Power-Aware Scheduling and Voltage Setting for Tasks Running on a Hard Real-Time System

Peng Rong and Massoud Pedram
Department of Electrical Engineering
University of Southern California

2006 ASP-DAC

Outline

- Introduction
- Problem Formulation
- Power Aware Scheduling
- Voltage Assignment
- Refinement
- Experimental Result

Introduction

- Dynamic Voltage Scaling (DVS)
  - Slow down underutilized resources and decrease their operating voltages
  - Effective for reducing power consumption of the CPU
- Dynamic Power Management (DPM)
  - Selective shut-off of idle system components
  - Effective for reducing power consumption of the CPU, peripheral components, and I/O devices
- Integrated DVS and DPM
  - Idle intervals are created by the scheduler which decides when and how fast to perform the tasks
  - May be beneficial to speed up task processing in order to create an idle period in which the system can sleep

Related Work

- Low Power Scheduling for DVS
  - List-scheduling algorithm
    - Task priority: reciprocal of the slack time [Luo, 2002]
    - High priority: high energy saving and low slack time [Gruian, 2001]
  - Early-deadline-first scheduling to assign a task to the best-fit processor [Zhang, 2002]
- DPM-based Task Scheduling
  - Online algorithm to reduce the number of on/off transitions of devices [Lu, 2000]
  - Offline branch-and-bound algorithm [Swaminathan, 2002]
- Approaches for integrated DVS and DPM
  - Stochastic Approach: CTMDP [Qinru, 1999], Renewal Theory with DVS [Simunic, 2001]
  - Competitive analysis based algorithm: 3-competitive for a processor with one sleep state [Irani, 2003]

Problem Description

- Targeted System
  - A real-time system having a single CPU and ***K*** other devices, e.g., I/O and memory
    - CPU (device 0) can operate at a number of ***m*** different voltage levels and supports power-down modes
    - Devices have only one active mode and support at least one low power mode
  - A set of ***n*** non-preemptive dependent tasks, periodically run on the system with a hard deadline; Task dependency captured by a DAG, **G(V,E)**
  - Every task has to be performed on the CPU, but may require support from only some of the devices
- Objective
  - Determine the optimal task scheduling and task-level voltage assignment that minimize the total system energy consumption while meeting the hard real-time constraint

Problem Formulation

- Energy consumption for execution of task *u*

$$ene_u = c_u \sum_{i=1}^{m} x(u,i) \cdot N_{u,i} \cdot V_i^2 + \sum_{k=1}^{K} Z_k(u) \cdot P_k \cdot dur_u$$

$$dur_u = \sum_{i=1}^{m} \frac{x(u,i) N_{u,i}}{f_i}$$

C_u : Switched capacitance per clock cycle

f_i : Clock frequency of the CPU at voltage V_i

P_k : Power dissipation of device k in active mode

- Precedence constraints

$$s(u) + dur_u \leq s(v) \quad \forall e(u,v) \in E$$

$x(u, i)$: Percentage of the workload of task *u* that is performed at CPU voltage V_i

$N_{u,I}$: Actual number of CPU cycles required to complete task *u* at operating voltage V_i

$Z_k(u)$: 0-1 integer variable; 1 iff task *u* requires service from device k

$s(u)$: Start time of task *u*

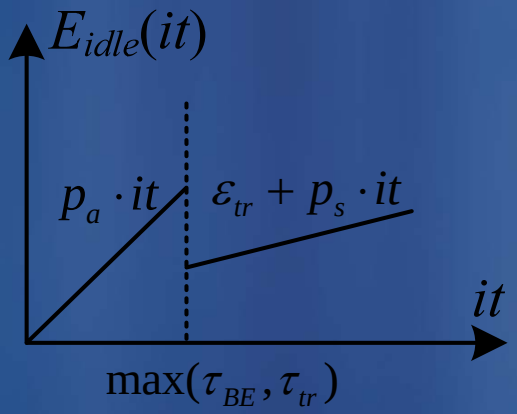
$Y_k(u, v)$: 0-1 integer variable; it is set to 1 iff task u is executed immediately before task *v* on device *k*

Problem Formulation (Cont'd)

- Energy consumption when device *k* is idle

$$idlene_k(it) = \begin{cases} \varepsilon_{tr} + p_s \cdot it, & it \geq \max(\tau_{BE}, \tau_{tr}) \\ p_a \cdot it, & \text{otherwise} \end{cases}$$

$$\tau_{BE} = \varepsilon_{tr} / (p_a - p_s)$$



- Idle time of device *k* just before task *v* is executed

$$it_{k,v} = s(v) - \sum_{u=0}^{n} (s(u) + dur_u) \cdot Y_k(u,v)$$

Problem Formulation (Cont'd)

- Total system energy consumption (ignoring the boundary intervals for each data instantiation period)

$$E_{sys} = \sum_{u=1}^{n} ene_u + \sum_{k=0}^{K} \sum_{v=1}^{n} idlene_k (Z_k(v) \cdot it_{k,v})$$

- Precedence constraint between a task and its immediate successor

$$\sum_{u=0}^{n} (s(u) + dur_u) \cdot Y_k(u,v) \leq s(v) \quad \forall v \in V, k \in devs_v$$

Solution Approach

- This problem formulation is a nonlinear non-convex integer program over variables $Y_k(u,v)$, $x(u,i)$, and $s(u)$
- Three-step heuristic approach
  - **Task Ordering**: Derive a linear ordering of tasks (i.e., calculate $Y_k(u,v)$ values) by performing an interactive minimum-cost matching on some appropriately constructed graph
  - **Voltage Assignment**: Given the task ordering implied by the schedule obtained in step 1, assign voltages and task durations (i.e., calculate $x(u,i)$ values) and exact start times (i.e., calculate $s(u)$ values) to each task so as to meet a target cycle time, T_d, for data initiation
  - **Solution Refinement**: Improve the task scheduling and voltage assignment of steps 1 and 2 to increase the energy efficiency of the resulting solutions

Task Ordering

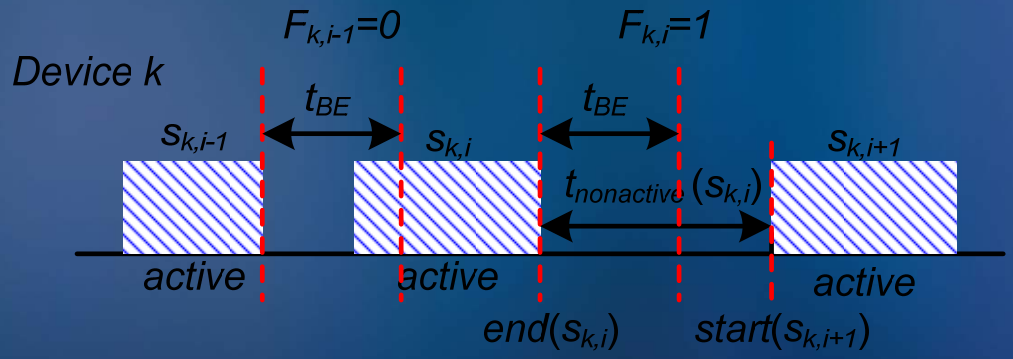
- Goal is to schedule the task graph with known task execution times on the CPU so as to minimize the total energy dissipation by maximizing the I/O device sleep times while accounting for the energy transition cost of devices going from ON to OFF states and vice versa
  - Notice that the summation of energy dissipations in all devices when these devices are in active states is fixed and independent of the task ordering. The ordering only changes the duration of the idle times and the number of ON to OFF transitions for the I/O devices

System Energy for Known Task Execution Schedule, Λ

- Lower-bound on the total system energy

$$totene_{LB} = \sum_{u=1}^{n} \sum_{k \in dev_u} funcpow_k \cdot dur_u + \sum_{k=0}^{\kappa} sleepow_k \cdot (T_d - \sum_{u \in tasks_k} dur_u)$$

- Total energy computation based on segment set representation of device activity, S_k



Let $tasks_k$ denote the set of tasks running on device k and dev_u denote the set of devices that are required by task u

$$t_{nonactive}(s_{k,i}) \equiv start(s_{k,i+1}) - end(s_{k,i})$$

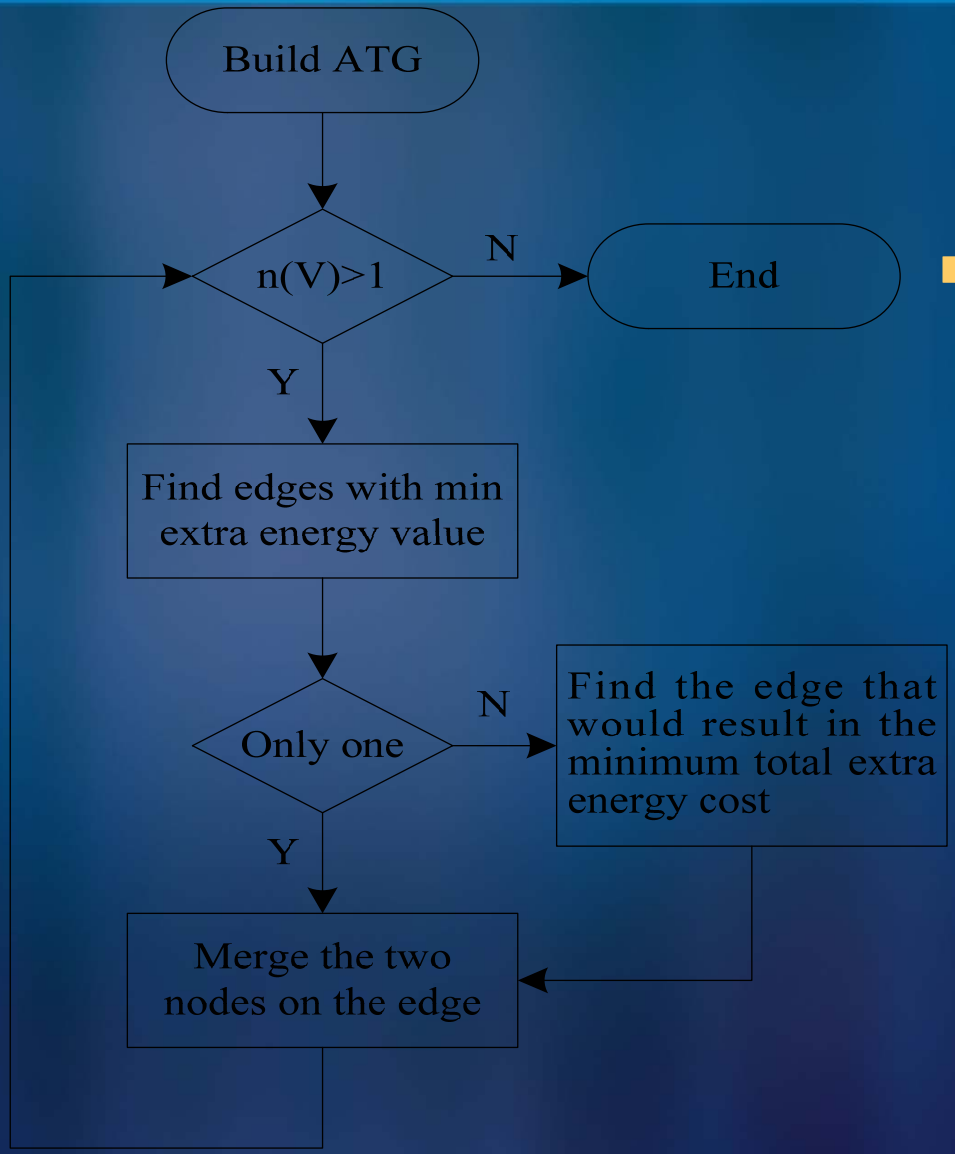
$$F_{k,i} \equiv F_k(s_{k,i}, s_{k,i+1}) = 1 \quad \text{iff} \quad t_{nonactive}(s_{k,i}, s_{k,i+1}) \geq t_{BE,k}$$

$$totene(\Lambda) = totene_{LB} + \sum_{k=1}^{\kappa} \sum_{i=1}^{|S_k|-1} (F_{k,i} \cdot transene_k + (1 - F_{k,i}) \cdot funcpow_k \cdot t_{nonactive}(s_{k,i}))$$

Augmented Task Graph (ATG)

- We construct an augmented task graph A(V,C) from the given task graph G(V,E) by copying G(V,E) and adding/deleting some edges to/from E
  - The new edge set *C* has the following properties:
    - It does NOT contain any directed edge *uv* if there exists another directed path from *u* to *v*
    - It contains undirected edges *qr* if tasks associated with *q* and *r* can be scheduled next to each other in some order
    - Each directed edge *qr* has an associated energy cost that captures the energy consumption due to devices staying in their ON state or device state transitions that occur when the two end nodes of the edge are executed in a sequence
  - Each node *q* in *V* has three attributes: task execution time dur_q, task energy consumption ene_q; device list required by the task dev_q

Scheduling Algorithm



- Node Merge Operation
  - Merge two nodes into a single node and remove the edge between these two nodes
  - Generate properties for the new node
  - Remove all edges that become invalid after this merge

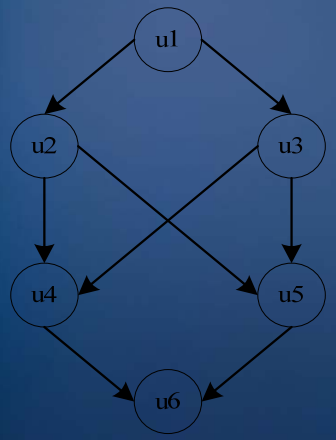
An Example

Example: Consider a task graph depicted on the left. Assume that there are four devices {0,1,2,3} with the following device utilization sets. Each device consumes 1 unit of energy for each transition to and from the sleep state

$dev(u1) = \{0\}$

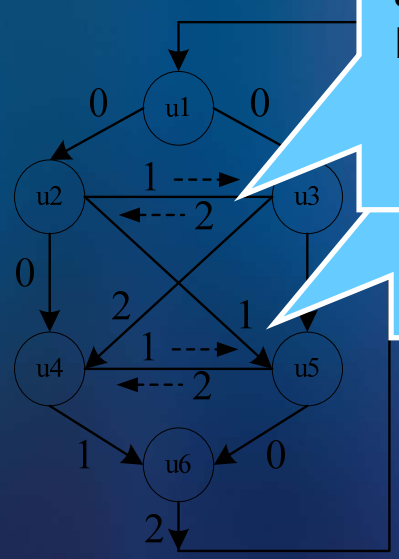
$dev(u2) = dev(u4) = \{0,1\}$

$dev(u3) = dev(u5) = dev(u6) = \{0,2,3\}$



Task Graph

Gen ATG

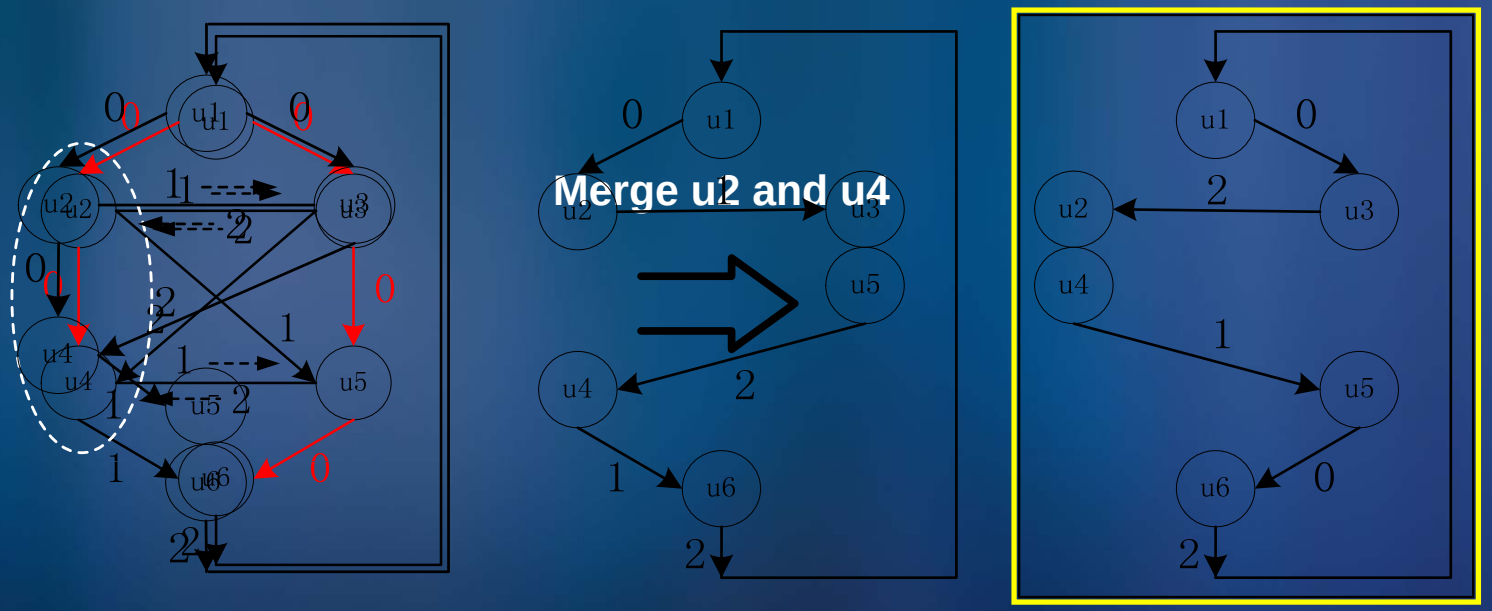


Undirected edge added between u2 and u3, because they can be scheduled next to each other in any order

sleep; the extra energy cost is 1

An Example (Cont'd)

- Select a pair of nodes to merge
  - There are five edges having the minimum *extraene* value: 0
  - We focus on three of the edge merges here: u_2u_4, u_3u_5, and u_5u_6
  - Pick the edge that results in minimum total extra energy for the ATG



From left to right, the total extra energy costs are 5.5, 6 and 5
Select the u_2u_4 edge

Voltage Assignment as an ILP

- Having generated the task schedule, we fix the ordering of tasks, but ignore task execution times and start times
  - Even with a fixed task ordering, the optimization problem cannot be solved efficiently due to the non-convex nature of device idle energy over time
- Our approach is to define a set of idle time ranges and map the actual idle time to one of these ranges
  - Define new 0-1 integer variables $W_k(v,h)$ to approximate idle time, $it_{v,k}$; Note that $W_k(v,h)$ is 1 iff $t_{k,h} \leq it_{k,v} < t_{k,h+1}$. $t_{k,h}$ and $t_{k,h+1}$ are chosen from a discrete set $\{ t_{k,1}, t_{k,2}, \ldots, t_{k,H} \}$

$$idlene_k(it_{v,k}) = \sum_{h=1}^{H} W_k(v,h) \cdot idlene_k(t_{k,h}) \quad \text{with} \quad \sum_{h=1}^{H-1} W_k(v,h) = 1$$

  - Precedence constraint between a task, u, and its immediate successor, v, becomes

$$\sum_{h=1}^{H-1} t_{k,h} W_k(v,h) \leq s(v) - \sum_{u=0}^{n} (s(u) + dur_u) \cdot Y_k(u,v) \leq \sum_{h=1}^{H-1} t_{k,h+1} W_k(v,h)$$

Solution Refinement

1. Move tasks closer together so as to minimize unnecessary positive slack times
2. Identify a task whose duration has a large impact on the system energy dissipation, e.g., a small change of its execution time can enable device transitions to low power states, and change its voltage assignment accordingly
3. Goto step 2 until no further improvement is possible

Experimental Setup

- The CPU operating voltage/frequency levels: 1V/200MHz, 1.1V/300MHz and 1.3V/400MHz
- Power and transition parameters

Device	Active Power	Sleep Power	Energy Overhead	Timing Overhead
SDRAM	0.3W	~0	~0	~0
HDD	2.1W	0.85W	0.6J	400ms
WLAN	0.7W	0.05W	0.04J	100ms
CPU	1.0W (200MHz)	0.05W	0.3J	400ms

Characteristics of Task Graphs

Task Graph	Task Count	CPU and device utilization factors at max CPU speed CPU; SDRAM; HDD; WLAN
G1	28	0.61 ; 0.61 ; 0.29 ; 0.42
G2	65	0.72 ; 0.72 ; 0.51 ; 0.39
G3	110	0.34 ; 0.34 ; 0.12 ; 0.23
G4	159	0.48 ; 0.48 ; 0.30 ; 0.25
G5	204	0.55 ; 0.55 ; 0.28 ; 0.36

Approaches Compared

- **M1**: No DVS, no DPM
  - The CPU always operates at the highest voltage level and devices are kept active during the whole execution time.
- **M2**: DPM without any task scheduling
  - Tasks are executed on the CPU (which has assumed its highest frequency and voltage setting) in an un-optimized order based on their ID numbers after they become available. The state transition sequences of all devices and the CPU are determined optimally.
- **M3**: DPM with task scheduling
  - This method is similar to M2, except that our proposed power-aware task scheduling algorithm is used to determine the task execution sequence.
- **M4**: Conventional cpu-driven DVS plus DPM
  - Similar to M2, except that the task operating voltage is assigned to minimize the CPU power consumption. More specifically, the operating voltage setting for each task is obtained without considering the energy consumption of devices.
- **M5**: Proposed system-aware DVS plus DPM (PSVS)
  - Task scheduling and operating voltage settings are determined through the proposed three-phase framework.

Experiment Results

- Energy consumption comparisons of different approaches (normalized as a ratio over M1 results)

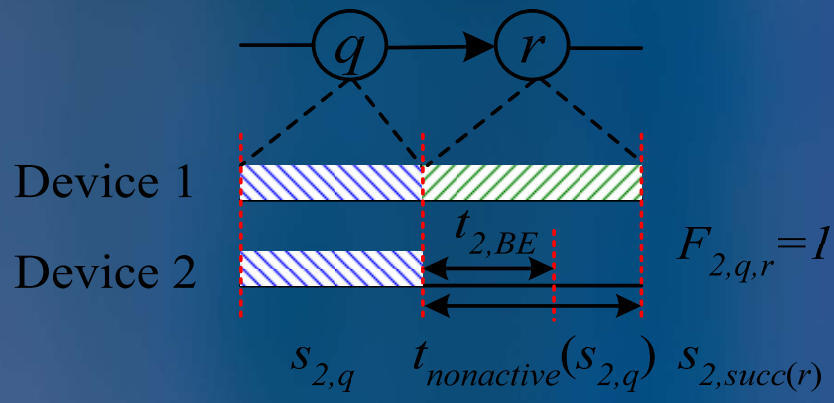
Task Graph	M2	M3	M4	M5
G1	0.54	0.50	0.58	0.47
G2	0.67	0.59	0.63	0.53
G3	0.28	0.26	0.32	0.25
G4	0.40	0.35	0.42	0.34
G5	0.43	0.37	0.39	0.33

Conclusion

- We addressed the problem of minimizing energy consumption of a computer system performing periodic hard real-time tasks with precedence constraints
- In our approach, dynamic power management and voltage scaling techniques were combined to reduce the energy consumption of the CPU and devices
- The optimization problem was first formulated as an integer non-convex programming problem. Next, a three-phase solution framework, which integrates power management scheduling and task voltage assignment, was presented
- Future work will focus on developing a more efficient algorithm for the voltage assignment step

Backup Slides

Edge Energy Cost Calculation



$$s_{k,q} \in [start(r) - dur_q, start(r)]; \quad s_{k,succ(r)} \in [start(r) + dur_r, T_d]$$

$$F_{k,q,r} \in F_k(s_{k,q}, s_{k,succ(r)})$$

$$extraene_{qr} = \sum_{k \in dev_q - dev_r} \sum_{i=1}^{|S_k|-1} (F_{k,q,r} \cdot transene_k + (1 - F_{k,q,r}) \cdot funcpow_k \cdot t_{nonactive}(s_{k,i}))$$