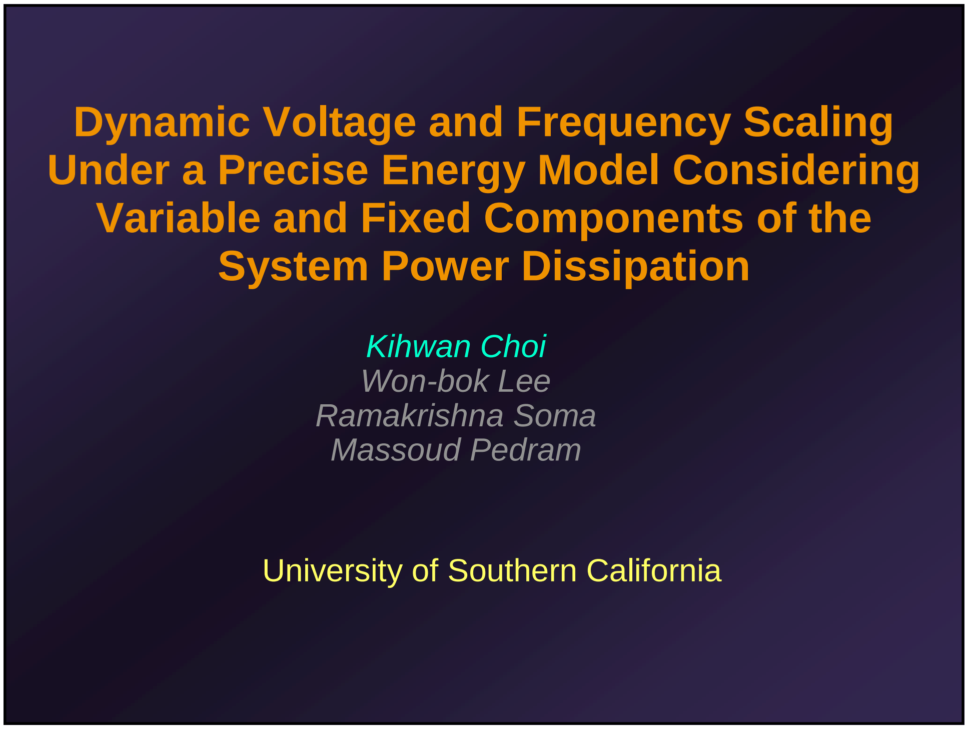
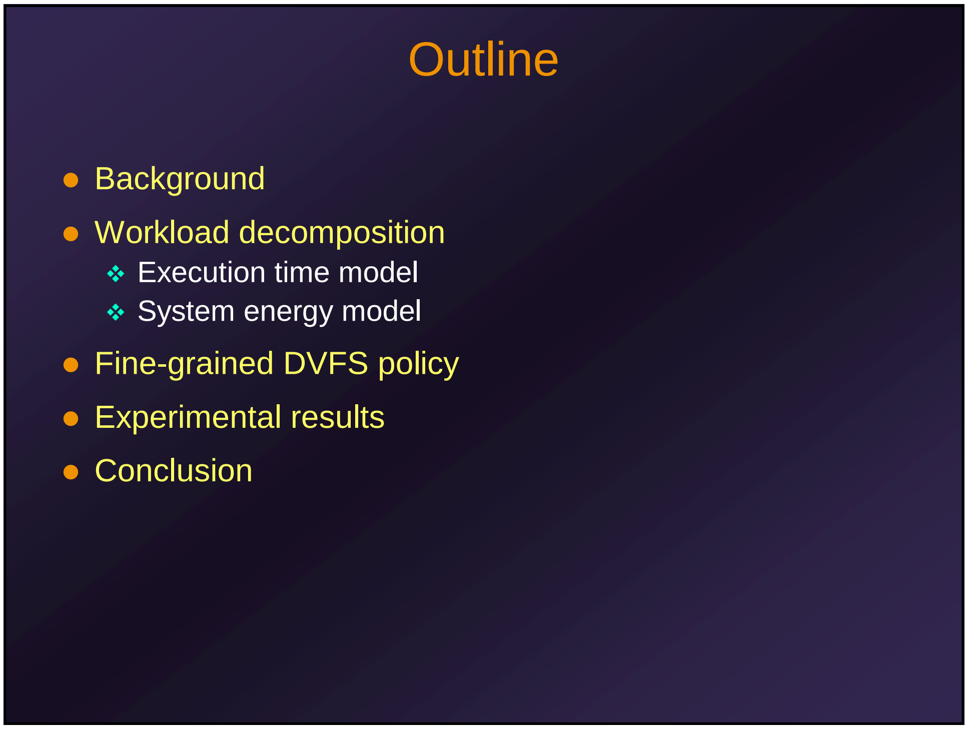


Dynamic Voltage and Frequency Scaling Under a Precise Energy Model Considering Variable and Fixed Components of the System Power Dissipation

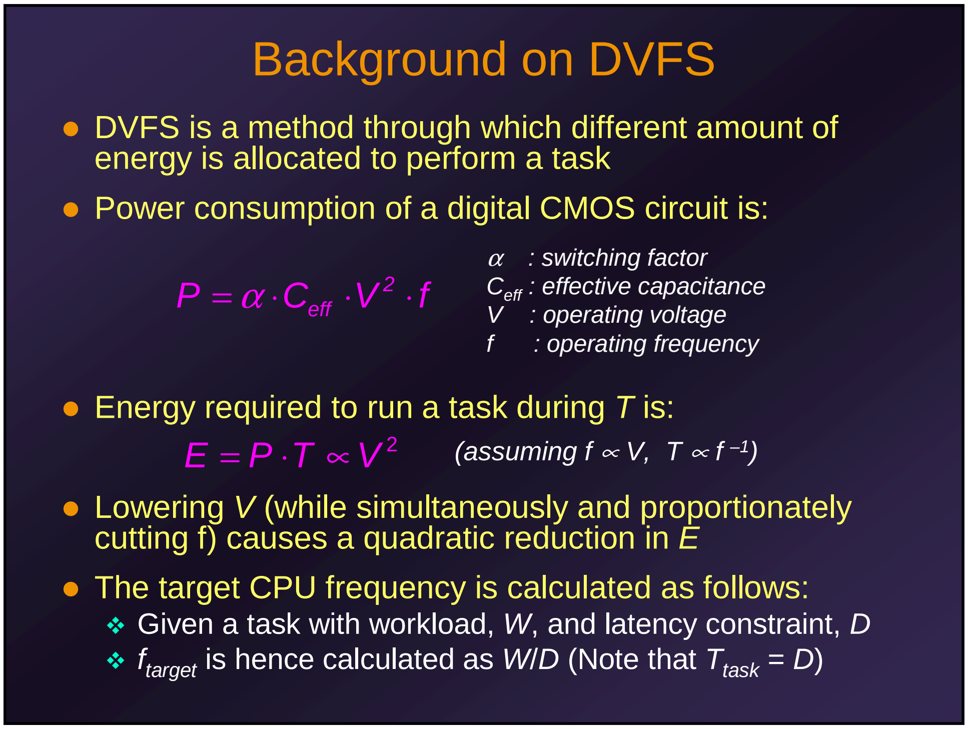
Kihwan Choi
Won-bok Lee
Ramakrishna Soma
Massoud Pedram

University of Southern California



Outline

- Background
- Workload decomposition
  - Execution time model
  - System energy model
- Fine-grained DVFS policy
- Experimental results
- Conclusion



Background on DVFS

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

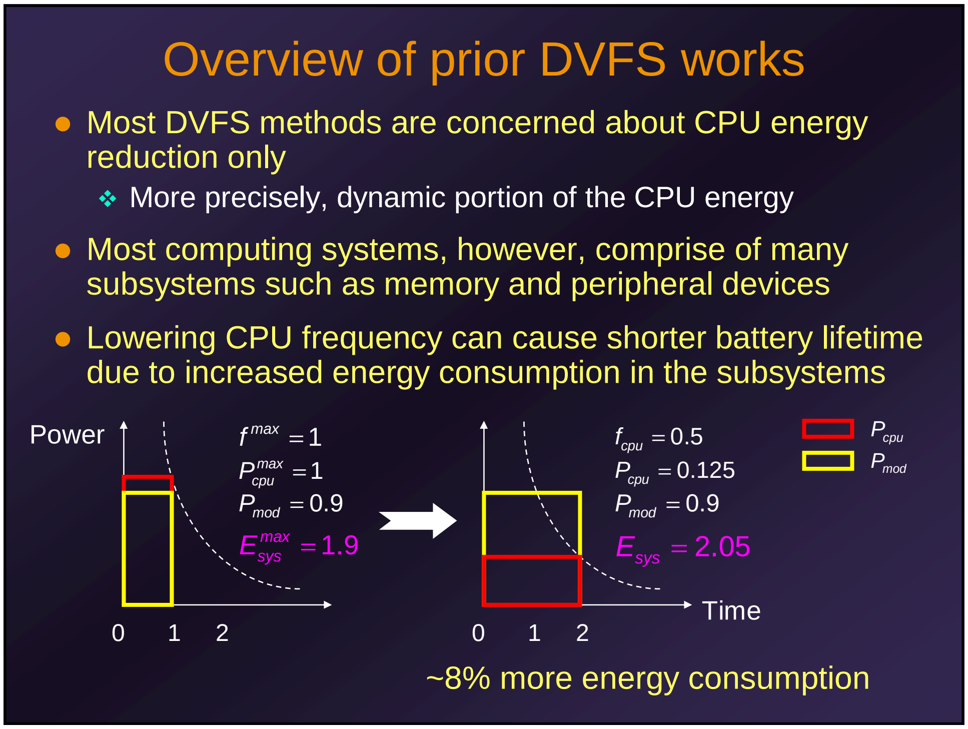
$$P = \alpha \cdot C_{eff} \cdot V^2 \cdot f$$

α : switching factor
C_{eff} : effective capacitance
V : operating voltage
f : operating frequency

- Energy required to run a task during *T* is:

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

- Lowering *V* (while simultaneously and proportionately cutting f) causes a quadratic reduction in *E*
- The target CPU frequency is calculated as follows:
  - Given a task with workload, *W*, and latency constraint, *D*
  - f_{target} is hence calculated as *W*/*D* (Note that $T_{task} = D$)



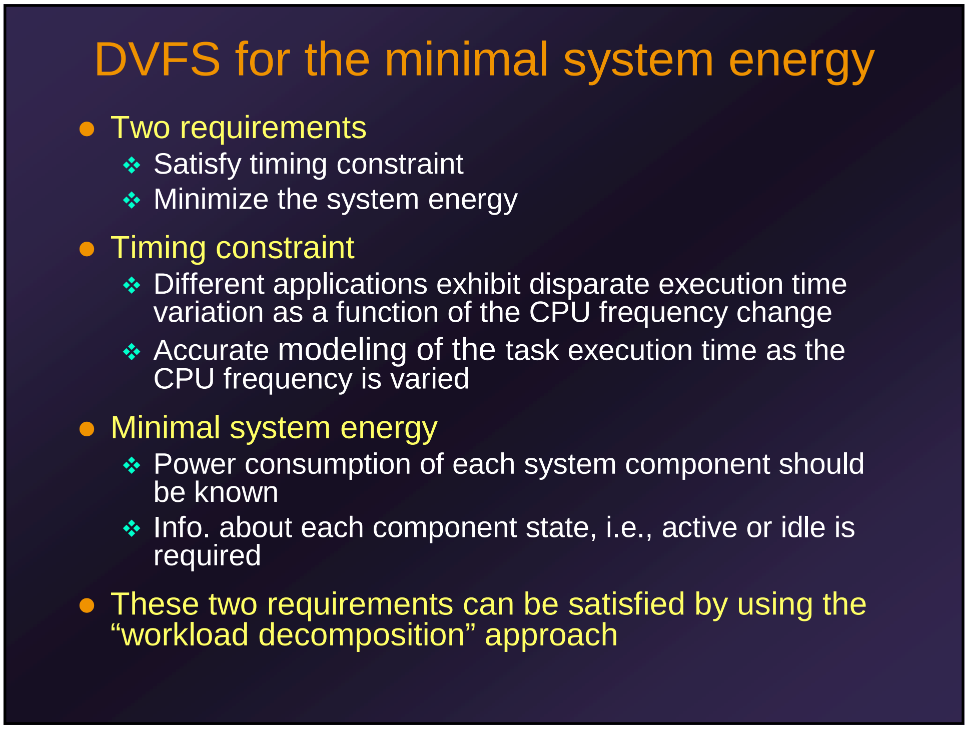
Overview of prior DVFS works

- Most DVFS methods are concerned about CPU energy reduction only
  - More precisely, dynamic portion of the 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

$f^{max} = 1$
$P_{cpu}^{max} = 1$
$P_{mod} = 0.9$
$E_{sys}^{max} = 1.9$

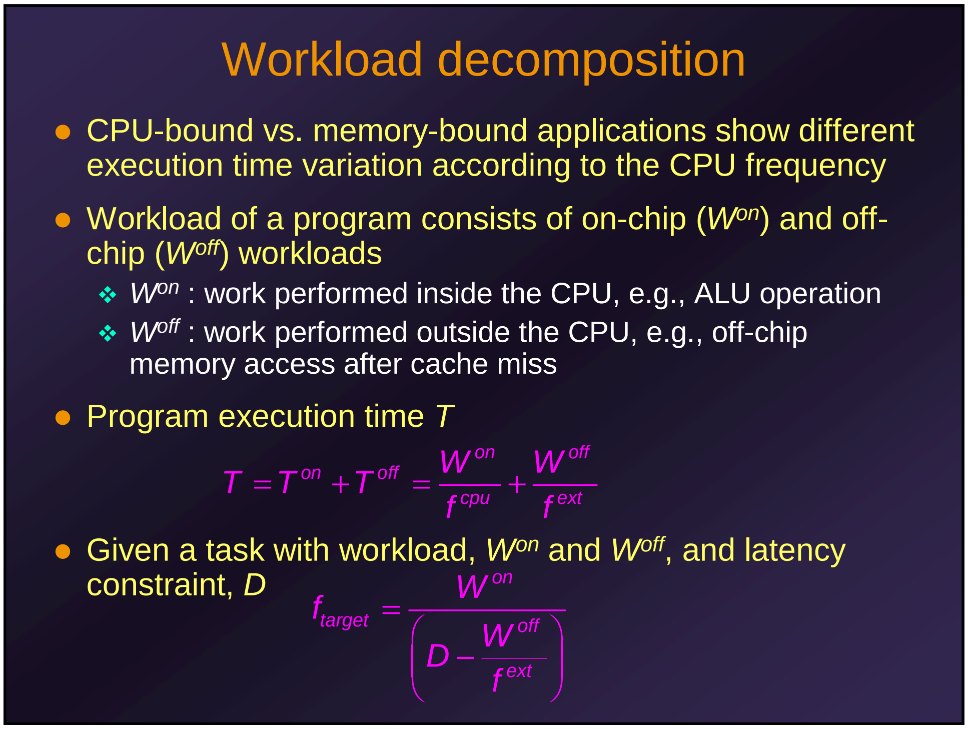
$f_{cpu} = 0.5$
$P_{cpu} = 0.125$
$P_{mod} = 0.9$
$E_{sys} = 2.05$

~8% more energy consumption



DVFS for the minimal system energy

- Two requirements
  - Satisfy timing constraint
  - Minimize the system energy
- Timing constraint
  - Different applications exhibit disparate execution time variation as a function of the CPU frequency change
  - Accurate modeling of the task execution time as the CPU frequency is varied
- Minimal system energy
  - Power consumption of each system component should be known
  - Info. about each component state, i.e., active or idle is required
- These two requirements can be satisfied by using the "workload decomposition" approach



Workload decomposition

- CPU-bound vs. memory-bound applications show different execution time variation according to the CPU frequency
- Workload of a program consists of on-chip (W^{on}) and off-chip (W^{off}) workloads
  - W^{on} : work performed inside the CPU, e.g., ALU operation
  - W^{off} : work performed outside the CPU, e.g., off-chip memory access after cache miss
- Program execution time T

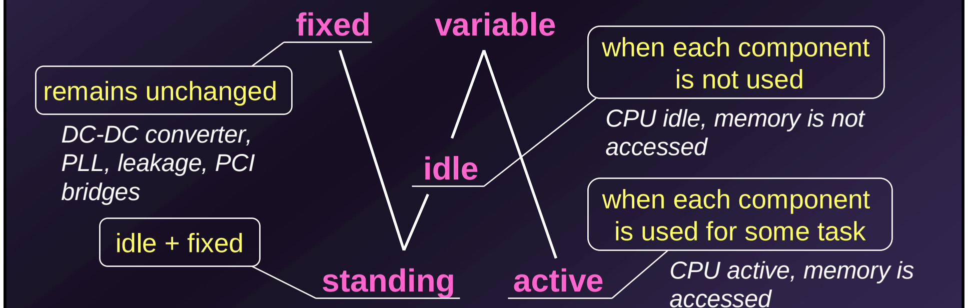
$$T = T^{on} + T^{off} = \frac{W^{on}}{f^{cpu}} + \frac{W^{off}}{f^{ext}}$$

- Given a task with workload, W^{on} and W^{off}, and latency constraint, D

$$f_{target} = \frac{W^{on}}{\left(D - \frac{W^{off}}{f^{ext}}\right)}$$

System power breakdown

- Power consumption profile fluctuates greatly due to alternate execution of W^{on} and W^{off}
  - W^{on} (W^{off}) requires the CPU (sub-module) power
- System power consumption can be broken into the following components:



fixed

variable

remains unchanged

DC-DC converter, PLL, leakage, PCI bridges

idle

when each component is not used

CPU idle, memory is not accessed

idle + fixed

standing

active

when each component is used for some task

CPU active, memory is accessed

obtained by *simple measurements* or using *values in the spec*

Performance monitoring unit (PMU)

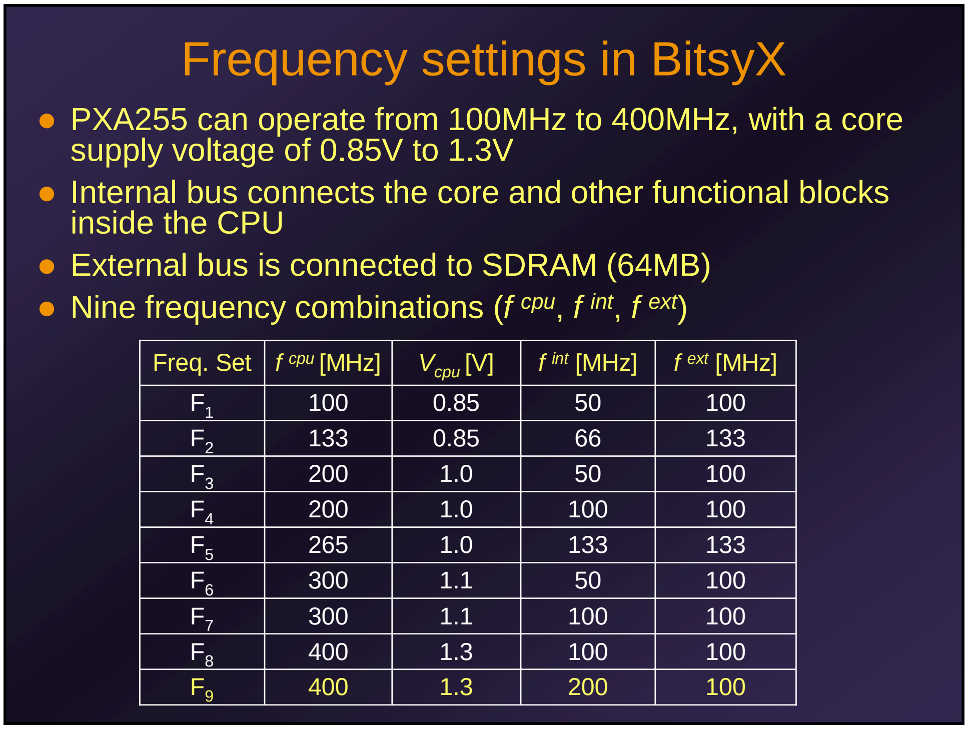
- W^{on} is modeled as:

$$W^{on} = \sum_{i=1}^{N} CPI_{on}^{i} = N \cdot CPI_{on}^{avg}$$

N : number of onchip instructions

CPI_{on} : CPU clocks per instruction

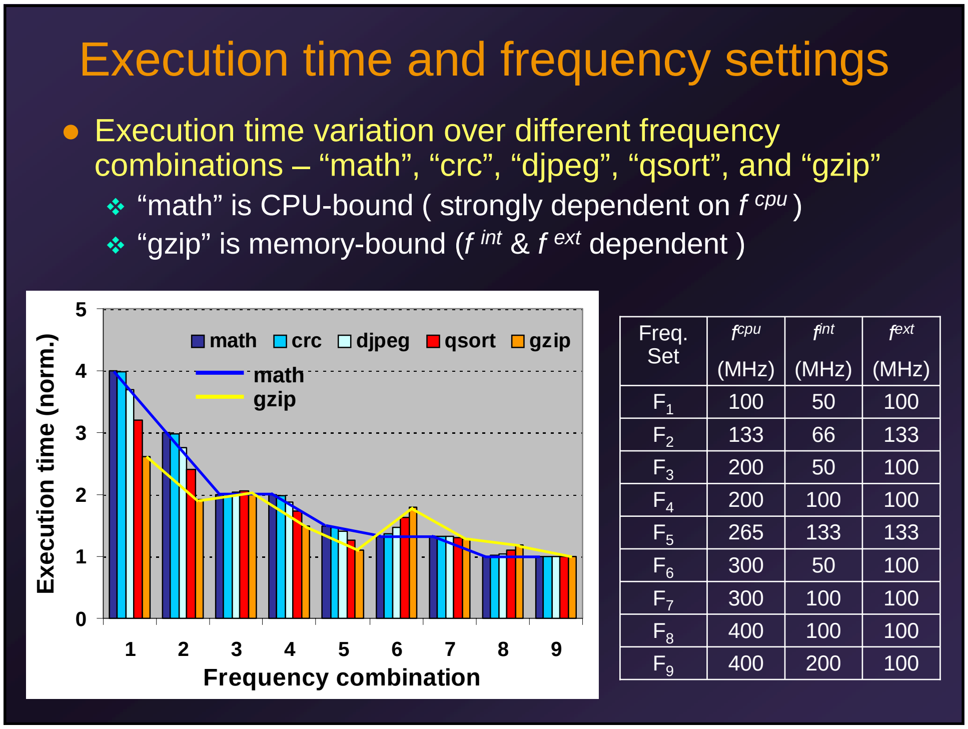
- PMU on the PXA255 processor chip can report up to 15 different dynamic events during execution of a program
  - Cache hit/miss counts, TLB hit/miss counts, No. of stall cycles, Total no. of instructions being executed, Branch misprediction counts
- For DVFS, we use the PMU to generate statistics for
  - Total no. of instructions being executed (INSTR)
  - No. of stall cycles due to on/off-chip data dependencies (STALL)
  - No. of Data Cache misses (DMISS)
- We also record the no. of clock cycles from the beginning of the program execution (CCNT)



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})

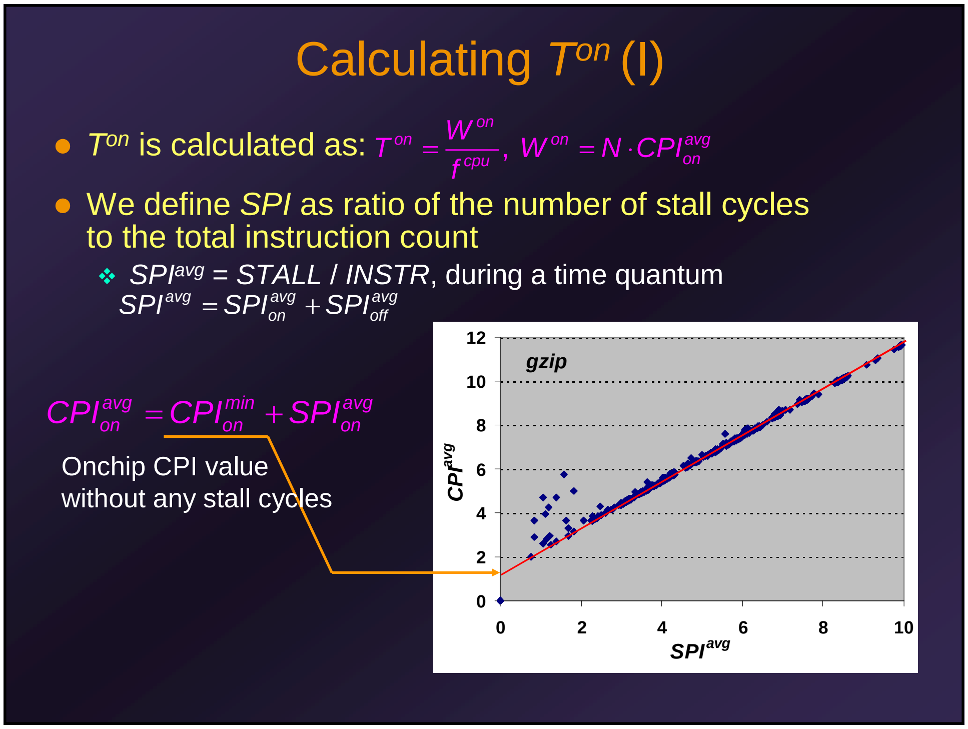
Freq. Set	f^{cpu} [MHz]	V_{cpu} [V]	f^{int} [MHz]	f^{ext} [MHz]
F_1	100	0.85	50	100
F_2	133	0.85	66	133
F_3	200	1.0	50	100
F_4	200	1.0	100	100
F_5	265	1.0	133	133
F_6	300	1.1	50	100
F_7	300	1.1	100	100
F_8	400	1.3	100	100
F_9	400	1.3	200	100



Execution time and frequency settings

- Execution time variation over different frequency combinations – “math”, “crc”, “djpeg”, “qsort”, and “gzip”
  - “math” is CPU-bound (strongly dependent on f^{cpu})
  - “gzip” is memory-bound (f^{int} & f^{ext} dependent)

Freq. Set	f^{cpu} (MHz)	f^{int} (MHz)	f^{ext} (MHz)
F_1	100	50	100
F_2	133	66	133
F_3	200	50	100
F_4	200	100	100
F_5	265	133	133
F_6	300	50	100
F_7	300	100	100
F_8	400	100	100
F_9	400	200	100

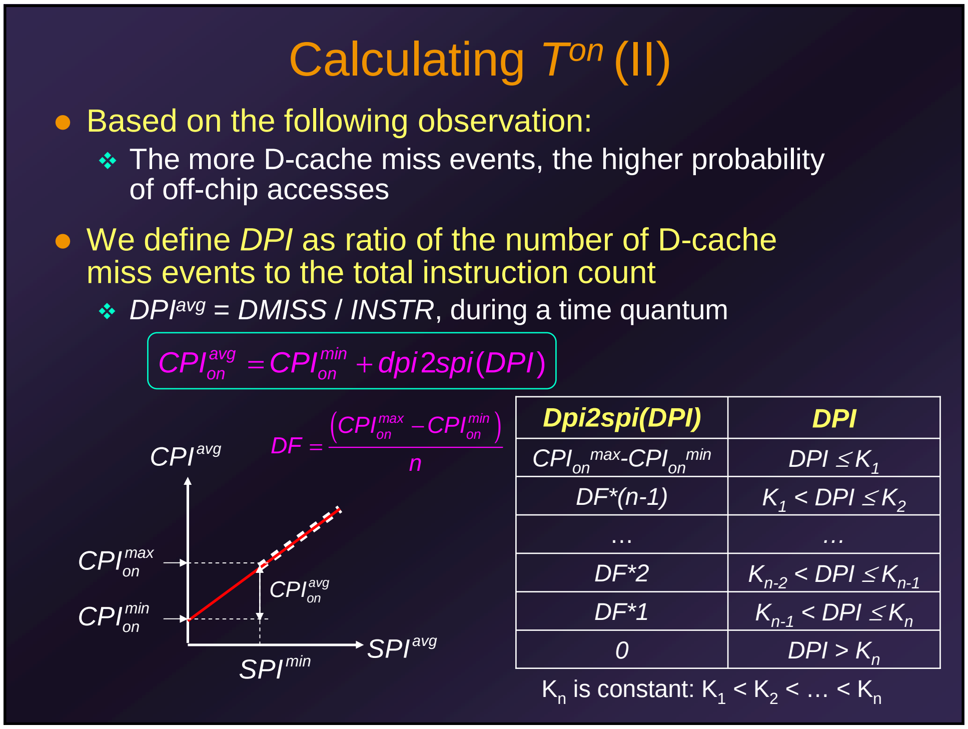


Calculating T^{on} (I)

- T^{on} is calculated as: $T^{on} = \frac{W^{on}}{f^{cpu}}$, $W^{on} = N \cdot CPI_{on}^{avg}$
- We define *SPI* as ratio of the number of stall cycles to the total instruction count
  - ❖ $SPI^{avg} = STALL / INSTR$, during a time quantum
    $SPI^{avg} = SPI_{on}^{avg} + SPI_{off}^{avg}$

$$CPI_{on}^{avg} = CPI_{on}^{min} + SPI_{on}^{avg}$$

Onchip CPI value without any stall cycles



Calculating T^{on} (II)

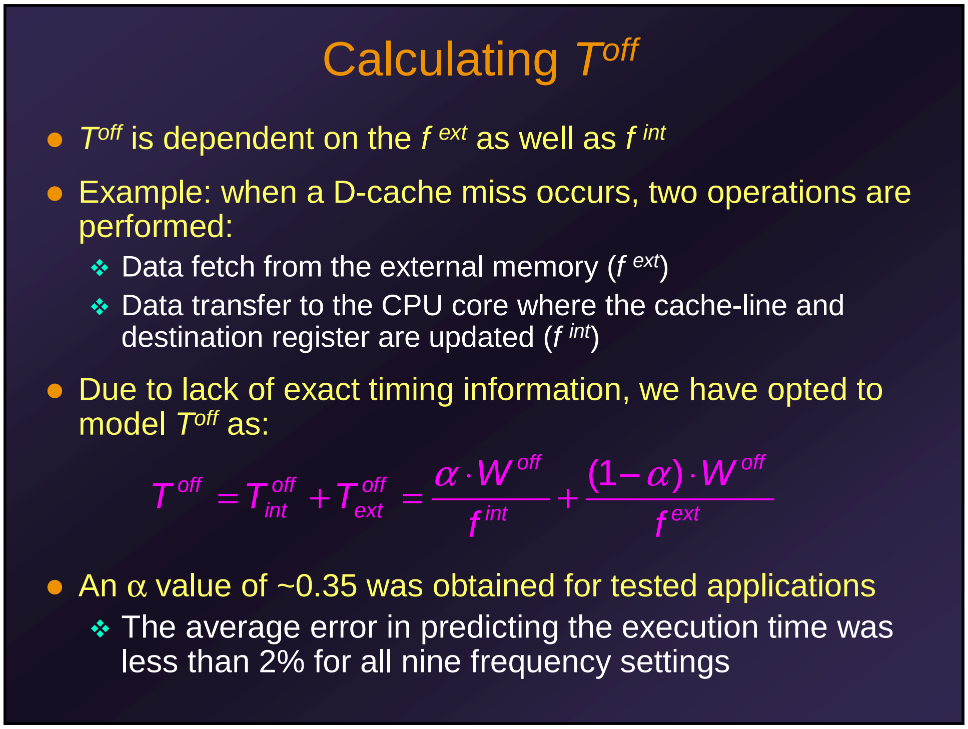
- Based on the following observation:
  - ❖ The more D-cache miss events, the higher probability of off-chip accesses
- We define *DPI* as ratio of the number of D-cache miss events to the total instruction count
  - ❖ $DPI^{avg} = DMISS / INSTR$, during a time quantum

$$\boxed{CPI_{on}^{avg} = CPI_{on}^{min} + dpi2spi(DPI)}$$

$$DF = \frac{(CPI_{on}^{max} - CPI_{on}^{min})}{n}$$

Dpi2spi(DPI)	*DPI*
CPI_{on}^{max}-CPI_{on}^{min}	$DPI \le K_1$
DF*(n-1)	$K_1 < DPI \le K_2$
…	…
DF*2	$K_{n-2} < DPI \le K_{n-1}$
DF*1	$K_{n-1} < DPI \le K_n$
0	$DPI > K_n$

K_n is constant: $K_1 < K_2 < … < K_n$

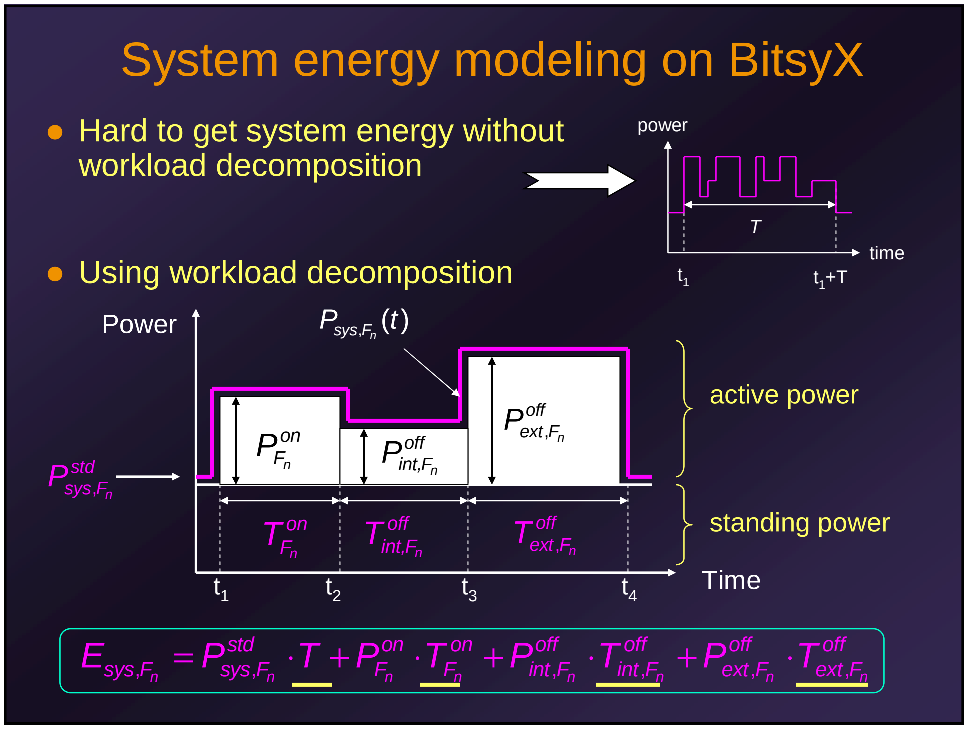


Calculating T^{off}

- T^{off} is dependent on the $f^{\,ext}$ as well as $f^{\,int}$
- Example: when a D-cache miss occurs, two operations are performed:
  - Data fetch from the external memory ($f^{\,ext}$)
  - Data transfer to the CPU core where the cache-line and destination register are updated ($f^{\,int}$)
- Due to lack of exact timing information, we have opted to model T^{off} as:

$$T^{off} = T^{off}_{int} + T^{off}_{ext} = \frac{\alpha \cdot W^{off}}{f^{\,int}} + \frac{(1-\alpha) \cdot W^{off}}{f^{\,ext}}$$

- An α value of ~0.35 was obtained for tested applications
  - The average error in predicting the execution time was less than 2% for all nine frequency settings



System energy modeling on BitsyX

- Hard to get system energy without workload decomposition

- Using workload decomposition

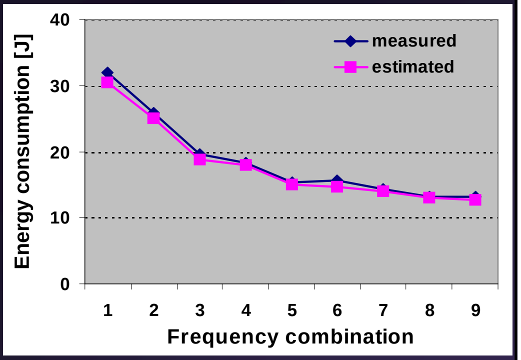
$$E_{sys,F_n} = P^{std}_{sys,F_n} \cdot \underline{T} + P^{on}_{F_n} \cdot \underline{T^{on}_{F_n}} + P^{off}_{int,F_n} \cdot \underline{T^{off}_{int,F_n}} + P^{off}_{ext,F_n} \cdot \underline{T^{off}_{ext,F_n}}$$

Accuracy of the system energy model

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

measured 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



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

$k_1 = 0.73$ [nF], $k_2 = 6.2$ [V²nF]

Determining the optimal frequency setting

- Consider timing constraint followed by system energy minimization
  - For a timing constraint, we used performance loss (*PF_{loss}*) which is defined as:

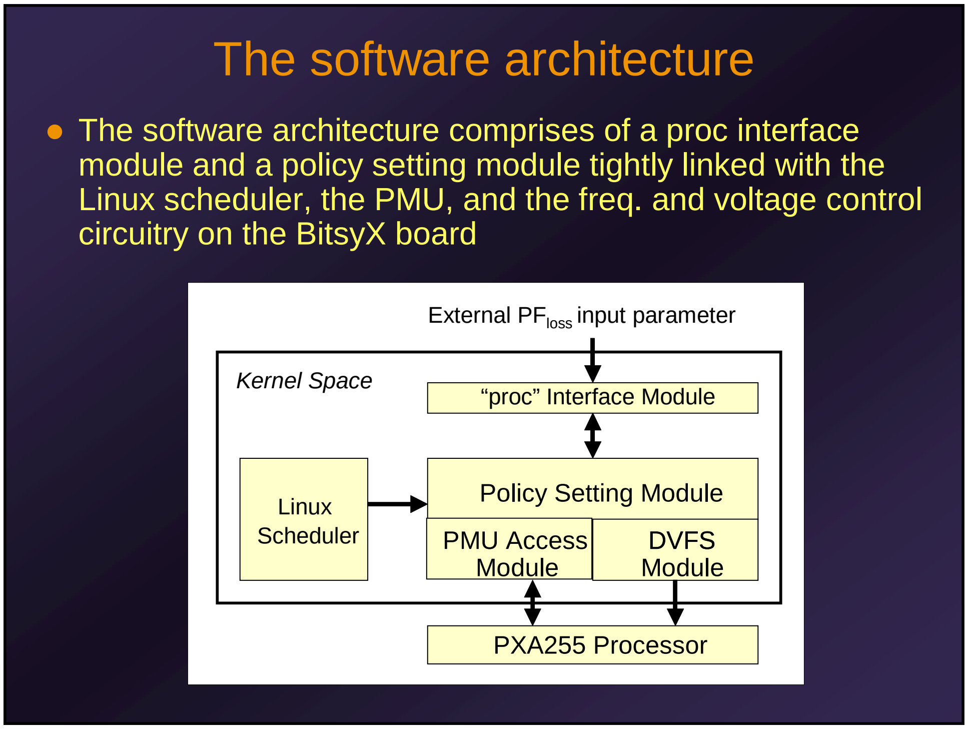
$$PF_{loss} = \frac{\left(T_{F_n} - T_{F_{max}}\right)}{T_{F_{max}}}$$

- Pseudo code for optimal frequency selection

Timing constraint → 3.

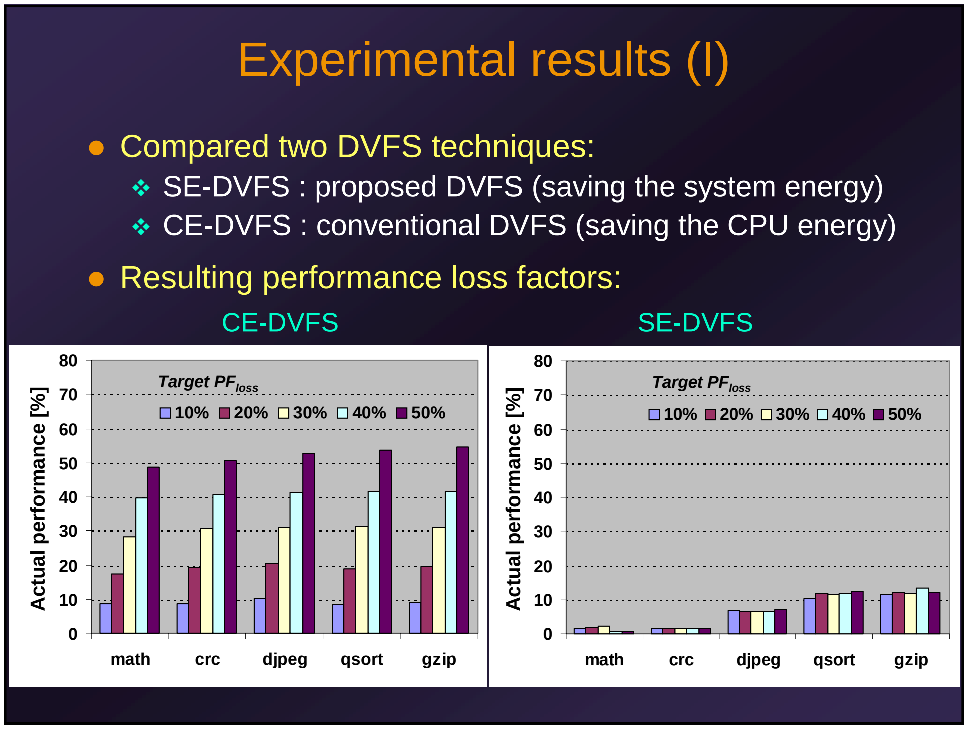
Energy minimization → 7.

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



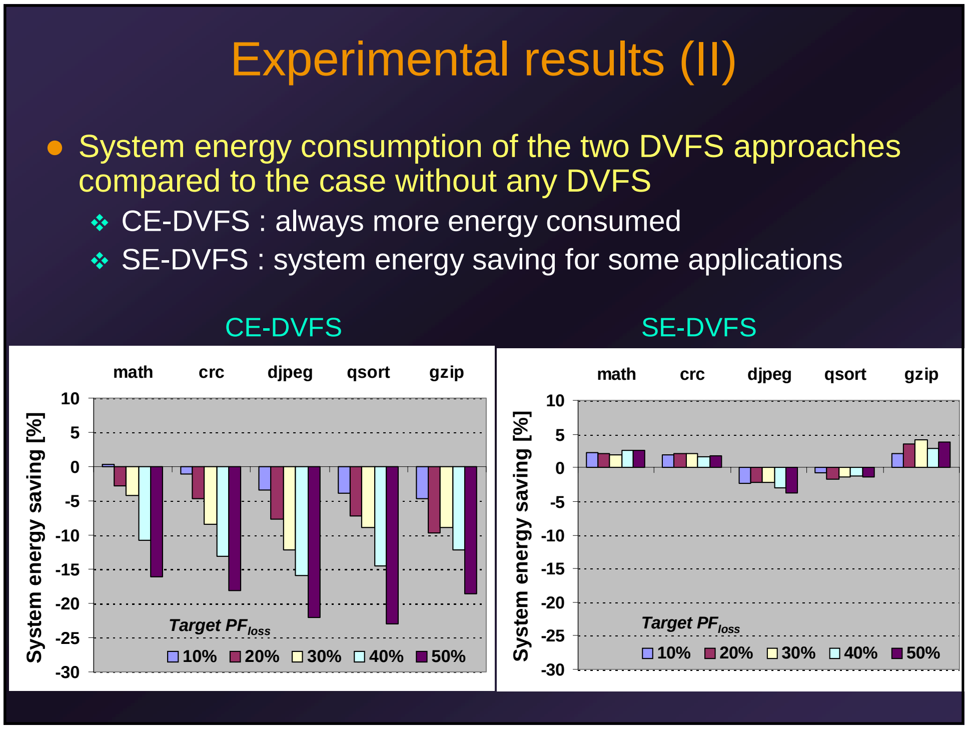
The software architecture

- The software architecture comprises of a proc interface module and a policy setting module tightly linked with the Linux scheduler, the PMU, and the freq. and voltage control circuitry on the BitsyX board



Experimental results (I)

- Compared two DVFS techniques:
  - SE-DVFS : proposed DVFS (saving the system energy)
  - CE-DVFS : conventional DVFS (saving the CPU energy)
- Resulting performance loss factors:

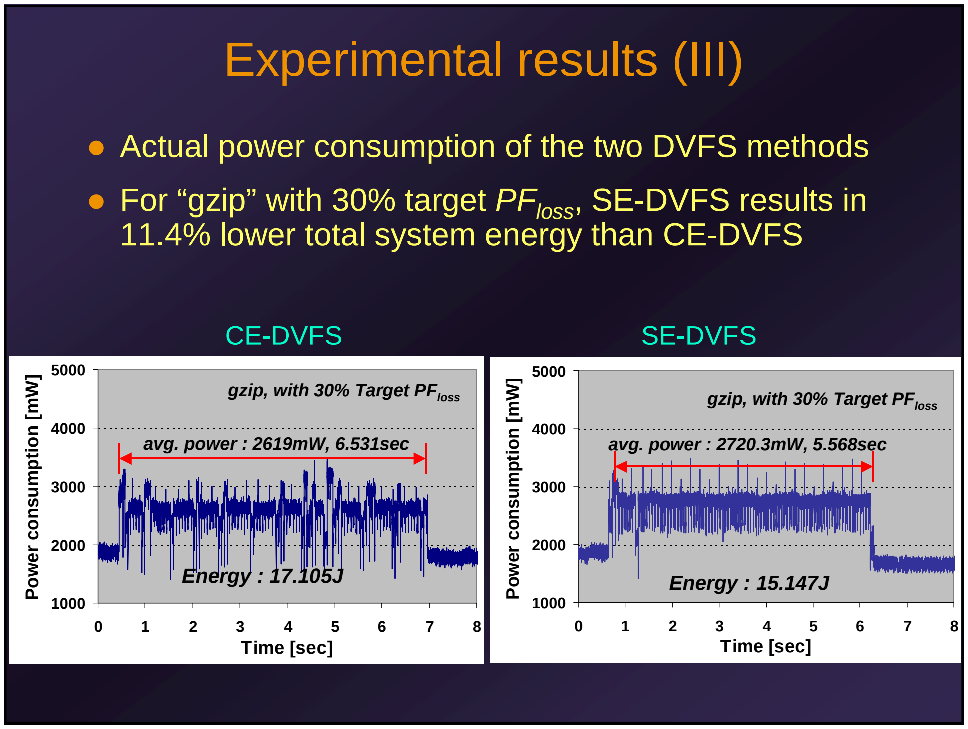


Experimental results (II)

- System energy consumption of the two DVFS approaches compared to the case without any DVFS
  - CE-DVFS : always more energy consumed
  - SE-DVFS : system energy saving for some applications

CE-DVFS

SE-DVFS

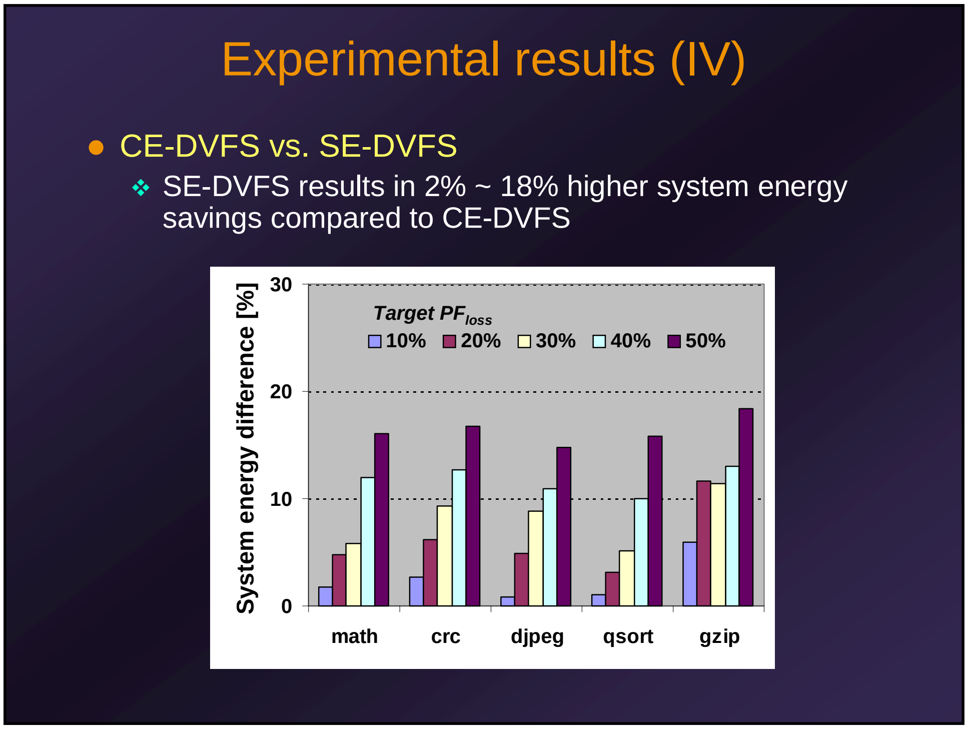


Experimental results (III)

- 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

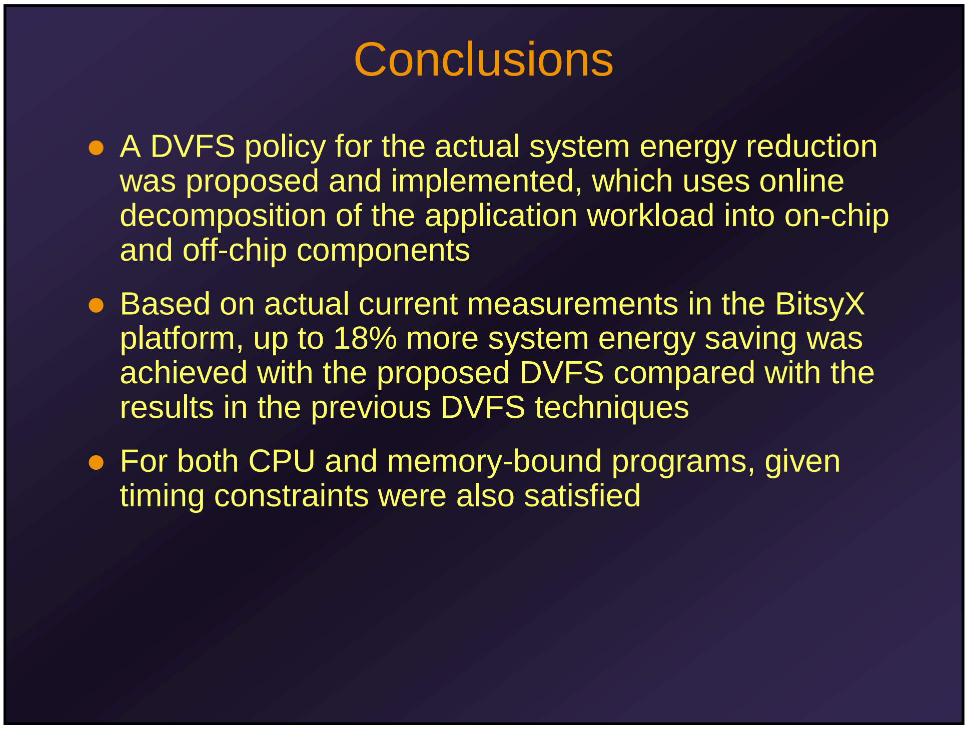
CE-DVFS

SE-DVFS



Experimental results (IV)

- CE-DVFS vs. SE-DVFS
  - SE-DVFS results in 2% ~ 18% higher system energy savings compared to CE-DVFS



Conclusions

- A DVFS policy for the actual system energy reduction was proposed and implemented, which uses online decomposition of the application workload into on-chip and off-chip components
- Based on actual current measurements in the BitsyX platform, up to 18% more system energy saving was achieved with the proposed DVFS compared with the results in the previous DVFS techniques
- For both CPU and memory-bound programs, given timing constraints were also satisfied