

Energy Aware Real Time Systems

Jennifer L. Wong
CSE591 Spring '09

1

Energy is Critical Resource!

- Task execution uses significant energy often subject to stringent timing constraints
- Reduce power by....
 - Hardware:
 - Limit parallelism & speculative execution
 - Improve circuit technology
 - Software
 - Perform fewer computations
 - Improve algorithms and mechanisms

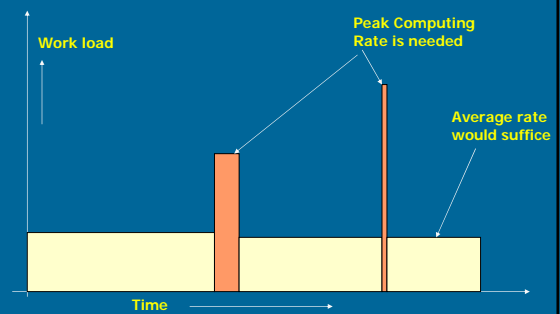
2

Important Facts

- The peak computing rate needed is much higher than the average throughput that must be sustained;
- High performance is needed only for a small fraction of time, while for the rest of time, a low-performance, a low-power processor would suffice.

3

Workload Profile



4

Variable Voltage Processors

- Modern processors operate at multiple frequency levels.
 - Crusoe Processor: Transmeta Corporation
 - PowerNow! Technology: AMD
 - Intel XScale: Intel
- Higher the frequency level higher the energy consumption

5

Crusoe processor

- What is the power consumption of a Crusoe processor? [1]

The extremely low power consumption delivered on multimedia applications can be directly attributed to a new feature called LongRun power management. **LongRun has the distinct ability to analyze the application workload dynamically and to adjust continuously the processor's speed (MHz) and voltage to provide the necessary performance.** This new feature promises to extend the battery life of all applications, most specifically those requiring the constant attention of the processor. This is a dramatic departure from today's ultra-light PCs, which are incapable of delivering over one and a half or two hours of runtime for DVD movies.

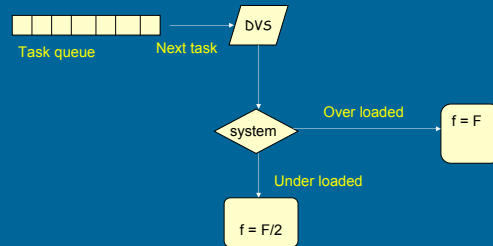
6

Dynamic Voltage Scaling (DVS)

- DVS scales the operating voltage of the processor along with the frequency.
- Since energy is proportional to f^2 , DVS can potentially provide significant energy savings through frequency and voltage scaling.

7

Simple DVS-Scheme



8

DVS-example

- Consider a task with a computation time 20 units.
- Energy of T_i without DVS:
 - $E_1 = K * 20 * F^2$
- Energy of T_i with DVS:
 - $E_2 = K * 20 * (F/2)^2$
- Clearly, $E_2 = (E_1)/4$

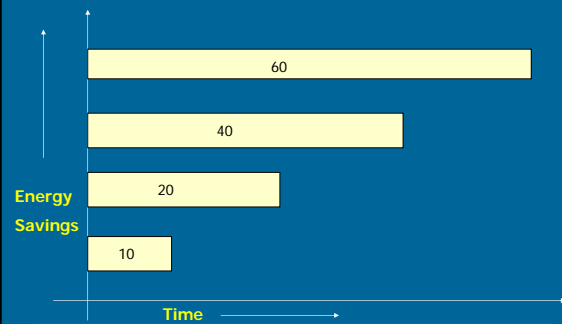
Time taken = t_1 (say)

Time taken = $t_2 = 2 * t_1$

Therefore, if we reduce the frequency we save energy but, we spend more time in performing the same computation

9

Energy-Time Tradeoffs



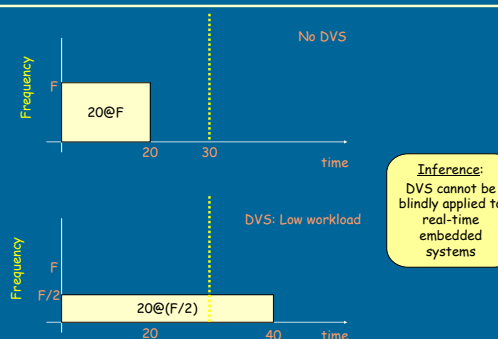
10

Simple DVS scheme handling RT-task

- Consider a real-time task $T_1 = (20, 30)$
- Applying the simple DVS scheme discussed earlier:
 - If there is peak load T_1 runs at maximum frequency (F) and meets the deadline with no energy savings
 - If there is less load T_1 runs at half the maximum frequency ($F/2$) and completes at time = 40 thereby missing its deadline

11

Simple DVS scheme handling RT-task



12

Energy aware scheduling in RT Systems

- Objectives:
 - Minimizing energy consumption
 - Meeting the deadlines.

13

Energy aware RTS Techniques

- OS Level Energy Management
 - The operating voltage/frequency decisions are made at scheduling phase
 - The operating voltage is decided at every task release and task completion
 - These schemes are also known as **Inter-task** schemes
- Compiler Level Energy Management
 - The operating voltage/frequency decisions are made at the compile time
 - The operating voltage is decided at the basic block level
 - These schemes are also known as **Intra-task** schemes

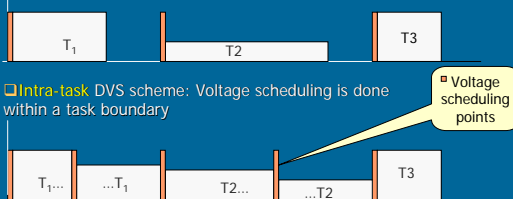
14

Real Time - DVS schemes

➤ The RT-DVS algorithms can be broadly classified based on the granularity at which voltage scheduling is performed as follows:

▣ **Inter-task** DVS scheme: Voltage scheduling is done on a task by task basis.

▣ **Intra-task** DVS scheme: Voltage scheduling is done within a task boundary



15

OS level energy management: Inter-task EDF

- Static voltage scaling EDF
- Cycle conserving RT-DVS
- Look-Ahead RT-DVS

16

Static Voltage Scaling EDF: Motivation

Pre-run schedule with holes

WC_i = worst case computation time @ F_{max}

Next arrival of T1



Holes in the pre-run schedule imply:

EDF Test:

$$\sum(wc_i/p_i) < 1 \text{ at frequency} = F_{max}$$

In other words, whenever $\sum(wc_i/p_i) < 1$ there are holes in the EDF schedule

17

Static Voltage Scaling EDF: exploiting holes

Pre-run schedule with holes

WC_i = worst case computation time @ F_{max}

Next arrival of T1



Processor typically idles during holes.
Instead, the holes can be exploited to slowdown the processor to save energy

18

Static Voltage Scaling EDF

EDF Test:
 $\sum (wc_i/p_i) < 1$ at maximum frequency = F_{max}

Static-VS EDF Test:
 $K * [\sum (wc_i/p_i)] = 1$ at frequency = F_{max}/K

19

Static voltage scaling: Example

- Task set: T1 = (1, 4) and T2 = (2, 8)
- $U = 1/4 + 2/8 = 0.5 (< 1) @ F_{max}$
- What is the "k" at which the task set is still schedulable @ (F_{max} / k):
 - Let $K = x$
 - $U = (1*x)/4 + (2*x)/8 = x*(0.5) = 1$
 - $X = 2$, that is $k = 2$
 - Therefore, we can operate at $f = F_{max} / 2$ and still meet the deadlines

20

Static voltage scaling: Example

Task set: T1 = (1, 4) and T2 = (2, 8)
 $U = 1/4 + 2/8 = 0.5 (< 1) @ F_{max}$

Finding the right frequency scaling parameter (say, k)
 $U = (1*k)/4 + (2*k)/8 = 0.5*k = 1 @ (F_{max}/k)$
 This gives, $k = 2$. Therefore, operating frequency = $F_{max}/2$

21

Static voltage scaling: Example

Modified Task set @ ($F_{max}/2$): T1 = (2, 4) and T2 = (4, 8)
 $U = 2/4 + 4/8 = 1 @ (F_{max}/2)$

Energy consumption:
 $1*F^2 + 2*F^2 = 3F^2$

Energy consumption:
 $1*(F/2)^2 + 2*(F/2)^2 = 1.5F^2$

22

What if ($C_i < WC_i$)??

Actual computation time

Next arrival of T1

More holes left unexploited

23

What if ($C_i < WC_i$)??

Actual computation time

Next arrival of T1

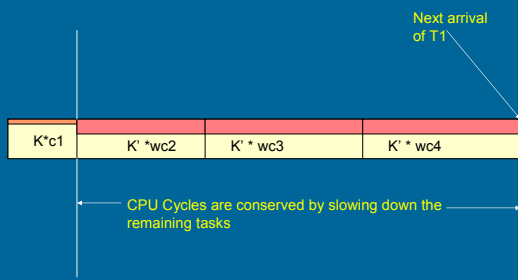
Task T1 completes

Hole of size = $(wc1 - c1)$

Slow down all these tasks proportionally

24

What if ($C_i < WC_i$)?? (contd..)



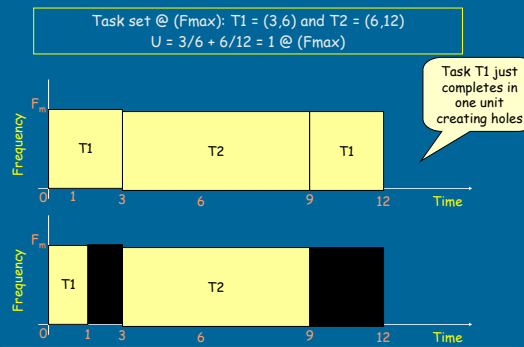
25

Cycle conserving EDF: Example

- Task set: T1 = (3, 6) and T2 = (6, 12)
- $U = 3/6 + 6/12 = 1 @ F_{max}$
- What is the “k” at which the task set is still schedulable @ (F_{max} / k):
 - Let $K = x$
 - $U = (3 \cdot x)/6 + (6 \cdot x)/12 = x \cdot (1.0) = 1$
 - $X = 1$, that is $k = 1$
 - Therefore, we should operate at $f = F_{max}$ in order to meet all the deadlines

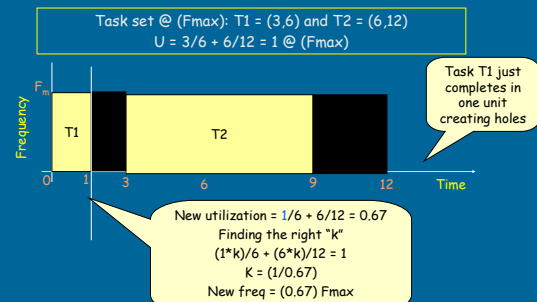
26

Cycle conserving EDF: Example



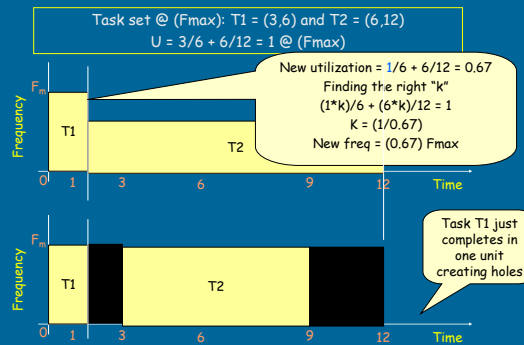
27

Static voltage scaling: Example



28

Cycle conserving EDF: Example



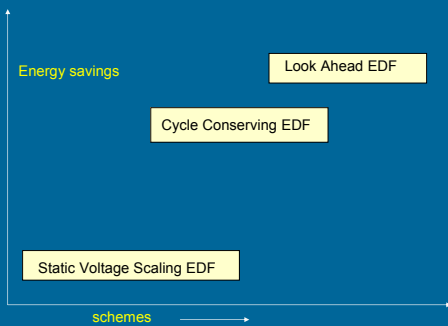
29

Look-Ahead EDF

- Defer as much work as possible.
- Sets the operating frequency to meet the minimum work that must be done now to ensure all future deadlines to be met.

30

Relative performance



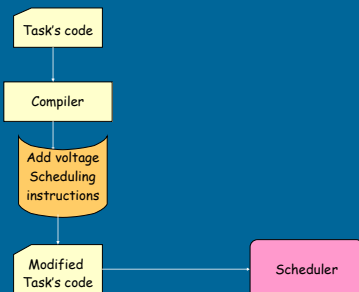
31

Compiler Level Energy Management

- Intra-task DVS: adjusts the voltage and clock speed within a task.
- Identifies the slack time generated within a task due to workload variation.
- Application code is preprocessed to enable the run-time clock/voltage adjustment.

32

Compiler level energy optimization framework



33

Intra-task Voltage Scheduling (Why..)

- Inter-task voltage scheduling: *run-calculate-assign-run*
- Requires OS modification
- Not suitable for 'single task' systems
- When the execution time of one task dominates total execution time, inter-task voltage scheduling may not be effective.

34

Intra-task Voltage Scheduling (What..)

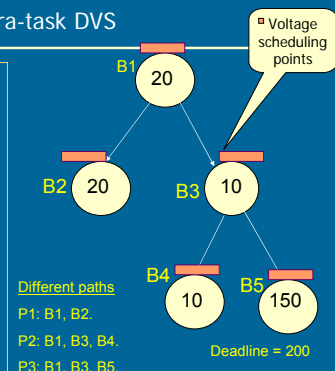
- Exploits all the slack time from runtime variations of different execution paths
- Voltage scaling decisions are made at compile time !!

35

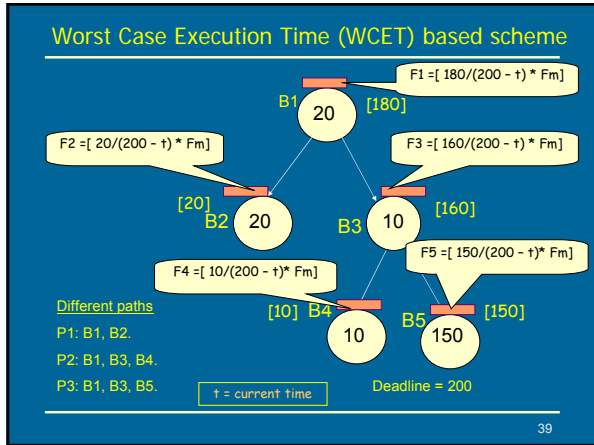
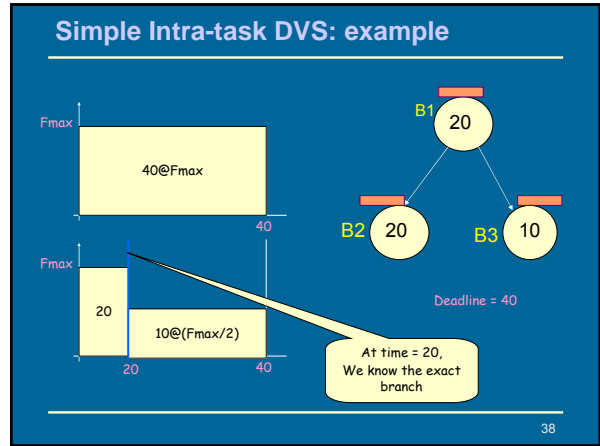
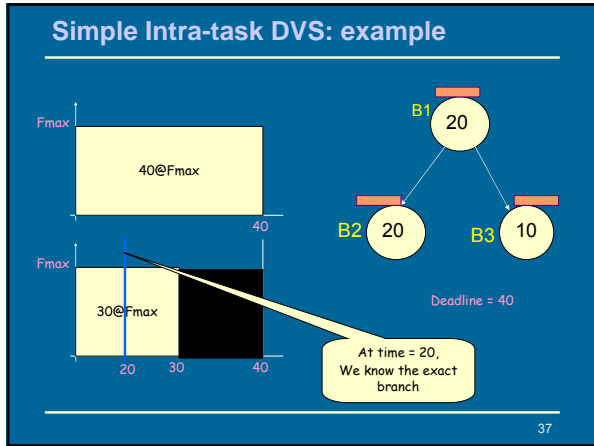
Intra-task DVS

Intra-task RT-DVS

- ✓ Intra-task DVS algorithms typically work with the control flow graph (CFG) of the real-time programs.
- ✓ Each node in the CFG denotes a basic block of computation.
- ✓ The edges in the CFG indicate the control dependency between the blocks.
- ✓ **Objective** is to assign proper clock frequency to each of the basic blocks so as to minimize the total energy consumption while meeting the task deadline.



36



Conclusions

- RT-DVS schemes are designed to ensure predictability while saving as much energy as possible in real time systems.
- Related issues:
 - Combined scheduling
 - Value based scheduling
 - Feedback based scheduling.
 - Fault Tolerant Scheduling.

40