Open-source RISC-V Input/Output Memory Management Unit (IOMMU) IP

Manuel Rodríguez  Francisco Marques  Bruno Sá  Sandro Pinto
Centro ALGORITMI/LASI, Universidade do Minho

Abstract

This work describes the design and implementation of an open-source IOMMU IP compliant with the ratified version of the RISC-V IOMMU specification (v1.0-rc3). So far, we have designed and implemented a basic IP encompassing only the mandatory features (of the spec) and support for virtualization, which has been successfully validated and evaluated on a single-core CVA6-based SoC. Moving forward, we plan to extend the IP with more advanced features, i.e., optional features such as a hardware performance monitor, memory-resident interrupt files support, etc. We will open-source our IP to the RISC-V community.

RISC-V IOMMU IP Overview

- Two IP versions: Basic and Advanced
- Target application: Embedded SoCs
  - Optional Process Context support
  - Does not support PCIe ATS and PRI
  - Single-endianness support (little-endian)
- Virtualization support
  - Two-stage address translation
  - MSI Translation (basic-translate mode)
  - Bare and Sv39/Sv39x4 translation
- WSI or MSI generation, parameterizable
- Basic IP to be open-sourced
  - Target -> Q2 2023. License: SHL-2.1
  - Advanced IP features (TBD)
    - MRIF support, HPM, debug interface

Evaluation

- Basic IOMMU IP and PULP iDMA integrated into a single-core CVA6-based SoC for validation
- Bare-metal tests application developed to validate basic IOMMU features
- Validated in Linux using RISC-V IOMMU platform driver + iDMA driver + user application to perform DMA mapping and memory transfers -> address translation
- Work-in-Progress: Add RISC-V IOMMU support on Bao Hypervisor -> virtualization validation

Functional Validation

- Basic IOMMU IP and PULP iDMA integrated into a single-core CVA6-based SoC for validation
- Bare-metal tests application developed to validate basic IOMMU features
- Validated in Linux using RISC-V IOMMU platform driver + iDMA driver + user application to perform DMA mapping and memory transfers -> address translation
- Work-in-Progress: Add RISC-V IOMMU support on Bao Hypervisor -> virtualization validation

Hardware Resources

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Resource</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>No IOMMU</td>
<td>LUT</td>
<td>76667 / 203800</td>
</tr>
<tr>
<td></td>
<td>FF</td>
<td>53017 / 407600</td>
</tr>
<tr>
<td>WSI generation only</td>
<td>LUT</td>
<td>93135 (+8.08%)</td>
</tr>
<tr>
<td></td>
<td>FF</td>
<td>61573 (+2.09%)</td>
</tr>
<tr>
<td>WSI and MSI generation</td>
<td>LUT</td>
<td>95417 (+10.29%)</td>
</tr>
<tr>
<td></td>
<td>FF</td>
<td>63143 (+2.90%)</td>
</tr>
<tr>
<td>WSI, MSI, and PC support</td>
<td>LUT</td>
<td>95816 (+10.41%)</td>
</tr>
<tr>
<td></td>
<td>FF</td>
<td>63464 (+2.98%)</td>
</tr>
</tbody>
</table>

Post-synthesis hardware utilization results for single core CVA6 SoC targeting a Genesys 2 FPGA. Each Address Translation Cache has 4 entries

- Most hardware resources are consumed by the Address Translation Caches (IOTLB, DDTC, PDC). As these were implemented using flip-flops
- DDTC with 4 entries represents up to 57.16% of IOMMU LUT utilization and up to 20.33% of FF utilization.
- MSI generation support increases remap HW consumption: +201.7% in LUTs and +126.16% in FFs