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

FCT Fundação para a Ciência e a Tecnologia
GOVERNO DE PORTUGAL | MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

CENTROALGORITMI

Universidade do Minho

Jun 8th, 2023

Agenda

01 Introduction
Contextualization

02 CVA6 Virtualization Support
Overview, Status and Implementation

03 CVA6 Virtualization Enhancements
Overview, L2 TLB and GTLB

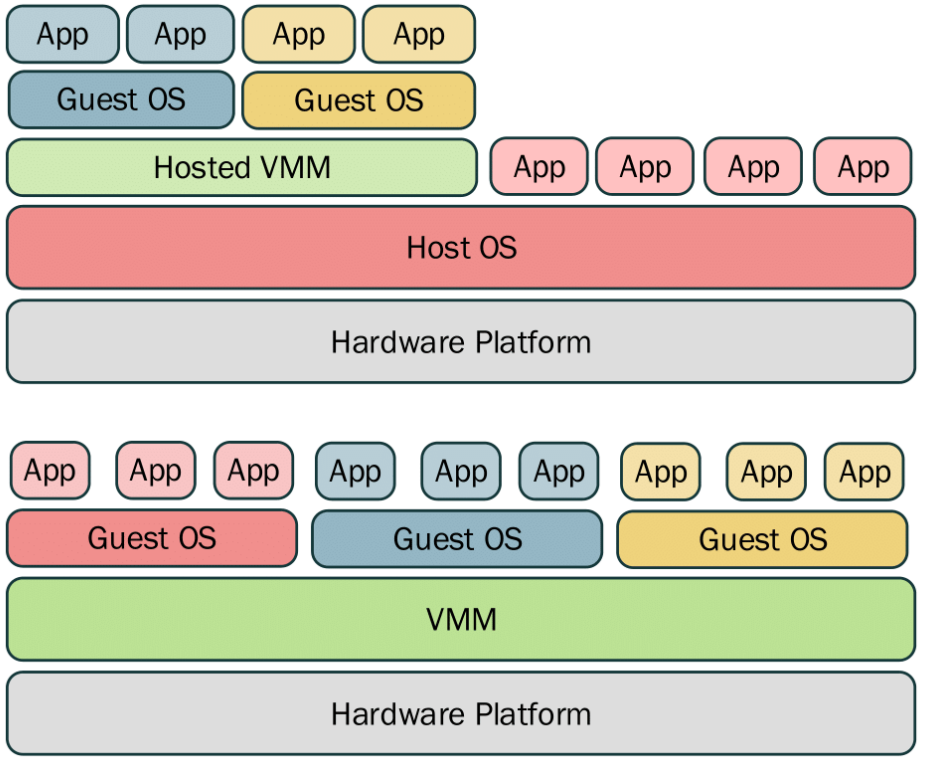
04 Evaluation
Design Space Exploration and Power, Performance and Area Analysis

Introduction

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)

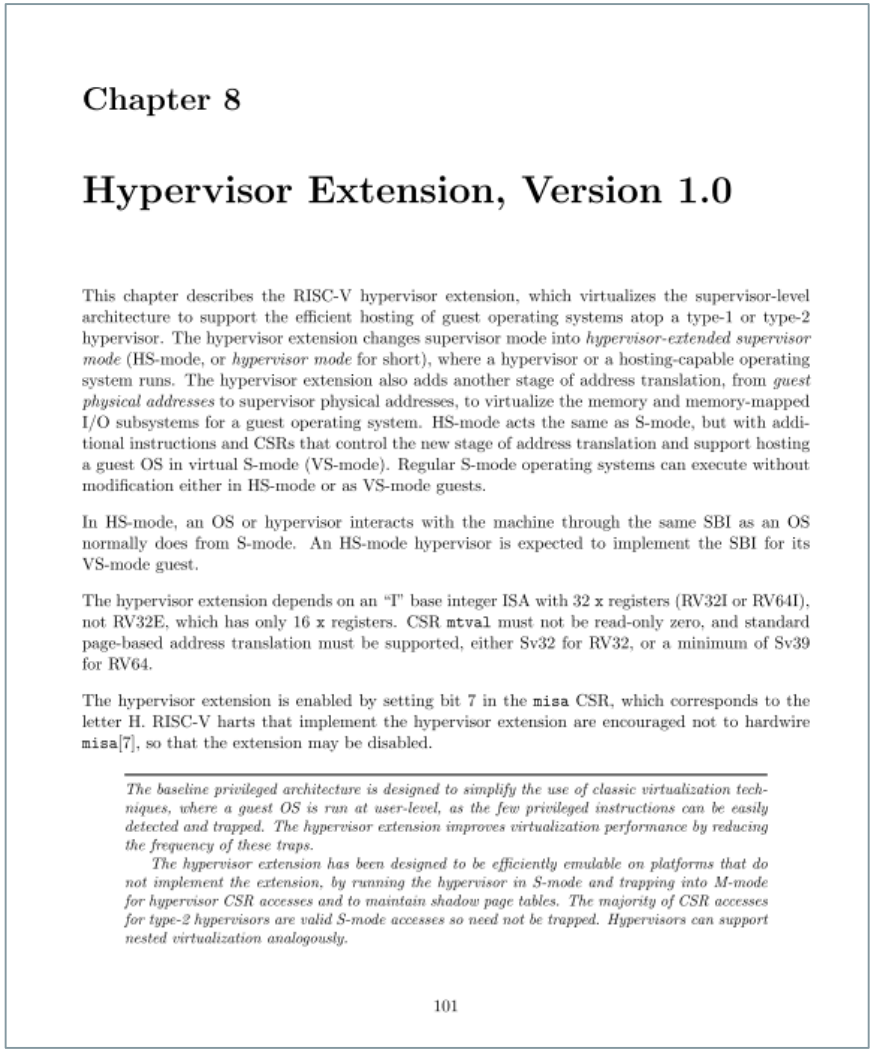


CVA6 Virtualization Support

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
- **CSRs 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 as VS-mode guests.

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

The hypervisor extension depends on an "I" base integer ISA with 32 `x` registers (RV32I or RV64I), not RV32E, which has only 16 `x` registers. CSR `mtval` must not be read-only zero, and standard page-based address translation must be supported, either Sv32 for RV32, or a minimum of Sv39 for RV64.

The hypervisor extension is enabled by setting bit 7 in the `misa` CSR, which corresponds to the letter H. RISC-V harts that implement the hypervisor extension are encouraged not to hardwire `misa`[7], so that the extension may be disabled.

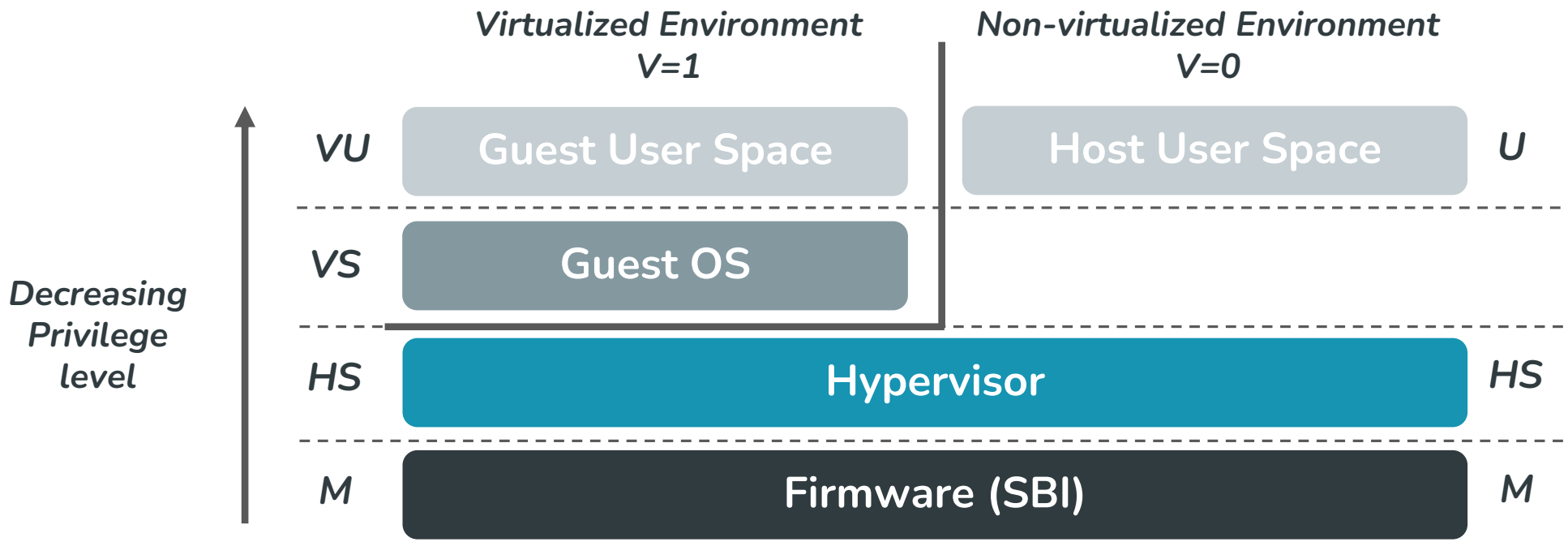
The baseline privileged architecture is designed to simplify the use of classic virtualization techniques, where a guest OS is run at user-level, as the few privileged instructions can be easily detected and trapped. The hypervisor extension improves virtualization performance by reducing the frequency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

The hypervisor extension has been designed to be efficiently emulable 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 valid S-mode accesses so need not be trapped. Hypervisors can support nested virtualization analogously.

101

https://github.com/riscv/riscv-isa-manual/releases/latest

Hypervisor Extension: Privilege modes



Adapted from: *Alistair Francis (WD), "Developing the RISC-V Hypervisor Extensions in QEMU", Embedded Linux Conference Europe, 2019*

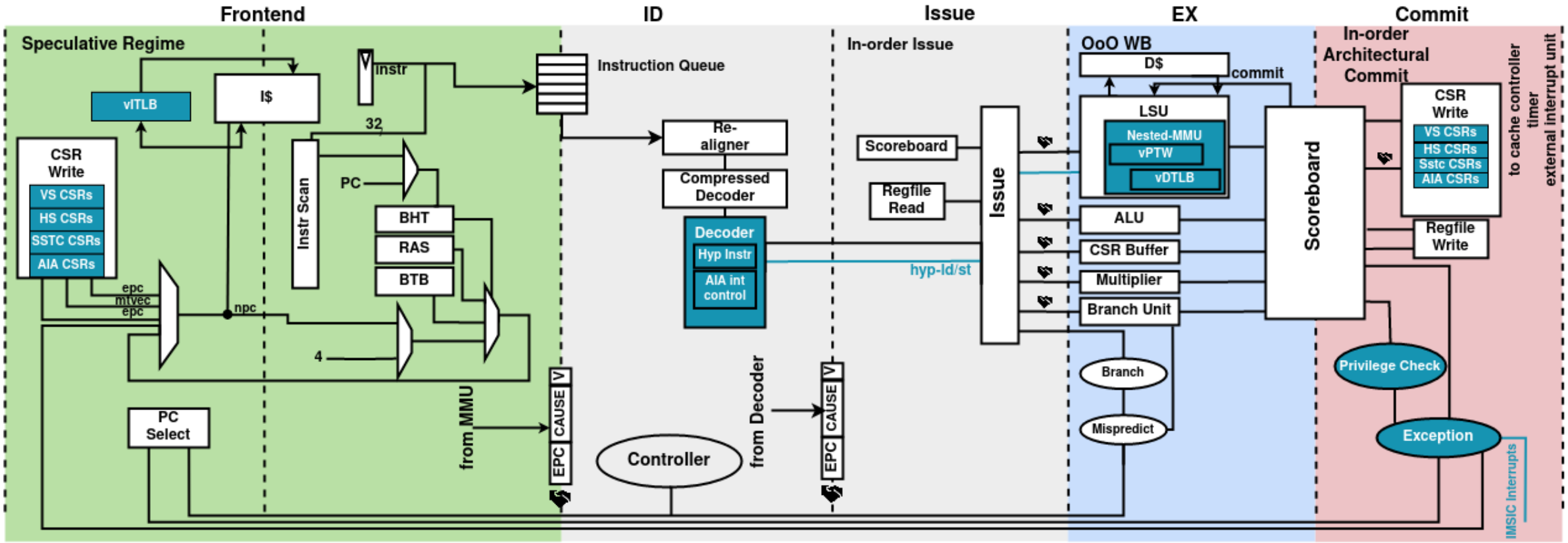
CVA6-H: Status and Features

- **H Extension, Version 1.0.0**
- **RV64 and sv39x4**
- **Nested-MMU uArch otimizations:**
  - GTLB -> G-Stage TLB in the nested-PTW
  - L2 TLB
- **Advanced Interrupt Architecture (AIA) support:**
  - APLIC -> wired Interrupts
  - IMSIC -> MSI interrupts
  - Supports interrupt virtualization
- **Sstc extension support, version 0.5.4**

CSRs	hstatus/mstatus	●
	hideleg/hedeleg/mideleg	●
	hvip/hip/hie/mip/mie	●
	hgeip/hgeie	◐
	hcounteren	●
	htimedelta	●
	henvcfg	◐
	mtval2/htval	●
	mtinst/htinst	●
	hgapt	◐
	vsstatus/vsip/vsie/vstvec/vsscratch vsepc/vscause/vstval/vsatp	●
Intructions	hlv/hlvx/hsv	●
	hfence.vvma/gvma	●
Exceptions & Interrupts	Environment call from VS-mode	●
	Instruction/Load/Store guest-page fault	●
	Virtual instruction	●
	Virtual Supervisor sw/timer/external interrupts	●
	Supervisor guest external interrupt	●

- **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**
- **Optional extension: enabled in the config file by setting the CVA6ConfigHExtEn**
- **~4087 sLOC modifications cva6. Mainly on:**
  - CSR
  - PTW
  - TLB
  - MMU

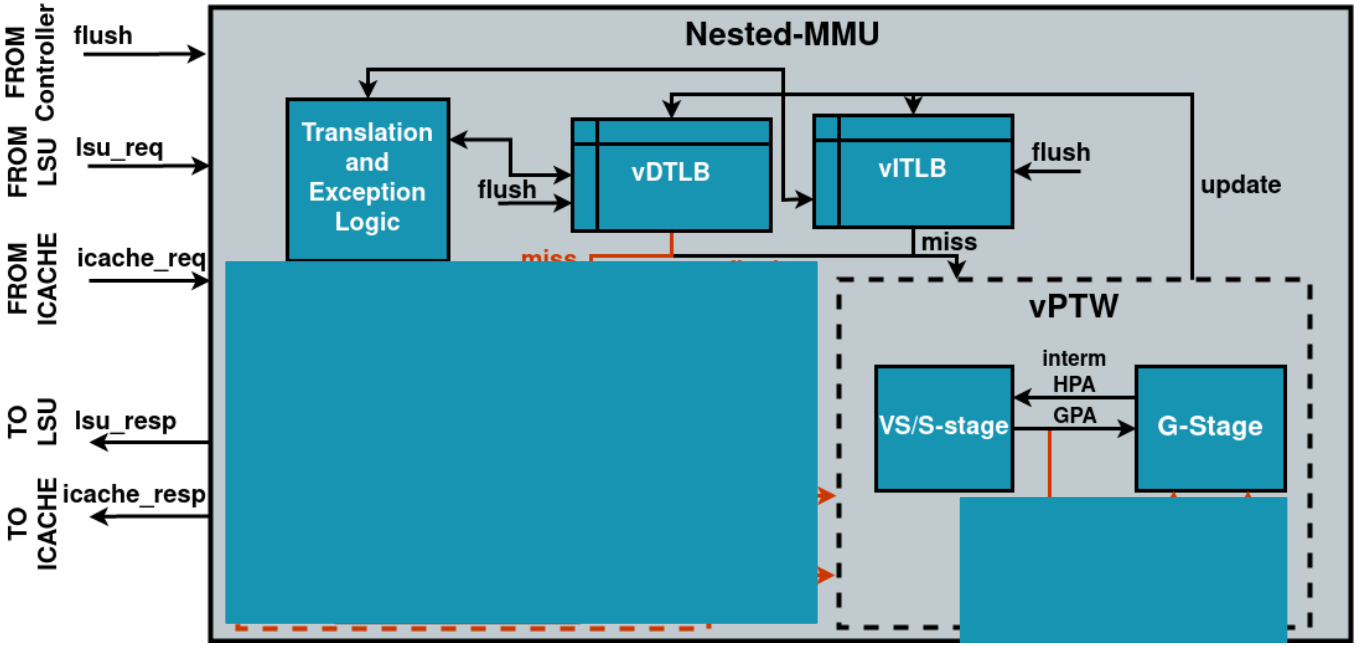
```
Bender.yml                                              |   3 +
Flist.ariane                                            |   3 +
core/Flist.cva6                                         |   5 +
core/branch_unit.sv                                     |   4 +
core/commit_stage.sv                                    |  39 +++++-
core/controller.sv                                      |  37 +++++-
core/csr_regfile.sv                                     | 995
core/cva6.sv                                            |  65
core/cvxif_fu.sv                                        |   6 +
core/decoder.sv                                         | 223
core/ex_stage.sv                                        |  84
core/frontend/frontend.sv                               |  17 ++-
core/frontend/instr_queue.sv                            |  21 +++
core/id_stage.sv                                        |  10 +-
core/include/ariane_pkg.sv                              | 111
core/include/cv32a60x_config_pkg.sv                     |   1 +
core/include/cv32a6_embedded_config_pkg.sv              |   1 +
core/include/cv32a6_ima_sv32_fpga_config_pkg.sv         |   1 +
core/include/cv32a6_imac_sv0_config_pkg.sv              |   1 +
core/include/cv32a6_imac_sv32_config_pkg.sv             |   1 +
core/include/cv32a6_imafc_sv32_config_pkg.sv            |   1 +
core/include/cv64a6_imafdc_sv39_config_pkg.sv           |   1 +
core/include/cv64a6_imafdc_sv39_openpiton_config_pkg.sv |   1 +
core/include/riscv_pkg.sv                               | 211
core/issue_read_operands.sv                             |  10 +-
core/issue_stage.sv                                     |   2 +
core/load_store_unit.sv                                 | 169
core/load_unit.sv                                       |  29 +++---
core/mmu_sv32/cva6_mmu_sv32.sv                          | 143
core/mmu_sv39/mmu.sv                                    | 144
core/mmu_sv39x4/cva6_mmu_sv39x4.sv                      | 703
core/mmu_sv39x4/cva6_ptw_sv39x4.sv                      | 642
core/mmu_sv39x4/cva6_tlb_sv39x4.sv                      | 393
core/store_unit.sv                                      |  10 +-
34 files changed, 3841 insertions(+), 246 deletions(-)
```

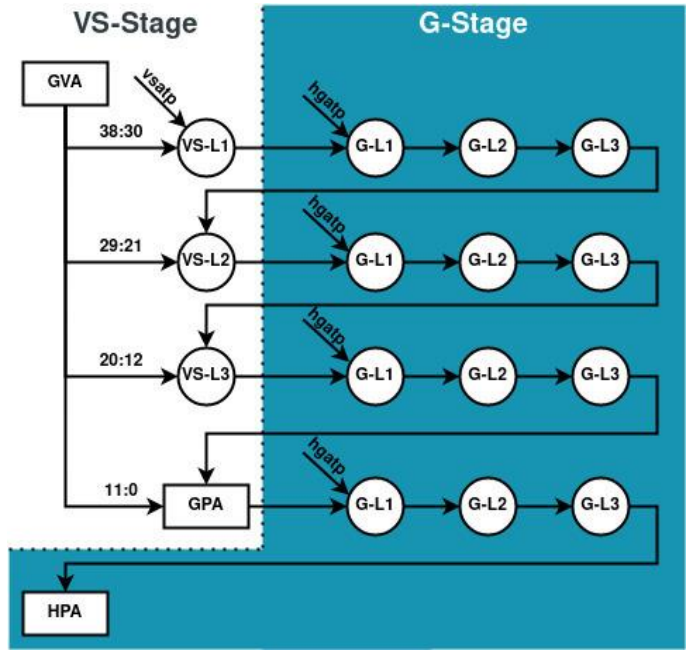
CVA6 Virtualization Enhancements

Overview, L2 TLB and GTLB

CVA6-H: Enhancements

How to improve the virtualization efficiency ?





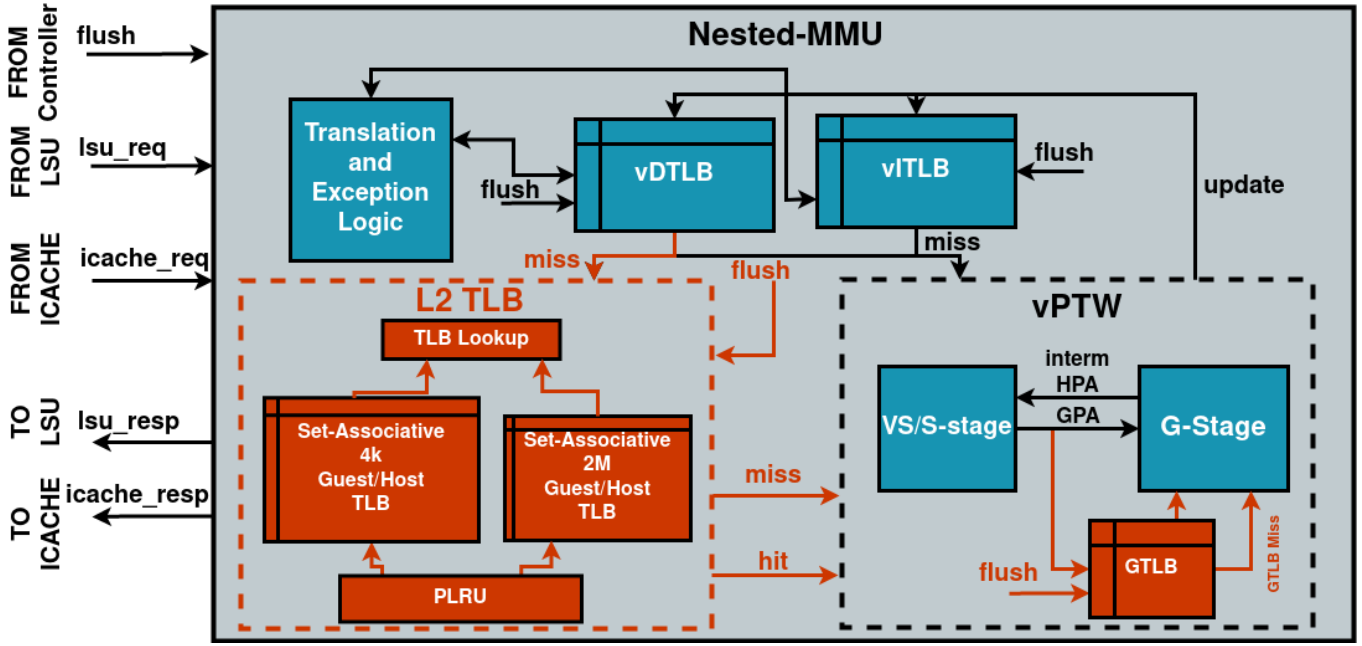
Example of a full two-stage translation for Sv39x4 scheme

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

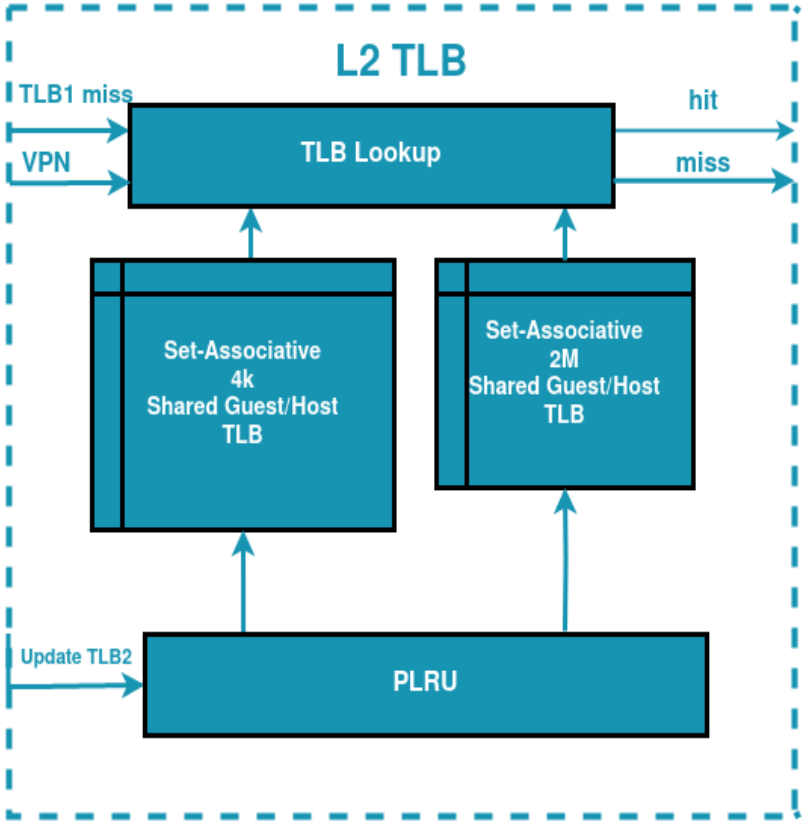
How to improve the virtualization efficiency ?

Our Solution:



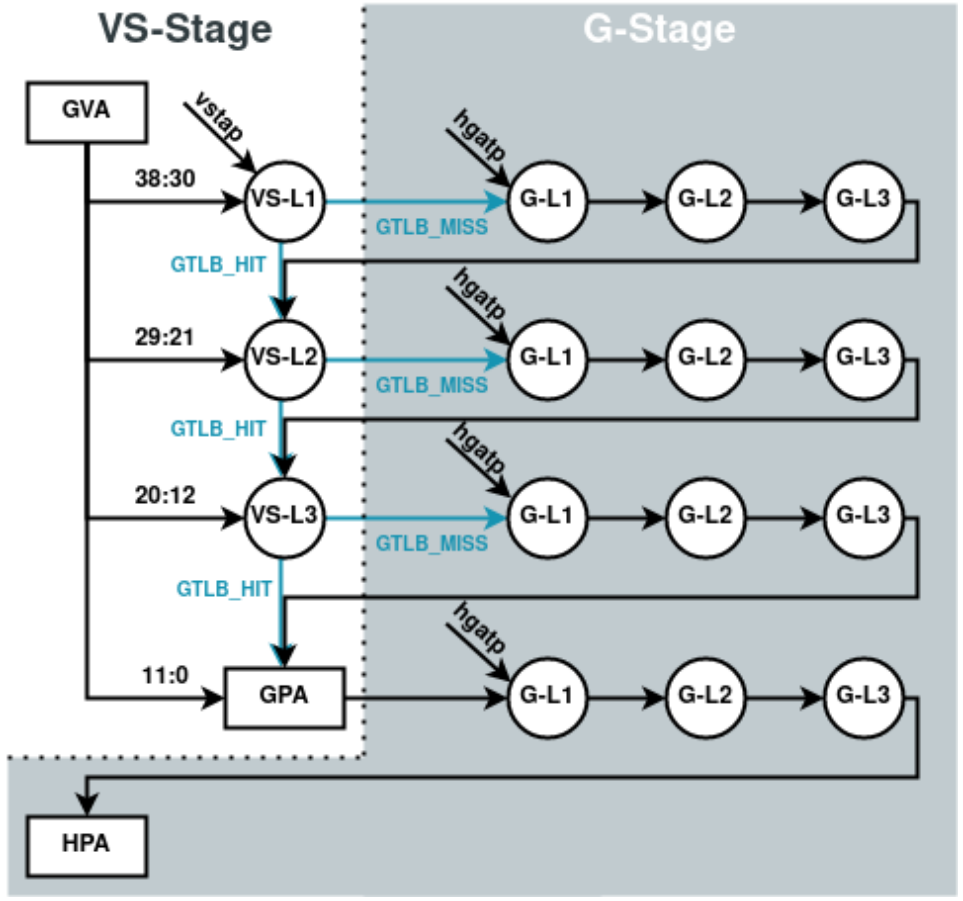
#1 Nested-MMU Enhancement: L2 TLB

- **Set-associative second-level TLB**
- **Main goal:**
  - Increase TLB reach
  - Decrease the TLB miss penality
- **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***



#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***



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)

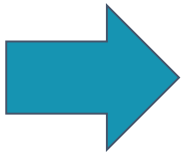
Benchmark

- Mibench (automotive subset)
- 1000 samples
- Higher results mean more performance
- non-otimized CVA6-H core serves as baseline

System Configurations

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

Module	Parameter	Configuration		
		#1	#2	#3
L1 TLB	size entries	16	32	64
GTLB	size entries	8	16	—
L2 TLB	page size	4KiB	2MiB	4KiB+2MiB
	associativity	4	8	—
	size entries for 4KiB	128	256	—
	size entries for 2MiB	32	64	—
SSTC	status	enabled	disable	—



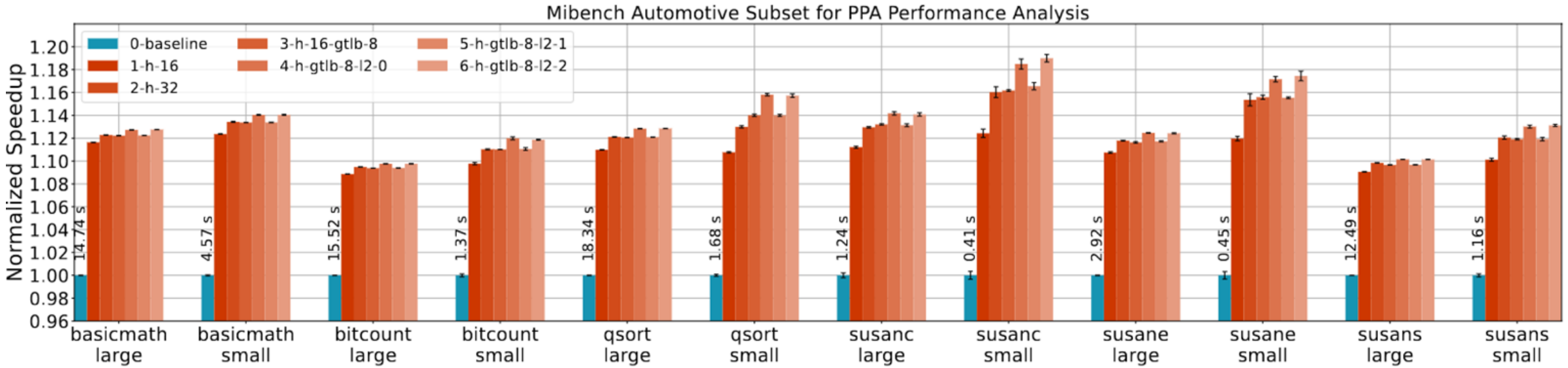
Designs selected for PPA Analysis

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

	SSTC support	I/DTLB #entries	8-entries GTLB	4k L2 TLB	2MB L2 TLB
0-vanilla	x	16	x	x	x
1-h-16	✓	16	x	x	x
2-h-32	✓	32	x	x	x
3-h-gtlb-8	✓	16	✓	x	x
4-h-gtlb-8-l2-0	✓	16	✓	✓	x
5-h-gtlb-8-l2-1	✓	16	✓	x	✓
6-h-gtlb-8-l2-2	✓	16	✓	✓	✓

TABLE V

Performance Speedup

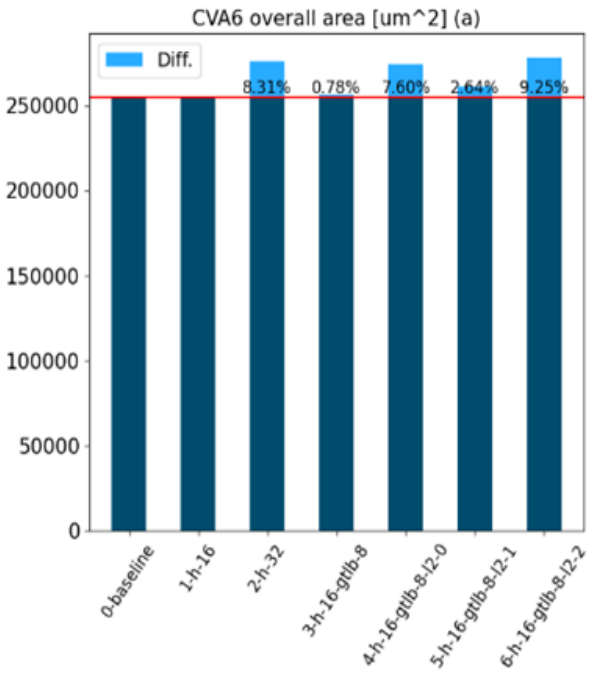


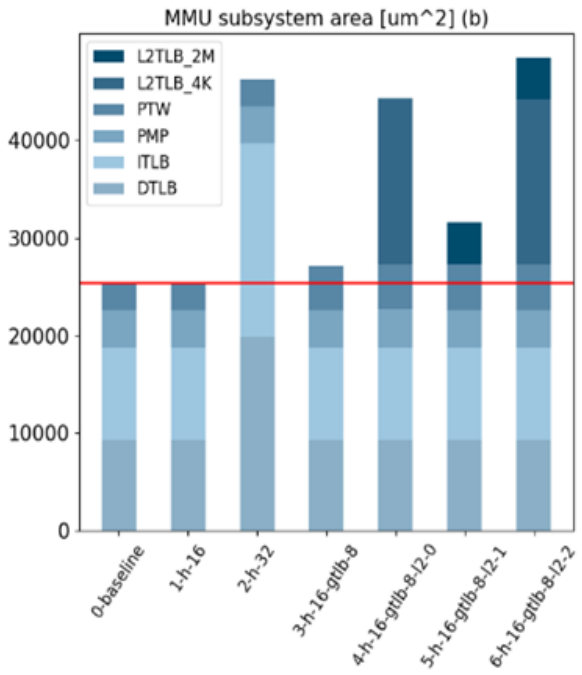
Sstc extension brings on average a **10% functional performance speed up**

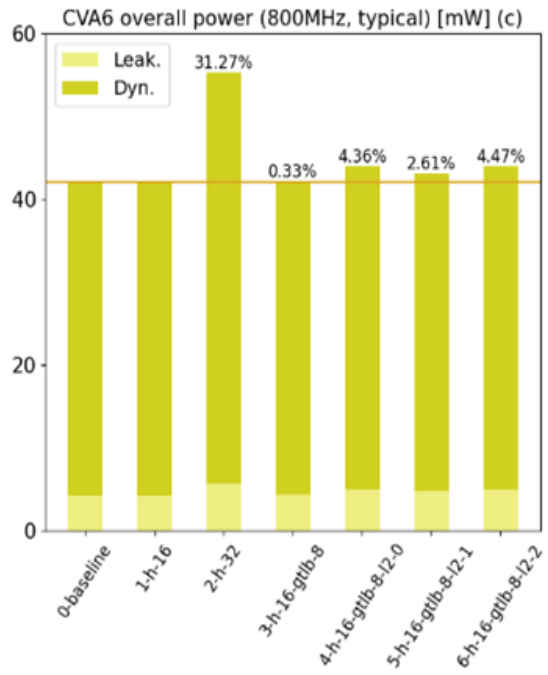
GTLB + Sstc support achieves a **minimum and a maximum performance** speedup of **9%** and **16%**

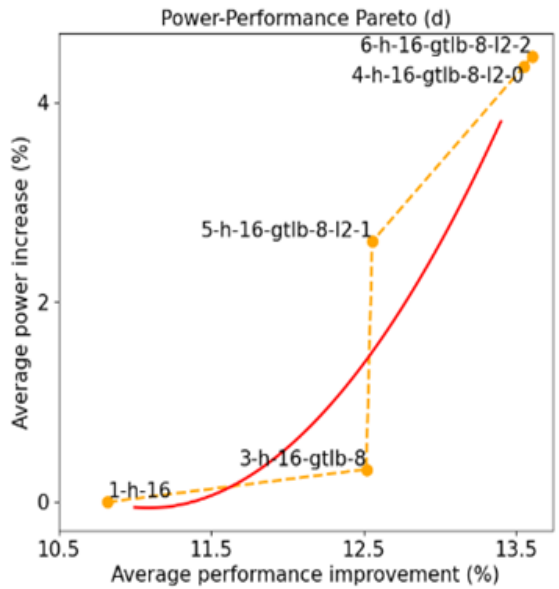
Full max optimized (GTLB + SSTC+L2 TLB) achieves a **17% max performance increase** for ***susanc-small***

Power, Performance and Area Analysis







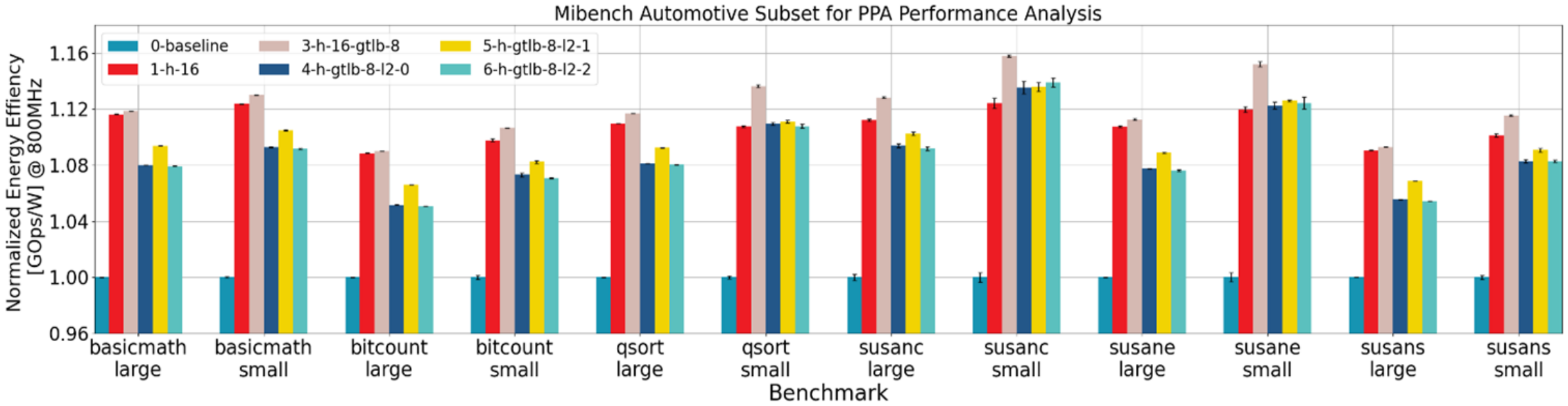


SSTC extension has a negligible impact on the area

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

Energy Efficiency



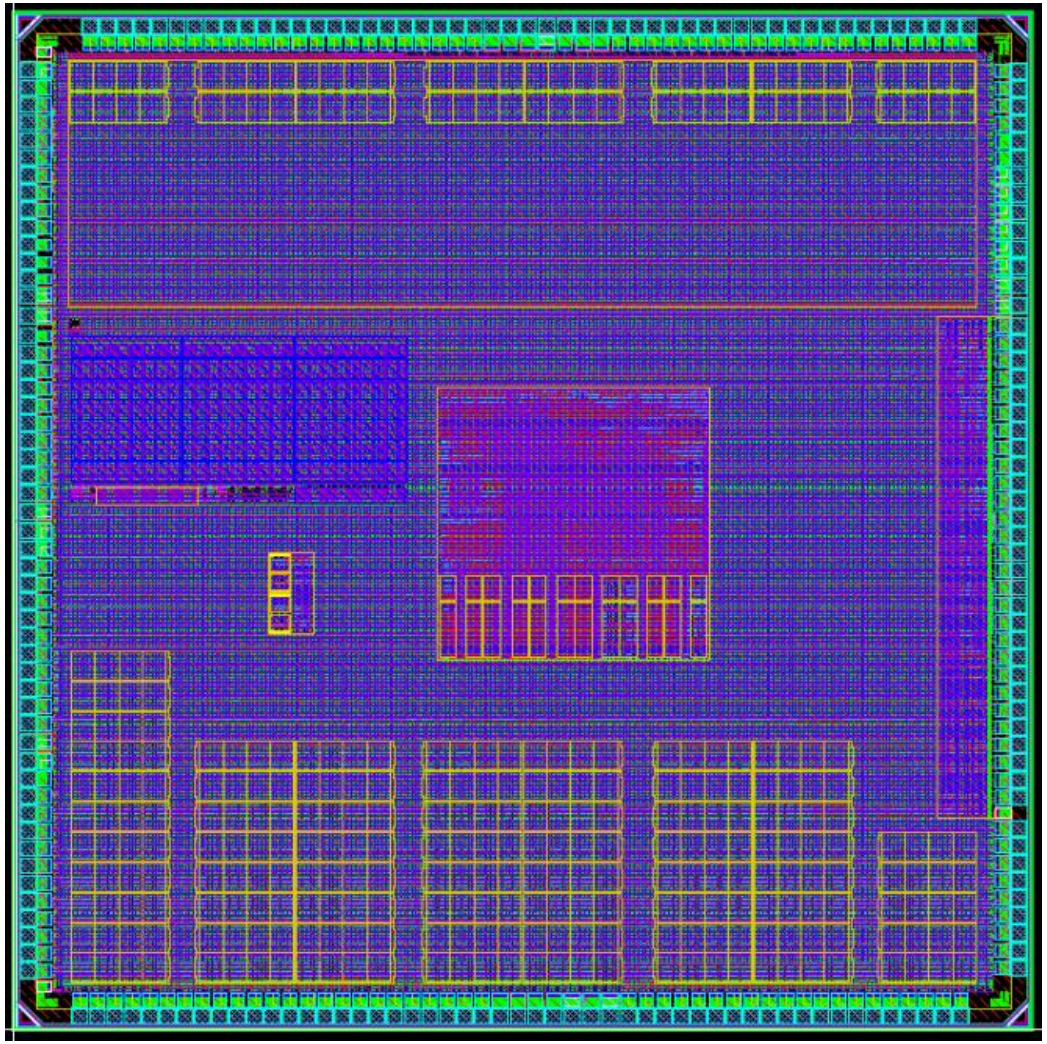
All configurations increase the energy efficient up to 16%

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**

Technology Innovation Institute



THANK YOU!

id10037@alunos.uminho.pt

Q&A

GET IN TOUCH WITH US!

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