AMVP – An ARM Multicore VP using Parallel SystemC

Jan Henrik Weinstock, 18.09.2018
Virtual Platforms (VPs)

- Simulator that mimics a real hardware platform
  - Accuracy depends on modeling detail
  - Executes unmodified target binary

- Benefits over a hardware prototype
  - Early availability: HW/SW Co-Design
  - Efficient redistribution to SW developers
  - Enables non-intrusive system introspection

- VPs are typically built using SystemC
  - Integrates C/C++ component models
  - High execution speed

Example VP Graphical User Interface
VP Simulation Performance

- Example: UC/MC VP performance (MIPS)
  - Target SW: COREMARK/Linux
  - Test single/dual/quad core systems
  - In-house Instruction Set Simulator (ISS)
  - OSCI/Accellera\(^1\) SystemC kernel

\(^1\) Open SystemC Initiative (OSCI) - http://www.accellera.org/
QEMU and SystemC

- QEMU only models conventional UP/SMP/HMP designs
  - Supports ARMv7, ARMv8, Thumb, Thumb2, NEON, VFPv2/3/4 ...
  - Boots Linux/Android
  - Open Source

- Approaches to couple QEMU and SystemC:

```
```
QEMU Multiple Instantiation

- Convert QEMU into a shared library
- Load multiple copies to replicate global state
  - `dlmopen` loads the same lib into separate namespaces
- Copies must be linked symbolically
  - Resolve symbols at link time (instead of load time)
Parallel Quantum

- Model built-in Parallelism
  - Activate via model property
  - Synchronous Transactions
  - Less code to maintain
  - 2.9k vs 40k LoC (SCope\textsuperscript{[1]})

Threads Controlflow

Exclusive Memory Access

- Thread-safe implementation of \texttt{ldrex} and \texttt{strex} needed
  - \texttt{ldrex} creates load-link (LL) at \texttt{addr}
  - LL breaks upon read/write to \texttt{addr}
  - \texttt{strex} only stores if LL still present

- \textbf{Problem}: Intel (x86) has no matching atomic instructions (only CAS)
  - Option 1: deny DMI access to LL regions $\rightarrow$ transaction tracing
  - Option 2: retain DMI and emulate \textit{as good as possible} $\rightarrow$ ABA Problem

<table>
<thead>
<tr>
<th>Load-Linked (\texttt{ldrex})</th>
<th>Store-Conditional (\texttt{strex})</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{uint32_t data;}</td>
<td>if (\texttt{addr} != \texttt{core0.link_addr})</td>
</tr>
<tr>
<td>\texttt{memcpy(&amp;data, ptr + addr, 4);}</td>
<td>return false;</td>
</tr>
<tr>
<td>\texttt{core0.link_addr = addr;}</td>
<td>else</td>
</tr>
<tr>
<td>\texttt{core0.link_data = data;}</td>
<td>return \texttt{cas(dmi_ptr + addr, core0.link_data, data);}</td>
</tr>
<tr>
<td>\texttt{return data;}</td>
<td></td>
</tr>
</tbody>
</table>
AMVP – Platform Overview

- Quad-core Cortex A9 SMP Design

- Software
  - Linux Kernel
  - Busybox initrd
  - Dhrystone
Single Threaded Performance

- Dhrystone Benchmark (scattered bare metal, 4,000,000 runs per core)
Single Threaded Performance

- SMP Linux Boot Benchmark (power on → login prompt)
Parallel Performance

- SCope\textsuperscript{[1]} (4 Threads), Parallel Quantum (4 + 1 Threads) vs. OSCI baseline

\begin{align*}
2.5x \\
2.8x \\
2.6x \\
3.2x
\end{align*}

Summary & Outlook

- **Sequential Operation**
  - AMVP is sufficiently detailed to boot and run a realistic OS (Linux)
  - AMVP reaches 100+ MIPS for computational workloads and large quanta (≥ 100us)
  - Future work: test ARMv8 (supported by QEMU, but ldrex/strex need more work)

- **Parallel Operation**
  - Using four worker threads, speedups between 2.5x and 3.2x have been demonstrated

<table>
<thead>
<tr>
<th>Methodology</th>
<th>SScope Kernel</th>
<th>Parallel Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td># Threads</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Speedup Dhrystone</td>
<td>2.8x</td>
<td>2.5x (~11% slower)</td>
</tr>
<tr>
<td>Speedup Linux</td>
<td>3.2x</td>
<td>2.6x (~19% slower)</td>
</tr>
<tr>
<td>LoC</td>
<td>~40.000</td>
<td>~2900 (~92% less)</td>
</tr>
</tbody>
</table>