RISC-V Virtualization: A Case Study on the CVA6

Bruno Sá, Luca Valente, José Martins, Dr. Davide Rossi, Dr. Luca Benini and Dr. Sandro Pinto

RISC-V Summit’23 @ Barcelona

Jun 8th, 2023
Agenda

Introduction
   Contextualization

CVA6 Virtualization Support
   Overview, Status and Implementation

CVA6 Virtualization Enhancements
   Overview, L2 TLB and GTLB

Evaluation
   Design Space Exploration and Power, Performance and Area Analysis
Virtualization Technology

Allows the execution of multiple Operating Systems into the same hardware platform. An **Hypervisor or VMM** is to an OS, as an OS is to a process.

- **Main functions**
  - Resource Management
  - Abstraction
  - Protection / Isolation

- **The hypervisor provides a Virtual Machine (VM) abstraction for guest OSes**

- **Well-established**
  - Servers (load balancing, power management)
  - Desktops (cross-platform, systems development)
  - Embedded / MCS (isolation, consolidation, security)
CVA6 Virtualization Support

Overview, Status and Implementation
Hypervisor Extension In a Nutshell

- Suitable for type-1 and type-2 hypervisors

- New/Extended privilege modes:
  - HS-mode (hypervisor-extended supervisor)
  - VS-mode & VU-mode (orthogonal execution)
  - Virtualization mode bit (V) to indicate a guest is executing

- CSR modifications:
  - HS-mode CSRs for hypervisor capabilities
  - HS-mode CSRs for accessing/managing Guest/VM state
  - VS-mode CSRs (replicas of the regular S-mode)

- Adds a second stage of translation (G-Stage)

- Supports nested virtualization

- Ratified in Q4 2021 version 1.0

Chapter 8
Hypervisor Extension, Version 1.0

This chapter describes the RISC-V hypervisor extension, which virtualizes the supervisor-level architecture to support the efficient hosting of guest operating systems atop a type-1 or type-2 hypervisor. The hypervisor extension changes supervisor mode into hypervisor-extended supervisor mode (HS-mode, or Hypervisor mode for short), where a hypervisor or a hosting-capable operating system runs. The hypervisor extension also adds another stage of address translation, from guest physical addresses to supervisor physical addresses, to virtualize the memory and memory-mapped I/O subsystems for a guest operating system. HS-mode acts the same as S-mode, but with additional instructions and CSRs that control the new stage of address translation and support hosting a guest OS in virtual S-mode (VS-mode). Regular S-mode operating systems can execute without modification either in HS-mode or in VS-mode guest.

In HS-mode, an OS or hypervisor interacts with the machine through the same SHL as an OS normally does from S-mode. An HS-mode hypervisor is expected to implement the SHL for its VS-mode guest.

The hypervisor extension depends on an “T” base integer ISA with 64 x registers (RV32I or RV44I), not RV64E, which has only 32 x registers. CSR entries must not be read-only zero, and standard page-based address translation must be supported, either RV32 or RV64, or a minimum of 512MB for RV64.

The hypervisor extension is enabled by setting bit 7 in the main CSR, which corresponds to the letter E. RISC-V books that implement the hypervisor extension are encouraged not to hardcline main7, so that the extension may be disabled.

The baseline privileged architecture is designed to simplify the use of classic virtualization techniques, where a guest OS is run at user-level, so the few privileged instructions can be easily detected and trapped. The hypervisor extension improves virtualization performance by reducing the frequency of these traps.

The hypervisor extension has been designed to be efficiently available on platforms that do not implement the extension, by running the hypervisor in S-mode and trapping into M-mode for hypervisor CSR accesses and to maintain shadow page tables. The majority of CSR accesses for type-2 hypervisors are small S-mode accesses so need not be trapped. Hypervisors can support nested virtualization transparently.

https://github.com/riscv/riscv-isa-manual/releases/latest
Hypervisor Extension: Privilege modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Environment</th>
<th>Privilege Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtualized</td>
<td>V=1</td>
<td>Decreasing</td>
</tr>
<tr>
<td>VS</td>
<td>Guest OS</td>
<td></td>
</tr>
<tr>
<td>HS</td>
<td>Hypervisor</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Firmware (SBI)</td>
<td></td>
</tr>
<tr>
<td>Non-virtualized</td>
<td>V=0</td>
<td>Non-decreasing</td>
</tr>
<tr>
<td>VU</td>
<td>Guest User Space</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>Host User Space</td>
<td></td>
</tr>
</tbody>
</table>

Adapted from: Alistair Francis (WD), “Developing the RISC-V Hypervisor Extensions in QEMU”, Embedded Linux Conference Europe, 2019
CVA6-H: Status and Features

- H Extension, Version 1.0.0
- RV64 and sv39x4
- Nested-MMU uArch optimizations:
  - GTLB -&gt; G-Stage TLB in the nested-PTW
  - L2 TLB
- Advanced Interrupt Architecture (AIA) support:
  - APLIC -&gt; wired Interrupts
  - IMSIC -&gt; MSI interrupts
  - Supports interrupt virtualization
- Sstc extension support, version 0.5.4
- Optional extension via config parameter CVA6ConfigHExtEn
- PR to the cva6 main repo open at: https://github.com/openhwgroup/cva6/pull/1112
CVA6-H: Overview
CVA6 Hypervisor Extension: Implementation

- PR to the cva6 main repo open at: [https://github.com/openhwgroup/cva6/pull/1112](https://github.com/openhwgroup/cva6/pull/1112)
- Optional extension: enabled in the config file by setting the `CVA6ConfigHExtEn`
- ~4087 sLOC modifications cva6. Mainly on:
  - CSR
  - PTW
  - TLB
  - MMU
CVA6 Virtualization Enhancements
CVA6-H: Enhancements

How to improve the virtualization efficiency?

A full nested-PTW walk takes 15 memory accesses, 5x more than a normal S-mode translation.

High TLB miss penalty leads to performance degradation in a virtualized system.
How to improve the virtualization efficiency?

Our Solution:
#1 Nested-MMU Enhancement: L2 TLB

- **Set-associative second-level TLB**
- **Main goal:**
  - Increase TLB reach
  - Decrease the TLB miss penalty
- **Follows split design:**
  - Larger TLB for 4KiB page size
  - Smaller TLB for 2MiB superpages
- **Size and associativity is configurable**
- **Page size support configurable (4KiB or/and 2MiB support)**
- **Available at:** [https://github.com/minho-pulp/cva6/tree/wip/hyp-opts](https://github.com/minho-pulp/cva6/tree/wip/hyp-opts)
#2 Nested-MMU Enhancement: GTLB

- **Second-level TLB:**
  - Stores only GPA -> HPA
- **Located at the PTW**
- **Fully associative TLB**
- **Speeds up the translation walk when G-Stage is enabled**
- **Configurable number of TLB entries (8 or 16)**
- **Optional: enabled through a parameter GTLB_EN**
- **Available at:** [https://github.com/minho-pulp/cva6/tree/wip/hyp-opts](https://github.com/minho-pulp/cva6/tree/wip/hyp-opts)

---

*Example of a full translation for Sv39x4 with GTLB support*
Evaluation

Design Space Exploration and Power, Performance and Area Analysis
CVA6-H Design Space Exploration: Evaluation

**System**
- Single core CVA6 SoC
- Genesys2 FPGA at 100MHz
- 16 KiB iL1$ and 32KiB dL1$

**Metrics**
- Functional performance
- FPGA resources

**Software Stack**
- OpenSBI (version 1.0)
- Bao Hypervisor (version 0.1)
- Linux OS (VM)

**Benchmark**
- Mibench (automotive subset)
- 1000 samples
- Higher results mean more performance
- non-optimized CVA6-H core serves as baseline
System Configurations

- Evaluated 23 out of 288 possible combinations
- Configurations and parameters as follow:

<table>
<thead>
<tr>
<th>Module</th>
<th>Parameter</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 TLB</td>
<td>size entries</td>
<td>#1  16  #2  32  #3  64</td>
</tr>
<tr>
<td>GTLB</td>
<td>size entries</td>
<td>8   16   --</td>
</tr>
<tr>
<td>L2 TLB</td>
<td>page size</td>
<td>4KiB 2MiB 4KiB+2MiB</td>
</tr>
<tr>
<td></td>
<td>associativity</td>
<td>4    8   --</td>
</tr>
<tr>
<td></td>
<td>size entries for 4KiB</td>
<td>128  256   --</td>
</tr>
<tr>
<td></td>
<td>size entries for 2MiB</td>
<td>32   64   --</td>
</tr>
<tr>
<td>SSTC</td>
<td>status</td>
<td>enabled disable</td>
</tr>
</tbody>
</table>

Designs selected for PPA Analysis

- Selected the 7 best designs based on the results for PPA analysis:

<table>
<thead>
<tr>
<th>Design</th>
<th>SSTC support</th>
<th>I/DTLB</th>
<th>8-entries GTLB</th>
<th>4k L2 TLB</th>
<th>2MB L2 TLB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-vanilla</td>
<td>x</td>
<td>16</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1-h-16</td>
<td>✓</td>
<td>16</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2-h-32</td>
<td>✓</td>
<td>32</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>3-h-gtlb-8</td>
<td>✓</td>
<td>16</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>4-h-gtlb-8-12-0</td>
<td>✓</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>5-h-gtlb-8-12-1</td>
<td>✓</td>
<td>16</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>6-h-gtlb-8-12-2</td>
<td>✓</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

*TABLE V*
Sstc extension brings on average a 10% functional performance speed up

GTLB + Sstc support achieves a minimum and a maximum performance speedup of 9% and 16%

Full max optimized (GTLB + SSTC+L2 TLB) achieves a 17% max performance increase for susanc-small
SSTC extension has a negligible impact on the area

All modules except the 2-h-32 config have less than 5% impact on power

Optimal design point achieved at 3-h-16-gtlb8 config: 12.5 % avg perf speedup, 0.78% area and 0.33% power
All configurations increase the energy efficient up to 16%

Optimal design point (3-h16-gtlb-8) is the most energy efficient
Shaheen Tapeout

- Shaheen Test-Chip
  - Secure RISC-V SoC
  - CVA6-based SoC w/ Hypervisor Extension
  - Target UAV/Drone applications
  - Shaheen -> Al Saqr
THANK YOU!

id10037@alunos.uminho.pt
GET IN TOUCH WITH US!

- **Address:** Universidade do Minho, Campus Azurém, 4800-058 Guimarães, PORTUGAL
- **Phone:** +351 253 510180
- **Email:** id10037@alunos.uminho.pt