Towards OS kernel acceleration in heterogeneous systems

Alex Kroh | Oliver Diessel
Accelerator candidates

• Traditional candidates
  – Long running operations
  – Highly parallel algorithms

• OS kernel operations
  – Typically short running
  – Little parallelism (any thread will usually be blocked on IO)
  – *Shared between all applications*
  – *Always on critical path*
  – *CPU execution time is non-deterministic*
    - Difficult WCET analysis of real-time systems

Towards OS kernel acceleration in heterogeneous systems
Case study: Kernel scheduler

- Zynq-7000 APSoC
  - Dual core ARM Cortex-A9 CPU
  - On-chip FPGA fabric accessible via MMIO over ARM AXI bus

- seL4 micro-kernel
  - Minimal code executing in privileged mode
  - High performance inter-process communication (IPC)

- Fixed-priority scheduler
  - Self contained
  - Scheduling is the most frequent kernel operation
  - Trivial to implement in hardware
Case study: OS kernel scheduler (SW)
Case study: OS kernel scheduler (HW)

• 256 FIFOs (1 per priority)
  – FIFO empty signals aggregated at a priority encoder
• MMIO via ARM AXI
• Address mapping matches SW scheduler
  – Additional bit 11 selects highest priority non-empty FIFO

<table>
<thead>
<tr>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>P = 1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

• WRITE := enqueue
• READ := dequeue
Case study: OS kernel scheduler (HW)
Case study: OS kernel scheduler (SW)

CPU execution cycles :=
 Kernel invocation +
 Kernel scheduling +
 Kernel reply to sender
CPU execution cycles :=
Kernel invocation +
Kernel scheduling +
Kernel reply to sender

Case study: OS kernel scheduler (SW)
Case study: OS kernel scheduler

CPU execution cycles :=
Kernel invocation +
Kernel scheduling +
Kernel reply to sender

![Graph showing CPU execution cycles against receiver thread priority.](image-url)
Case study: OS kernel scheduler

Schedule()

ChooseThread()

Execution delay

Communication delay

Schedule()

Hardware
Theoretical limits

CPU execution cycles :=

Kernel invocation +
Kernel scheduling +
Kernel reply to sender
Future work

• Investigation of cache coherent AXI port (ACP) to reduce delay
• Acceleration of other kernel functions
• Zynq Ultrascale+  
  • OS kernel acceleration for Cortex-R  
  • FPGA resource virtualisation for use by virtual machines on Cortex-A53
Towards OS kernel acceleration in heterogeneous systems
Branch predictor anomalies

IPC execution cycles against receiver thread priority

CPU execution cycles @ 666 MHz

IPC receiver thread priority

Branch mispredicts against receiver thread priority

Mispredicted or not predicted branch

IPC receiver thread priority

Towards OS kernel acceleration in heterogeneous systems