Dynamic Voltage Scaling for Real-Time Multi-task Scheduling Using Buffers

June 12th, 2004
Chaeseok Im and Soonhoi Ha
Seoul National University, Korea
Outline

🌟 Our target and contribution
🌟 Buffering technique – ISLPED (2001)
🌟 Buffering technique extension for EDF/RM
🌟 Experiments
🌟 Conclusions and future work
Our Target and Contribution

- Dynamic voltage scaling (DVS)
- Contribution: enhancing the effectiveness of the DVS technique for EDF/RM
Dynamic Voltage Scaling (DVS)

- Popular low power design technique
- \( E = P \times T \propto V_{dd}^2 \cdot f \cdot T = V_{dd}^2 \)

Processor

\[ \begin{align*}
V_{dd} & \quad \text{variable} \\
\quad & \quad f \\
\end{align*} \]

\(5V\)

\(5V\)

\(50MHz\)

\(5V\)

\(3V\)

\(30MHz\)

slack time
Slack Time: WCST vs. VST

- **Worst-case slack time (WCST)**
  - when assuming worst-case execution time (WCET)

- **Workload-variation slack time (VST)**
  - due to the run-time variation of the actual execution time (AET ≤ WCET)

- **Key concern of the real-time DVS research**
  - Increasing the slack time (WCST + VST) utilization
Target Applications

- There is latency constraint, but...
- Latency constraint (deadline) > period
  - Latency increase beyond the period is OK to some extent
- Multimedia applications
  - e.g. video phone
    - We can tolerate a slight latency increase if it gives us a great energy saving
Buffering technique – ISLPED (2001)
- increases the slack time (WCST + VST) utilization by using buffers
- makes the processor execute a buffered task instead of going to the power-down mode
- proposed for single-task execution

Buffering technique extension for EDF/RM
- targets for priority-based preemptive real-time multi-task scheduling
- implemented on top of the typical kernel implementation with slight modification
- easily applicable to various low-power EDF/RM scheduling algorithms
Buffering Technique – ISLPED (2001)

🌟 Single-task example
Buffering Technique – ISLPED (2001)

🌟 Single-task example

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Period $(T_P)$</td>
<td>8</td>
</tr>
<tr>
<td>Worst-case execution time (WCET)</td>
<td>6</td>
</tr>
<tr>
<td>Best-case execution time (BCET)</td>
<td>2</td>
</tr>
<tr>
<td>Average execution time (AVET)</td>
<td>4</td>
</tr>
</tbody>
</table>

🌟 Applying the DVS technique **without** input buffer
Single-task example

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Period ($T_p$)</td>
<td>8</td>
</tr>
<tr>
<td>Worst-case execution time (WCET)</td>
<td>6</td>
</tr>
<tr>
<td>Best-case execution time (BCET)</td>
<td>2</td>
</tr>
<tr>
<td>Average execution time (AVET)</td>
<td>4</td>
</tr>
</tbody>
</table>

Applying the DVS technique without input buffer
Buffering Technique – ISLPED (2001)

Single-task example

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Period ($T_p$)</td>
<td>8</td>
</tr>
<tr>
<td>Worst-case execution time (WCET)</td>
<td>6</td>
</tr>
<tr>
<td>Best-case execution time (BCET)</td>
<td>2</td>
</tr>
<tr>
<td>Average execution time (AVET)</td>
<td>4</td>
</tr>
</tbody>
</table>

Applying the DVS technique without input buffer

![Graph showing workload variation and time]
Buffering Technique – ISLPED (2001)

Single-task example

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Period ($T_P$)</td>
<td>8</td>
</tr>
<tr>
<td>Worst-case execution time (WCET)</td>
<td>6</td>
</tr>
<tr>
<td>Best-case execution time (BCET)</td>
<td>2</td>
</tr>
<tr>
<td>Average execution time (AVET)</td>
<td>4</td>
</tr>
</tbody>
</table>

Applying the DVS technique without input buffer
**Single-task example**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Period ($T_p$)</td>
<td>8</td>
</tr>
<tr>
<td>Worst-case execution time (WCET)</td>
<td>6</td>
</tr>
<tr>
<td>Best-case execution time (BCET)</td>
<td>2</td>
</tr>
<tr>
<td>Average execution time (AVET)</td>
<td>4</td>
</tr>
</tbody>
</table>

**Applying the DVS technique without input buffer**

- $P_{\text{reduced}}$
- Task$_{1,1}$
Single-task example

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Period ($T_p$)</strong></td>
<td>8</td>
</tr>
<tr>
<td><strong>Worst-case execution time (WCET)</strong></td>
<td>6</td>
</tr>
<tr>
<td><strong>Best-case execution time (BCET)</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>Average execution time (AVET)</strong></td>
<td>4</td>
</tr>
</tbody>
</table>

Applying the DVS technique without input buffer
Applying the DVS technique without input buffer

Applying the DVS technique with input buffer

- e.g. latency constraint (deadline) = $2T_P$

Buffering delay
Applying the DVS technique without input buffer

- Power down

Applying the DVS technique with input buffer

- e.g. latency constraint (deadline) = 2T_P

Buffering delay
Applying the DVS technique **without** input buffer

Applying the DVS technique **with** input buffer

- e.g. latency constraint (deadline) = $2T_P$

---

Power down

Buffer delay

$P_{\text{reduced}}$

Task 1,1

Task 1,2

Task 1,3

0 8 16 24

time

---

Buffering Technique - ISLPED (2001)
Buffering Technique – ISLPED (2001)

Applying the DVS technique **without** input buffer

- Power down

Applying the DVS technique **with** input buffer

- e.g. latency constraint (deadline) = 2T_p, assuming AVET

---

\[ P_{\text{reduced}} \]

\[ \begin{align*}
\text{Task}_{1,j-1} & \quad \text{Power down} \\
\text{Task}_{1,j} & \quad \text{Power down} \\
\text{Task}_{1,j+1} & \quad \text{Power down}
\end{align*} \]

---

\[ P_{\text{opt}} \]

\[ \begin{align*}
\text{Task}_{1,j-1} & \quad \text{Power down} \\
\text{Task}_{1,j} & \quad \text{Power down} \\
\text{Task}_{1,j+1} & \quad \text{Power down}
\end{align*} \]
Buffering Technique – ISLPED (2001)

- Increases the latency: period($T_P$) < latency($T_L$)
- The latency increase is tolerable in our target applications: period($T_P$) < latency constraint($T_{LC}$)
- Buffering is permitted until the latency constraint ($T_L \leq T_{LC}$)
Motivation for Extension

🌟 Base buffering technique
- achieves significant energy saving with slight latency increase
- proposed mainly for single-task execution

🌟 Extended buffering technique
- targets for priority-based preemptive real-time multi-task scheduling (EDF/RM)
Buffering Technique Extension for EDF/RM

- Priority-based preemptive real-time multi-task scheduling (EDF, RM)
- Low-power EDF/RM scheduling algorithms
- Buffering technique extension for EDF/RM
EDF & RM

❖ Priority-based preemptive real-time scheduling
  ■ Earliest Deadline First (EDF)
    ▪ The priority changes according to the deadline
  ■ Rate Monotonic (RM)
    ▪ The priority is fixed according to the period

❖ Kernel implementation
  ■ Task state
    ▪ Active: when the CPU is dispatched to the task
    ▪ Ready: when a new task instance is released
    ▪ Waiting: when a new task instance is not released
  ■ Queue
    ▪ Ready queue: sorted by the priority
    ▪ Wait queue: sorted by the next arrival time
  ■ Active task ← ready_queue.head
Low-power priority-based scheduling (LPPS) algorithm – Y. Shin et al. (2000)
- When active task ≠ φ and ready queue ≠ φ → executes the active task at the full speed
- When active task ≠ φ and ready queue = φ → executes the active task at a reduced speed by using the VST to the earliest next arrival time of all the tasks

Dynamic reclaiming algorithm (DRA) – H. Aydin et al. (2001)
- When the prior task does not take the WCET → executes the active task at a reduced speed by reclaiming the VST

Limitation
- When active task = φ and ready queue = φ → power down the CPU
LPPS (RM)

<table>
<thead>
<tr>
<th></th>
<th>$T_i$</th>
<th>$D_i$</th>
<th>$C_i$</th>
<th>$AC_i$</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
</tr>
</tbody>
</table>

When WCET

When AET < WCET

→ Slack time increase
### LPPS (RM)

<table>
<thead>
<tr>
<th></th>
<th>$T_i$</th>
<th>$D_i$</th>
<th>$C_i$</th>
<th>$AC_i$</th>
<th>Priority</th>
<th>Next arrival time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>50 → 100</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
<td>100</td>
</tr>
</tbody>
</table>

At time 50:
AET($\tau_3$) < WCET ($\tau_3$)
Run $\tau_1$ at a reduced speed to the earliest next arrival time (80)
At time 65:
AET(τ₁) < WCET (τ₁)
Power down until the earliest next arrival time(80)
Buffering Technique Extension

- allocates buffer for each task
- adds another task state
- adds the buffer queue
- sorted by the next buffer full time

![Diagram of task states: ACTIVE, READY (Buffers=MAX), BUFFERED (Buffers<MAX), WAITING (Buffers=0) with transitions: Activate, Preempt, Complete, Release]
Algorithm Modification

☀ When active task = φ and ready queue = φ

- Previous algorithm (LPPS, DRA): power down the CPU
- Extended buffering algorithm:
  - buffer queue ≠ φ → active task ← buffer_queue.head
  - execute the active task at a reduced speed by using the VST to the earliest next buffer full time of all the tasks

L1: if (active_task is completed) {
L2:   active_task.buffers--; 
L3:   if (active_task.buffers = 0) 
L4:     move active_task to the wait_queue;  
L5:   else 
L6:     move active_task to the buffer_queue; 
L7:  }
L8:  for (∀task ∈ wait_queue or buffer_queue) {
L9:    if (task.release_time ≤ current_time) {
L10:       task.buffers++; 
L11:       if (task.buffers = task.max_buffers) 
L12:         move the task to the ready_queue;  
L13:       else 
L14:         move the task to the buffer_queue; 
L15:     }
L16: if (active_task ≠ φ and ready_queue ≠ φ) 
L17:   if (ready_queue.head.priority > active_task.priority) {
L18:     set active_task.executed_time; 
L19:     context switch; 
L20:   }
L21: else if (active_task = φ) {
L22:    if (ready_queue ≠ φ) 
L23:       set active_task = ready_queue.head; 
L24:    else if (buffer_queue ≠ φ) 
L25:       set active_task = buffer_queue.head; 
L26:    if (active_task ≠ φ and ready_queue = φ) 
L27:      do DVS to the time when ready_queue ≠ φ; 
L28:    else if (active_task = φ) {
L29:      set timer = wait_queue.head.release_time; 
L30:      enter power_down mode; 
L31:   }}
Buffering Technique Extension

<table>
<thead>
<tr>
<th>Task</th>
<th>$T_i$</th>
<th>$D_i$</th>
<th>$C_i$</th>
<th>$AC_i$</th>
<th>Priority</th>
<th>Buffer size</th>
<th>Next buffer full time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>0 $\rightarrow$ 1</td>
<td>50</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
<td>0 $\rightarrow$ 1</td>
<td>80</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
<td>0 $\rightarrow$ 1</td>
<td>100</td>
</tr>
</tbody>
</table>

At time 0:
Increment buffer size for all tasks

At time 0:
Increment buffer size for all tasks
Buffering Technique Extension

### Table

<table>
<thead>
<tr>
<th></th>
<th>( T_i )</th>
<th>( D_i )</th>
<th>( C_i )</th>
<th>( AC_i )</th>
<th>Priority</th>
<th>Buffer size</th>
<th>Next buffer full time</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>50 ( \rightarrow ) 100</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

### Diagram

- \( \tau_1 \) Active task
- \( \tau_2 \) Ready queue
- \( \tau_3 \) Buffer queue
- \( \tau_1,1 \) Wait queue

At time 0:
- Update the next buffer full time of \( \tau_1 \)
- Run \( \tau_1 \) at a reduced speed to the earliest next buffer full time (80)
### Buffering Technique Extension

**Buffering Technique Extension**

<table>
<thead>
<tr>
<th></th>
<th>$T_i$</th>
<th>$D_i$</th>
<th>$C_i$</th>
<th>$AC_i$</th>
<th>Priority</th>
<th>Buffer size</th>
<th>Next buffer full time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>1 $\rightarrow$ 0</td>
<td>100</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

---

![Diagram](image)

- **Active task**
- **Ready queue**
- **Buffer queue**
- **Wait queue**

At time 40:
Decrement the buffer size of $\tau_1$
Buffering Technique Extension

<table>
<thead>
<tr>
<th></th>
<th>$T_i$</th>
<th>$D_i$</th>
<th>$C_i$</th>
<th>$AC_i$</th>
<th>Priority</th>
<th>Buffer size</th>
<th>Next buffer full time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>80 → 160</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

At time 40:
Update the next buffer full time of $\tau_2$
Run $\tau_2$ at a reduced speed to the earliest next buffer full time(100)
Buffering Technique Extension

<table>
<thead>
<tr>
<th>( \tau_1 )</th>
<th>( T_i )</th>
<th>( D_i )</th>
<th>( C_i )</th>
<th>( AC_i )</th>
<th>Priority</th>
<th>Buffer size</th>
<th>Next buffer full time</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>0 ( \rightarrow ) 1</td>
<td>100</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1 ( \rightarrow ) 0</td>
<td>160</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

At time 70:
- Increment the buffer size of \( \tau_1 \)
- Decrement the buffer size of \( \tau_2 \)
### Buffering Technique Extension

#### Table

<table>
<thead>
<tr>
<th></th>
<th>$T_i$</th>
<th>$D_i$</th>
<th>$C_i$</th>
<th>$AC_i$</th>
<th>Priority</th>
<th>Buffer size</th>
<th>Next buffer full time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>100 → 150</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>80</td>
<td>80</td>
<td>20</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>160</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>35</td>
<td>3</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

#### Diagram

- **Active task**: $\tau_1$
- **Ready queue**
- **Buffer queue**: $\tau_3$
- **Wait queue**: $\tau_2$

At time 70:
- Update the next buffer full time of $\tau_1$
- Run $\tau_1$ at a reduced speed to the earliest next buffer full time (100)
Comparison

LPPS

\[ \tau_1, \tau_2, \tau_3 \]

LPPS + Buffering technique

\[ \tau_1, \tau_2, \tau_3 \]

\[ \tau_{1,1}, \tau_{1,2}, \tau_{1,3}, \tau_{2,1}, \tau_{2,2}, \tau_{3,1}, \]
Experiments

- Experimental environment
- Energy consumption according to the number of tasks
- Energy consumption according to the additional buffer size
- Comparison between LPPS and DRA over the buffering technique extension
Target schemes
- lppsEDF, lppsRM – Y. Shin et al. (2000)
- ccEDF, ccRM, laEDF – P. Pillai and K. G. Shin (2001)
- DRA, AGR – H. Aydin et al. (2001)
- bufEDF, bufRM

- Energy model of the CPU
  - 10 speed levels
- Input: 30 random task sets
  - Actual execution time: gaussian distribution
Energy vs. Number of Tasks

- Normalized to the power-down approach
- Improvement: EDF) 6~56%, RM) 20~62%
Normalized to the power-down approach
Only 1 or 2 additional buffer is sufficient
LPPS vs. DRA over the Buffering

- Energy saved by the buffering
  - LPPS: 45~64%, DRA: 13~46%

- Slack time utilization increased by the buffering
  - LPPS: 29~40%, DRA: 13~37%
Conclusions and Future Work
Conclusions and Future Work

🌟 Conclusions

- Buffering technique extension for EDF/RM algorithms
  - achieves significant energy saving: EDF) 6~56%, RM) 20~62%
  - by using only 1 or 2 additional buffer for each task
  - with slight modification effort
  - easily applied to various low-power EDF/RM scheduling algorithms

🌟 Future work

- Inventing a new algorithm that uses buffer characteristics directly