A Configurable Hardware Scheduler (CHS) for Real-Time Systems

Pramote Kucharoen, Mohamed A. Shalan and Vincent J. Mooney III

Center for Research on Embedded Systems and Technology
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA

23 June 2003
Outline

- Introduction
- Related work
- CHS architecture
- CHS commands
- CHS interface
- Software support
- Automatic customization of CHS
- Experiments and results
- Conclusion
ptcb = OSTCBList; /* Point at first TCB in TCB list */
while (ptcb->OSTCBId != OS_TASK_IDLE_ID) { /* Go through all TCBs in TCB list */
    OS_ENTER_CRITICAL();
    if (ptcb->OSTCBDly != 0) { /* Delayed or waiting for event with TO */
        if (--ptcb->OSTCBDly == 0) { /* Decrement nbr of ticks to end of delay */
            if (!(ptcb->OSTCBStat & OS_STAT_SUSPEND)) /* Is task suspended? */
                OSSched(ptcb, RDY);
            else /* Yes, leave 1 tick to prevent loosing */
                ptcb->OSTCBDly = 1; /* the task when the suspension is removed. */
        }
    }
    ptcb = ptcb->OSTCBNext; /* Point at next TCB in TCB list */
    OS_EXIT_CRITICAL();
}
Overhead in \(\mu\text{C/OS II Scheduler}\)

![Graph showing Overhead vs. Time tick resolution (usec) for different numbers of tasks: 64, 32, 16, 8, and 4 tasks. The graph highlights the Overhead % ranging from 0% to 45%. The Time tick resolution (usec) axis ranges from 10 to 1000. The graph indicates a trend where the overhead decreases as the time tick resolution increases.](image-url)
Related Work

- Inflexible
- Only One Scheduling Discipline
- Packet Scheduler
- FASTCHART
- FASTHARD
- Not for Adaptive Systems
Why do we need the CHS?

- To reduce the scheduling overhead from the real-time operating system; hence, improve the system response time.
- To support a wide range of applications by supporting multiple scheduling disciplines that can be changed during system execution time.
  - Priority
  - Earliest Dead Line First (EDF)
  - Rate Monotonic (RM)
Interrupt Controller

Tasks Table

Current Task

Control Unit

Bus Interface Signals

Control Signals

SQ

PQ

CHS Architecture (1)
CHS Architecture (2)

Priority Queue (Ready Queue)

REG + Counter
Comparator
LOGIC

REG + Counter
Comparator
LOGIC

REG + Counter
Comparator
LOGIC

ID Register Counter

Comparison Results
New Data
Comparison results from the right block
Sleep Queue

- Used to store the Sleeping Tasks (YIELD/SLEEP).
- The Tasks are sorted according to their remaining sleep time.
- Once The Sleep Time expires it is moved to the PQ.

<table>
<thead>
<tr>
<th>ID</th>
<th>Counter</th>
</tr>
</thead>
</table>

Task Table

- Store Information about the existing tasks
- Indexed by the Task ID

<table>
<thead>
<tr>
<th>PRI</th>
<th>Period</th>
<th>WCET</th>
<th>TYPE</th>
<th>PRE</th>
<th>STATUS</th>
</tr>
</thead>
</table>

## CHS Commands

<table>
<thead>
<tr>
<th>Command</th>
<th># of Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduler Related</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>1</td>
</tr>
<tr>
<td>RUN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIGURE</td>
<td>1</td>
</tr>
<tr>
<td>Task Related</td>
<td></td>
</tr>
<tr>
<td>CREATE Task</td>
<td>1</td>
</tr>
<tr>
<td>MODIFY Task</td>
<td>2</td>
</tr>
<tr>
<td>SLEEP</td>
<td>2</td>
</tr>
<tr>
<td>SSLEEP</td>
<td>1</td>
</tr>
<tr>
<td>YIELD</td>
<td>1</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>1</td>
</tr>
<tr>
<td>RESUME</td>
<td>1</td>
</tr>
<tr>
<td>DELETE</td>
<td>1</td>
</tr>
</tbody>
</table>
The CHS Hardware is designed to be able to interface easily to any microprocessor core:

- As a memory mapped I/O Port,
- As a co-processor, or
- As instruction-set accelerator
Software Support

<table>
<thead>
<tr>
<th>APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
</tr>
<tr>
<td>– createTask</td>
</tr>
<tr>
<td>– suspendTask, resumeTask</td>
</tr>
<tr>
<td>– changePriority, changeWCET, changePeriod</td>
</tr>
<tr>
<td>– Yield</td>
</tr>
<tr>
<td>– ssleep, sleep</td>
</tr>
<tr>
<td><strong>Scheduler</strong></td>
</tr>
<tr>
<td>– configureScheduler</td>
</tr>
<tr>
<td>– enableScheduler, disableScheduler</td>
</tr>
</tbody>
</table>
Automatic Customization of CHS

Scheduler Configurator (SCon)

Processor: ARMSTM
Tasks: 0
External Interrupts: ?
Timer Resolution: 10 usec

VPP  HW  DB

Customized HW

Customized RTOS

DC Synthesis Script
Experiments and Results (1)

Simulation Environment

VCS -> Seamless CVE -> XRAY

MPC750 <-> Hardware Scheduler

Interrupt

Address/Data Bus

Memory
Experiments and Results (2)

Assembly instruction execution comparison

<table>
<thead>
<tr>
<th></th>
<th>Micro C/OS II</th>
<th>Hardware Scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduler*</td>
<td>69</td>
<td>0</td>
</tr>
<tr>
<td>Time-tick processing</td>
<td>47 + 47* (number of tasks)</td>
<td>0</td>
</tr>
</tbody>
</table>

* Priority Scheduler

Number of PowerPC instruction of the APIs

<table>
<thead>
<tr>
<th>API</th>
<th># of PPC Assembly Instructions</th>
<th>WCET (# of cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>configureScheduler</td>
<td>37</td>
<td>230</td>
</tr>
<tr>
<td>SuspendTask</td>
<td>21</td>
<td>125</td>
</tr>
</tbody>
</table>

CHS Requires One PPC Instruction to be Configured and One Instruction to Suspend a Task which means over 100x Speedup.
Experiments and Results (3)

- Fixed-Cycle Operations
- Improve Response Time
- Scheduling in Software
- Time tick
- background processing

CHS
### CHS Synthesis Results

<table>
<thead>
<tr>
<th>Number of standard cells</th>
<th>Area (mm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1115</td>
<td>0.24</td>
</tr>
</tbody>
</table>

Using HP 0.35µ process

<table>
<thead>
<tr>
<th>Number of Logic Elements</th>
<th>Number of Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>421</td>
<td>564</td>
</tr>
</tbody>
</table>

Using Altera Quartus II for EP20K

The Synthesized CHS Supports
- 16 Tasks and
- up to 8 interrupt sources
Conclusion

- We implemented a configurable hardware scheduler that supports 3 scheduling algorithms
- We developed software interface for the configurable hardware scheduler and a tool to generate a customized synthesizable CHS
- The configurable hardware scheduler eliminated the time spent by the processor for background time tick processing and scheduling