Power Management Technologies

Lecturer:
Ing. Patrick Bellasi
Politecnico di Milano - DEI
bellasi@elet.polimi.it

Outline

- Why Low Power Design?
- VLSI IC's Technology Trends
- Where Power Goes?
  - Power Leakage Sources
  - Power, Delay and Reliability Models
- Power Optimization
  - Hardware solutions

    Power islands, Clock Gating, DVFS
  - Software/System solutions

    APM/ACPI, OS level PM

Why Low Power Design

Why should we interest on *Power Management*?

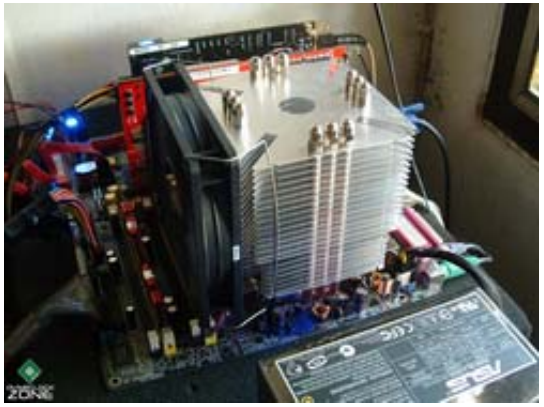
AP / Paul Sakuma

It make us happy!?

The need of more and more complex portable and wireless applications requires better effort on designing low power system solutions

Low Power Design Concerns

Battery lifetime



Cooling and energy costs

System reliability

Environmental concerns

Motivation

Power has become a major consideration in VLSI design

- power consumption will increase significantly in next few years due to increase of complexity of new device functionalities
- high power consumption increases the packaging and cooling cost and decreases the system reliability
- the battery technology cannot keep pace with the VLSI technology
- environmental concerns require both a better usage of limited energy resource and the reduction of unnecessary energy wastage

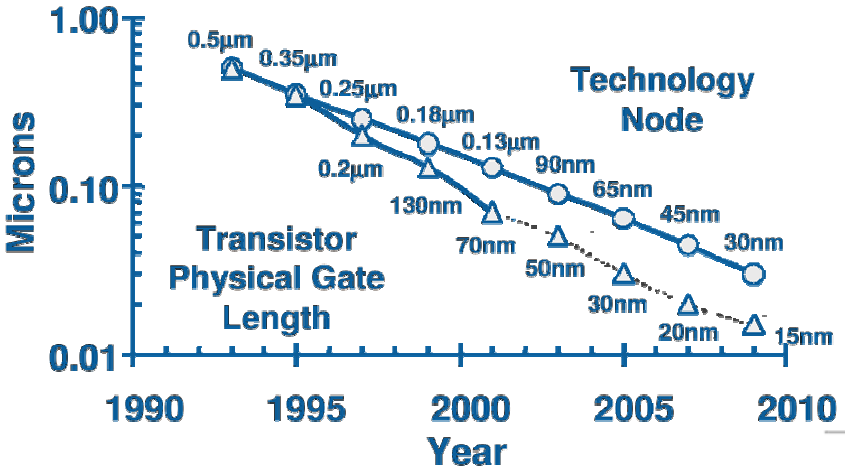
Outline

- Why Low Power Design?
- VLSI IC's Technology Trends
- Where Power Goes?
  - Power Leakage Sources
  - Power, Delay and Reilability Models
- Power Optimization
  - Hardware solutions

    Power islands, Clock Gating, DVFS
  - Software/System solutions

    APM/ACPI, OS level PM

Technology scaling



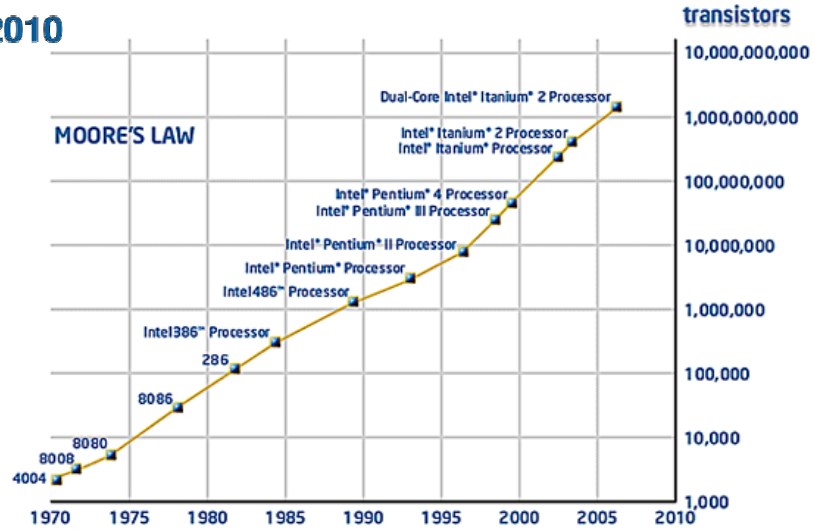
Physical Gate Length decrease

New IC are manufactured using sub-micro production technology process

Transistors number increase

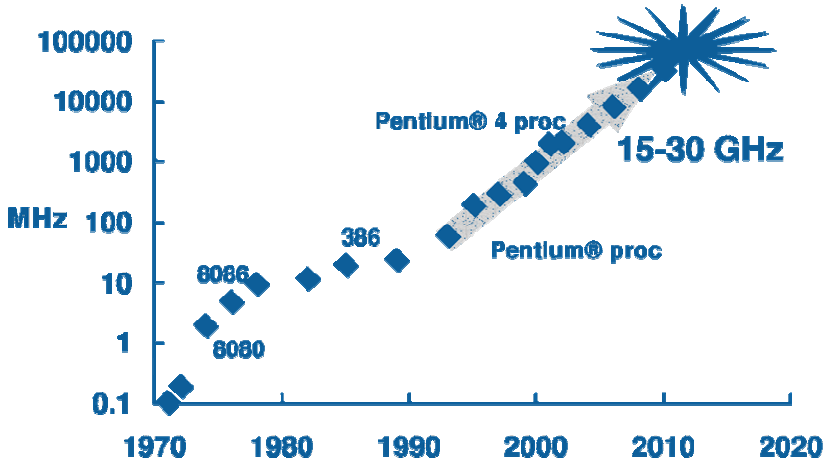
more and more transistors are integrated within single chips

higher device density



Moore's Law Means More Performance. Processing power, measured in millions of instructions per second (MIPS), has steadily risen because of increased transistor counts.

System requirements

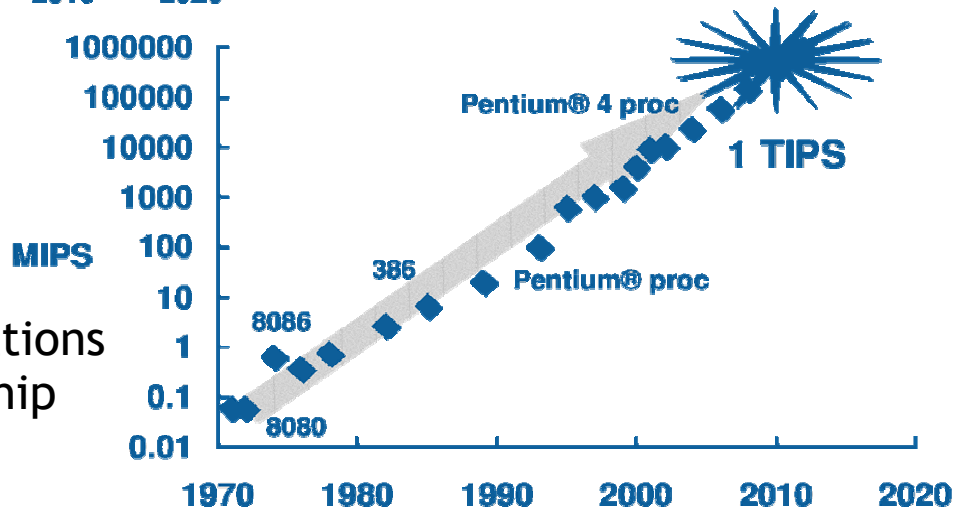


Frequency increase

modern embedded systems could operate at hundreds MHz

Performances increase

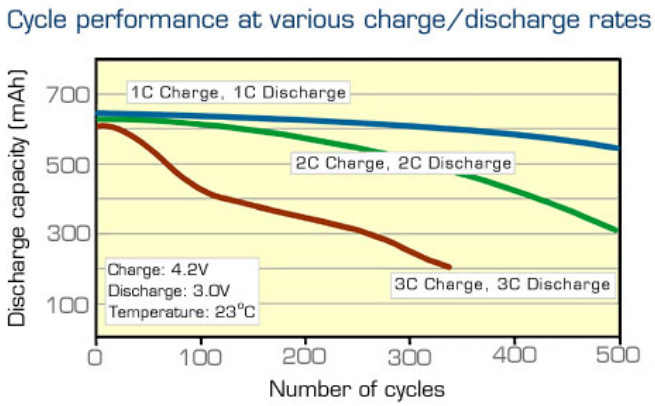
more and more complex operations are delivered by single chip embedded systems



Battery Technologies

The battery discharge rate is *super-linearly* related to the average power consumption in the VLSI circuits

- discharging rate and operational temperature have a profound effect on capacity and lifetime
- high discharge voltages, excessive discharge rate and extreme load conditions will have a negative effect and shorten the battery life



http://www.batteryuniversity.com/

Outline

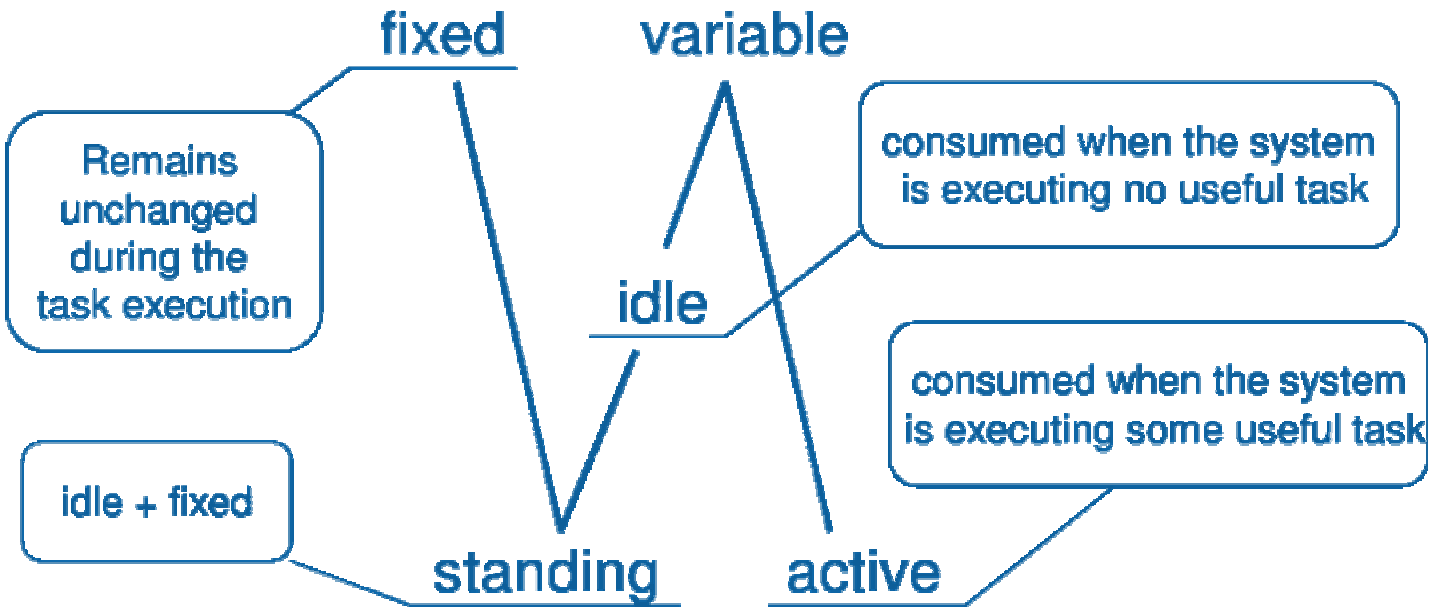
- Why Low Power Design?
- VLSI IC's Trends
- Where Power Goes?
  - Power Leakage Sources
  - Power, Delay and Reilability Models
- Power Optimization
  - Hardware solutions

    Power islands, Clock Gating, DVFS
  - Software/System solutions

    APM/ACPI, OS level PM

System Power Breakdown

System power consumption can be broken into the following components



Basic Principles of Low Power Design

$$P = 0.5V_{DD}^2 f_{clock} C_L E_{sw} + t_{sc} V_{DD} I_{peak} f_{0\to1} + V_{DD} I_l$$

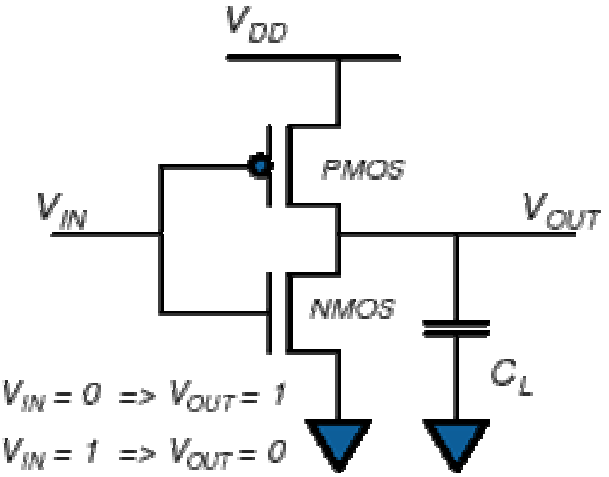
Switching (or dynamic) power

- E_{sw} represents the probability that the output node makes a transition at each clock cycle
- models the fact that, in general, switching does not occur at the clock frequency
- it is called the switching activity of the gate

Short-circuit power

Leakage (or stand-by) power

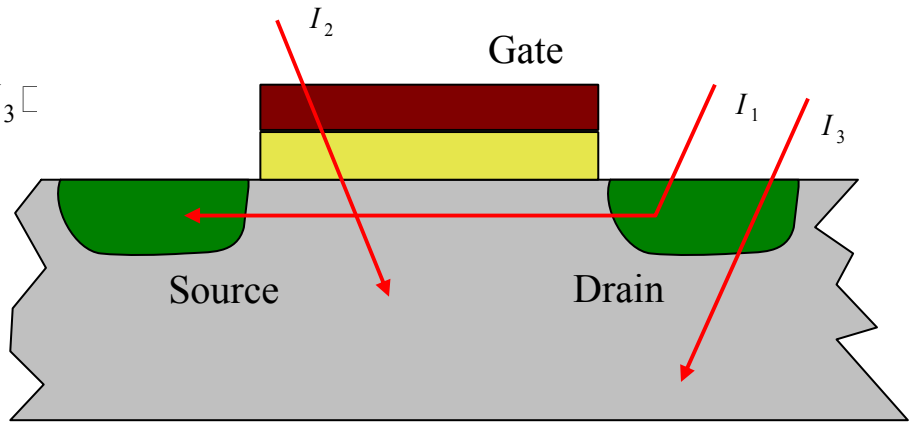
- in older technologies (0.25um and above) was marginal w.r.t. switching power
- in deep sub-micron processes becomes critical accounting for about 35-50% of power budget at 90nm

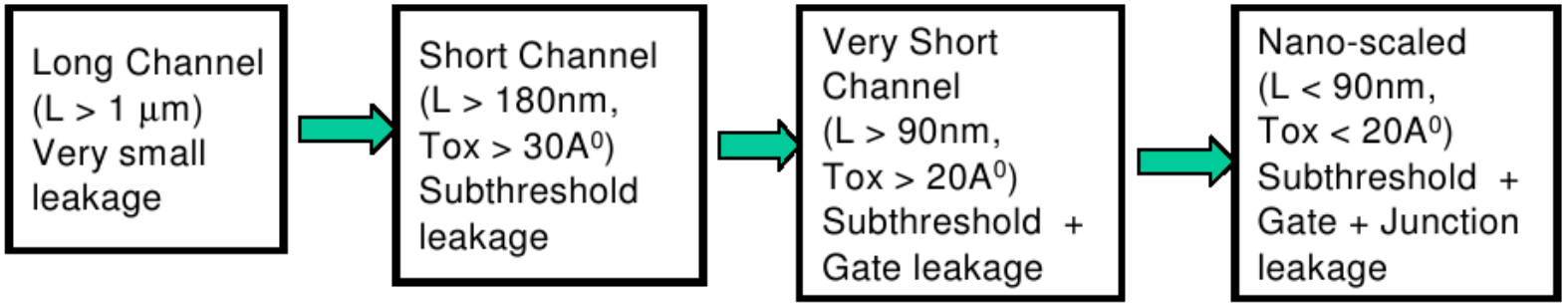


Leakage Components in bulk CMOS

Leakage Components

- *Subthreshold Leakage* I_1
- *Gate Leakage* I_2
- ***Junction Leakage*** I_3
- Gate Induced Drain Leakage
- Impact Ionization current

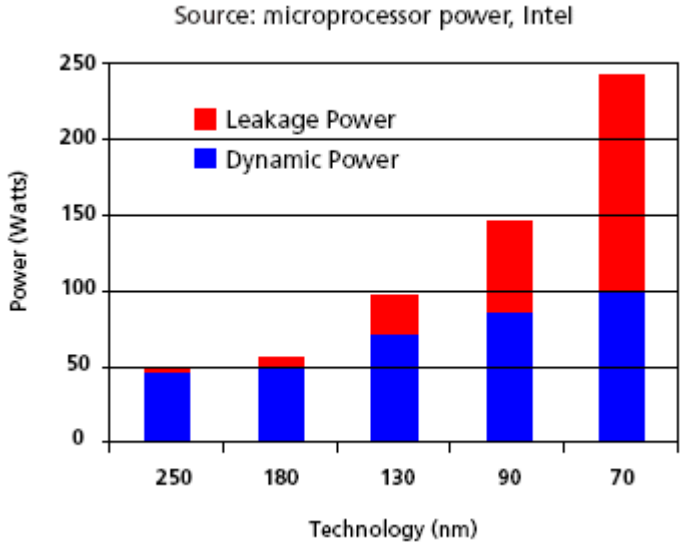




Power Dissipation Contributions

With the migration to Deep Sub-Micron (DSM) process technologies, the static power (leakage) has become the major contributor to the design's overall power consumption

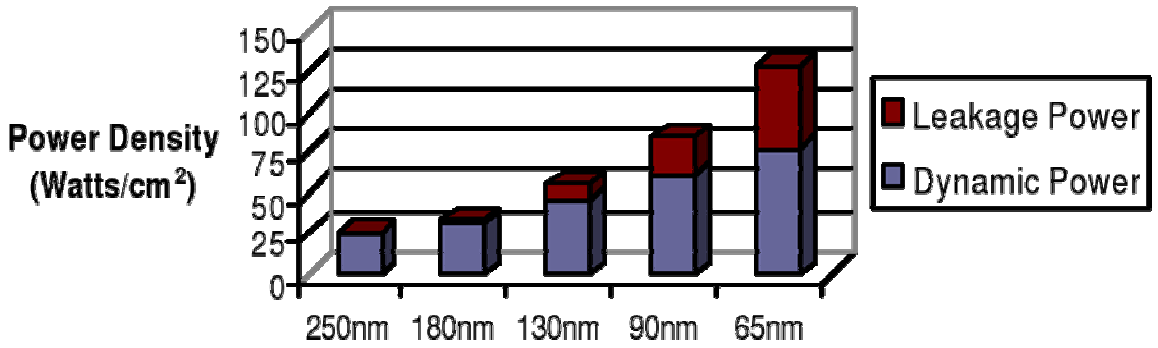
Illustrate the significant increase in the ratio of the leakage to the total power



Power Dissipation Contributions (cont)

Increasing contribution of leakage power

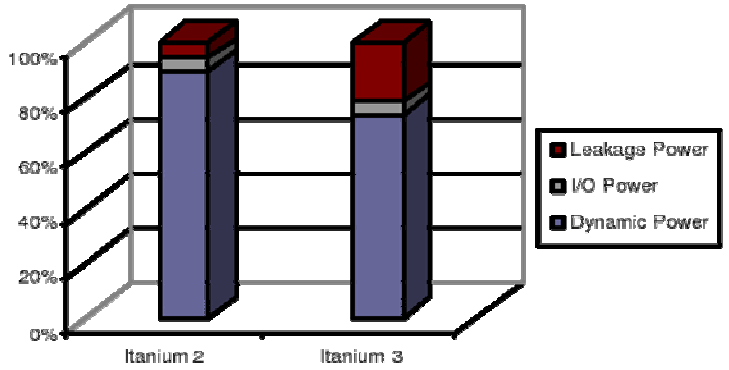
Example: ASICs [source: STMicroelectronics].



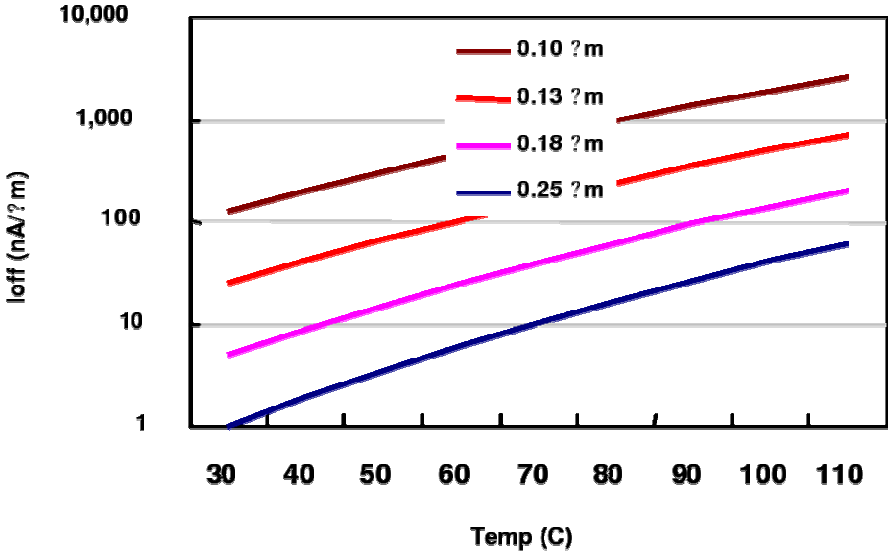
Example: Microprocessors [source: Intel].

Itanium 2:
180nm, 1.5V, 1.0GHz,
221MTx (core+cache)

Itanium 3:
130nm, 1.3V, 1.5GHz,
410MTx (core+cache)

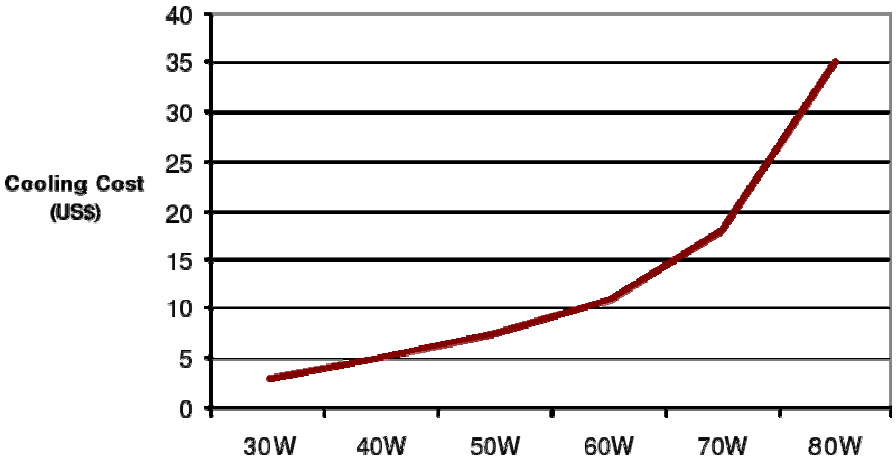


Temperature Concerns



Temperature effects on sub-threshold leakage

Trends in heat removal cost



Scaling effects on power consumption

Technology scaling

- higher device densities
  smaller capacitance per gate to be charged and discharged, but many more gates per chip, higher switched capacitance
- higher clock frequencies
  - *Higer switching power*
- lower supply voltages
  lower switching power, lower speed, lower threshold voltages
  - *Higer leakage power*

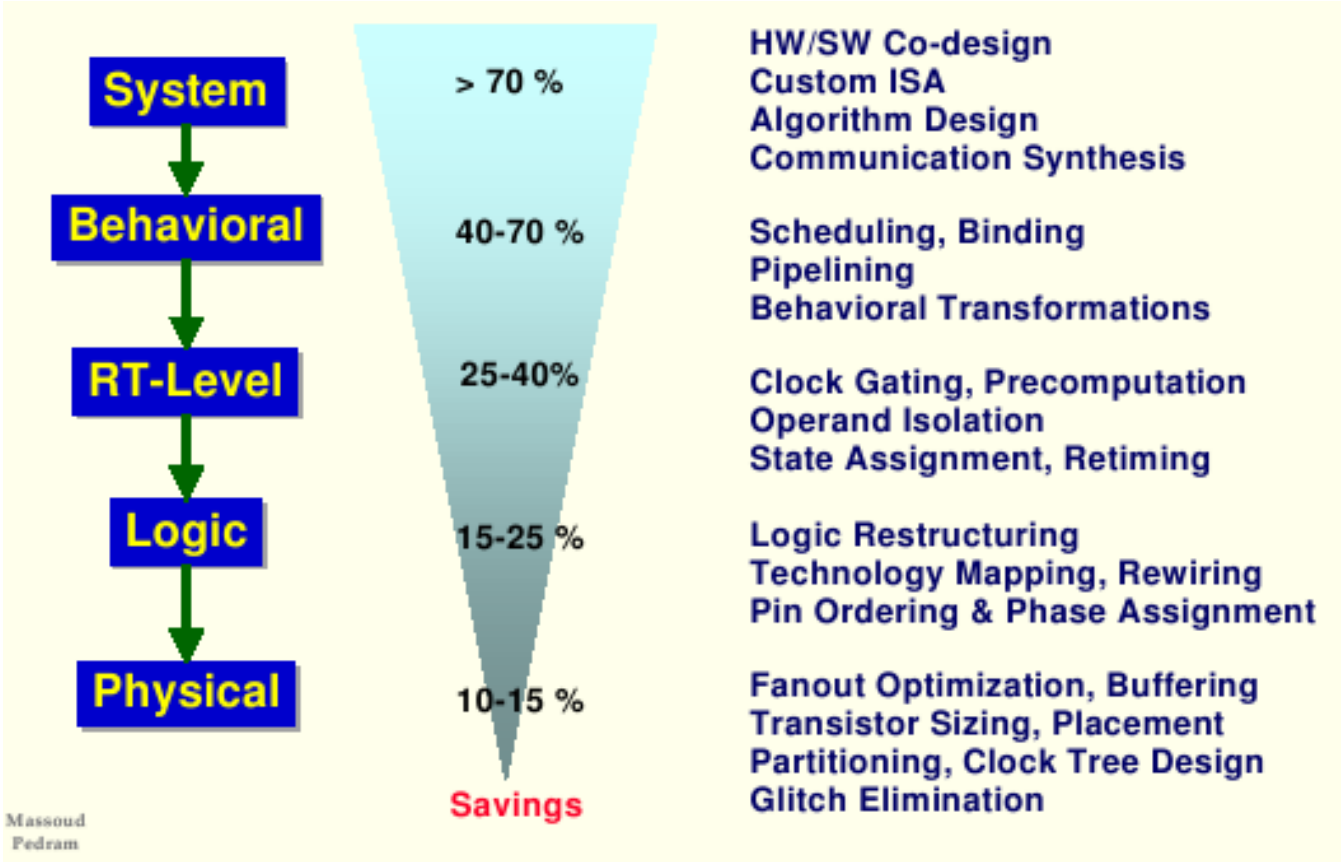
Power density increases as technology scales

- Why Low Power Design?
- VLSI IC's Trends
- Where Power Goes?
  - Power Leakage Sources
  - Power, Delay and Reilability Models
- Power Optimization
  - Hardware solutions

    Power islands, Clock Gating, DVFS
  - Software/System solutions

    APM/ACPI, OS level PM

Power Saving Opportunities



Dynamic Power Optimization

- Circuit 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
  - clock gating
  - guarded evaluation
  - bus encoding
- Design tools
  - power-conscious synthesis and design tools
  - power-aware compiler and architecture design
- Power control and management techniques
  - dynamic voltage scaling based on workload

- Why Low Power Design?
- VLSI IC's Trends
- Where Power Goes?
  - Power Leakage Sources
  - Power, Delay and Reilability Models
- Power Optimization
  - Hardware solutions

    Power islands, Clock Gating, DVFS
  - Software/System solutions

    APM/ACPI, OS level PM

Basic Principles of Low Power Design

$$P = 0.5V_{DD}^2 f_{clock} C_L E_{sw} + t_{sc} V_{DD} I_{peak} f_{0\to1} + V_{DD} I_l$$

Switching (or dynamic) power

- Reduce the supply voltage
  - Quadratic effect -> dramatic savings
  - Negative effect on performance
- Reduce clock frequency
- Reduce switched capacitance
- Reduce wasteful switching

Short-circuit power

Leakage (or stand-by) power

Leakage Reduction Techniques

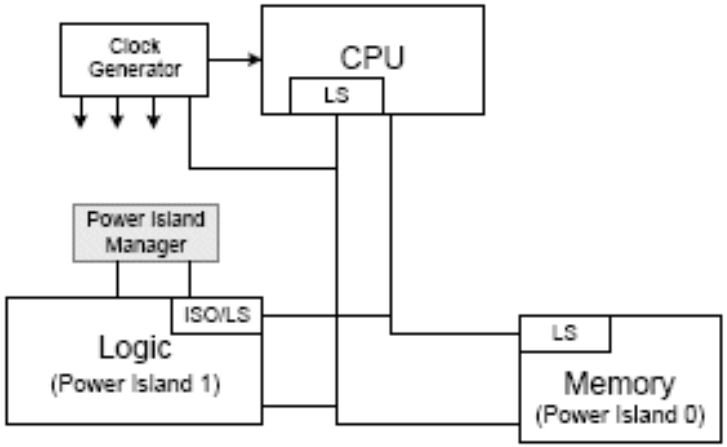
- Lowering Vdd (voltage islands, dynamic voltage scaling)
- SOI technology
- Cooling and/or refrigeration
- Gate-level dual-Vth design
- Mixed-Vth (MVT) CMOS design
- Transistor sizing (shorter W, longer L)
- Transistor stacking
- Body bias control (static and/or adaptive)
- Input vector control during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

Power Islands

minimizes the leakage in the circuit by partitioning it into islands

each island

- is a cluster of logic whose power can be controlled independent from the rest of the circuit
- can be completely *powered down* when all the logic contained within it is idling



Power Islands Considerations

Pro

- elimination of leakage in inactive components during the power down cycles of the islands
- high-level design/synthesis method

Cons

- suspend/resume device states
- needs for optimization allocation algorithms

DOI: http://doi.ieeecomputersociety.org/10.1109/ISQED.2006.103

Clock Gating

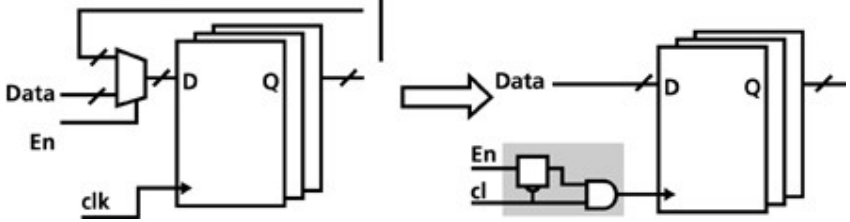
minimizes the leakage in the circuit by cutting-off clock tree where not needed

each gate

- drive a cluster of logic whose clock can be controlled independent from the rest of the circuit
- can be completely *suspended* when all the logic contained within it is idling

```
module test (En, Data, clk, out);
input En, clk;
input [7:0] Data;
output [7:0] out;

always @ (posedge clk)
        if (En) out <= Data;
endmodule
```



Clock Gating Considerations

Pro

- reduce leakage in inactive components during the suspended cycles of the gate
- decrease the overall clock tree capacity load
- device state is preserved
- high-level design/synthesis method

Cons

- needs for optimization allocation algorithms

DPM vs. DVFS

Dynamic power management

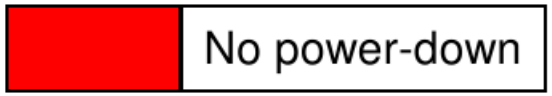
- Changes the power state of system components to lower the energy consumption depending on the performance constraints
- Components transition between active and low-power states
- Power manager observes system and responds at run-time

Dynamic voltage and frequency scaling

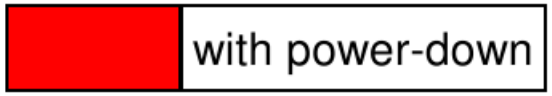
- Adjust performance and energy consumption levels while the device is active
- Key is to meet users performance needs while saving energy
- Reduce processor frequency and voltage to obtain quadratic energy savings

DPM vs. DVFS (II)

- With nothing



- DPM



- DVFS

# of cycle	Voltage	Energy	Saving
12.5×10^6	5	125mJ	-
5×10^6	5	50mJ	60%
5×10^6	2	8mJ	93.6%

$$E \propto V^2$$
$$f \propto V$$
$$E \propto V^3$$

Dynamic Voltage and Frequency Scaling

DVFS's 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:

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

α switching factor
C_{eff} effectivecapacitance
V operating voltage
f operating frequency

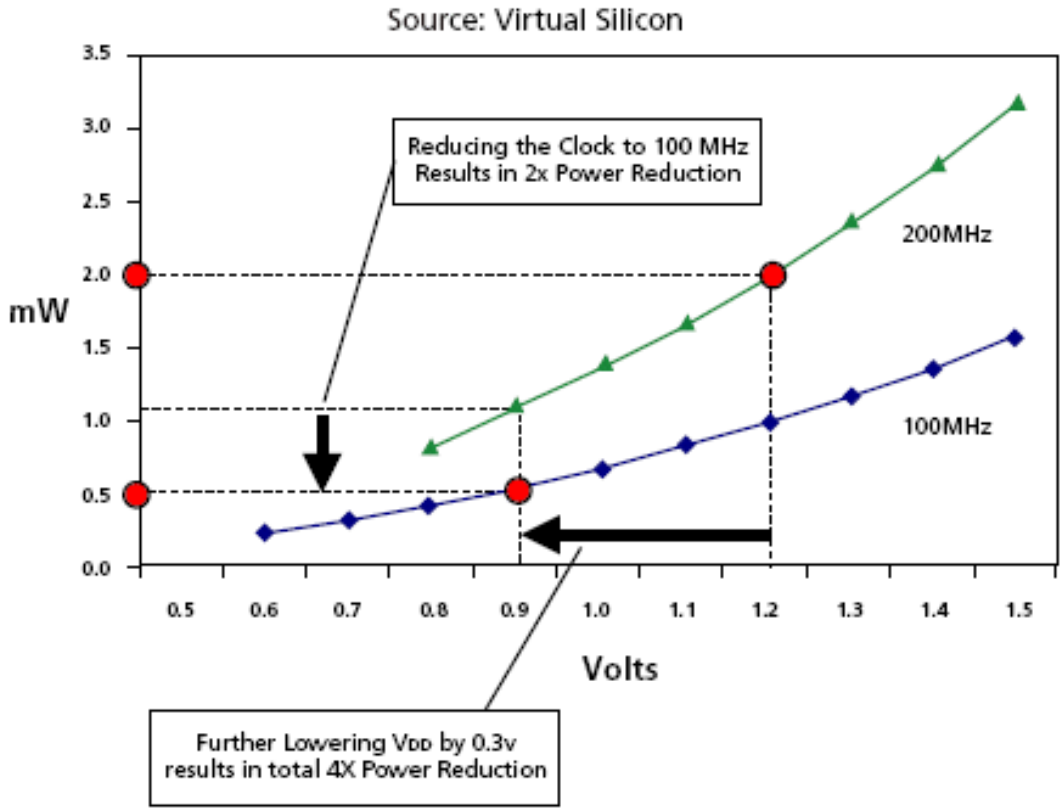
Energy required to run a task during T is:

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

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

DVFS effects

Effects on Power of Varying Voltage and Frequency



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

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

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}

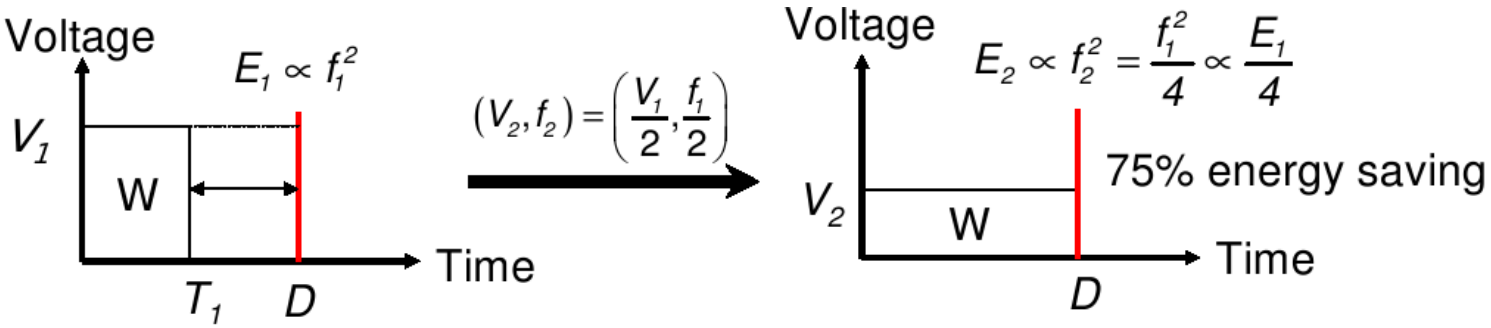
$$T_{task} = \frac{W_{task}}{f_{cpu}}$$

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

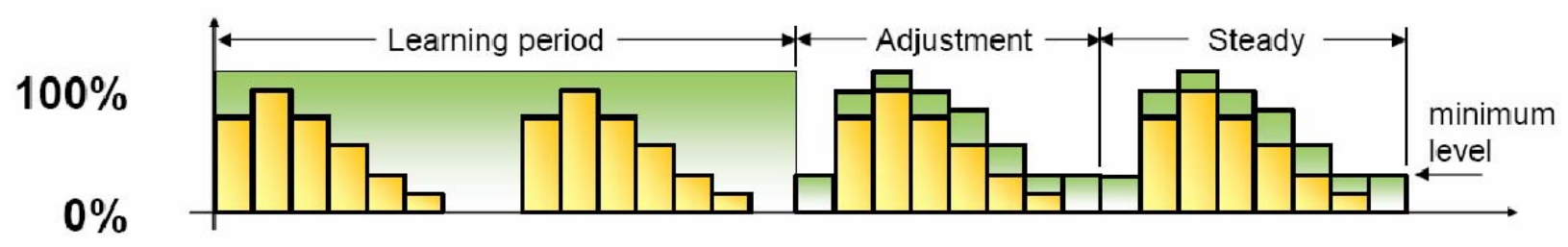


DVFS is an effective way of reducing the CPU energy consumption by providing “just-enough” computation power

DVFS Approaches

Reactive – Hardware Solution

Predictive – Hardware & Software Solution



DVFS issues

Requirements

- HW support for voltage and frequency setting
- accurate workload prediction (or fixed deadline knowledge)
- an error compensation method for workload prediction

Must not be concerned only about CPU energy

- most computing systems, however, comprise of many subsystems such as memory and peripheral devices
- lowering CPU frequency can cause shorter battery lifetime due to increased energy consumption in the subsystems

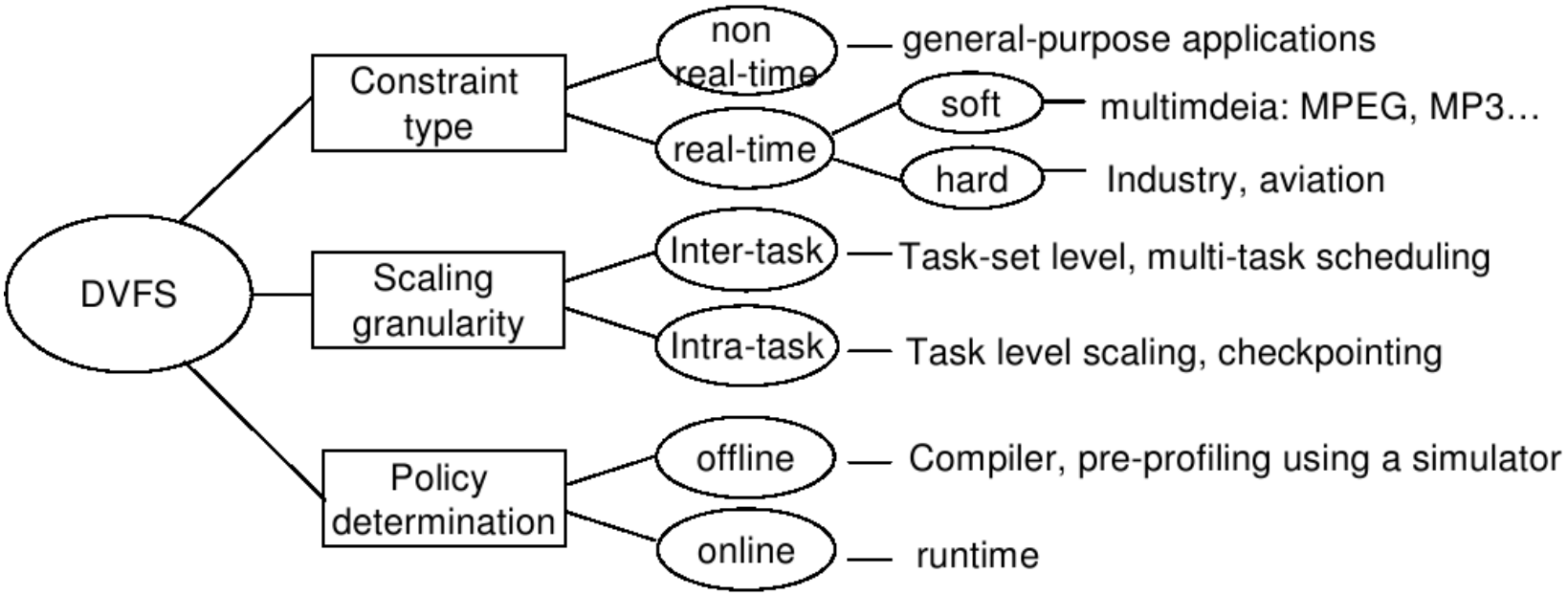
The Power vs Performances Tread-off

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

Inter-task DVFS

- Scaling occurs at the start of a task
  - It is unchanged until the task is completed
- Use worst-case slack time (Deadline_task >= WCET_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

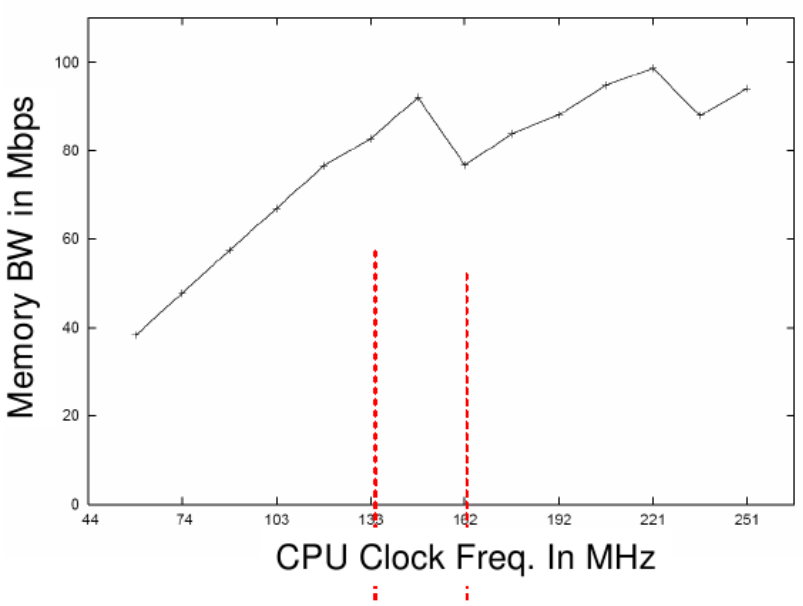
"Dynamic voltage scaling on a low-power microprocessor", Pouwelse et al., 2001

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

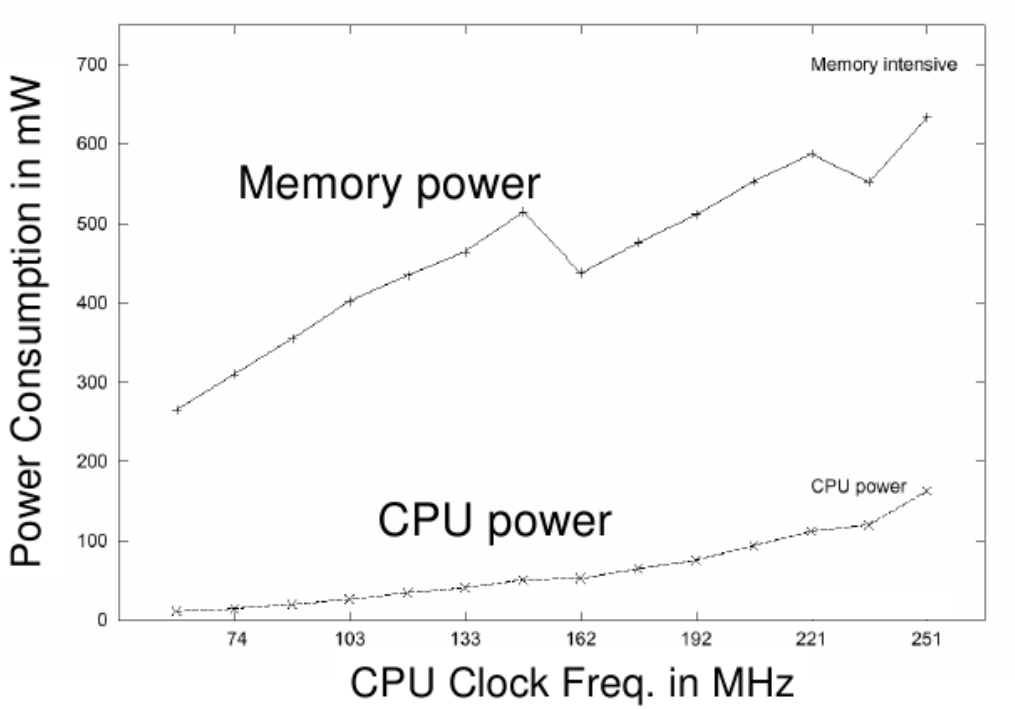
faster at 133MHz than at 162MHz in case of memory bound applications



Performance vs. CPU Speed -2

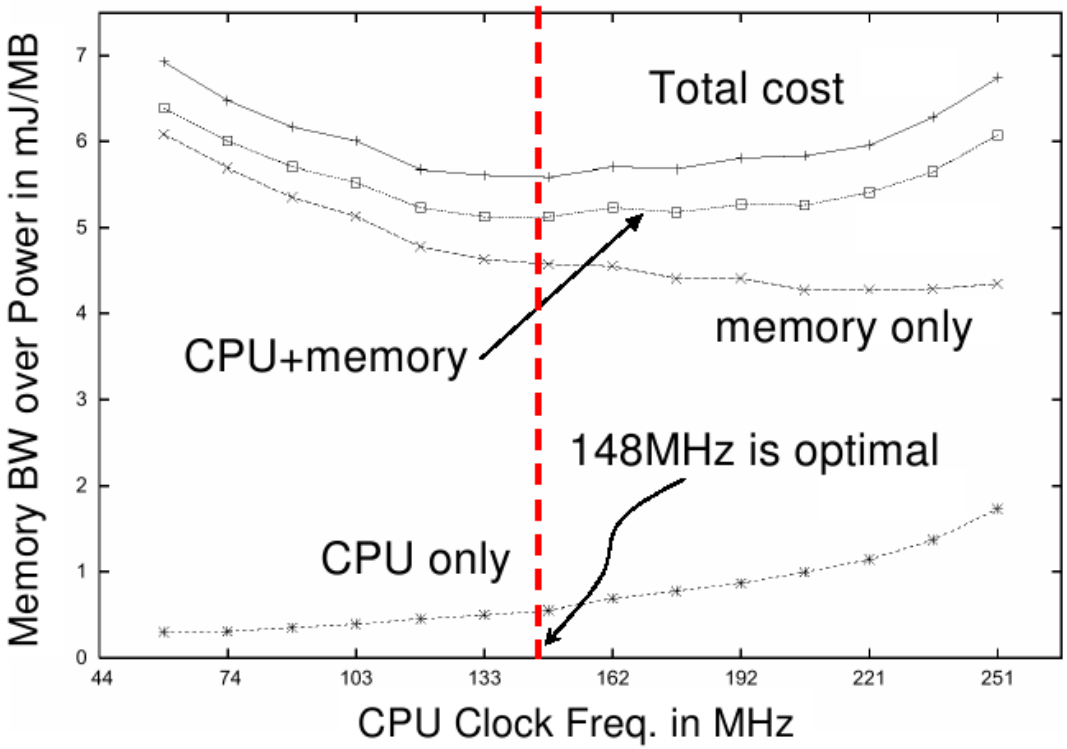
Non-linearity of memory bandwidth also affects the power consumption

Memory chip typically operates at 3.3V



Performance vs. CPU Speed -3

Energy breakdown for memory read



lmbench tool kit

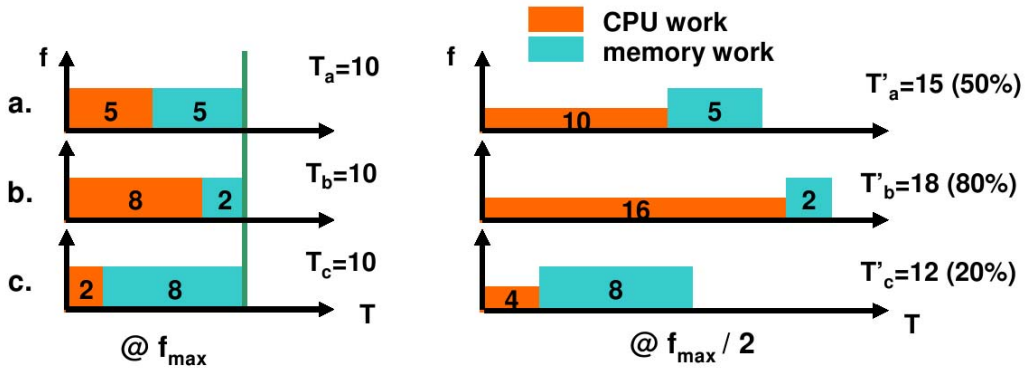
DVFS with Workload Decomposition

"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 for Workload Decomposition

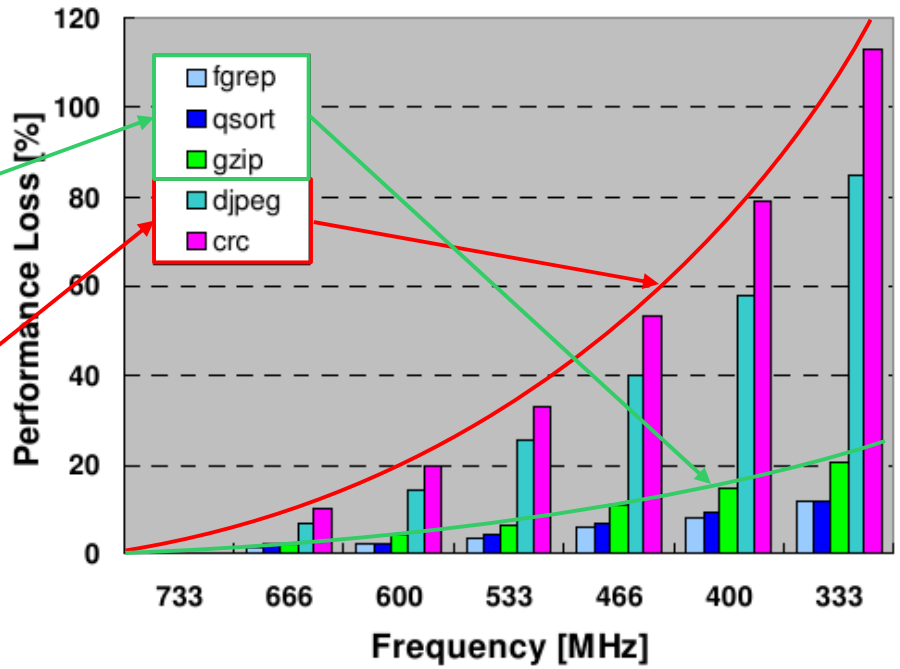
CPU-bound vs. memory-bound applications

- different execution time variation according to the CPU frequency

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

Memory bound

CPU bound



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)

$$T = T_{on} + T_{off}$$

Ton: varies with the CPU frequency

- Cache hit
- Stall due to data dependency
- TLB hit...

Toff: 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

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

m number of offchip events
CPI_{off} memory clocks per offchip event
f_{cpu} memory clock frequency – fixed –

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

n number of onchip instructions
CPI_{on} CPU clocks per instruction
f_{cpu} CPU clock frequency – varied –

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

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

Performance Monitoring Unit (PMU)

Present on some processor chip can report different dynamic events during execution of a program

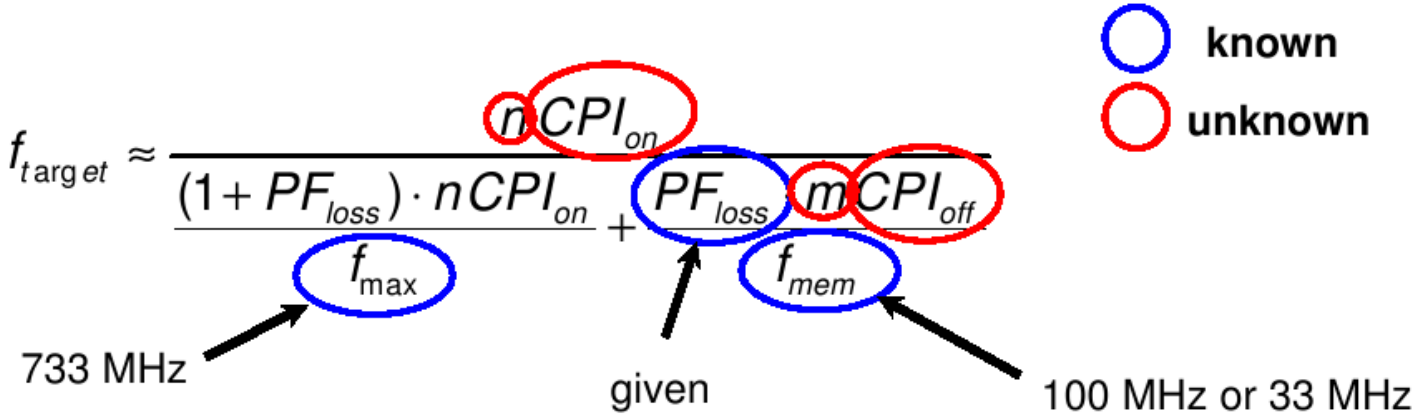
- on XScale 80200: up to 20 different dynamic events
- only two events can be monitored and reported at any given time
  - Cache hit/miss counts
  - TLB hit/miss counts
  - No. of external memory accesses
  - Total no. of instructions being executed
  - Branch misprediction counts

For DVFS, could be used to generate statistics for:

- Total no. of instructions being executed (INSTR)
- No. of external memory accesses (MEM)
- no. of clock cycles from the beginning of the program execution (CCNT)

How the PMU Data Is Used in DVFS

Target frequency for a given PF_{loss}



unknown parameters must be calculated from CCNT and the PMU's reported values (INSTR & MEM)

- n (no. of executed instructions) ← INSTR
- m (no. of offchip events) ← MEM
- CPI_{on} ← ?
- CPI_{off} ← ?

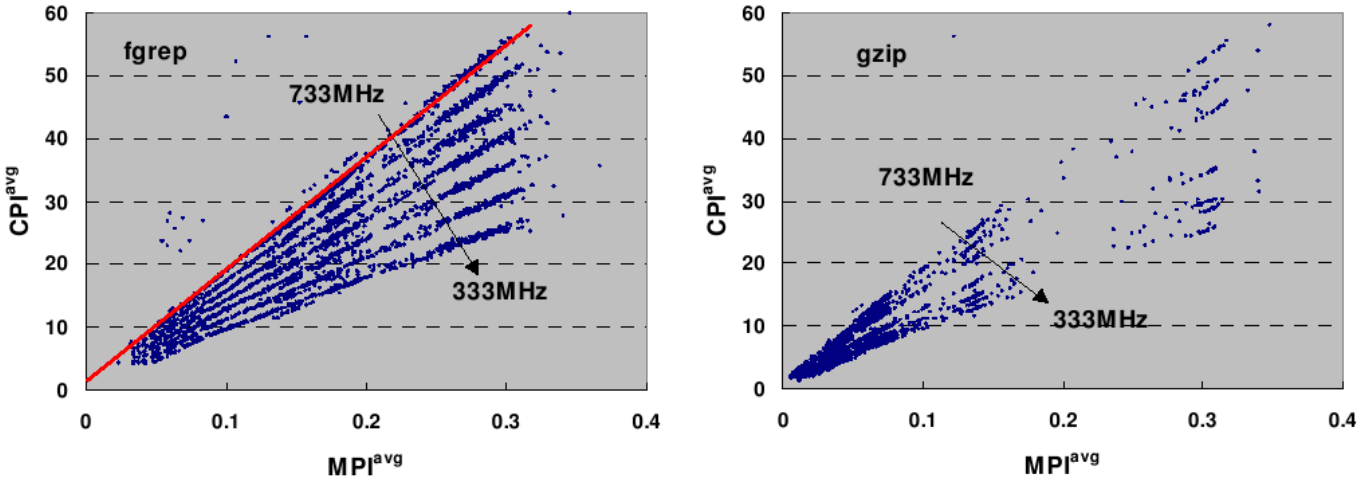
Plot of CPI vs. MPI

PMU is read at every OS quantum (~50msec)

We define MPI as the ratio of memory access count to the total instruction count:

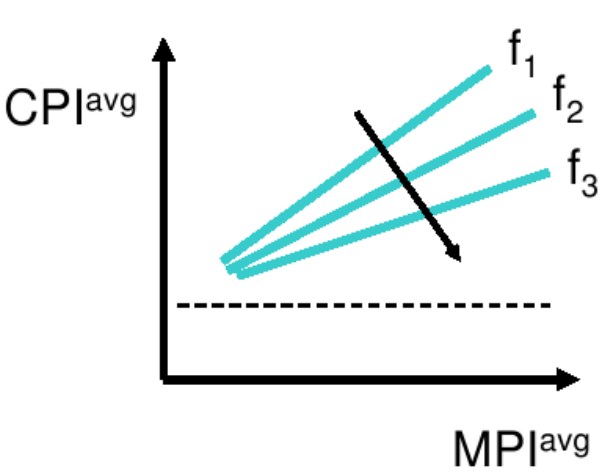
$$CPI^{avg} = CCNT / INSTR \text{ , during a quantum}$$
$$MPI^{avg} = MEM / INSTR \text{ , during a quantum}$$

A plot of CPIavg vs. MPIavg with changing frequency



Regression Equation Modeling

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



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

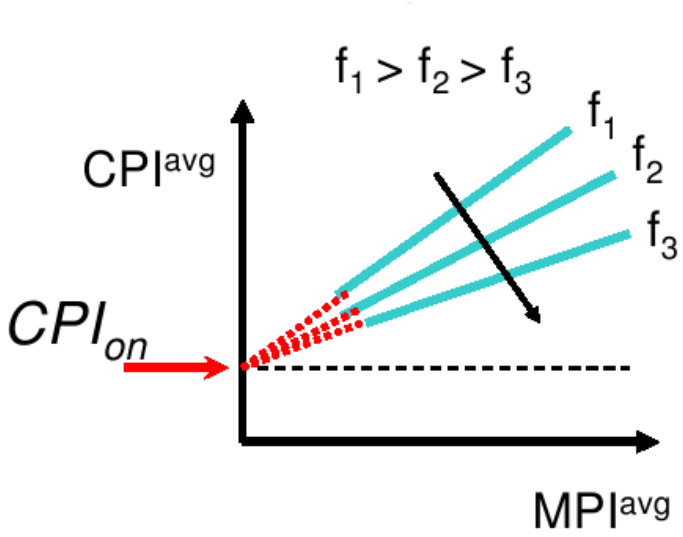
$$b = \frac{N \cdot (\sum_{i=t}^{t-N+1} x_i \cdot y_i) - (\sum_{i=t}^{t-N+1} x_i) \cdot (\sum_{i=t}^{t-N+1} y_i)}{N \cdot (\sum_{i=t}^{t-N+1} x_i^2) - (\sum_{i=1}^{t-N+1} x_i)^2},$$

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

N : No. of regression points, e.g., 25
x_i : MPI^{avg} for the i^{th} point
y_i : CPI^{avg} for the i^{th} point

Calculating on-chip CPI

Notice that CPI_on denotes the CPI value without the off-chip access; so it is equal to the y intercept of the CPI vs. MPI plot



$$CPI_{on} = c$$

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

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

Calculating off-chip CPI

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 off-chip events

Recall that CPI_{off} is only needed to calculate T_{off}

We can calculate T_{off} directly as:

$$T = T_{on} \square T_{off} = CCNT / f_{cpu}$$
$$T_{off} = CCNT / f_{cpu} - T_{on}$$

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:

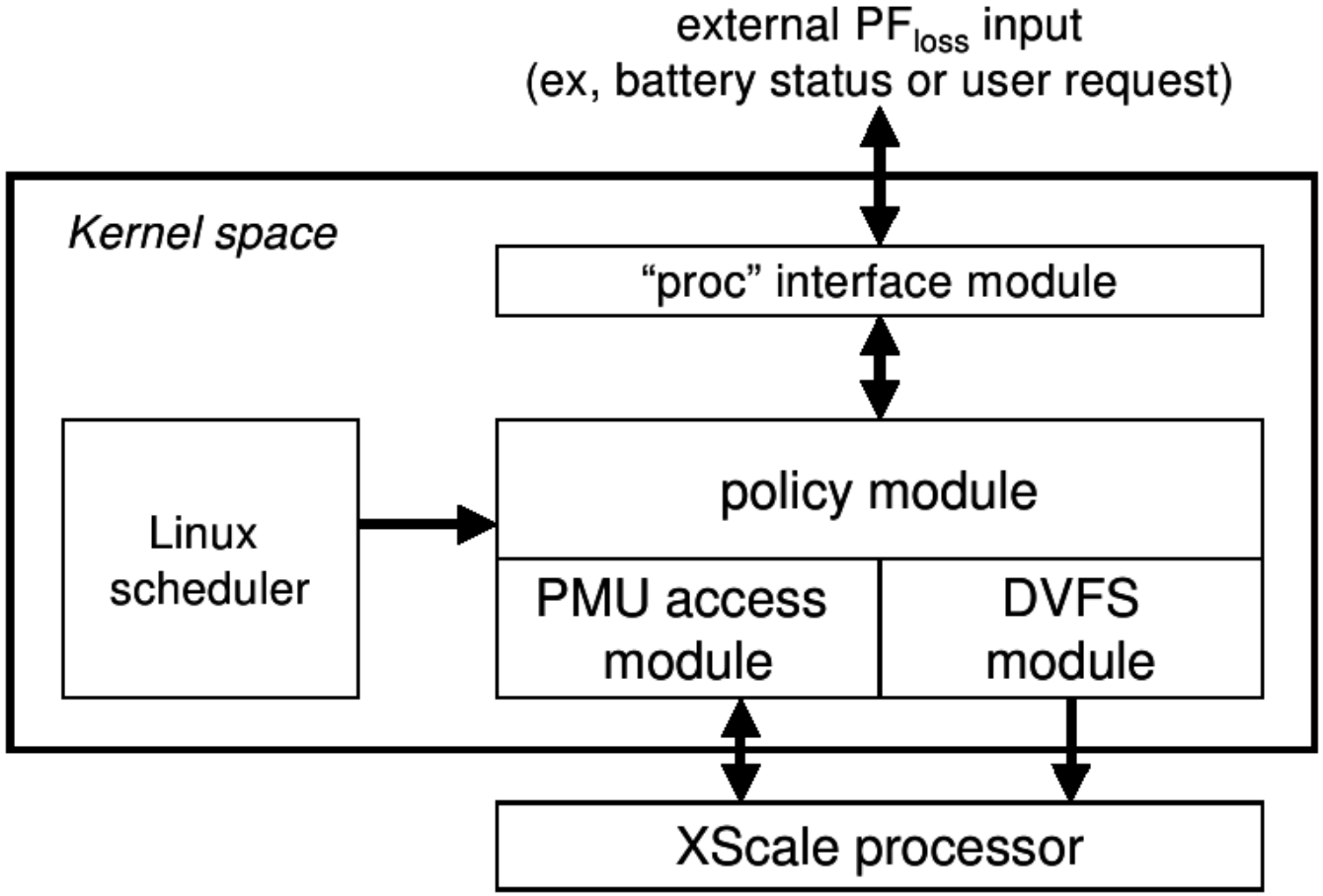
$$T_{on} = \frac{\sum_{i=1}^{n} CPI_{on}^{i}}{f_{cpu}} = \frac{n \cdot CPI_{on}}{f_{cpu}} \qquad T_{off} = \frac{\sum_{j=1}^{m} CPI_{off}^{j}}{f_{mem}} = T - T_{on}$$

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

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

Implementation (I)

Software architecture



Implementation (II)

A voltage is mapped to each CPU frequency

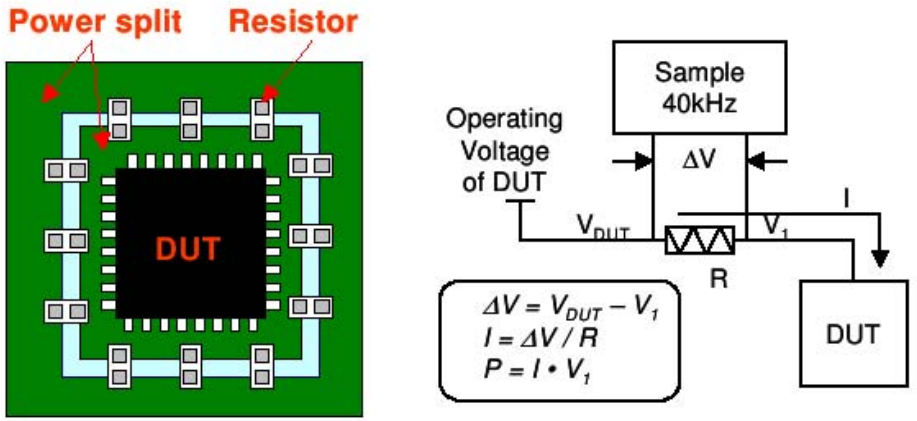
Voltage control circuitry is on-board

Power measurement with DAQ (DATA Acquisition)

CPU Freq. vs. Volt. Relation

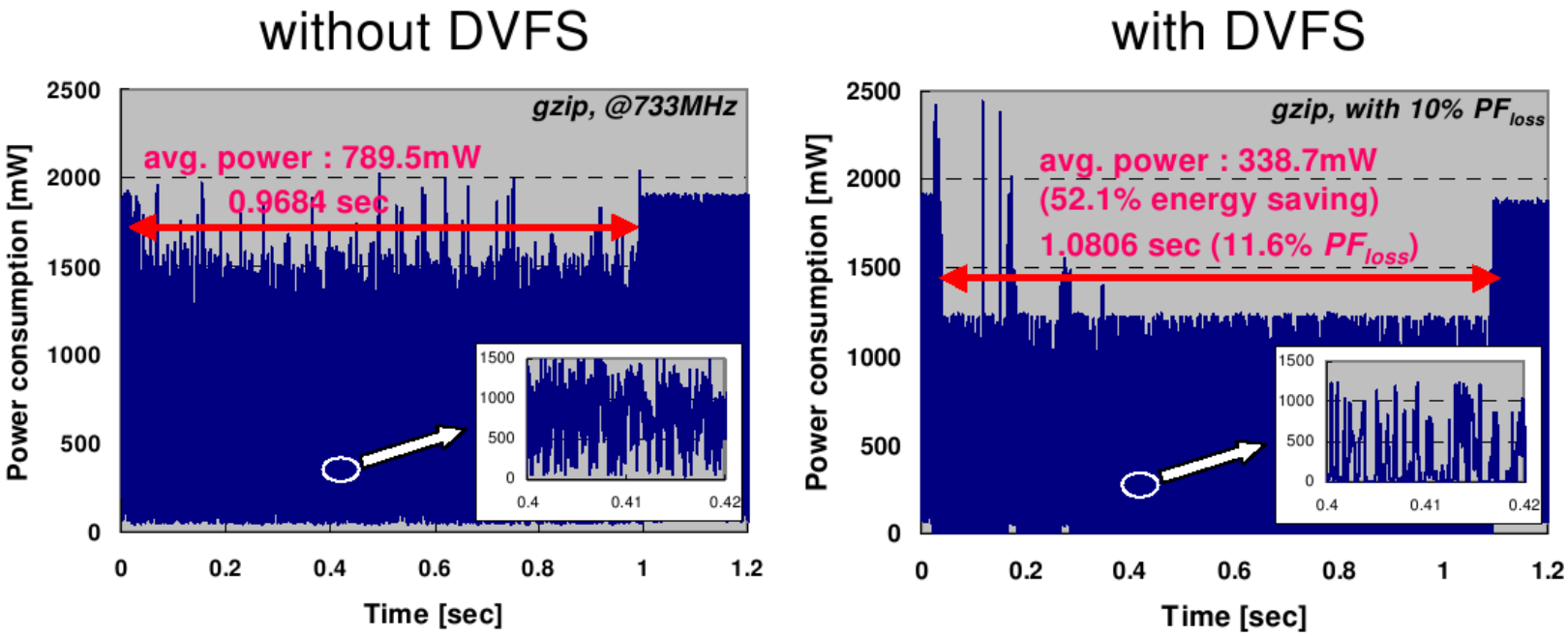
Frequency (MHz)	Voltage (V)
333	0.91
400	0.99
466	1.05
533	1.12
600	1.19
666	1.26
733	1.49

Data Acquisition system



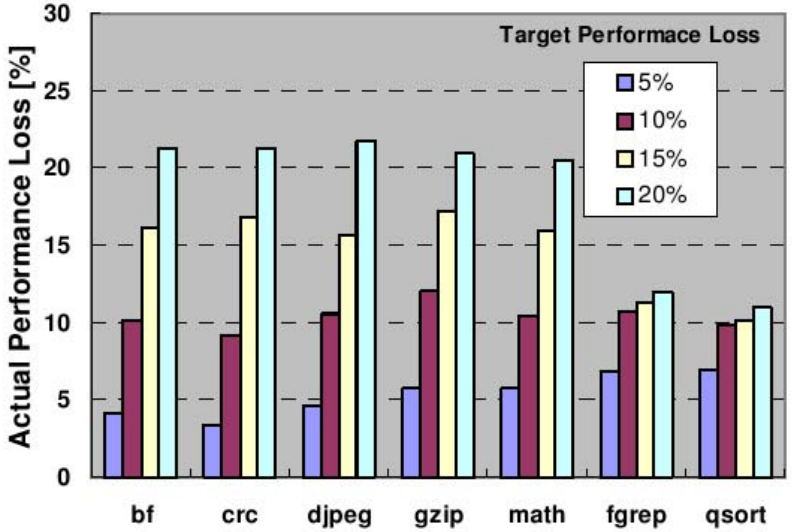
Experimental Results (I)

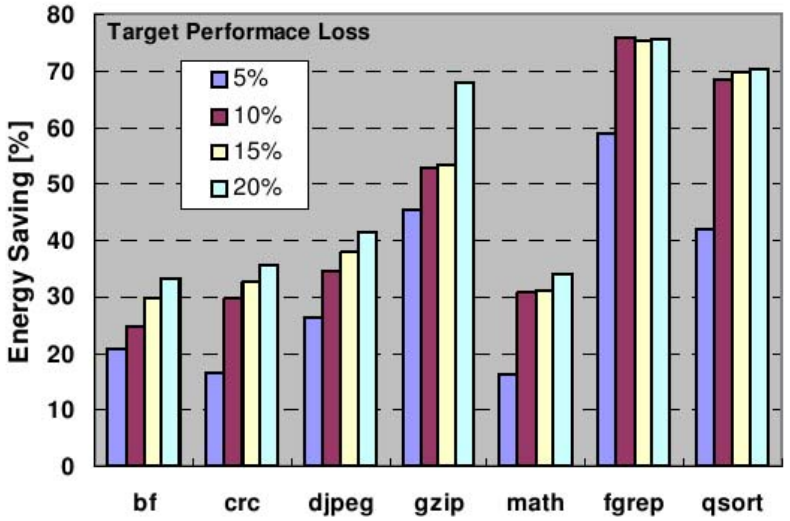
Power consumption vs. performance degradation



Experimental Results (II)

Measured PF_{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

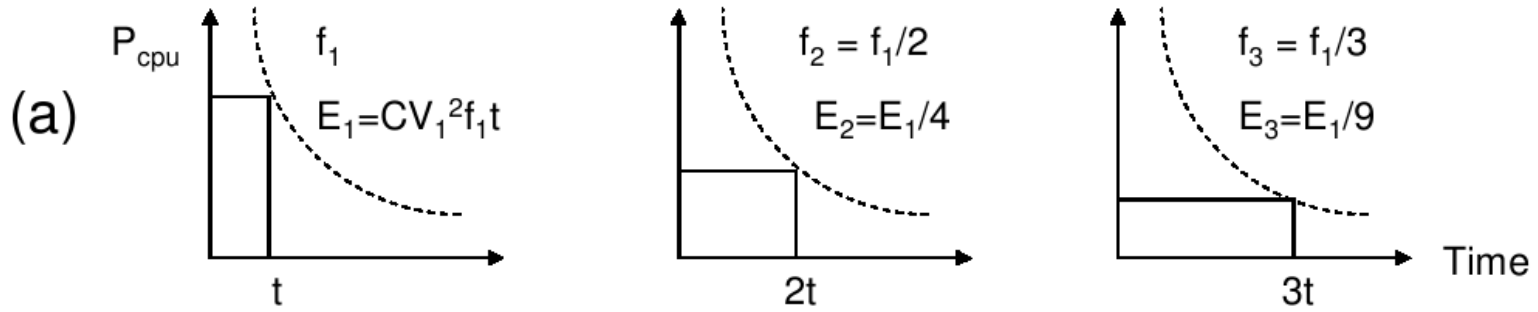
- more precisely, dynamic portion of the CPU energy

However, most systems comprise of many subsystems such as memory and peripheral devices

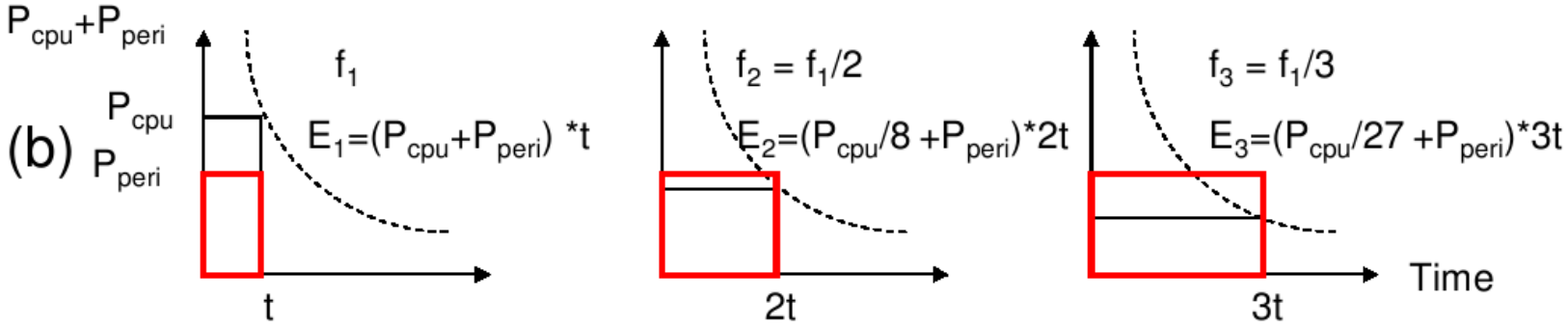
- battery lifetime also depends on power consumption in subsystems, which is not affected by CPU frequency changes
- lowering CPU frequency can cause shorter battery lifetime due to an increase in the standing and idle portions of the system energy consumption

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 γ denotes $P_{peripheral} / P_{cpu} \quad @ f_{max}$

$$@ f_{max} \quad : E_1 = P_{cpu} \cdot (1+\gamma) \cdot t$$

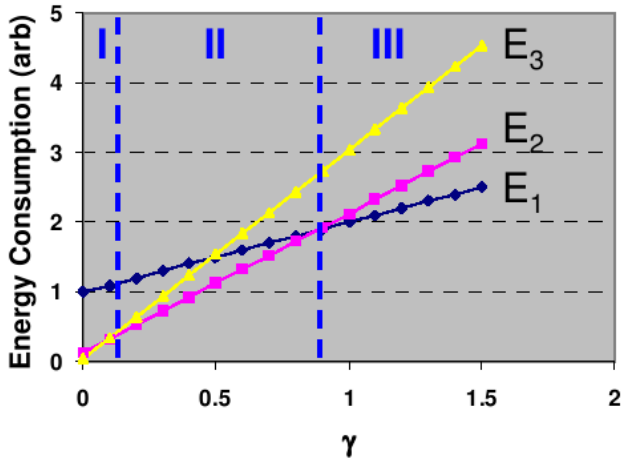
$$@ f_{max}/2 \quad : E_2 = P_{cpu} \cdot (1/4 + 2 \cdot \gamma) \cdot t$$

$$@ f_{max}/3 \quad : E_3 = P_{cpu} \cdot (1/9 + 3 \cdot \gamma) \cdot t$$

Based on the γ 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}

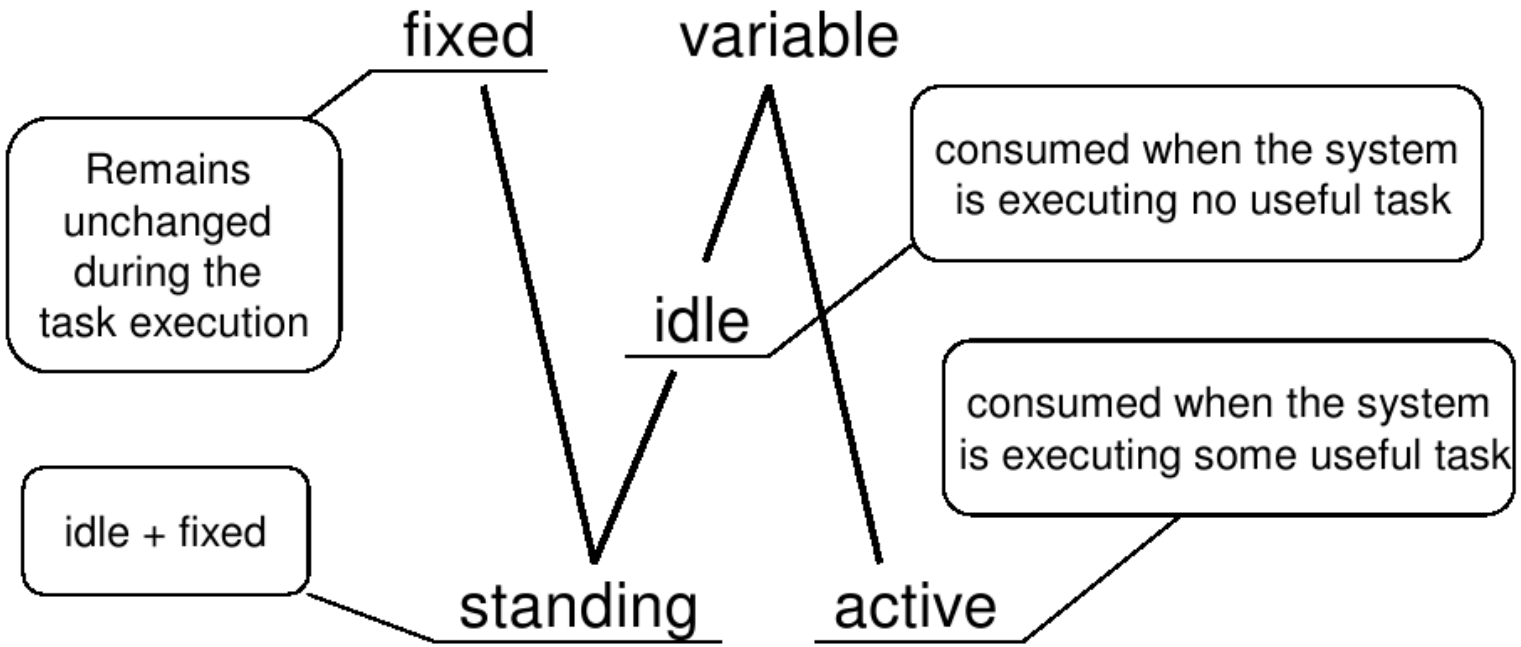


Previous DVFS techniques only consider the case when γ is 0

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



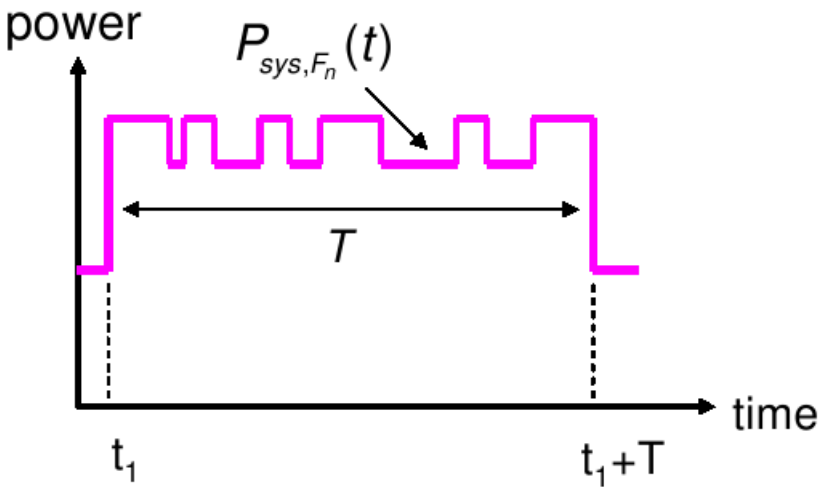
Total System Power Dissipation

Time-varying system power consumption

$$P_{sys,F_n}(t) = P_{sys}^{fix} + P_{sys,F_n}^{idle} + P_{sys,F_n}^{act}(t) = P_{sys,F_n}^{std} + P_{sys,F_n}^{act}(t)$$

$$= P_{cpu,F_n}^{std} + P_{cpu,F_n}^{act}(t) + \sum_{i=1}^{N} P_{mod,i}^{std} + \sum_{i=1}^{N} P_{mod,i}^{act}(t)$$

$$E_{sys,F_n} = \int_{t_1}^{t_1+T} P_{sys,F_n}(t) \cdot dt$$



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

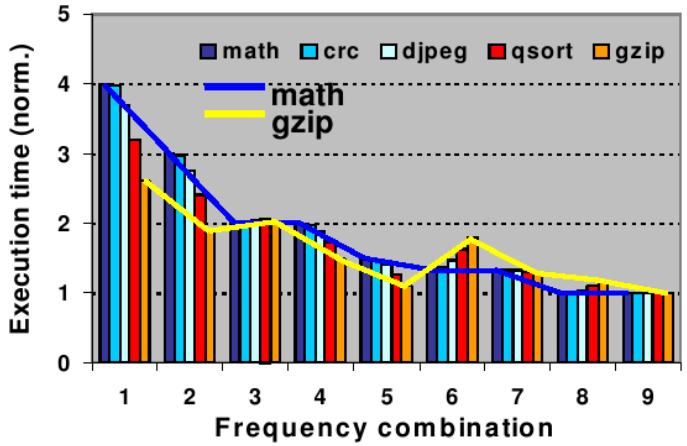
Frequency Settings in BitsyX

PXA255 core

- internal bus connects the core and other functional blocks inside the CPU
- external bus is connected to SDRAM

Nine frequency combinations

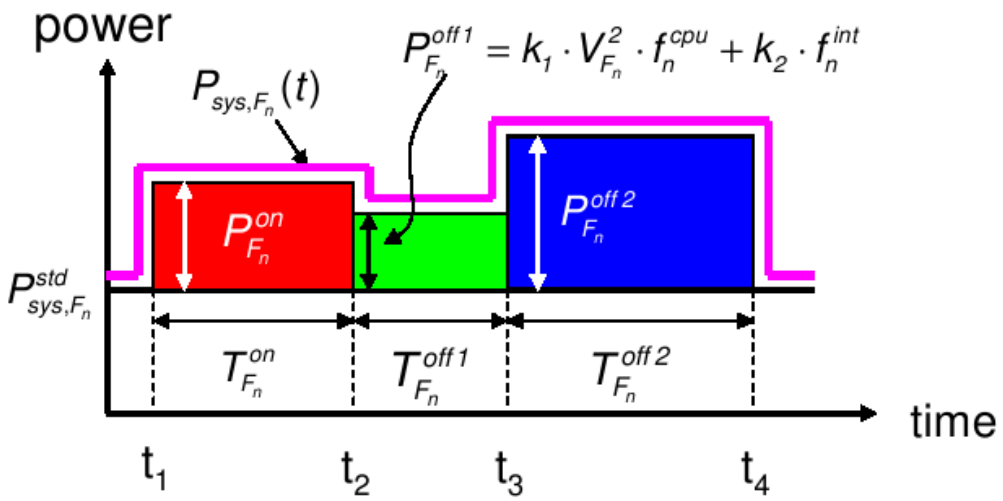
Freq. Set	f_{cpu} [MHz]	V_{cpu} [V]	f_{int} [MHz]	f_{ext} [MHz]
F_1	100	0.85	50	100
F_2	133	0.85	66	133
F_3	200	1.0	50	100
F_4	200	1.0	100	100
F_5	265	1.0	133	133
F_6	300	1.1	50	100
F_7	300	1.1	100	100
F_8	400	1.3	100	100
F_9	400	1.3	200	100



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 is:

$$E_{sys,F_n} = P^{std}_{sys,F_n} \cdot T + P^{on}_{F_n} \cdot T^{on}_{F_n} + P^{off\,1}_{F_n} \cdot T^{off\,1}_{F_n} + P^{off\,2}_{F_n} \cdot T^{off\,2}_{F_n}$$

Accuracy of the System Energy Model

Estimated energy consumption for “djpeg”

The average error rate is less than 4%

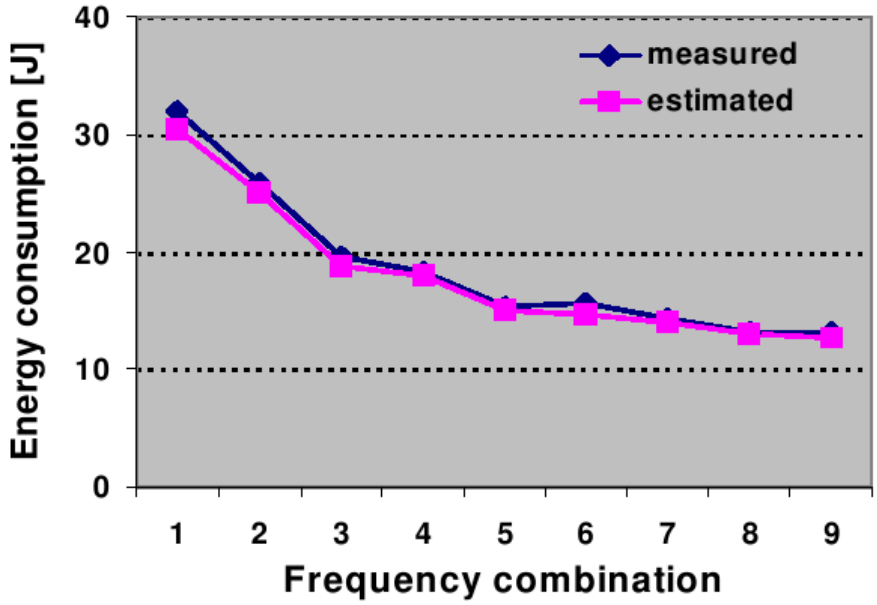
Extracted parameters

Freq. set	P^{std}_{sys,F_n} (mW)	$P^{on}_{F_n}$ (mW)	P^{off}_{int,F_n} (mW)	P^{off}_{ext,F_n} (mW)
F_1	1665	89	363	785
F_2	1757	148	479	785*1.33
F_3	1699	218	456	785
F_4	1728	217	766	785
F_5	1836	336	1018	785*1.33
F_6	1732	344	575	785
F_7	1778	378	885	785
F_8	1869	673	1113	785
F_9	1963	675	1733	785

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

$$P^{off}_{int,F_n} \sim k_1 \cdot V^2_{F_n} \cdot f^{cpu}_n + k_2 \cdot f^{int}_n$$

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



Optimal Frequency Setting

Consider both *timing* and *minimal system energy* constraint

- For timing constraint, using the performance loss factor defined as:

$$PF_{loss} = \frac{\lceil T_{F_n} - T_{F_{max}} \rceil}{T_{F_{max}}}$$

T_{F_n} task execution time at F_n

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

optimal frequency selection Pseudo code for

Timing constraint

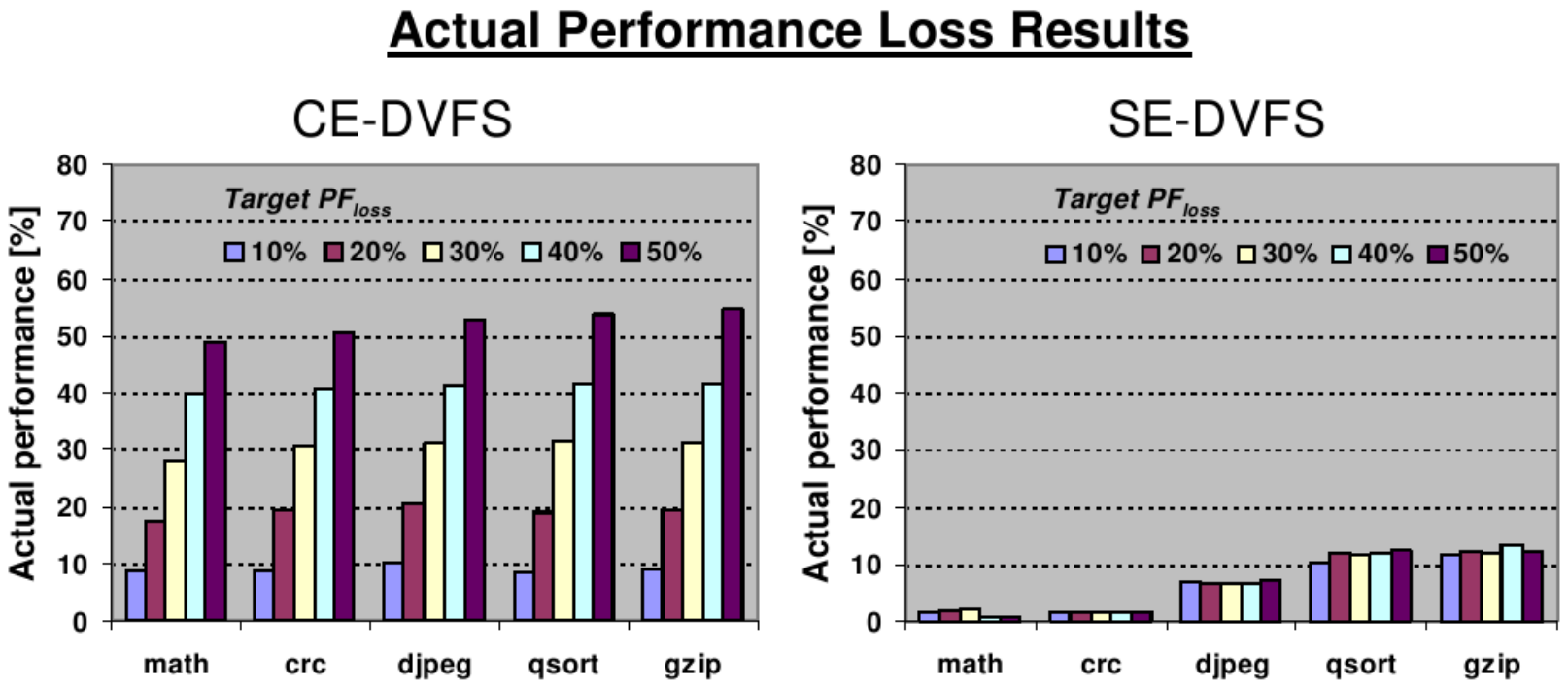
Energy optimization

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

Experimental Results (I)

Comparing two DVFS methods:

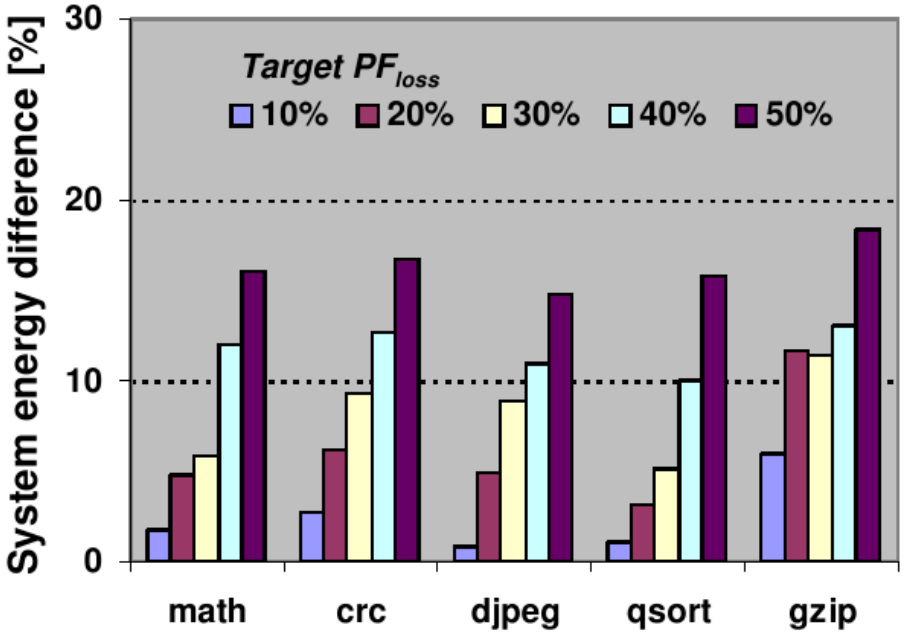
- SE-DVFS: targets the total system energy saving
- CE-DVFS: targets CPU energy savings only



Experimental Results (II)

SE-DVFS vs. CE-DVFS

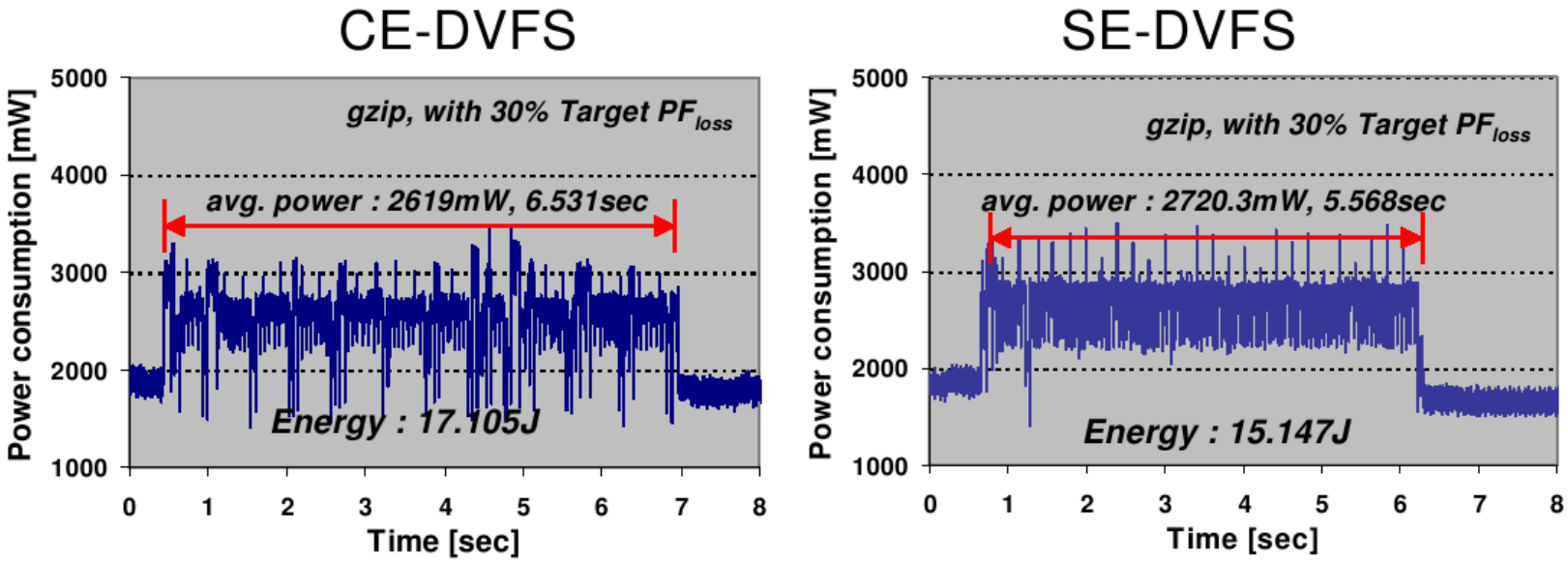
- SE-DVFS results in 2% ~ 18% lower total system energy consumption compared to CE-DVFS



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



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

Usual Power Management Features

- Power Domains
  - MCU, ARM11 core, MMU, L1 Caches
  - L2 cache
  - Peripherals
  - PLLs and pre-multiplier
- Power Gating
- Low Power Modes
  - Stop - clocks off
  - Doze - clocks/PLL off
  - State retention - ultra-low voltage
  - Hibernate - power down
- Multiple levels of clock hierarchy
- Dynamic Voltage Frequency Scaling (DVFS)

- Why Low Power Design?
- VLSI IC's Trends
- Where Power Goes?
  - Power Leakage Sources
  - Power, Delay and Reilability Models
- Power Optimization
  - Hardware solutions

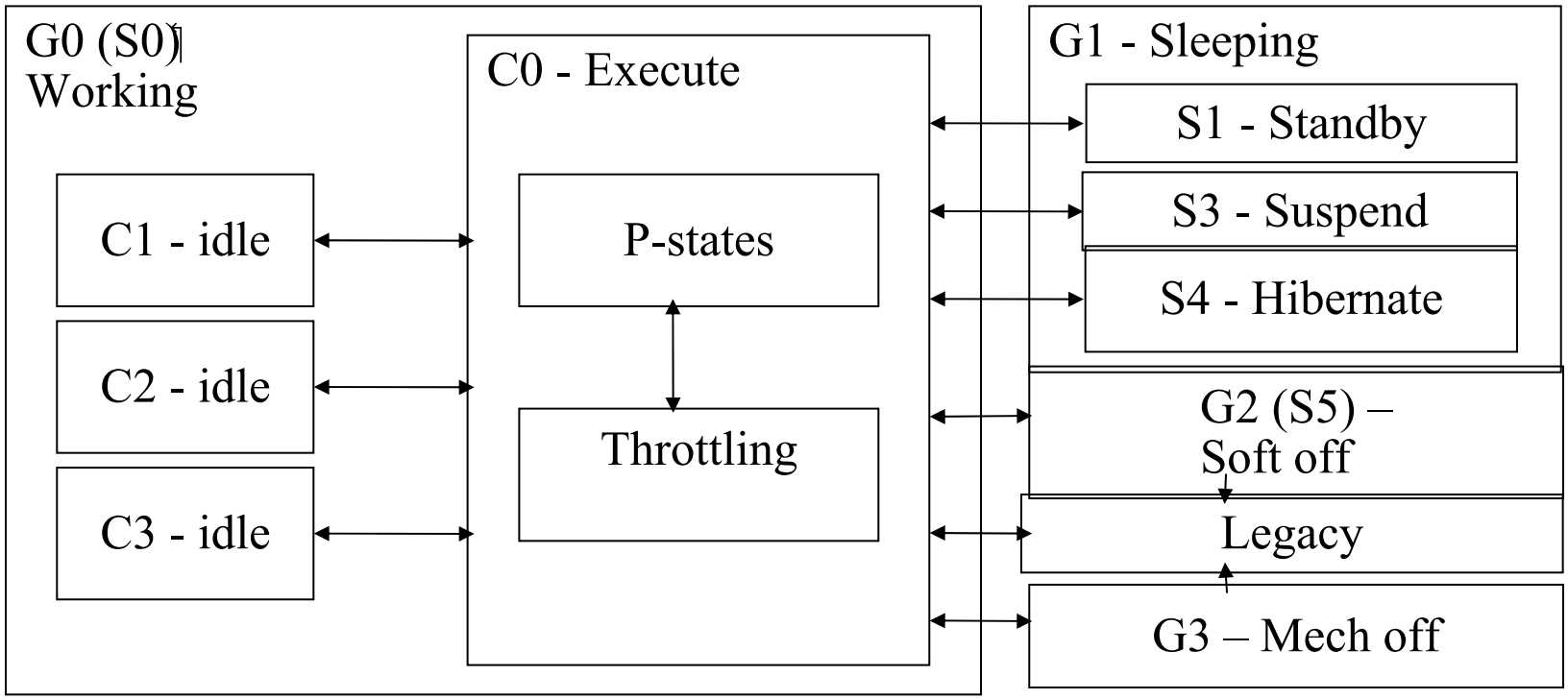
    Power islands, Clock Gating, DVFS
  - Software/System solutions

    APM/ACPI, OS level PM

ACPI: An Industrial Standard

- Advanced Configuration and Power Interface (ACPI) [Intel, Microsoft, Toshiba]
- Interface between OS and hardware
- Abstract, hierarchical
- finite-state machine
- Each state represents power and performance levels
- Global power states:
  - Mechanical off: power failure
  - Soft off: OS reboot needed
  - Sleep: entire system sleep, waiting for wake-up signals
  - Working: system is operational
  - Legacy: backward compatability
- Policy selection:
  - No concrete proposal (left to the engineer's decision)

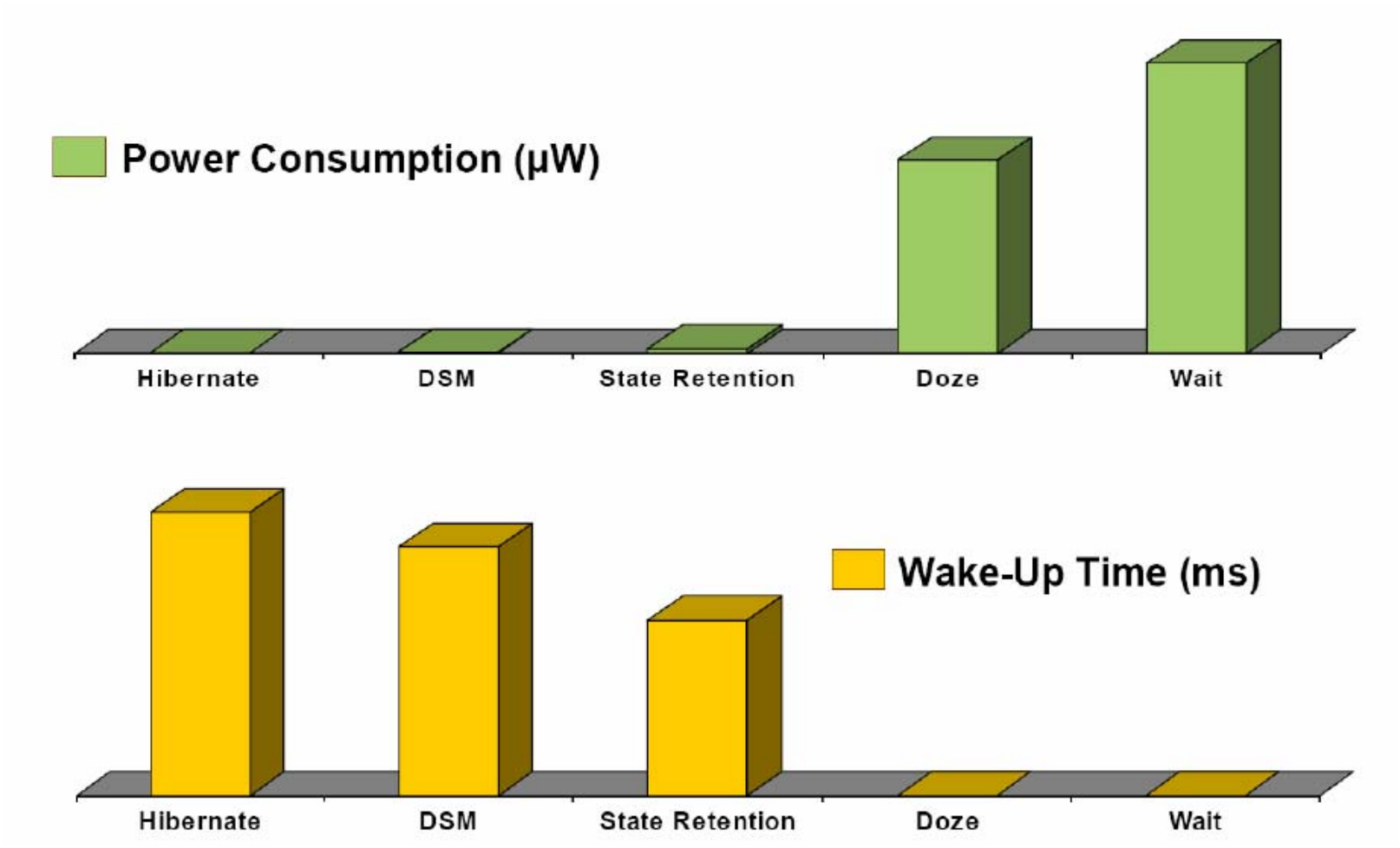
Linux ACPI Power States



WinCE Power States

- **Full on (D0)**
  - device on and running, receiving full power from the system delivering full functionality to the user
- **Low on (D1)**
  - device fully functional at a lower power or performance state than D0. D1 is applicable when the device is being used, but where peak performance is unnecessary and power is at a premium
- **Standby (D2)**
  - device partially powered with automatic wakeup on request. A device in state D2 is effectively standing by
- **Sleep (D3)**
  - device is partially powered with device-initiated wakeup if available. A device in state D3 is sleeping but capable of raising the System Power State on its own. It consumes only enough power to be able to do so; which must be less than or equal to the amount of power used in state D2
- **Off (D4)**
  - device no powered. A device in state D4 should not be consuming any significant power. Some peripheral busses require static terminations that intrinsically use non-zero power when a device is physically

Power Modes



OS-Directed Power Management

Motivations

- Digital circuitry has been researched a lot
- Idle or under-utilized resources can be shut down or slowed down

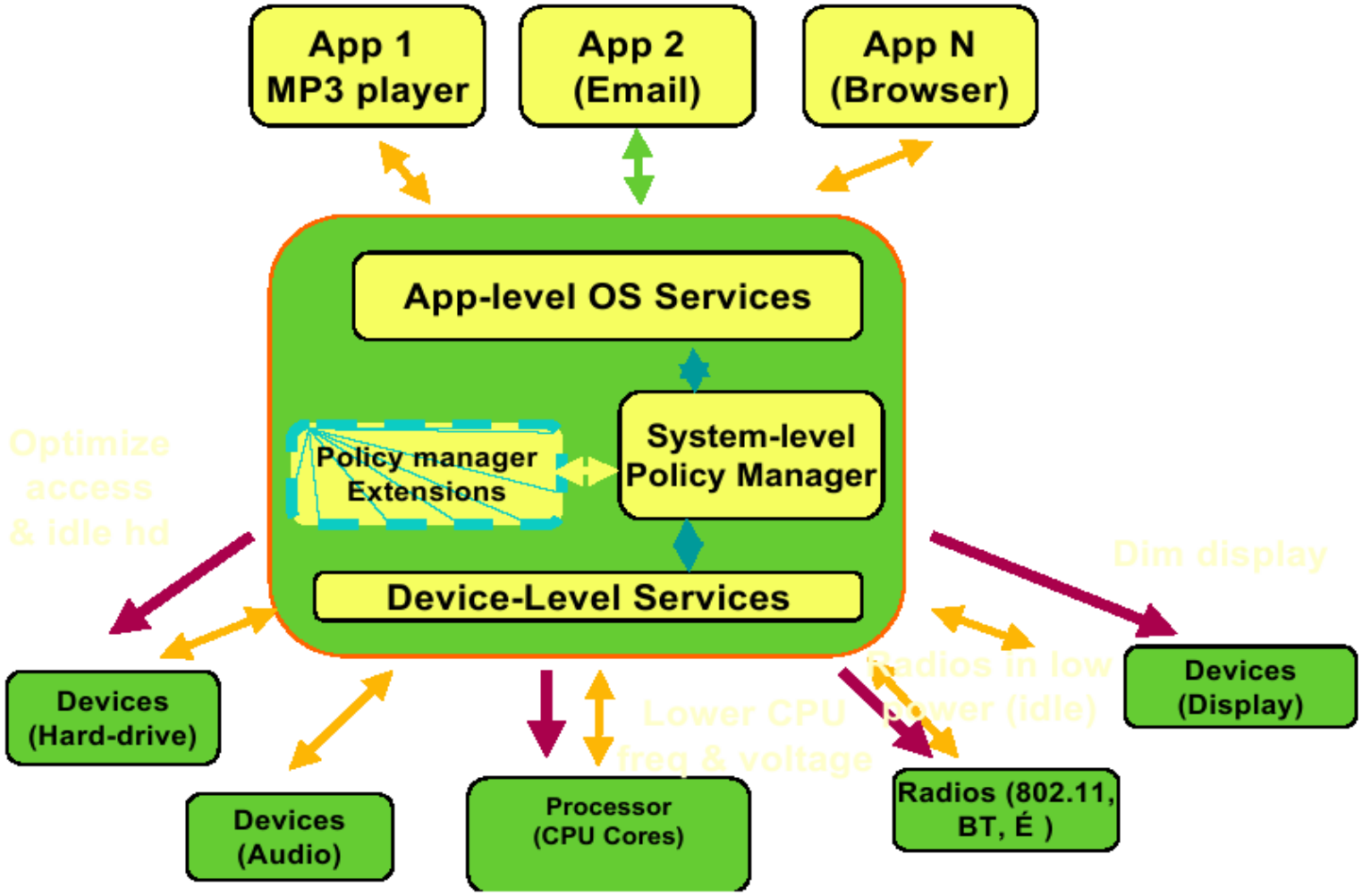
Why use OS

- OS knows all activities (state of system: running, sleep or waiting)
- Can communicate with devices
- Can include DPM as a module

DPM module

- Observer — collecting data from devices
- Policy — decision management
- Controller — steer hardware

Case Study - An MP3 Player



Low Power Challenges and Directions

- 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

Low Power Challenges and Directions

- 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

Five Key Technology Areas

- Fine-Grain Power Management
- I/O Optimization
- System Power Conversion
- Client Sensor Architecture
- Power Policy Management

Bibliography

- **[kihwan2004dynamic]** - "Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation", **K. Choi, W. Lee, R. Soma, M. Pedram**, Proccedings of the International Conference on Computer-Aided Design (ICCAD'04), November 2004, pp 29-34. http://doi.acm.org/10.1145/1112239.1112251
- **[roy2003leakage] -** "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicron CMOS Circuits", **K. Roy, et. al.,** Proceedings of the IEEE, February 2003, pp. 305-327
- **[dal2006powerislands]** - "Power Islands: A High-Level Technique for Counteracting Leakage in Deep Sub-Micron", **D. Dal, A. Nunez, N. Mansouri**, ISQED 2006

