Energy-Efficient Real-Time Task Scheduling in Multiprocessor DVS Systems

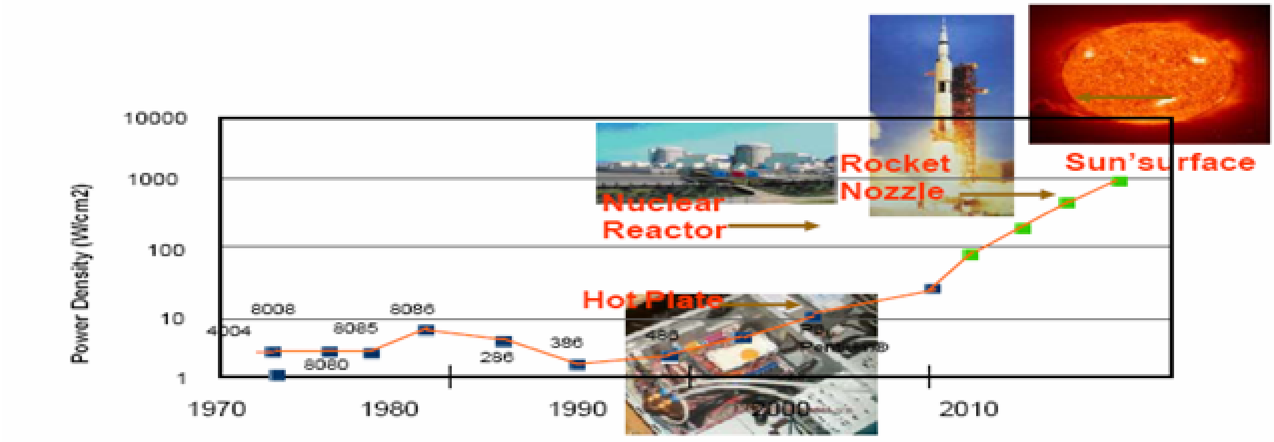
Jian-Jia Chen[*], Chuan Yue Yang,
Tei-Wei Kuo, and Chi-Sheng Shih

Embedded Systems and Wireless Networking Lab.
Department of Computer Science and Information Engineering,
National Taiwan University
*Fisheries Agency, Council of Agriculture,
Executive Yuan, Taiwan

Agenda

- Introduction
- Scheduling Algorithms
  - Energy-Efficient Scheduling for Homogeneous Multiprocessor Systems
    - Negligible leakage power consumption
    - Non-negligible leakage power consumption
  - Energy-Efficient Scheduling for Heterogeneous Two-Processor Systems
  - Allocation Cost Minimization for Multiprocessor Synthesis under Energy Constraints
- Conclusion and Open Issues

Motivations for Power Saving



- Rapid Increasing of Power Consumption
  - Modern hardware design increases the power consumption of circuits.
  - The power consumption of processors increases dramatically.
- Slow Increasing of the Battery Capacity
  - The battery capacity increases about 5% per year
  - Battery life time is a major concern for embedded systems
- Embedded Systems vs Servers

  The reduction of power is also needed to cut the power bill off

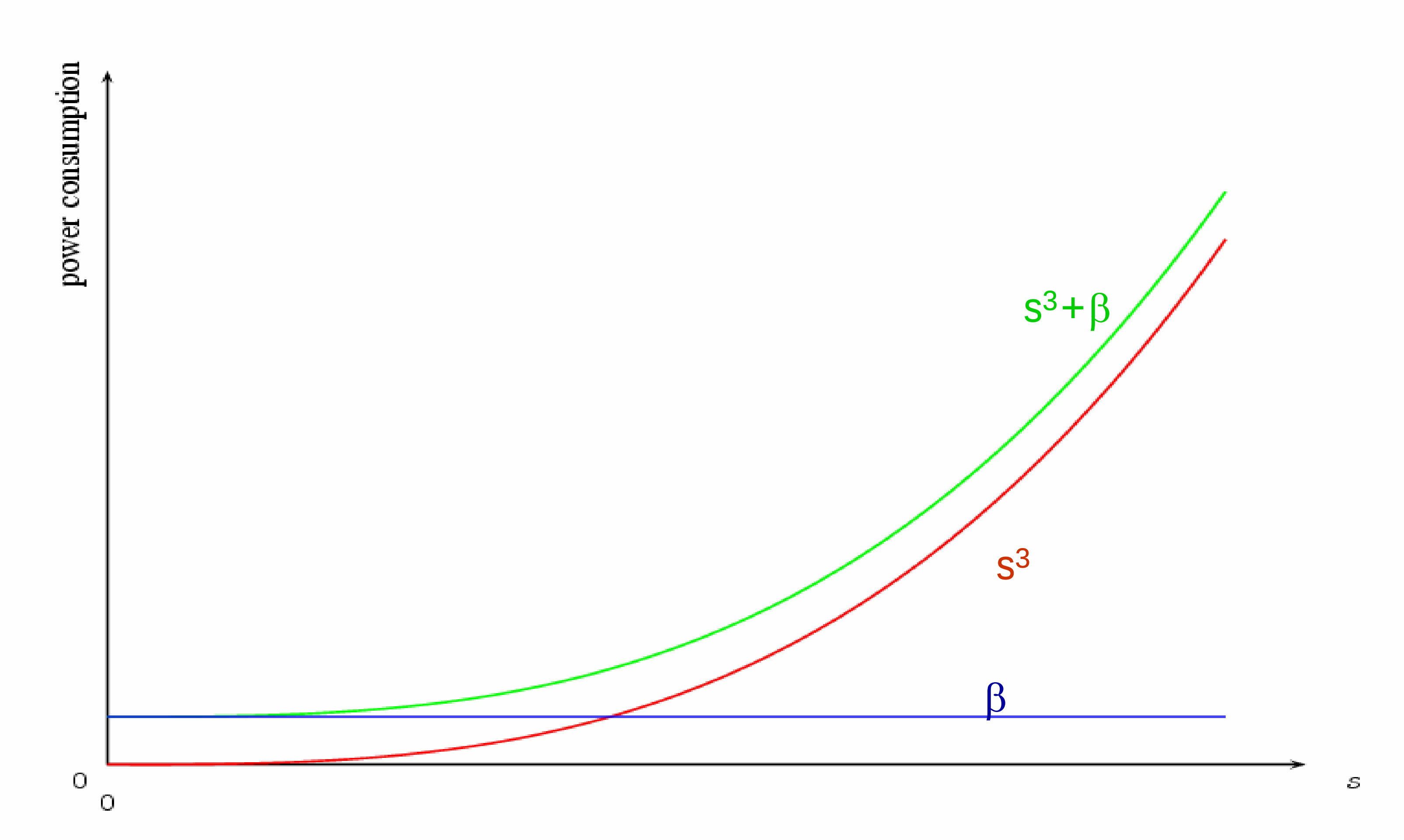
Hardware Methodology for Power Saving

- Dynamic power management (DPM)
  - The operation mode of the system
  - ACPI
- Micro-architecture technique
  - Adaptive architecture
  - Cache management
- Dynamic voltage scaling (DVS)
  - Supply voltage scaling
    - Intel Xscale, StrongARM; Transmeta Crusoe, Intel Pentium 4
    - Intel SpeedStep, AMD PowerNow!
  - Threshold voltage scaling

Dynamic Voltage Scaling

- A higher supply voltage usually results in a higher frequency (or higher execution speed)
  - $s = k * (V_{dd}-V_t)^2/(V_{dd})$, where
    - s is the corresponding speed of the supply voltage V_{dd} and
    - V_t is the threshold voltage
- The dynamic power consumption function *P()* of the execution speeds of a processor is a convex function:
  - $P(s) = C_{ef}\, V_{dd}^2\, s$, in which C_{ef} is the switch capacitance related to tasks under executions
  - $P(s) = C_{ef}\, s^3/k^2$, when $V_t = 0$
- The static power consumption comes from the leakage current
  - A constant or
  - A sub-linear function of speed s

An Example of Power Consumption Functions

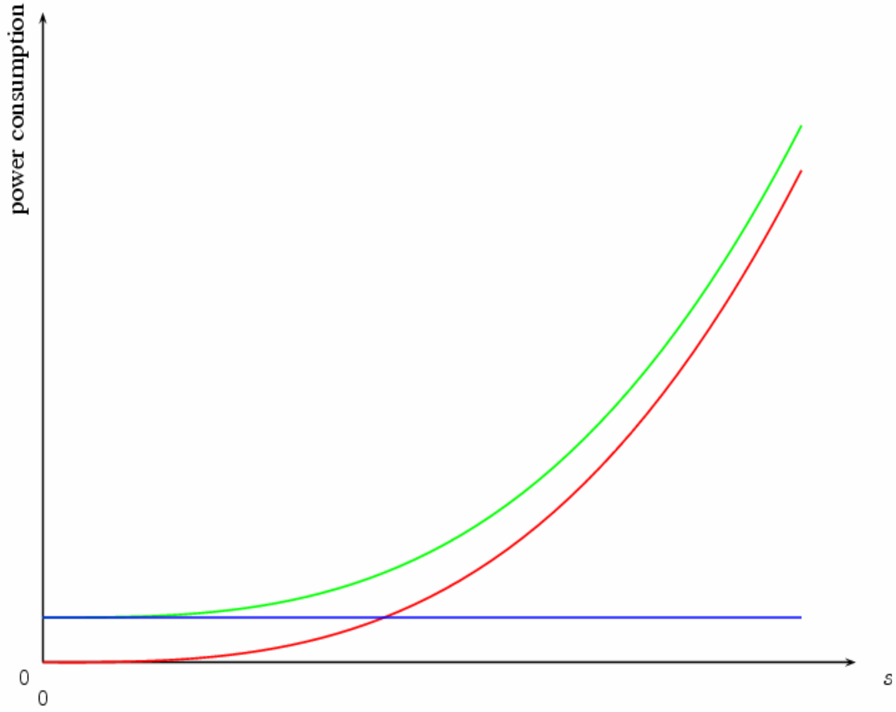


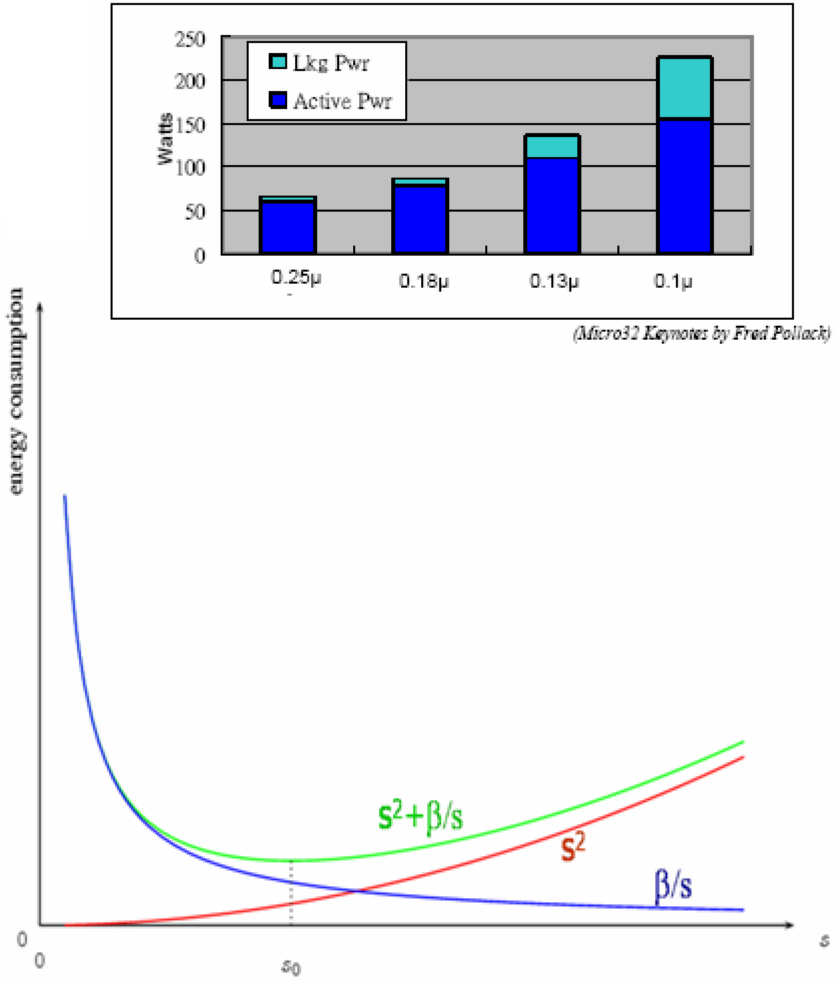
Energy Efficiency

Energy-efficient scheduling is to minimize the energy consumption while the performance index or the timing constraint is guaranteed

- To minimize the energy consumption resulting from the dynamic voltage scaling circuits
  - *Slow down the execution speed* while the timing constraints could be met
- To minimize the energy consumption resulting from the leakage current
  - *Turn the circuit off* whenever needed

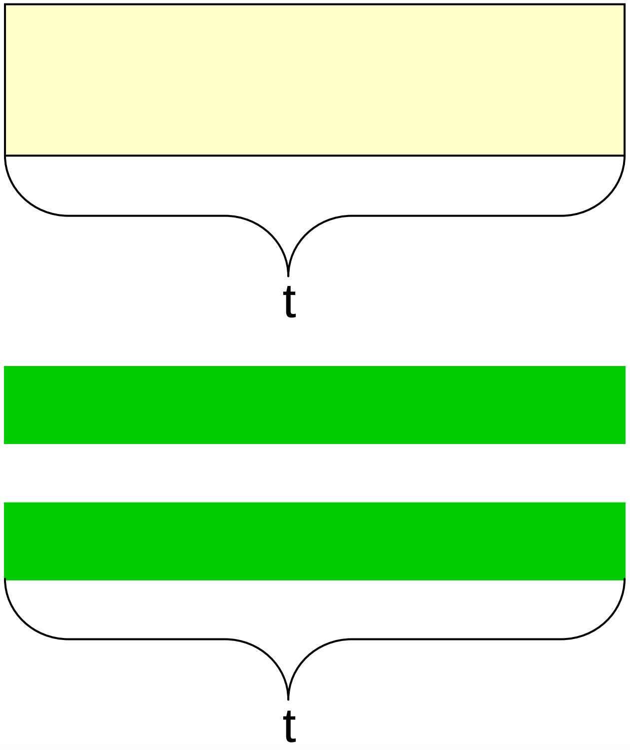
Non-Negligible Leakage Power





(Micro32 Keynotes by Fred Pollack)

Why Multiprocessor?



Case 1

Energy = $t \times s^3$

Case 2

Energy = $2t \times (0.5s)^3 = 0.25\ s^3$

Related Work

- [Gruian et al. ASP-DAC’01, Zhang et al. DAC’02]:

  Heuristic algorithms based on the well-known list-scheduling
- [Mishra et al. IPDPS’03]:

  Heuristic algorithms based on the well-known list-scheduling for tasks with precedence constraints with communication costs
- [Anderson and Baruah ICDCS’04]:

  Heuristic partition algorithms to trade the number of processors with the energy consumption
- [Aydin and Yang IPDPS’03, AlEnawy and Aydin RTAS’05]:

  Heuristic algorithms based on traditional bin packing strategies

Agenda

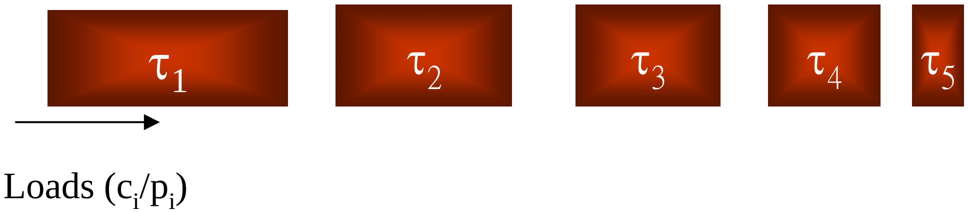
- Introduction
- Scheduling Algorithms
  - Energy-Efficient Scheduling for Homogeneous Multiprocessor Systems
    - Negligible leakage power consumption
    - Non-negligible leakage power consumption
  - Energy-Efficient Scheduling for Heterogeneous Two-Processor Systems
  - Allocation Cost Minimization for Multiprocessor Synthesis under Energy Constraints
- Conclusion and Open Issues

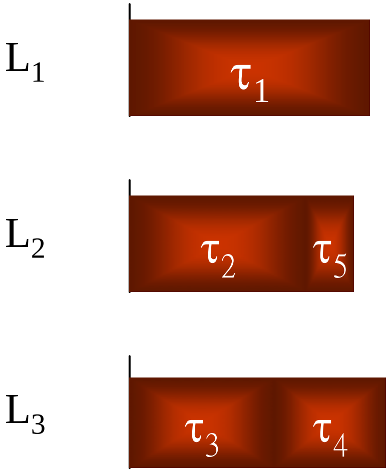
Problem Definition

- Given a set T of n periodic real-time tasks:
  - τ_i is characterized by its
    - arrival time: 0
    - computing requirement: c_i cycles
    - period: p_i
    - relative deadline: p_i
  - A homogeneous multiprocessor environment with M processors
- The objective is to derive a feasible schedule so that
  - each task is on a processor,
  - each task completes in time, and
  - the energy consumption is minimized

Algorithm Largest-Task First (LTF)

M = 3





[*Aydin et al. RTSS'01*]: EDF schedule by executing tasks at a constant speed with 100% utilization is optimal for energy-efficiency when tasks are with an identical power consumption function

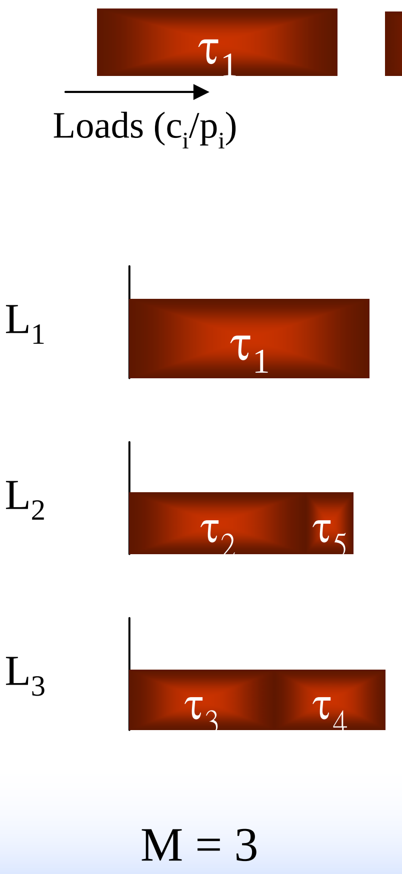
1. Sort tasks in a non-increasing order of c_i/p_i
2. Assign tasks in a greedy manner to the processor with the smallest load
3. Execute tasks on a processor at the speed with 100% utilization

Algorithm LTF is a 1.13-approximation algorithm

Jian-Jia Chen, He … Tei-Wei Kuo,
"Multiprocessor E … RTS 2004.

Jian-Jia Chen, He … heduling of
Real-Time Tasks … Multiprocessor Systems", in RTAS 2006.

Leakage-Aware Largest-Task-First (LA+LTF)



1. Sort tasks in a non-increasing order of their loads (c_i/p_i)
2. Assign tasks in a greedy manner to the processor with the smallest total estimated utilizations
3. Decide execution speeds

Algorithm LA+LTF is a 1.283-approximation algorithm when the overhead on turning processors on/off is negligible

Scheduling Scheme Non-negligible Overhead on Turning Processors on/off

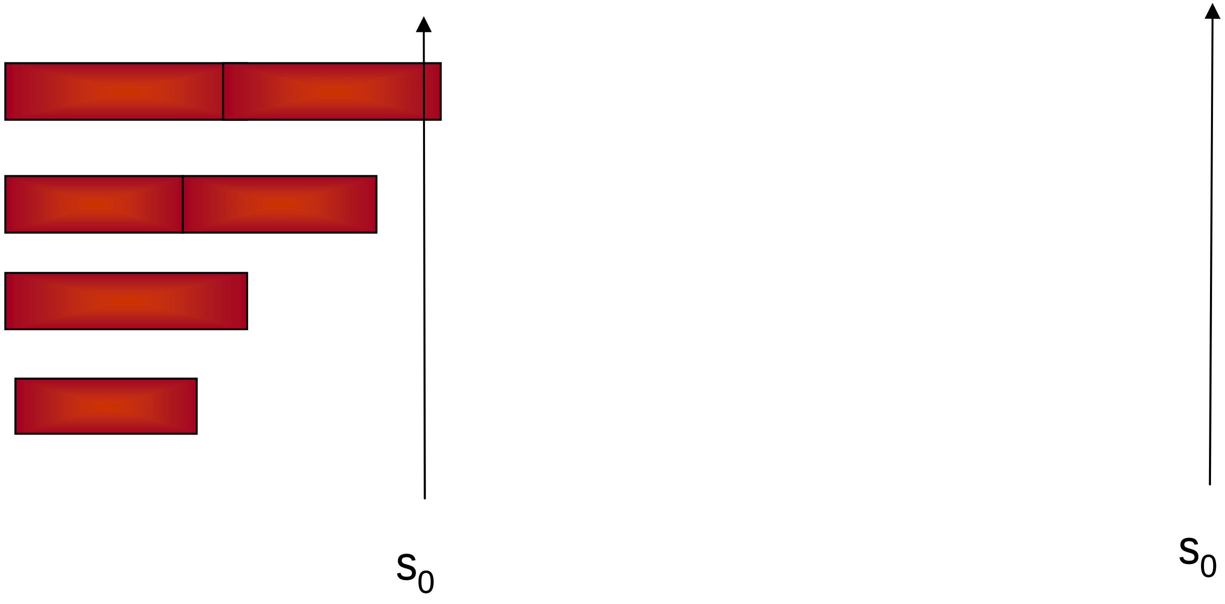
- The total load of tasks in ***T*** is no more than s_0
  - An optimal solution will execute tasks on only one processor
  - It becomes a uniprocessor scheduling problem

    Apply the 2-approximation algorithm for uniprocessor EDF scheduling strategy by Irani et al. in SODA 2003
- The total load of tasks in ***T*** is greater than s_0
  - Apply Algorithm LA+LTF for task assignment
  - Apply Algorithm FF (first-fit) for task re-assignment

Algorithm FF (First-Fit)

Execute tasks in an EDF order

When a processor is idle, idle at speed S_{min}

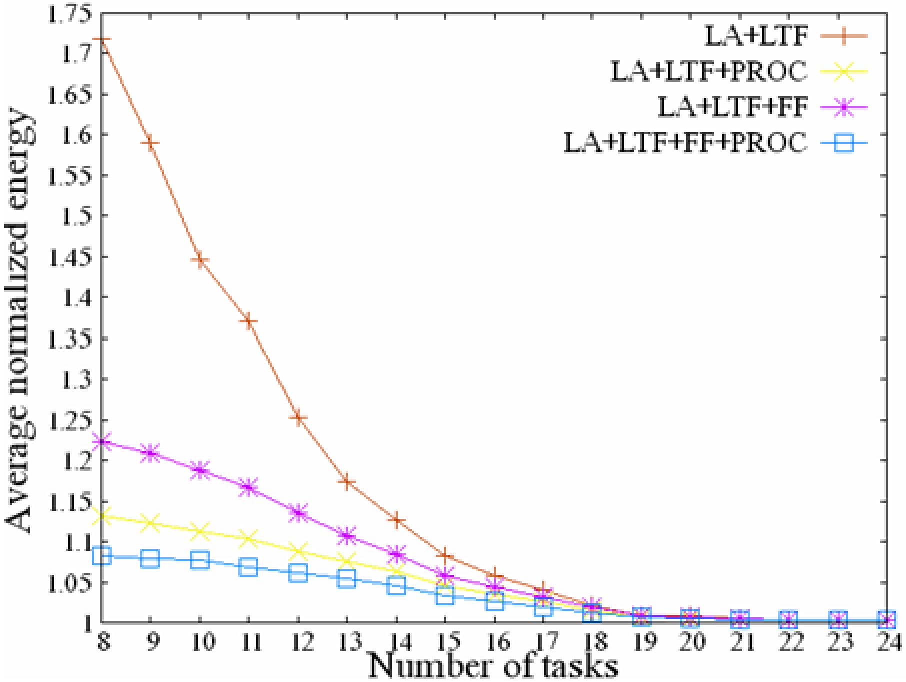


$SC_{\mathrm{LA+LTF+FF}}$

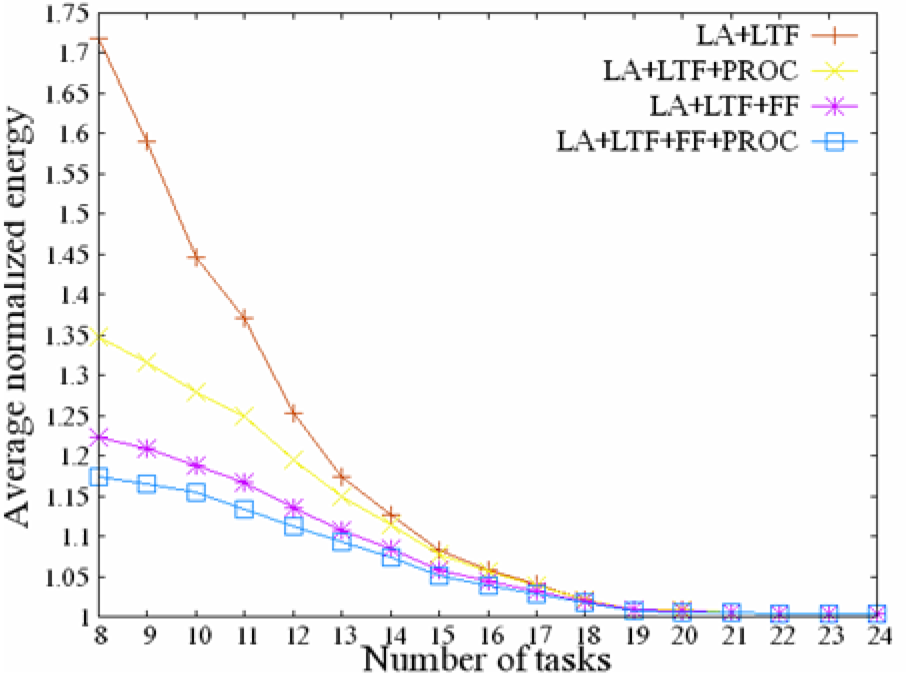
M=4

The energy consumption of $SC_{LA+LTF+FF}$ is at most twice of the optimal solution

Simulation Results



$E_{sw} = 0.1$



$E_{sw} = 0.3$

- Normalized energy: the energy consumption of the derived schedule divided by a lower bound of the input instance
- M=8

Agenda

- Introduction
- Scheduling Algorithms
  - Energy-Efficient Scheduling for Homogeneous Multiprocessor Systems
    - Negligible leakage power consumption
    - Non-negligible leakage power consumption
  - Energy-Efficient Scheduling for Heterogeneous Two-Processor Systems
  - Allocation Cost Minimization for Multiprocessor Synthesis under Energy Constraints
- Conclusion and Future Work

System Models

- Processing element models
  - DVS PE
    - Ideal PE ($S_{min} \sim S_{max}$) vs. Non-Ideal PE ($S_{min} = S_1, S_2, \ldots, S_M = S_{max}$)
    - Dynamic power consumption vs. static power consumption
    - Power consumption: $P_1(s)$
  - Non-DVS PE
    - Workload-independence vs. workload-dependence
    - A networking device or an FPGA
    - Power consumption: P_2
- Task models: a set T of n tasks
  - Period: p_i
  - Worst-case execution cycles on the DVS PE: c_i
  - Execution requirement on the non-DVS PE: u_i
  - Feasibility constraints:

$$\sum_{\tau_i \text{ on the non}-\text{DVS PE}} u_i \leq 1$$

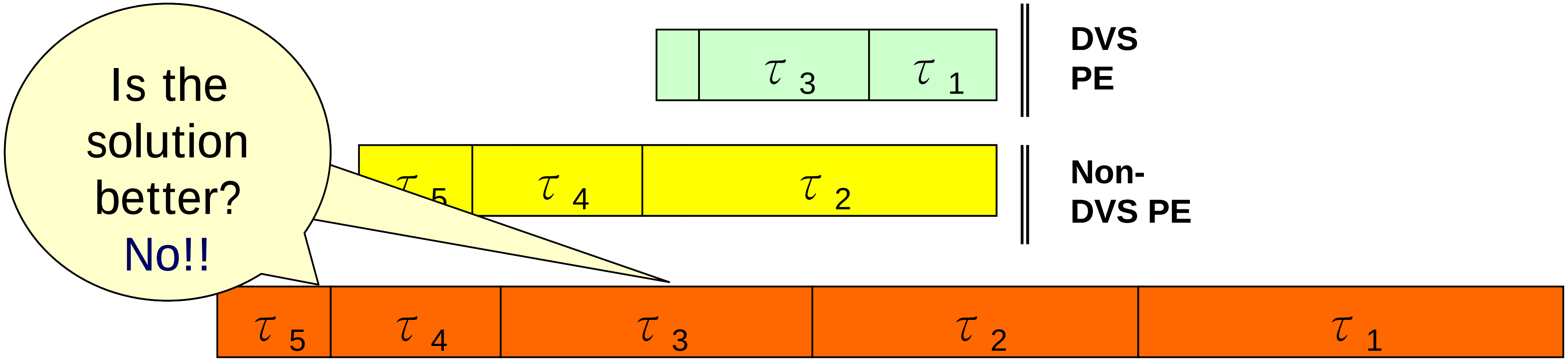
and

$$\sum_{\tau_i \text{ on the DVS PE}} \frac{c_i}{p_i} \leq S_{\max}$$

Chia-Mei Hung, Jian-Jia Chen, and Tei-Wei Kuo, "Energy-Efficient Real-Time Task Scheduling for a DVS System with a Non-DVS Processing Element", in IEEE Real-Time Systems Symposium (RTSS) 2006

Algorithm E-GREEDY

- Sort the tasks in a non-increasing order of $\dfrac{u_i}{c_i / p_i}$
- Put all the tasks on the non-DVS PE initially
- Consider tasks in the order to move to the DVS PE:



E-GREEDY: an 8-approximation algorithm

	τ_5	τ_4	τ_3	τ_2	τ_1
c_i / p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

$$\text{Minimize} \sum_{\tau_i \text{ on the DVS PE}} c_i / p_i$$

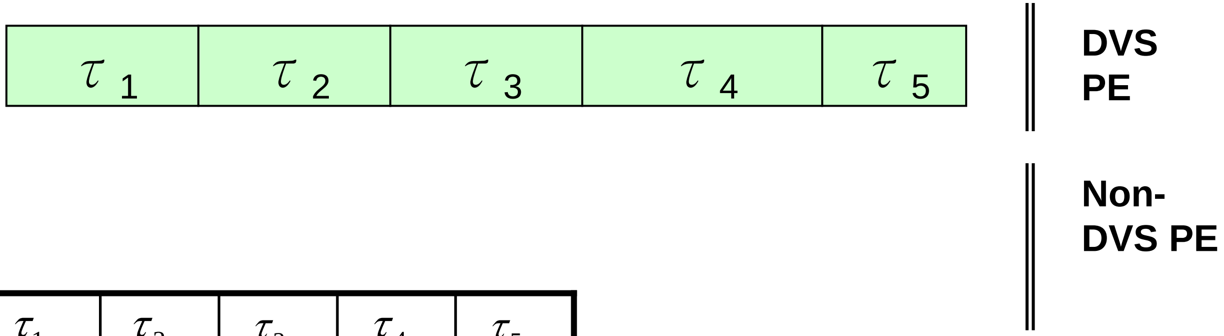
$$\text{while} \sum_{\tau_i \text{ on the DVS PE}} u_i \geq 1$$

S-GREEDY

Steps

1. Sort the tasks non-increasingly according to $\frac{c_i / p_i}{u_i}$
2. Put all the tasks on the DVS PE
3. Generate a solution (A): go through from the first task

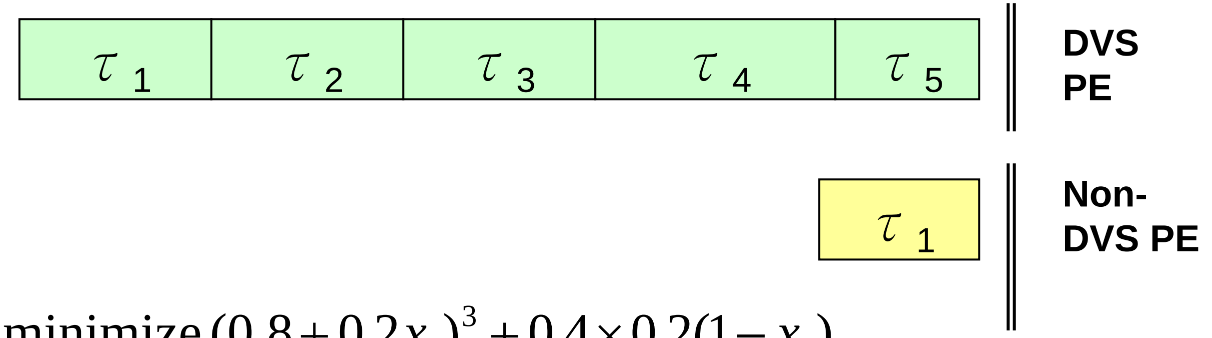
   If a task keeps saving more energy during its migration, then move it to the non-DVS PE if feasible; otherwise fix it on the DVS PE



	τ_1	τ_2	τ_3	τ_4	τ_5
c_i / p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

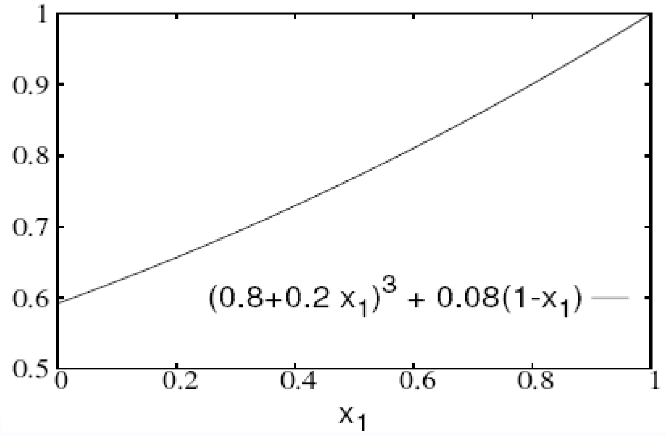
$$(P_1(s) = s^3, P_2 = 0.4, S_{\max} = 1, S_{\min} = 0)$$

S-GREEDY



$$\text{minimize } (0.8 + 0.2x_1)^3 + 0.4 \times 0.2(1 - x_1)$$

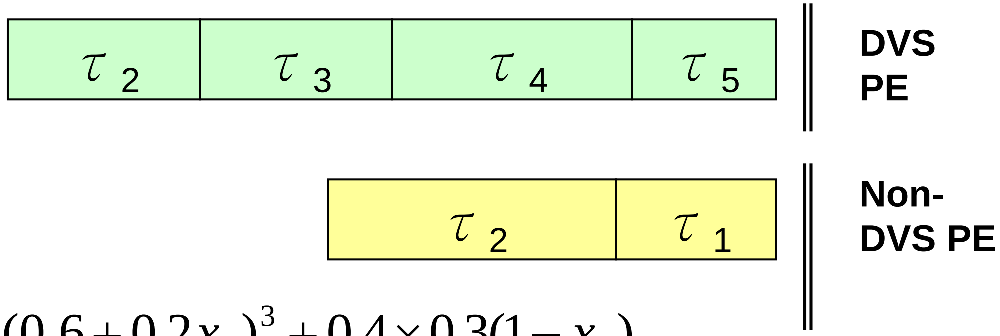
$$=> x_1 = 0$$



	τ_1	τ_2	τ_3	τ_4	τ_5
c_i / p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

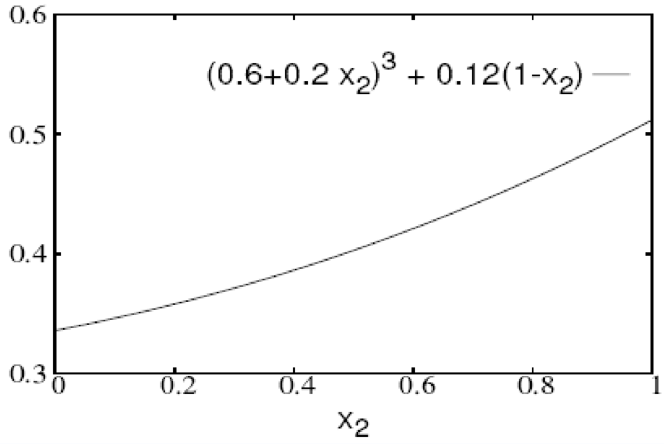
$$(P_1(s) = s^3, P_2 = 0.4, S_{\max} = 1, S_{\min} = 0)$$

S-GREEDY



$$\text{minimize}\ (0.6+0.2x_2)^3+0.4\times 0.3(1-x_2)$$

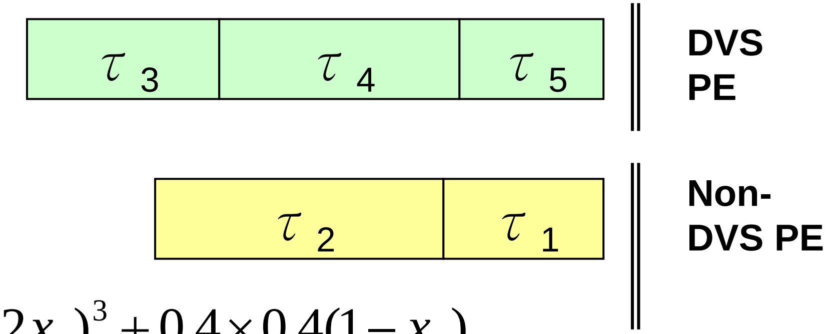
$$=> x_2 = 0$$



	τ_1	τ_2	τ_3	τ_4	τ_5
c_i/p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

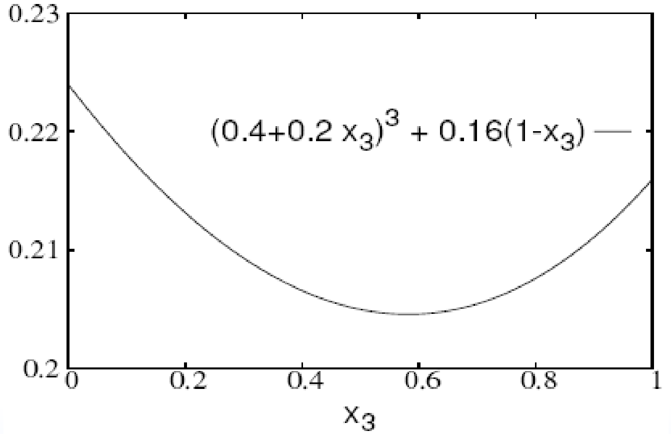
$$(P_1(s)=s^3, P_2=0.4, S_{\max}=1, S_{\min}=0)$$

S-GREEDY



$$\text{minimize}\,(0.4+0.2x_3)^3+0.4\times 0.4(1-x_3)$$

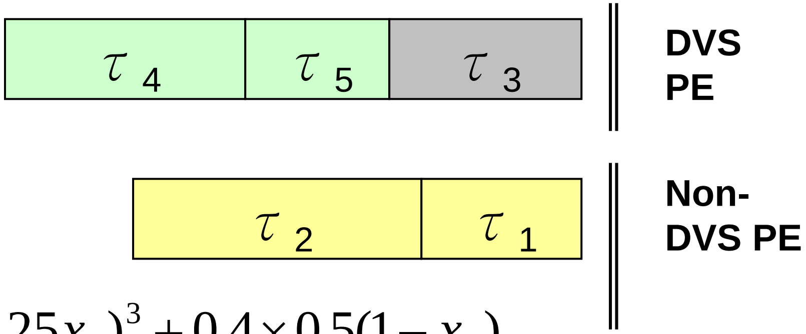
$$\Rightarrow x_3 = 0.582$$



	τ_1	τ_2	τ_3	τ_4	τ_5
c_i/p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

$$(P_1(s)=s^3,\ P_2=0.4,\ S_{\max}=1,\ S_{\min}=0)$$

S-GREEDY



τ_4 τ_5 τ_3

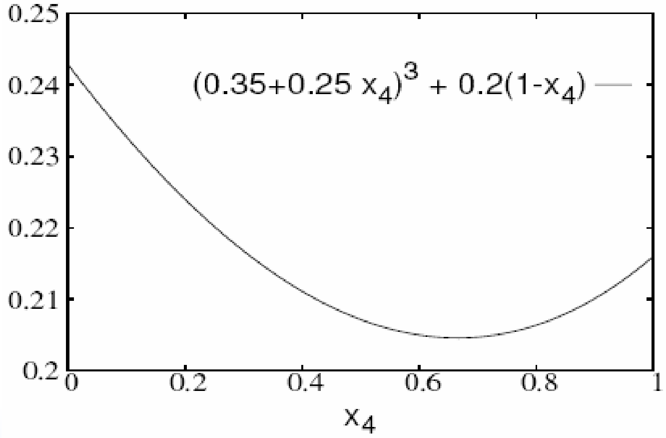
DVS PE

τ_2 τ_1

Non-DVS PE

$$\text{minimize}\ (0.35+0.25x_4)^3+0.4\times0.5(1-x_4)$$

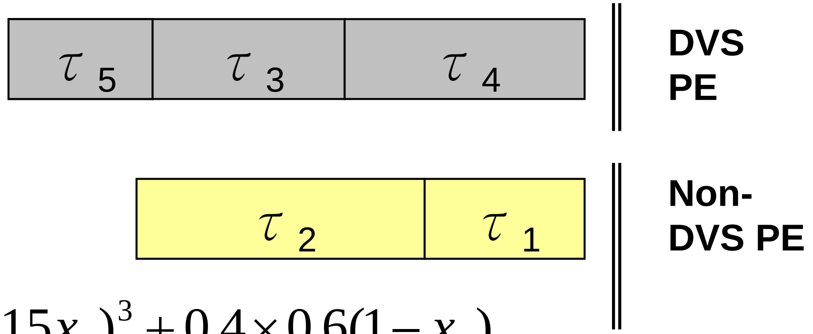
$$=> x_4=0.666$$



	τ_1	τ_2	τ_3	τ_4	τ_5
c_i/p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

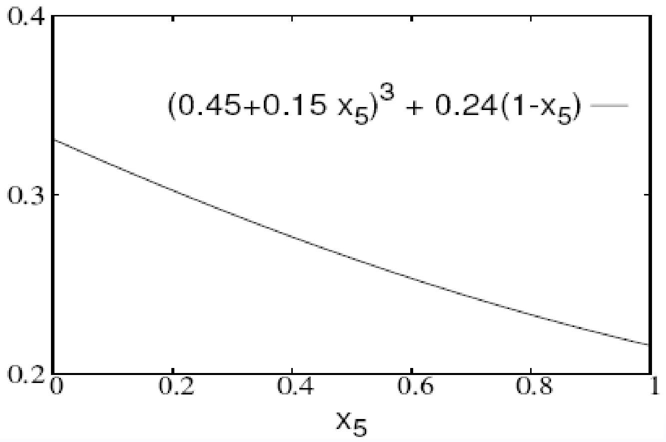
$$(P_1(s)=s^3,\ P_2=0.4,\ S_{\max}=1,\ S_{\min}=0)$$

S-GREEDY



$$\text{minimize } (0.45 + 0.15x_5)^3 + 0.4 \times 0.6(1 - x_5)$$

$$=> x_5 = 1$$



	τ_1	τ_2	τ_3	τ_4	τ_5
c_i / p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

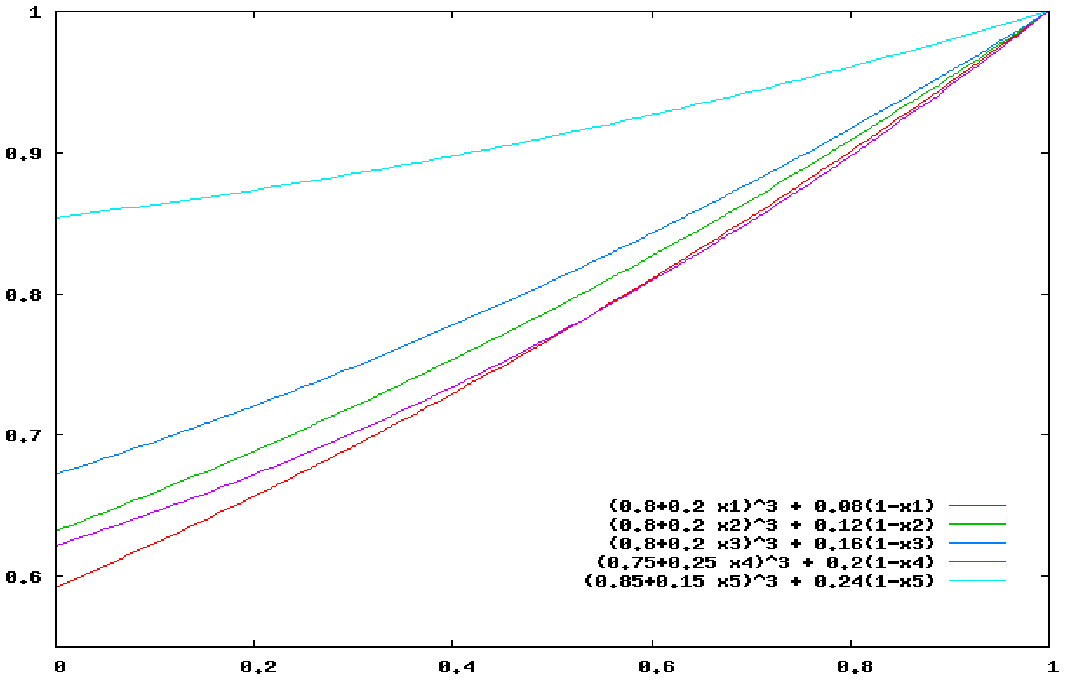
$$(P_1(s) = s^3, P_2 = 0.4, S_{\max} = 1, S_{\min} = 0)$$

S-GREEDY

Steps

1. Sort the tasks non-increasingly according to $\frac{c_i/p_i}{u_i}$
2. Put all the tasks on the DVS PE
3. Generate a solution (A): go through from the first task, if a task keeps saving more energy during its migration, then move it to the non-DVS PE; otherwise fix it on the DVS PE
4. Generate a solution (B): move the most energy-saving task to the non-DVS PE

S-GREEDY



	τ_1	τ_2	τ_3	τ_4	τ_5
c_i/p_i	0.2	0.2	0.2	0.25	0.15
u_i	0.2	0.3	0.4	0.5	0.6

Only τ_1 is moved to the non-DVS PE

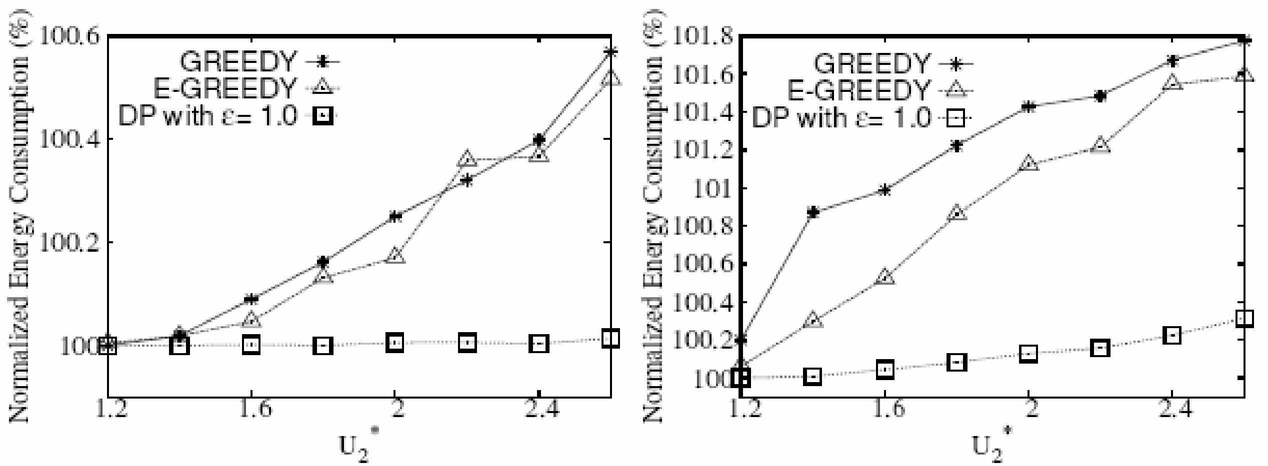
$$(P_1(s) = s^3, P_2 = 0.4, S_{\max} = 1, S_{\min} = 0)$$

S-GREEDY

- Steps of S-GREEDY
  1. Sort the tasks non-increasingly according to $\frac{c_i/p_i}{u_i}$
  2. Put all the tasks on the DVS PE
  3. Generate a solution (A): go through from the first task, if a task keeps saving more energy during its migration, then move it to the non-DVS PE; otherwise fix it on the DVS PE
  4. Generate a solution (B): move the most energy-saving task to the non-DVS PE
  5. *Return the better solution between (A) and (B)*
- A 0.5-approximation ratio

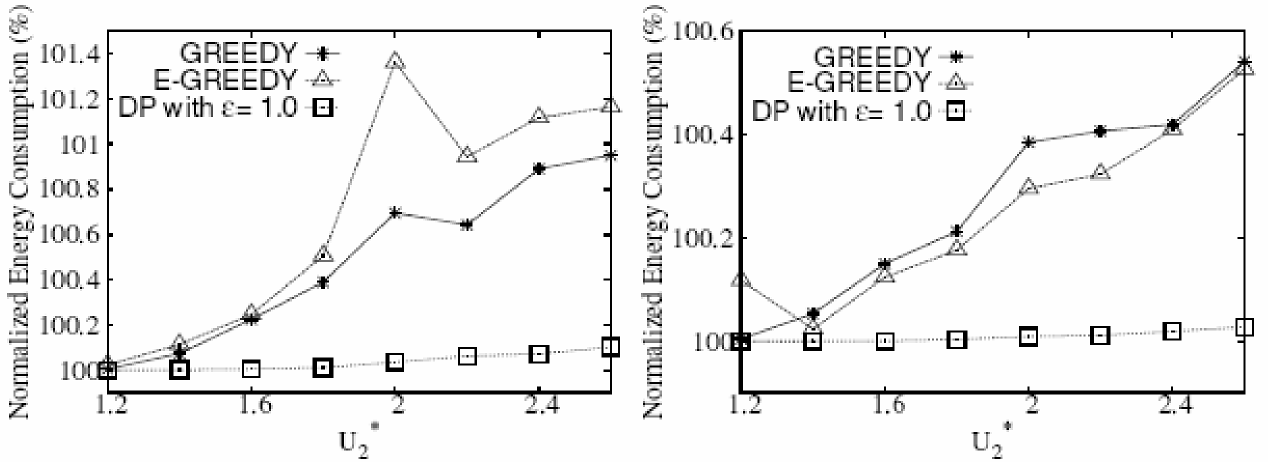
Evaluation Results (1)

Non-Ideal DVS PE & workload-independent non-DVS PE:



(a) inverse

(b) proportional



(c) independent

(d) 50% inverse, 50% independent

n=10,

P_2=558mW

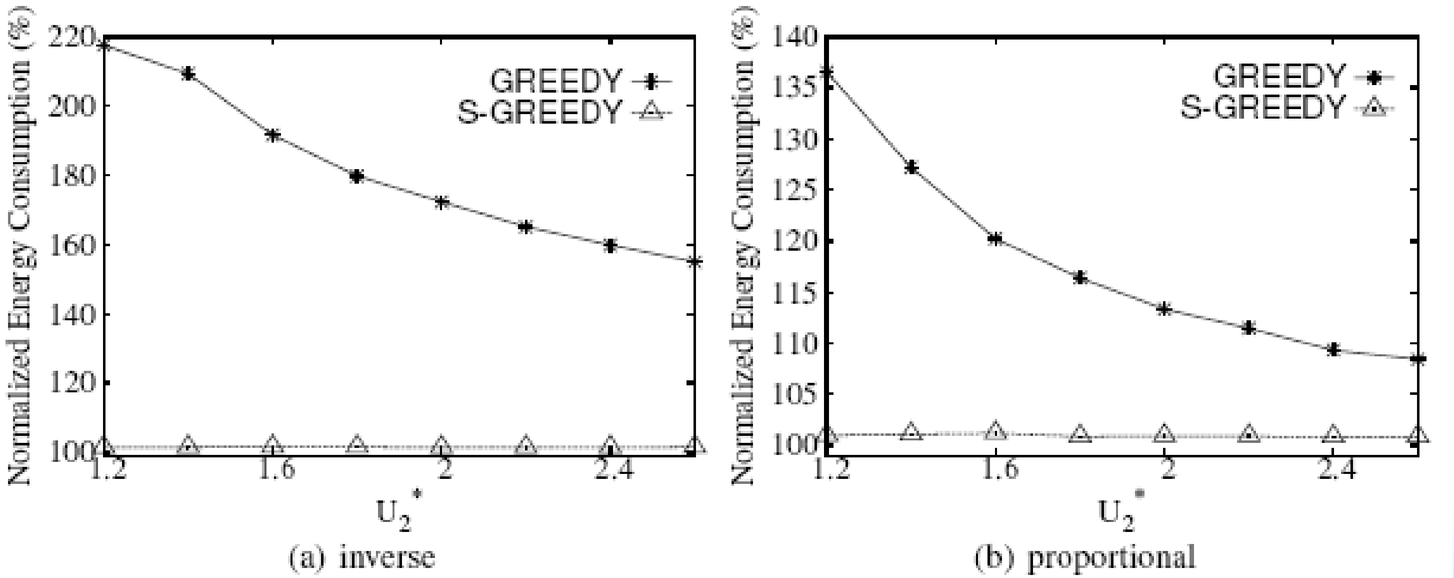
U_1^*=1

ε = 1

Evaluation Results (2)

Ideal DVS PE & workload-dependent non-DVS PE:

n=10, P_2=558mW, U_1^*=1



(a) inverse

(b) proportional

Agenda

- Introduction
- Scheduling Algorithms
  - Energy-Efficient Scheduling for Homogeneous Multiprocessor Systems
    - Negligible leakage power consumption
    - Non-negligible leakage power consumption
  - Energy-Efficient Scheduling for Heterogeneous Two-Processor Systems
  - Allocation Cost Minimization for Multiprocessor Synthesis under Energy Constraints
- Conclusion and Open Issues

Problem Definition

- Input
  - m processor types: M_j with cost C_j , $j = 1 .. m$
  - n independent periodic real-time tasks:
    - Period of task τ_i: p_i
    - Relative deadline of task τ_i: p_i
    - Required cycles of a task instance of task τ_i at M_j : $c_{i,j}$
  - An energy budget in the hyper-period L of tasks: E_{budget}
- Output
  - Select a multisubset of these m processor types
  - Assign each task to one allocated processor
  - Determine execution speeds of tasks
  - Consume no more than E_{budget} in energy
  - Minimize the allocation cost of allocated processors

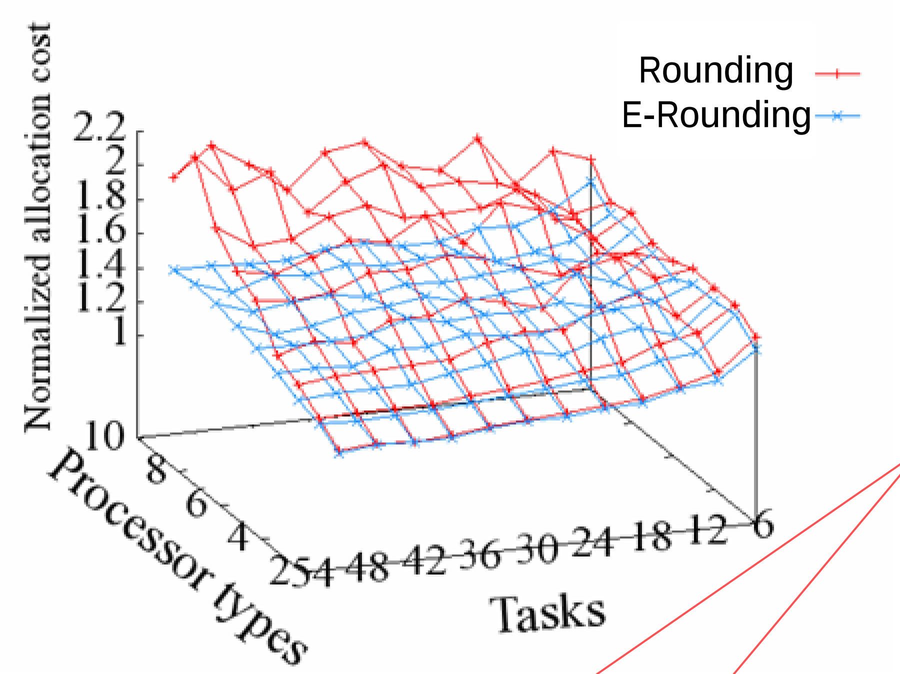
J.-J. Chen and T.-W. Kuo. Allocation cost minimization for periodic hard real-time tasks in energy-constrained DVS systems. In *ICCAD* 2006.

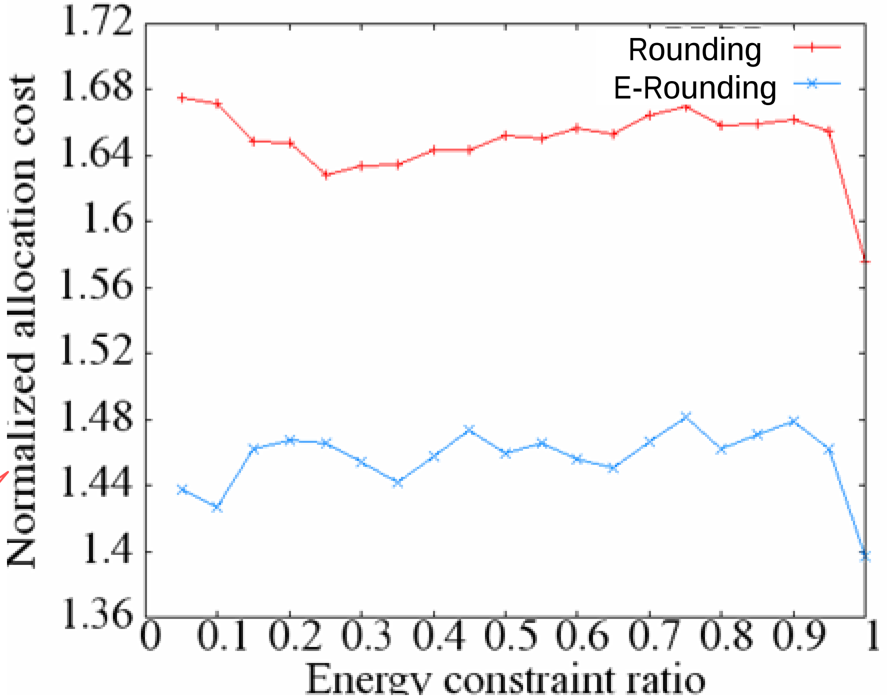
Approaches for Non-Ideal Processors

Formulate the problem into an integer linear programming problem

- Relax the problem into a naïve linear programming (LP) problem → unbounded in worst cases
- Relax the problem into m linear programming
  - Sort processor types from cheap to expensive ones
  - Restrict no processor type with index larger than j is used for each j=1,2,…,m
- Find the solution with the minimum objective value among the m linear programming relaxations with feasible solutions
  - Algorithm ROUNDING
    - Assign tasks onto processors based on the optimal LP solution
    - Have (m+2)-approximation
  - Algorithm E-ROUNDING

Evaluation Results



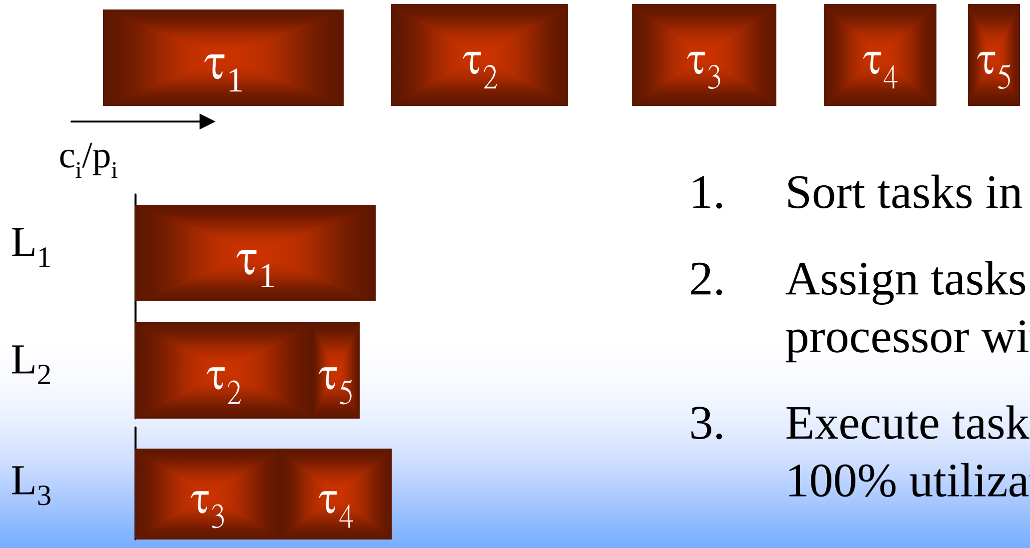


The allocation cost of the derived solution is normalized to a lower bound

Energy constraint is E_{min} + (E_{max} – E_{min})*(constraint ratio)

Approaches for Ideal Processors

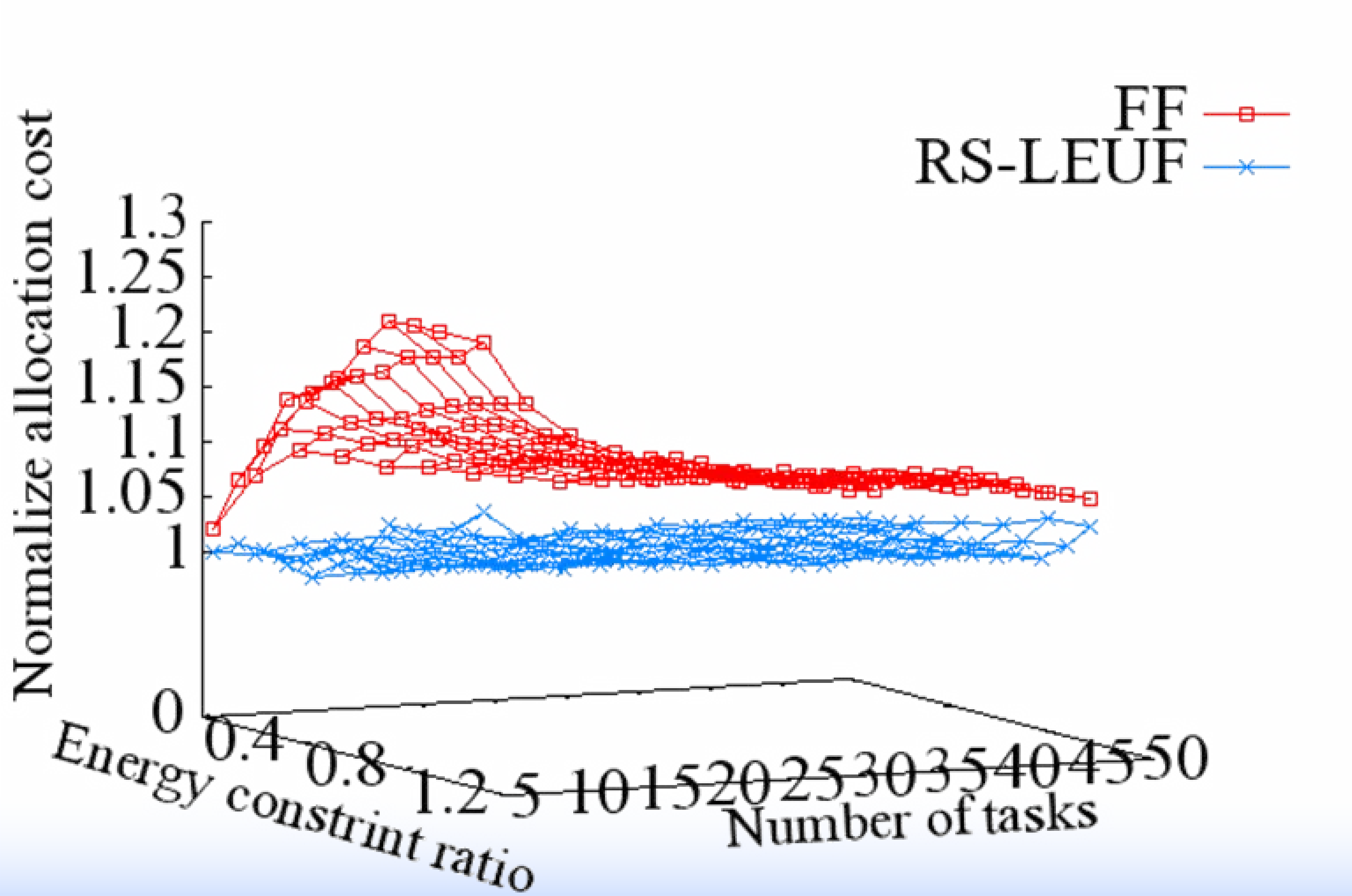
- One processor type:
  - Incrementally allocate processors until the energy consumption is satisfied
  - At most use $(1.189)m^* + 1$ processors, where m^* is the optimal cost
- Multiple processor types:
  - Determine the (virtual) discrete speeds of processors manually
  - Apply algorithms for non-ideal processors to assign tasks and energy constraint on each processor type
  - Adopt the incremental approach to allocate processors in each processor type



$M = 3$

1. Sort tasks in a non-increasing order of c_i/p_i
2. Assign tasks in a greedy manner to the processor with the smallest load
3. Execute tasks on a processor at the speed with 100% utilization

Evaluation Results



Summary

- Energy-Efficient Multiprocessor Scheduling
  - Homogeneous multiprocessor systems
    - Negligible-leakage: A 1.13-approximation algorithm for homogeneous systems
    - Non-negligible leakage
      - A 1.283-approximation algorithm with negligible overhead in turning on/off processors
      - 2-approximation algorithms for the other case
  - Heterogeneous multiprocessor systems
    - Approximation algorithms for systems with two processors
- Energy-Constrained DVS Synthesis
  - Approximation algorithms based on parametric linear programming relaxations
    - Ideal processor types
    - Non-ideal processor types

Additional Material

- Chip-multiprocessor (CMP) architecture [Yang, Chen, and Kuo in DATE 2005]
- Multiprocessor energy-efficient scheduling for tasks with different power characteristics [Chen and Kuo in ICPP 2005]
- Energy-efficient slack reclamation for multiprocessor systems [Chen, Yang, and Kuo in SUTC 2006]
- Energy-efficient scheduling with task rejection [Chen, Kuo, Yang, and King in DATE 2007]
- Multiprocessor instantaneous temperature minimization [Chen, Hung and Kuo in RTAS 2007]

Open Issues

- Energy-Efficient Scheduling for
  - DVS systems and I/O devices
  - Heterogeneous multiprocessors
  - Tasks with precedence constraints
  - Tasks with uncertain execution paths
- Energy-Aware Synthesis
  - Multi-dimensional floor-planning
  - Tasks with precedence constraints
- Thermal-Aware Issues
  - Thermal-constrained scheduling
  - Thermal-constrained synthesis

Questions and Suggestions?