMOS technique appear to be the most effective solutions for minimizing the subthreshold leakage current
- With lower  $V_{dd}$ 's, lower  $V_{th}$  values, and environmental and/or process technology parameter variations, the task of controlling subthreshold leakage will become even more difficult
- Logic synthesis for leakage control (esp. in light of statistical parameter variations) and physical design to support multiple  $V_{dd}$ 's or MTCMOS need to be developed further

## 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.

## Dynamic Voltage and Frequency Scaling

**Massoud Pedram**

*University of Southern California*

*Dept. of EE-Systems*




*Los Angeles, CA 90089*

pedram@ceng.usc.edu

## DPM vs. DVFS (I)

- Dynamic power management
  - Changes the power state of system components to lower the energy consumption depending on the performance constraints
  - Components transition between active and low-power states
  - Power manager observes system and responds at run-time
- Dynamic voltage and frequency scaling
  - Adjust performance and energy consumption levels while the device is active
  - Key is to meet users performance needs while saving energy
  - Reduce processor frequency and voltage to obtain quadratic energy savings

## DPM vs. DVFS (II)

	# of cycle	Voltage	Energy	Saving
• With nothing  No power-down	12.5x10 <sup>6</sup>	5	125mJ	-
• DPM  with power-down	5x10 <sup>6</sup>	5	50mJ	60%
• DVFS 	5x10 <sup>6</sup>	2	8mJ	93.6%

$$E \propto V^2$$

$$f \propto V$$

$$E \propto V^3$$

## Key Idea

- DVFS is a method through which variable amount of energy is allocated to perform a task
- Power consumption of a digital CMOS circuits is:

$$P = \alpha \cdot C_{eff} \cdot V^2 \cdot f$$

$\alpha$  : switching factor  
 $C_{eff}$  : effective capacitance  
 $V$  : operating voltage  
 $f$  : operating frequency

- Energy required to run a task during T is:

$$E = P \cdot T \propto V^2 \quad (\text{assuming } f \propto V, T \propto f^{-1})$$

- Lowering V (while simultaneously and proportionately cutting f) causes a quadratic reduction in E

## Choosing a frequency in DVFS

- Workload of a task,  $W_{task}$ , is defined as the total number of CPU clock cycles required to finish the task

$$W_{task} \triangleq \sum_{i=1}^n CPI_i$$

$n$  : total number of instructions in a task  
 $CPI$  : clock cycles per instruction

- The task execution time,  $T_{task}$ , is a function of CPU frequency,  $f_{cpu}$

$$T_{task} = \frac{W_{task}}{f_{cpu}}$$

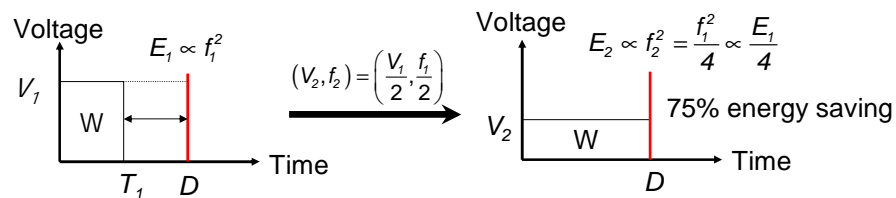
- To save CPU energy using DVFS for a given deadline,  $D$ , choose a  $f_{cpu}$  at which  $T_{task}$  can be closest to  $D$

$$T_{task} = D, \quad f_{cpu} = \frac{W_{task}}{D}$$



## DVFS by Example

- Suppose that a task with workload  $W$  should be finished by deadline  $D$



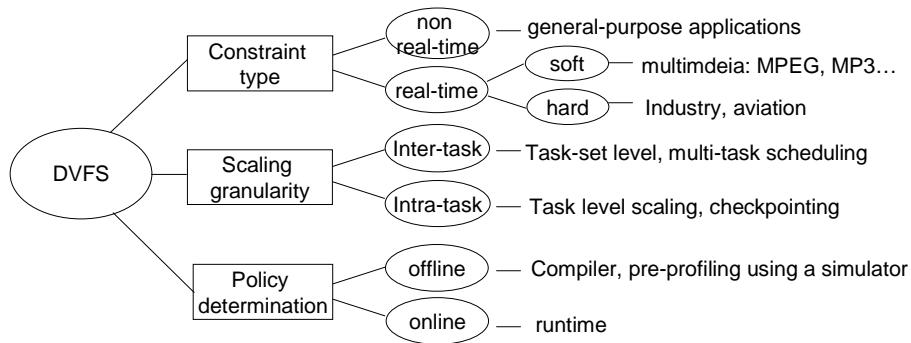
- DVFS is an effective way of reducing the CPU energy consumption by providing “just-enough” computation power

## CPUs Equipped with DVFS Functionality

	Voltage Range	Frequency Range
IBM PowerPC 405LP	1.0V-1.8V	153M-333M
Transmeta Crusoe TM5800	0.8V-1.3V	300M-1G
Intel SA1110	1.1V-1.7V	59M-221M
Intel XScale 80200	0.95V-1.5V	266M-733M
Intel PXA255	0.85V-1.3V	100M-400M

## DVFS Classification

- DVFS techniques may be classified based on three factors:



## Inter-task vs. Intra-task DVFS

- Classification is based on the scaling granularity
- Inter-task DVFS
  - Scaling occurs at the start of a task
    - It is unchanged until the task is completed
  - Use worst-case slack time ( $= \text{Deadline}_{\text{task}} - \text{WCET}_{\text{task}}$ )
  - Usually used in multi-task scheduling scenario at OS level
- Intra-task DVFS
  - Scaling occurs at the sub-task level
    - Different frequency is set for each sub-task
  - Use workload-variation slack time
    - Average-Case Execution Time (ACET) rather than Worst-Case Execution Time (WCET)
  - Much finer granularity than inter-task
  - Fully exploits the slack time arising from task execution time variation
  - Requires off-line profiling and source code modification
  - Can achieve higher energy saving compared to inter-task
  - Energy and delay overheads of voltage switching must be carefully considered

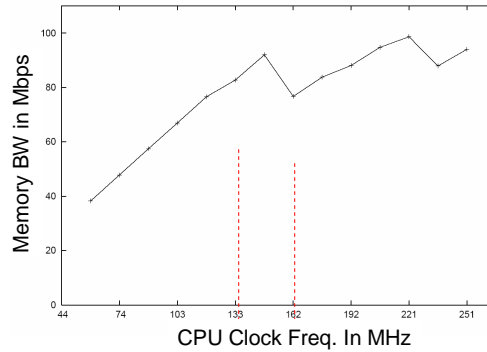
## Performance vs. CPU Speed -1

- Dynamic voltage scaling on a low-power microprocessor,” Pouwelse et al., 2001
- Considered memory power/performance
- Memory bandwidth is not linearly proportional to the CPU frequency

### LART system

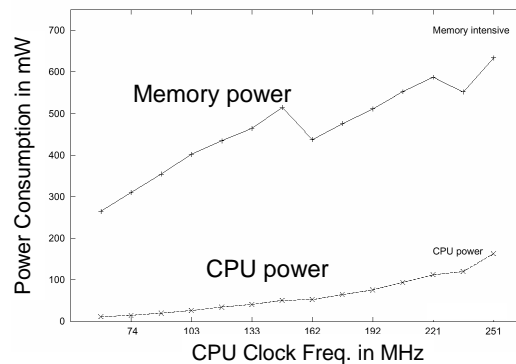
- SA1100 processor-based
- 32MB EDO-DRAM
- 60ns access time

*faster at 133MHz than at 162MHz in case of memory bound applications*



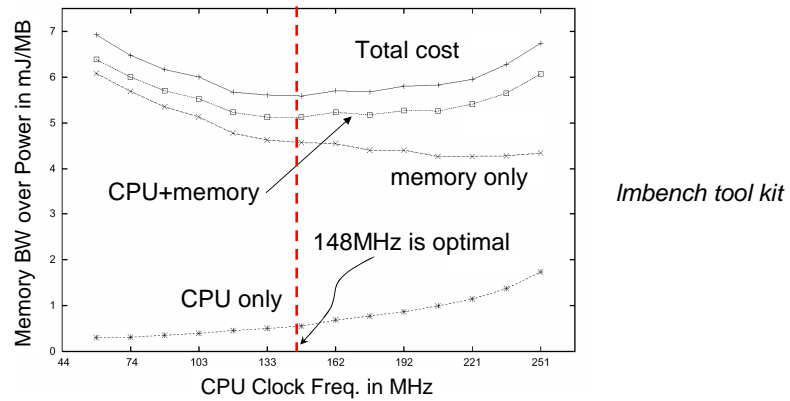
## Performance vs. CPU Speed -2

- Non-linearity of memory bandwidth also affects the power consumption
- Memory chip typically operates at 3.3V



## Performance vs. CPU Speed -3

- Energy breakdown for memory read

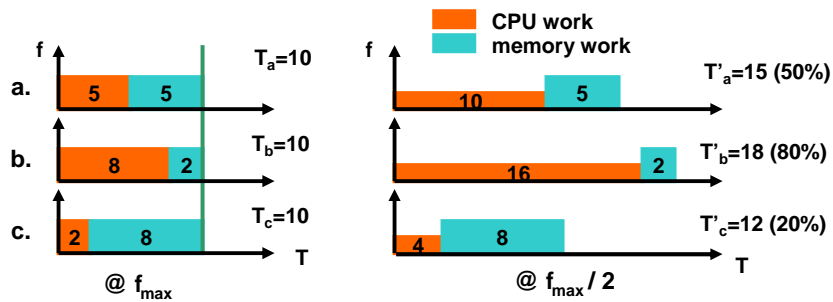


## DVFS Methods exploiting the Memory Asynchrony

- “Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors,” Hsu et al., 2001
- Use microarchitecture to reduce the CPU frequency during memory access
- “On the use of microarchitecture-driven dynamic voltage scaling,” D. Marculescu 2000
  - L1-cache miss drives voltage scaling
- “VSV: L2-miss-driven variable supply-voltage scaling for low power,” Li et.al., 2003
  - Transit to lower power mode when L2-cache miss is detected

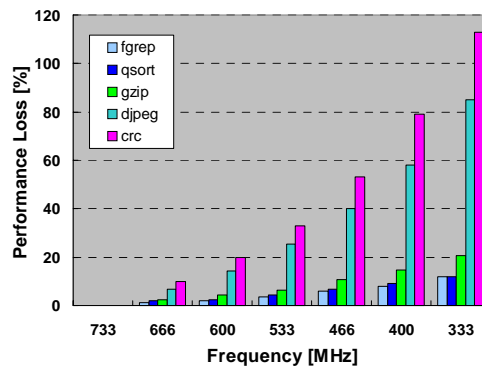
## DVFS with Workload Decomposition

- “Fine-Grained Dynamic Voltage and Frequency Scaling for Precise ...,” Choi et.al., 2004
- A program execution sequence consists of CPU and memory instructions
- The CPU has to stall until the external memory access is completed
  - With DVFS, CPU energy is saved with little performance loss



## Motivation

- Performance degradation for different frequencies
- More CPU energy saving is possible with a given performance loss target for memory-intensive applications



## The Program Execution Time

- The amount of CPU and memory workload for an application program must be determined
- Execution time of a program is the sum of the On-chip(CPU work) and the Off-chip Latency (memory work)
- $T = T_{on} + T_{off}$
- $T_{on}$  : varies with the CPU frequency
  - Cache hit
  - Stall due to data dependency
  - TLB hit ...
- $T_{off}$  is invariant with the CPU frequency
  - Access to external memory such as SDRAM and frame buffer memory through the PCI, which is in turn due to a cache miss

## Calculating the Program Execution Time

- $T = T_{onchip} + T_{offchip}$

$$T_{on} = \frac{\sum_{i=1}^n CPI_{on}^i}{f_{cpu}}$$

$$T_{off} = \frac{\sum_{j=1}^m CPI_{off}^j}{f_{mem}}$$

n : number of onchip instructions  
 $CPI_{on}$  : CPU clocks per instruction  
 $f_{cpu}$  : CPU clock frequency (varied)

m : number of offchip events  
 $CPI_{off}$  : memory clocks per offchip event  
 $f_{mem}$  : memory clock frequency (fixed)

- When all parameters are know and the target performance loss( $PF_{loss}$ ) is specified, then CPU frequency is calculated by:

$$f_{target} = \frac{\sum_{i=1}^n CPI_{on}^i}{(1 + PF_{loss}) \cdot \sum_{i=1}^n CPI_{on}^i + \frac{PF_{loss} \cdot \sum_{j=1}^m CPI_{off}^j}{f_{mem}}} \cdot f_{max}$$

$$PF_{loss} = 0 \Rightarrow f_{target} = f_{max}$$

$$PF_{loss} \uparrow \Rightarrow f_{target} \downarrow$$

$$PF_{loss} \downarrow \Rightarrow f_{target} \uparrow$$

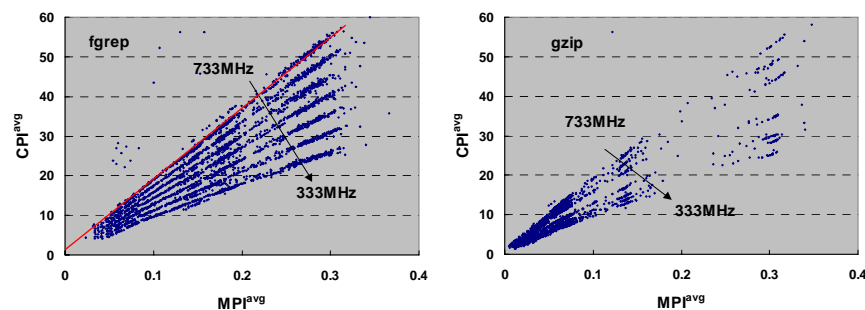
$$T_{offchip} \uparrow \Rightarrow f_{target} \downarrow$$

## Performance Monitoring Unit (PMU)

- PMU on the XScale 80200 processor chip can report up to 20 different dynamic events during execution of a program
  - Cache hit/miss counts
  - TLB hit/miss counts
  - No. of external memory accesses
  - Total no. of instructions being executed
  - Branch misprediction counts
- However, only two events can be monitored and reported at any given time
- For DVFS, we use PMU to generate statistics for
  - Total no. of instructions being executed (INSTR)
  - No. of external memory accesses (MEM)
- We also record the no. of clock cycles from the beginning of the program execution (CCNT)

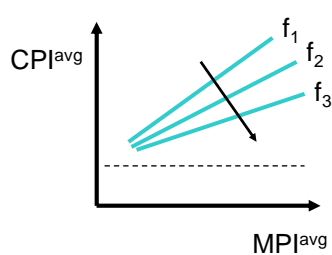
## Plot of CPI vs. MPI

- PMU is read at every OS quantum (~50msec)
- We define MPI as the ratio of memory access count to the total instruction count
  - $CPI^{avg} = CCNT / INSTR$ , during a quantum
  - $MPI^{avg} = MEM / INSTR$ , during a quantum
- A plot of  $CPI^{avg}$  vs.  $MPI^{avg}$  with changing frequency



## Regression Equation Modeling

- A linear regression equation can be generated for each CPU clock frequency



$$CPI^{avg} = b(f) * MPI^{avg} + c$$

$$b = \frac{N \cdot (\sum_{i=1}^{t-N+1} x_i \cdot y_i) - (\sum_{i=1}^{t-N+1} x_i) \cdot (\sum_{i=1}^{t-N+1} y_i)}{N \cdot (\sum_{i=1}^{t-N+1} x_i^2) - (\sum_{i=1}^{t-N+1} x_i)^2}$$

$$c = \frac{\sum_{i=1}^{t-N+1} y_i}{N} - b \cdot \frac{\sum_{i=1}^{t-N+1} x_i}{N}$$

N : No. of regression points, e.g., 25  
 $x_i$  : MPI<sup>avg</sup> for the  $i^{\text{th}}$  point  
 $y_i$  : CPI<sup>avg</sup> for the  $i^{\text{th}}$  point

## How the PMU Data Is Used in DVFS

- Target frequency for a given  $PF_{loss}$

$$f_{target} \approx \frac{n \cdot CPI_{on}}{(1 + PF_{loss}) \cdot n \cdot CPI_{on} + PF_{loss} \cdot m \cdot CPI_{off}} + f_{mem}$$

$f_{max}$  (733 MHz) points to  $f_{max}$  in the denominator.  
 given points to  $PF_{loss}$ .  
 $f_{mem}$  (100 MHz or 33 MHz) points to  $f_{mem}$  in the denominator.

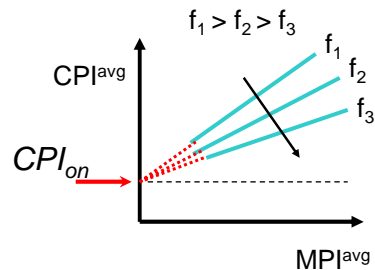
Legend: ○ known, ○ unknown

- The four unknown parameters (circled in red) must be calculated from CCNT and the two reported values by the PMU (INSTR & MEM)
  - $n$  (no. of executed instructions) ← INSTR
  - $m$  (no. of offchip events) ← MEM
  - $CPI_{on}$  ← ?
  - $CPI_{off}$  ← ?



## Calculating $CPI_{on}$

- Notice that  $CPI_{on}$  denotes the CPI value without the offchip access; So it is equal to the y intercept of the CPI vs. MPI plot



$$CPI_{on} = c$$

$$c = \frac{\sum_{i=t}^{t-N+1} y_i}{N} - b \cdot \frac{\sum_{i=t}^{t-N+1} x_i}{N}$$

$$b = \frac{N \cdot \left( \sum_{i=t}^{t-N+1} x_i \cdot y_i \right) - \left( \sum_{i=t}^{t-N+1} x_i \right) \cdot \left( \sum_{i=t}^{t-N+1} y_i \right)}{N \cdot \left( \sum_{i=t}^{t-N+1} x_i^2 \right) - \left( \sum_{i=t}^{t-N+1} x_i \right)^2}$$

## Calculating $CPI_{off}$

- It is difficult to get  $CPI_{off}$  directly from the PMU events
  - $CPI_{off}$  accounts for both the SDRAM access (100MHz) and the PCI device access (33MHz) in AT2
  - MEM captures both offchip events
- Recall that  $CPI_{off}$  is only needed to calculate  $T_{off}$
- We can calculate  $T_{off}$  directly as shown below
  - $T = T_{on} + T_{off} = CCNT / f_{cpu}$
  - $T_{off} = CCNT / f_{cpu} - T_{on}$

## Fine-grained DVFS Policy

- Scaling is performed at every OS quantum (~50msec)
- Optimal frequency for the next quantum is chosen based on the statistics of the previous quanta
- $T_{on}$  and  $T_{off}$  are calculated as :

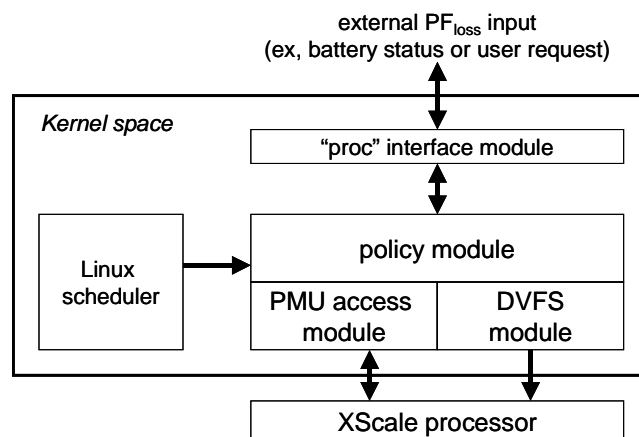
$$T_{on} = \frac{\sum_{i=1}^n CPI_{on}^i}{f_{cpu}} = \frac{n \cdot CPI_{on}}{f_{cpu}} \quad T_{off} = \frac{\sum_{j=1}^m CPI_{off}^j}{f_{mem}} = T - T_{on}$$

- Frequency for the next quantum (t+1),  $f^{t+1}$ , is calculated as:

$$f^{t+1} = \frac{f_{max}}{1 + PF_{loss} \cdot \left[ 1 + \beta^t \cdot \left( \frac{f_{max}}{f^t} \right) + \frac{S^t}{PF_{loss} \cdot T_{on}} \cdot \left( \frac{f_{max}}{f^t} \right) \right]}$$

## Implementation (I)

- Software architecture



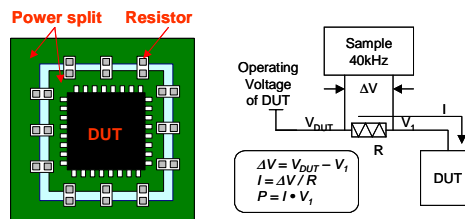
## Implementation (II)

- A voltage is mapped to each CPU frequency
- Voltage control circuitry is on-board
- Power measurement with DAQ (DATA Acquisition)

CPU Freq. vs. Volt. Relation

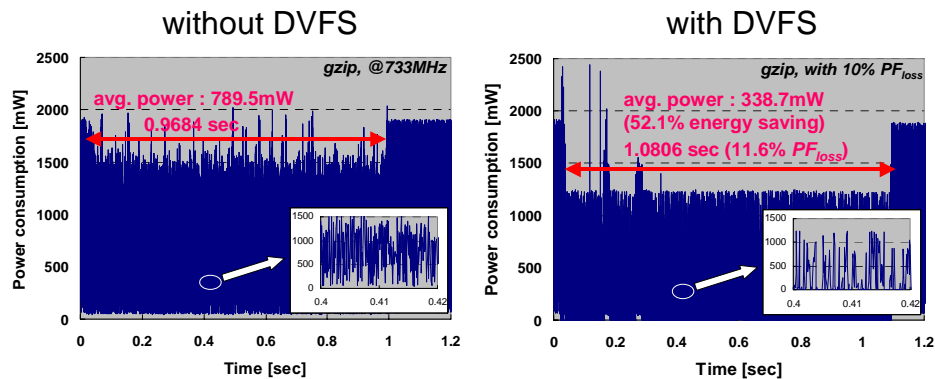
Frequency (MHz)	Voltage (V)
333	0.91
400	0.99
466	1.05
533	1.12
600	1.19
666	1.26
733	1.49

Data Acquisition system



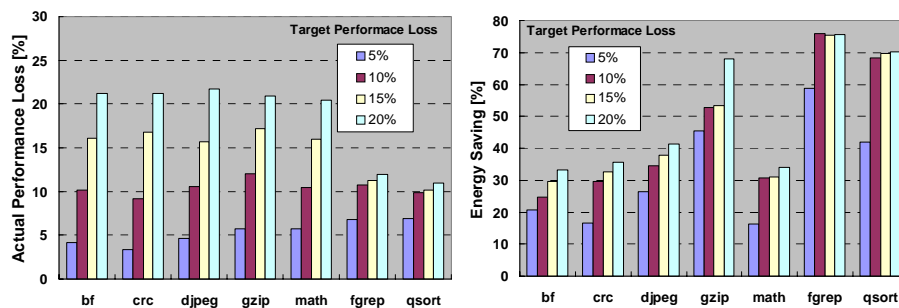
## Experimental Results (I)

- Power consumption vs. performance degradation



## Experimental Results (II)

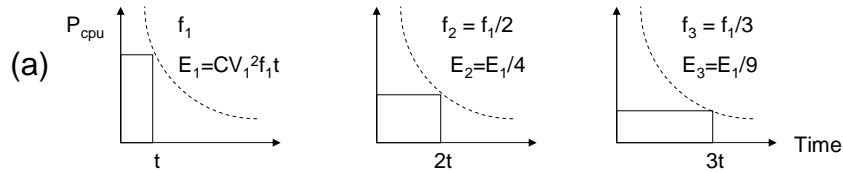
- Measured  $PF_{\text{loss}}$  with a variable performance loss target ranging from 5% to 20%



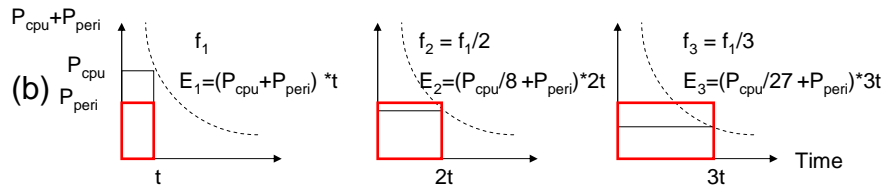
## DVFS Considering the Total System Energy

- “DVFS Considering Variable and Fixed Components of the System Power Dissipation,” Choi et al., 2004
- Most DVFS methods are concerned about the CPU energy reduction only
  - More precisely, dynamic portion of the CPU energy
- However, most systems comprise of many subsystems such as memory and peripheral devices
  - Battery lifetime also depends on power consumption in subsystems, which is not affected by CPU frequency changes
  - Lowering CPU frequency can cause shorter battery lifetime due to an increase in the standing and idle portions of the system energy consumption

## Total System Energy Variation in DVFS



**For CPU, lower frequency gives lower energy consumption**



**For a system, lower frequency does not give lower energy consumption**

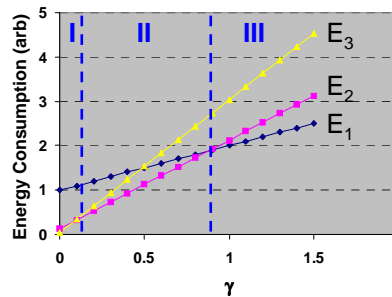
## CPU Freq. for Minimum System Energy

- Let  $\gamma$  denotes  $P_{peripheral}/P_{cpu}$  @  $f_{max}$ 
  - @  $f_{max}$  :  $E_1 = P_{cpu} \cdot (1 + \gamma) \cdot t$
  - @  $f_{max}/2$  :  $E_2 = P_{cpu} \cdot (1/4 + 2 \cdot \gamma) \cdot t$
  - @  $f_{max}/3$  :  $E_3 = P_{cpu} \cdot (1/9 + 3 \cdot \gamma) \cdot t$
- Based on the  $\gamma$  value, minimum system energy is obtained at different CPU freq.

Frequency setting to achieve minimum energy:

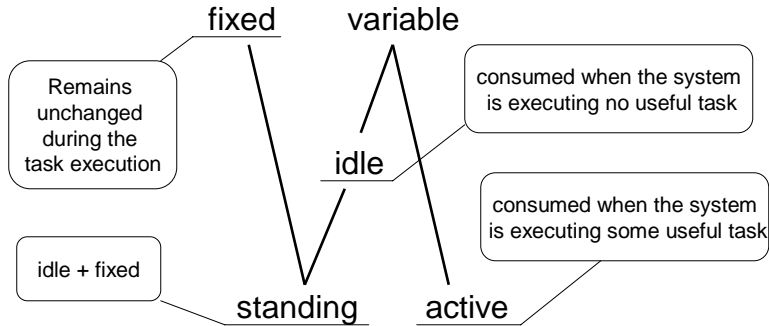
- Region (I) :  $f_{max}/3$
- Region (II) :  $f_{max}/2$
- Region (III) :  $f_{max}$

Previous DVFS techniques only consider the case when  $\gamma$  is 0



## System Power Breakdown

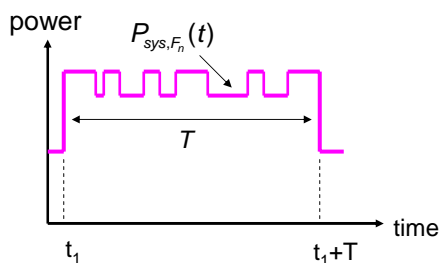
- A system consists of the CPU and other sub-modules such as memory
- System power consumption can be broken into fixed vs. variable or standing vs. active components



## Total System Power Dissipation

- Time-varying system power consumption

$$\begin{aligned}
 P_{sys,F_n}(t) &= P_{sys}^{fix} + P_{sys,F_n}^{idle} + P_{sys,F_n}^{act}(t) = P_{sys,F_n}^{std} + P_{sys,F_n}^{act}(t) \\
 &= P_{cpu,F_n}^{std} + P_{cpu,F_n}^{act}(t) + \sum_{j=1}^N P_{mod,i}^{std} + \sum_{i=1}^N P_{mod,i}^{act}(t) \\
 E_{sys,F_n} &= \int_{t_1}^{t_1+T} P_{sys,F_n}(t) \cdot dt
 \end{aligned}$$

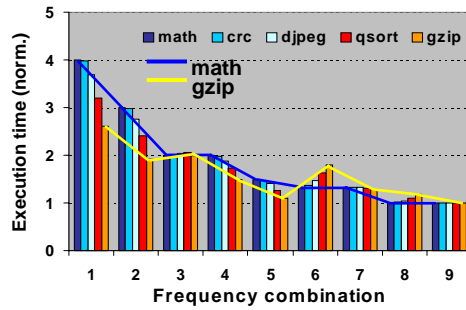


$F_n$	The nth frequency setting, $F_n(f_n^{cpu}, f_n^{int}, f_n^{ext})$
$P_{cpu,F_n}^{std}$	standing component in the CPU power at CPU frequency $F_n$
$P_{cpu,F_n}^{act}(t)$	active component in the CPU power at CPU frequency $F_n$
$P_{mod,i}^{std}$	standing component in the $i^{th}$ module power
$P_{mod,i}^{act}$	active component in the $i^{th}$ module power

## Frequency Settings in BitsyX

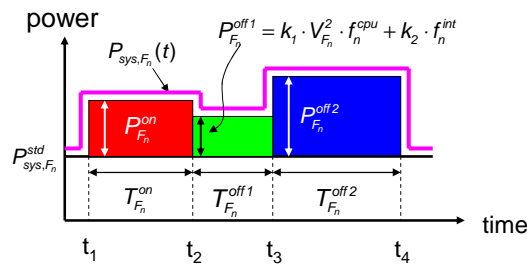
- PXA255 can operate from 100MHz to 400MHz, with a core supply voltage of 0.85V to 1.3V
- Internal bus connects the core and other functional blocks inside the CPU
- External bus is connected to SDRAM (64MB)
- Nine frequency combinations ( $f_{cpu}$ ,  $f_{int}$ ,  $f_{ext}$ )

Freq. Set	$f_{cpu}$ [MHz]	$V_{cpu}$ [V]	$f_{int}$ [MHz]	$f_{ext}$ [MHz]
F <sub>1</sub>	100	0.85	50	100
F <sub>2</sub>	133	0.85	66	133
F <sub>3</sub>	200	1.0	50	100
F <sub>4</sub>	200	1.0	100	100
F <sub>5</sub>	265	1.0	133	133
F <sub>6</sub>	300	1.1	50	100
F <sub>7</sub>	300	1.1	100	100
F <sub>8</sub>	400	1.3	100	100
F <sub>9</sub>	400	1.3	200	100



## System Power Modeling in BitsyX

- Using workload decomposition
  - $T = T^{on} + T^{off} = T^{on} + T^{off1} + T^{off2}$



- The system energy for a task at  $F_n$ ,  $E_{sys,F_n}$ , is given as

$$E_{sys,F_n} = P_{sys,F_n}^{std} \cdot T + P_{F_n}^{on} \cdot T_{F_n}^{on} + P_{F_n}^{off1} \cdot T_{F_n}^{off1} + P_{F_n}^{off2} \cdot T_{F_n}^{off2}$$

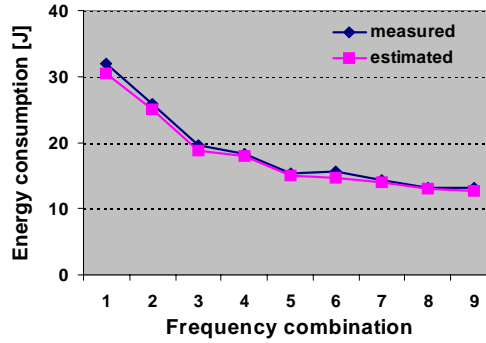
## Accuracy of the System Energy Model

- The estimated energy consumption for “djpeg”
  - The average error rate is less than 4%

### Extracted parameters

Freq. set	$P_{sys,F_n}^{std}$ (mW)	$P_{F_n}^{on}$ (mW)	$P_{int,F_n}^{off}$ (mW)	$P_{ext,F_n}^{off}$ (mW)
F <sub>1</sub>	1665	89	363	785
F <sub>2</sub>	1757	148	479	785*1.33
F <sub>3</sub>	1699	218	456	785
F <sub>4</sub>	1728	217	766	785
F <sub>5</sub>	1836	336	1018	785*1.33
F <sub>6</sub>	1732	344	575	785
F <sub>7</sub>	1778	378	885	785
F <sub>8</sub>	1869	673	1113	785
F <sub>9</sub>	1963	675	1733	785

$$E_{sys,F_n} = P_{sys,F_n}^{std} \cdot T + P_{F_n}^{on} \cdot T_{F_n}^{on} + P_{F_n}^{off1} \cdot T_{F_n}^{off1} + P_{F_n}^{off2} \cdot T_{F_n}^{off2}$$



$$P_{int,F_n}^{off} \sim k_1 \cdot V_{F_n}^2 \cdot f_n^{cpu} + k_2 \cdot f_n^{int}$$

$k_1 = 0.73 [nF], k_2 = 6.2 [V^2nF]$

## Determining the Optimal Frequency Setting

- Consider both *timing* and *minimal system energy* constraint
  - For a timing constraint, we use the performance loss factor (PF<sub>loss</sub>) which is defined as:

$$PF_{loss} = \frac{(T_{F_n} - T_{F_{max}})}{T_{F_{max}}}$$

$T_{F_n}$ : task execution time at  $F_n$   
 $T_{F_{max}}$ : task execution time at  $F_{max}$

- Pseudo code for optimal frequency selection

1.  $\Psi = \{F_{min}, \dots, F_{max}\}, \Gamma = \{\emptyset\}$ , and  $E_{min} = \infty$
2. for every frequency setting  $F_n$  in  $\Psi$
3. if ( $T_{F_n}^{i+1} \leq (1 + PF_{loss}) \cdot T_{F_{max}}^i$ )
4.  $\Gamma = \Gamma \cup F_n$ ;
5. for every frequency setting  $F_n$  in  $\Gamma$
6. calculate system energy using proposed model
7. if ( $E_{sys,F_n} \leq E_{min}$ )
8.  $E_{min} = E_{sys,F_n}$ ;  $F_{i+1}^{opt} = F_n$ ;

Timing constraint → (points to step 3)

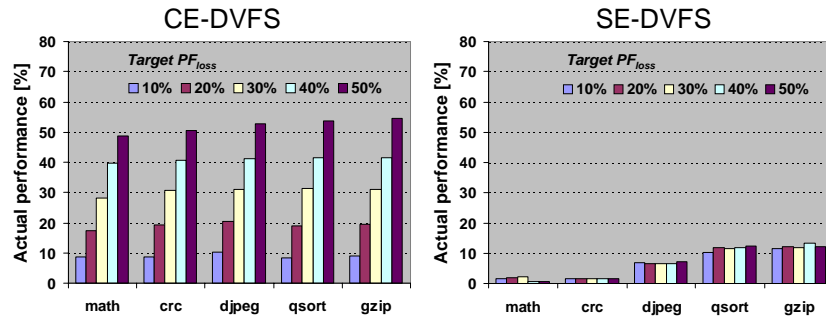
Energy optimization → (points to step 7)



## Experimental Results (I)

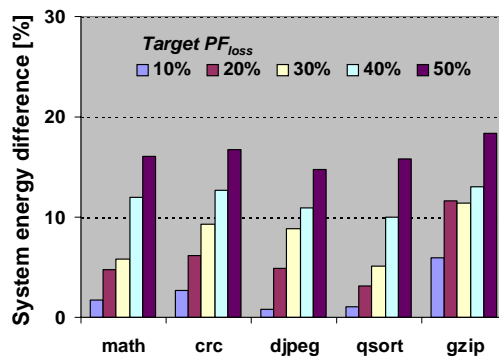
- Compared two DVFS methods:
  - SE-DVFS : the proposed DVFS (targets the total system energy saving)
  - CE-DVFS : conventional DVFS (targets CPU energy savings only)

### Actual Performance Loss Results



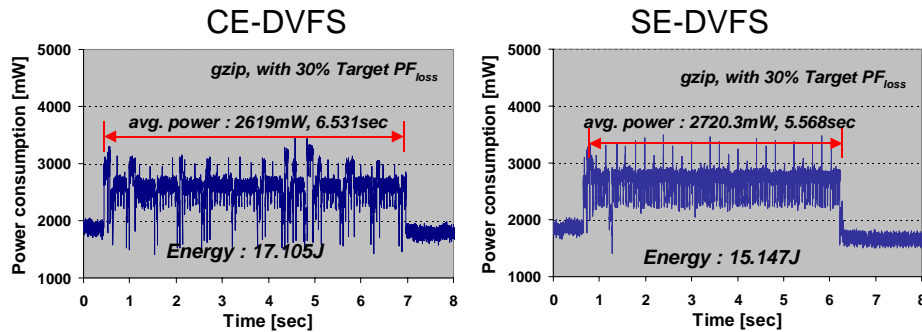
## Experimental Results (II)

- SE-DVFS vs. CE-DVFS
  - SE-DVFS results in 2% ~ 18% lower total system energy consumption compared to CE-DVFS



## Experimental Results (III)

- Actual power consumption of the two DVFS methods
- For “gzip” with 30% target  $PF_{loss}$ , SE-DVFS results in 11.4% lower total system energy than CE-DVFS



## Summary

- DVFS technique has been proven to be a highly effective technique for power minimization subject to a performance constraint
- DVFS techniques for both intra-task and inter-task optimizations have been proposed
- DVFS with workload decomposition is very successful for high-performance CPU's with a built-in PMU
- DVFS should consider not only the CPU power but also the total system power dissipation

## Challenges and Directions in Low Power Design

---

- Subthreshold leakage control in standby as well as sleep modes without adversely effecting the circuit performance and cost
- Statistical parameter variation and its impact on leakage current modeling and calculation
- Physical design tools that support multiple voltage islands, body bias control, and power gating
- Full-chip DVFS considering the total system energy consumption
- Combining DVFS and DPM at system level
- Combining Dual- $V_{th}$  and MTCOMS (power gating) at RT-level and below

## Challenges and Directions in Low Power Design

---

- Need to address all major consumers of power in microelectronic systems, including the on-chip drivers, memory, I/O drivers, TFT LCD, radio transceiver, etc.
- Power-aware bus encoding techniques (for off-chip buses as well as on-chip buses in SoC designs) which account for various noise sources e.g., the capacitive and inductive crosstalk, and power plane variations
- Taking the battery characteristics (rate-capacity curve, energy recovery effect, and battery aging) into consideration when developing energy-aware design solutions
- Power delivery issues