Scheduling and Mapping of Periodic Tasks on Multi-Core Embedded Systems with Energy Harvesting

Jun Lu and Qinru Qiu
Department of Electrical and Computer Engineering
Binghamton University, State University of New York
Binghamton, New York
Outline

- Background
  - System model
  - Existing works
- Task scheduling and management in EH-RTES
  - Motivational Example
  - Utilization based task speed selection
  - Avoid energy overflow and shortage
  - Task slack management
  - Utilization based task mapping for multi-core system
  - Simulation results
- Conclusions
Power of Embedded Systems

- A typical battery-powered portable device lasts no more than several days in active mode
  - Frequent battery change is not desirable for many applications such as:
    - Remote sensor and monitoring, unmanned vehicle, etc.
- The goal of power reduction in embedded systems is to extend the system lifetime

<table>
<thead>
<tr>
<th>Device</th>
<th>Current</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>SmartMesh M2510</td>
<td>24 mA</td>
<td>10.4 days</td>
</tr>
<tr>
<td>Freescale BeeKit</td>
<td>100 mA</td>
<td>2.5 days</td>
</tr>
<tr>
<td>Crossbow MICAz</td>
<td>28 mA</td>
<td>8.9 days</td>
</tr>
</tbody>
</table>

Energy available from 1 AA Lithium battery: 3000 mAh
Solutions for Low Power

- **Energy Harvesting**
  - Draw parts or all of its operating energy from its physical surroundings

- **Dynamic voltage frequency scaling**
  - Lowering V (while simultaneously and proportionately cutting f) causes a quadratic reduction in E

- **Energy aware task scheduling**
  - Effectively reduces the deadline miss and increases the system reliability
Energy Harvesting Real-Time Embedded System (EH-RTES)

- **Energy Harvesting Module**
  - Solar panels are chosen to be the primary energy harvesting technology for EH-RTES.
  - Harvested power is time-variable, energy is calculated as:
    \[ E_H(t_1, t_2) = \int_{t_1}^{t_2} P_H(t) dt \]

- **Real-Time Embedded system module**
  - DVFS-enabled single-core or multi-core processor running real-time applications.
Energy Harvesting Real-Time System

- **Energy Storage Module**
  - To continue operation even when there is no energy to harvest
  - Energy buffer, provide a large energy-performance trade-off space

- **Conversion Monitoring Circuitry**
  - Measure and modulate the output from the EH Module and direct the energy flow among the EH, ES and RTES modules
  - Capable of performing automatic maximum power point (MPP) tracking (MPPT) to yield the maximum output power
DVFS-enable processor & Real-Time Task

- **DVFS-enable processor**
  - $N$ discrete operating frequencies $f_n: \{ f_n | 1 \leq n \leq N, \ f_{\text{min}} = f_1 < f_2 < \ldots < f_N = f_{\text{max}} \}$
  - $P_n$: The power consumption at frequency $f_n$
  - Slowdown factor $S_n : S_n = f_n/f_{\text{max}}$

- **Real-time task set** $T \{ T_1, \ldots, T_P \}$
  - The worst case execution time of a task $T_p$ is denoted by $w_p$, $D_p$ is denoted as the relative deadline of $T_p$, which is also equal to the period of $T_p$.
  - All tasks are independent and ready at the beginning of their period.
  - Slowdown factor $S_n$, then its actual execution time at frequency $f_n$ is $w/S_n$. 
Existing Works Overview

- Adaptive Scheduling DVFS (AS-DVFS) algorithm, schedules all the tasks in task queue and assigns frequency and voltage levels to them for evenly distributed workload for the processor.

- By assigning tasks to the processing element with lower operating frequency, migrate tasks among processors to further reduce energy consumption.
Motivational Example

- Three Tasks $T_1$, $T_2$ and $T_3$
  - weight: 2, 3, 1
  - period: 5, 10, 20

$U = 0.4 + 0.3 + 0.05 = 0.75$

### XSCALE PROCESSOR POWER AND FREQUENCY LEVELS

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>150</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage (V)</td>
<td>0.75</td>
<td>1.0</td>
<td>1.3</td>
<td>1.6</td>
<td>1.8</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>80</td>
<td>170</td>
<td>400</td>
<td>900</td>
<td>1600</td>
</tr>
<tr>
<td>Normalized Speed</td>
<td>0.15</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1.0</td>
</tr>
</tbody>
</table>

AS-DVFS: Total energy consumption is 18850mJ

Total energy consumption is 16875mJ
Utilization Based Task Speed Selection

- Selects the lowest possible speed that is higher than or equal to the summation of utilization of all tasks.

- Tasks are scheduled at the same speed based on the total utilization, it also reduces the time and energy overhead of voltage and frequency changing.

- Complexity: $O(P)$

<table>
<thead>
<tr>
<th>Speed Selection Based on Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. util=0;</td>
</tr>
<tr>
<td>2. for $i = 1:P$ {</td>
</tr>
<tr>
<td>3. util += get_util($T_i$);</td>
</tr>
<tr>
<td>4. }</td>
</tr>
<tr>
<td>5. choose lowest $S$ from {$S_f, \ldots S_N$}, such that util $\leq S$</td>
</tr>
<tr>
<td>6. for $i = 1:P$ {</td>
</tr>
<tr>
<td>7. $ft_i = st_i + w_i/S$;</td>
</tr>
<tr>
<td>8. }</td>
</tr>
</tbody>
</table>
Avoid Energy Overflow and Shortage

- Limited energy storage capacity and the uncertainty of the harvested energy.
- Task $T_p$ is scheduled to execute at time interval $[st_p, ft_p]$ with speed $S_p$:
  \[ E_O = E_C(st_p) + E_H(st_p, ft_p) - E_D(st_p, ft_p) - E_{cap} \]
- Round up the execution speed of task $T_p$ to the a higher speed $S_{p,\text{new}}$:
  \[ E_D(st_p, w_p / S_{p,\text{new}}) - E_D(st_p, w_p / S_p) \geq E_O \]

Deal with Energy Overflow and Shortage

Require: maintain P tasks in Q;
1. if (overflow energy) {
2.   calculate new operating frequency for current task based on
3.   update the finish time of the current task;
5. } elsif (energy shortage) {
6.   remove the task;
7. }
Task Slack

- The CPU will not be fully occupied.
  - A finite number of discrete speeds, and our algorithm selects the lowest possible speed
  - Some tasks will be executed at a higher speed and finished earlier if energy overflow is predicted.
  - Some tasks will be removed when energy underflow occurs

- Utilize the task slack
  - Insert idle period to let the system harvest more energy
  - Further slowing down the future tasks for lower energy consumption
Task Slack Management (TSM)

- ASAP-TSM: insert the idle period as soon as the battery is below 80% of the full capacity
- ALAP-TSM: holds the slack until there is no more task to be executed and an idle period is automatically inserted.
- MSTF-TSM policy will not consume the slack unless the energy harvesting rate exceeds a threshold $E_{th}$.

**Slack Reclamation**

Require: Task slack is available after $T_p$ is executed

1. for $i = p+1: P$
2. $st_i = \min(st_i, ft_{i-1})$
3. if $(st_i+w_i/S_{id[i]-1} < ft_i \& \& (w_i/(w_i+\text{slack}) \leq S_{id[i]-1})$;
4. slack = slack - $(w_i/S_{id[i]-1} - w_i/S_{id[i]})$;
5. $S_i = S_{i,\text{slow}}$
6. }$
7. $ft_i = st_i + w_i/S$;
8. }$

- Slack Reclamation algorithm utilizes the slack to further slow down the future tasks for more energy saving.
Overall Utilization-based Task Scheduling

---

**Overall Utilization Based Scheduling and DVFS**

Require: maintain P tasks in Q;
1. while (true) {
2.   if (incoming new task)
3.     push new task into Q, sort all task based on their deadlines;
4.   schedule task in Q according speed selection Algorithm;
5. }
6. check energy available;
7. manage the task slack according to Task Slack Management Algorithm
8. execute current task in the task queue
9. remove finished task from Q;
10. }
11. }
Task Mapping for Multi-Core system

- Multi-core scheduling
  - Global scheduling: assign task to each core and allows task migration
  - Partitioned scheduling: allocates each task to one core permanently

- Partitioning periodic real-time tasks in a multiprocessor is NP-Hard problem, worst-Fit-Decreasing is best heuristic


<table>
<thead>
<tr>
<th>Multi-Core UTB Partitioned Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Require: maintain a ready task queue Q</td>
</tr>
<tr>
<td>1. sorted periodic tasks based on non-increasing order of their utilization;</td>
</tr>
<tr>
<td>2. for $i = 1:P$ {</td>
</tr>
<tr>
<td>3. find the core $C_j$ with the lowest utilization;</td>
</tr>
<tr>
<td>4. allocate the task $T_i$ to the core $C_j$;</td>
</tr>
<tr>
<td>5. }</td>
</tr>
<tr>
<td>6. execute Utilization-based Task Scheduling;</td>
</tr>
</tbody>
</table>
Experiment Setup

- Energy source: Solar
- Periodic tasks, number of tasks is arbitrary
- Task period distributed randomly from 10s to 100s
- Intel Xscale and PowerPC 405PL
- Average core utilization $U_{ave}$

$$U_{ave} = \frac{U_{tot}}{M} = \frac{\sum W_i}{\sum D_i} \leq 1$$

### PowerPC PROCESSOR POWER AND FREQUENCY LEVELS

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>33</th>
<th>100</th>
<th>266</th>
<th>333</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage (V)</td>
<td>1.0</td>
<td>1.0</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>19</td>
<td>72</td>
<td>600</td>
<td>750</td>
</tr>
<tr>
<td>Normalize d Speed</td>
<td>0.1</td>
<td>0.3</td>
<td>0.8</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Single-core Deadline miss rate comparison

- Our approach provides the most improvement at workloads with medium utilization (i.e. 0.4~0.6)
Task Slack Management Algorithms

- TSM1~4 represent the ASAP-TSM, ALAP-TSM, MSTF-TSM and the slack reclamation policy.

- TSM-4 and TSM-3 provides up to 10.91% and 8.86% improvement compared to TSM-1 and TSM-2 respectively.
Multi-core with different partition algorithms

- Compare our proposed *Utilization Based (UTB)* multi-core partition algorithm with the random partition algorithm and task movement algorithm.
Comparison of Single-core and Multi-core Processor

- The multi-core system is able to execute more tasks under the same solar panel and energy storage.
- The multi-core system is able to utilize the overflowed energy that is wasted due to the limited energy storage capacity.

<table>
<thead>
<tr>
<th>Core Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{ave}$ (Average Utilization %)</td>
<td>60</td>
<td>30</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>Average DMR (%)</td>
<td>8.61</td>
<td>6.52</td>
<td>5.21</td>
<td>4.19</td>
</tr>
<tr>
<td>Normalized Consumed Energy</td>
<td>1</td>
<td>1.06</td>
<td>1.09</td>
<td>1.13</td>
</tr>
<tr>
<td>Normalized Overflowed Energy</td>
<td>1</td>
<td>0.94</td>
<td>0.89</td>
<td>0.84</td>
</tr>
</tbody>
</table>
Conclusions

- We proposed a low-complexity and effective task scheduling algorithm for EH-RTES based on task utilization
- We proposed the Utilization Based (UTB) partitioned methods to schedule periodic tasks on multi-core scheduling
- Achieved less DMR than random and TMA multi-core scheduling
- Illustrated the trend of DMR with different TSM algorithms
- Compared the DMR, consumed energy and overflowed energy of single-core and multi-core processor.
Thanks!

Questions?