Some elements on RISC-V and cache mechanisms

SCRATCHS project kickoff

Pascal Cotret, pascal.cotret@ensta-bretagne.fr
$^{\alpha}$ ENSTA Bretagne

November 22, 2021

Outline

1. Some elements on RISC-V + comparison
2. LiteX: a framework to build SoCs in Python
3. Some cache mechanisms that may be interesting

Some elements on RISC-V + comparison

RISC-V [7]

Unprivileged architecture:

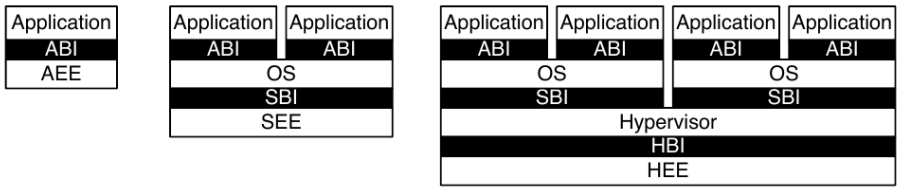
- Modular design
- Base and extensions
- Standard modules can be combined without conflict

Base	Version	Status
RVWMO	2.0	**Ratified**
RV32I	**2.1**	**Ratified**
RV64I	**2.1**	**Ratified**
RV32E	*1.9*	*Draft*
RV128I	*1.7*	*Draft*
Extension	Version	Status
Zifencei	**2.0**	**Ratified**
Zicsr	**2.0**	**Ratified**
M	**2.0**	**Ratified**
A	*2.0*	Frozen
F	**2.2**	**Ratified**
D	**2.2**	**Ratified**
Q	**2.2**	**Ratified**
C	**2.0**	**Ratified**
Ztso	*0.1*	*Frozen*
Counters	*2.0*	*Draft*
L	*0.0*	*Draft*
B	*0.0*	*Draft*
J	*0.0*	*Draft*
T	*0.0*	*Draft*
P	*0.2*	*Draft*
V	*0.7*	*Draft*
N	*1.1*	*Draft*
Zam	*0.1*	*Draft*

RISC-V [7]

Privileged architecture:

- Machine ISA
- Supervisor ISA
- Supports full virtualization



Custom instructions and ongoing extensions

inst[4:2] / inst[6:5]	000	001	010	011	100	101	110	111 ($> 32b$)
00	LOAD	LOAD-FP	*custom-0*	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	$48b$
01	STORE	STORE-FP	*custom-1*	AMO	OP	LUI	OP-32	$64b$
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2/rv128*	$48b$
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	*reserved*	*custom-3/rv128*	$\geq 80b$

- Cache Management Operations for RISC-V:
  https://github.com/riscv/riscv-CMOs

Existing “application-class” RISC-V cores (mainly 64-bit)[1]

	Rocket	BOOM	CVA6	SHAKTI
Bits	32/64	64	64	32/64
Stages	5	10	6	5
Extensions	MAFDC	MAFDC	MAFDC	MAFDC
OoO exec	no	yes	no	no
Funct. Units	4	8	6	3
Inferfacing	TileLink	TileLink	AXI4	AXI4/TL
HDL	Chisel	Chisel	SV	BSV
License	BSD	BSD	SolderPad	BSD
Framework	Chipyard	Chipyard	OpenPiton	shakti-soc
Commit	1872f5d[1]	d77c2c3[2]	1793be6[3]	884fc43[4]

[1] https://github.com/chipsalliance/rocket-chip/
[2] https://github.com/riscv-boom/riscv-boom/
[3] https://github.com/openhwgroup/cva6/
[4] https://gitlab.com/shaktiproject/cores/c-class/

[1] Alexander Dörflinger et al. “A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations”. In: *Proceedings of the 18th ACM International Conference on Computing Frontiers*. CF ’21. Virtual Event, Italy: Association for Computing Machinery, 2021, 12–20. ISBN: 9781450384049. DOI: 10.1145/3457388.3458657. URL: https://doi.org/10.1145/3457388.3458657.

Small-scale cores [2] (MECO 2019)

Name of processor	Instruct. Set Architecture and Data Width	No.of Pipeline Stages	Bus Architecture	MMU	FPU	HDL	Compiler	Debug Support	License	Last Update	Multi-Core	In Order	Cache	JTAG	Peripherals Included
Amber	ARM v2a, 32 bit	5	WB	N	N	Verilog	GCC	Y	LGPL	2017	N	Y	Y	N	Y
Lattice Mico 32	LatticeMico32, 32 bit	6	WB	N	N	Verilog	GCC	Y	GPL	2017	N	Y	Y	Y	Y
openrisc	ORBIS, 32 bit	5	WB	Y	Y	Verilog	GCC	Y	LGPL	2019	N	Y	Y	N	Y
Leon3	SPARC V8, 32 bit	7	AHB	Y	Y	VHDL	GCC	Y	GPL	2018	Y	Y	Y	Y	Y
freedom	RISC-V, 32 bit	5	TL/AXI	N	Y	Chisel	GCC	Y	BSD	2018	N	Y	Y	Y	Y
ORCA	RISC-V, 32 bit	5	WB/AXI	N	N	VHDL	GCC	N	BSD	2019	N	Y	Y	N	N
RI5CY	RISC-V, 32 bit	4	AXI	N	Y	Verilog	GCC	Y	Solderpad	2018	N	Y	N	Y	Y
zero-riscy	RISC-V, 32 bit	2	AXI	N	N	Verilog	GCC	Y	Solderpad	2018	N	Y	N	N	Y
OPenV	RISC-V, 32 bit	3	AXI	N	N	Verilog	GCC	N	MIT	2018	N	Y	N	N	Y
VexRiscv	RISC-V, 32 bit	5	AXI	Y	N	SpinalHDL	GCC	Y	MIT	2019	N	Y	Y	Y	Y
Roa Logic RV12	RISC-V, 32 bit	6	AHB/WB	N	N	Verilog	GCC	Y	Non	2018	N	Y	Y	N	N
SCR1	RISC-V, 32 bit	4	AXI	N	N	Verilog	GCC	Y	Solderpad	2019	N	Y	N	Y	N
Hummingbird E200	RISC-V, 32 bit	2	AXI	N	N	Verilog	GCC	Y	Apache	2019	N	Y	Y	Y	Y
Shakti	RISC-V, 32 bit	3	AXI	N	N	Bluespec	GCC	N	BSD	2019	N	Y	Y	N	Y
ReonV	RISC-V, 32 bit	7	AHB	Y	Y	VHDL	GCC	Y	GPL v3	2018	Y	Y	Y	Y	Y
PicoRV32	RISC-V, 32 bit	0	AXI	N	N	Verilog	GCC	N	ISC	2018	N	Y	N	N	Y
SweRV EH1	RISC-V, 32 bit	9	AXI	N	N	Verilog	GCC	Y	Apache	2019	N	Y	Y	Y	N
Taiga	RISC-V, 32 bit	3±	AXI	Y	N	Verilog	GCC	N	Apache	2018	N	Y	Y	N	N
potato	RISC-V, 32 bit	5	WB	N	N	VHDL	GCC	N	BSD	2018	N	Y	Y	N	Y

OpenHWgroup implementations - CVA6 [4]

Core	License	Language	ISA	Pipeline structure
CVA6	Solderpad	SystemVerilog	RV64IMAC	6-stage
CV32E40P	Solderpad	SystemVerilog	RV32IM[F]C	4-stage

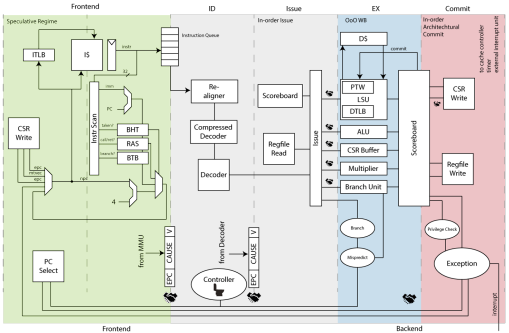


Figure: `https://cva6.readthedocs.io/en/latest/index.html`

OpenHWgroup implementations - CV32E40P [5]

Core	License	Language	ISA	Pipeline structure
CVA6	Solderpad	SystemVerilog	RV64IMAC	6-stage
CV32E40P	Solderpad	SystemVerilog	RV32IM[F]C	4-stage

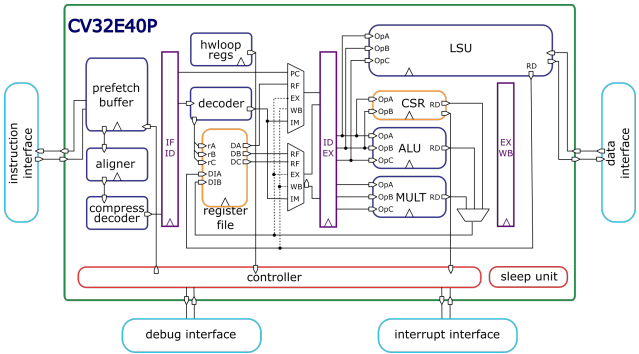


Figure: https://cv32e40p.readthedocs.io/en/latest/

LiteX: a framework to build SoCs in Python

Building RISC-V based SoCs

Methodology	Maintainability	Language	Dev. cost	SW layer
By hand	-	TBD	+++	TBD
Chipyard[2]	+	Bash	-	Simulation with Spike
LiteX[3]	+	Python	-	Spike simulation HDL simulation Buildroot/OpenSBI

[2] https://github.com/ucb-bar/chipyard
[3] https://github.com/enjoy-digital/litex

LiteX example

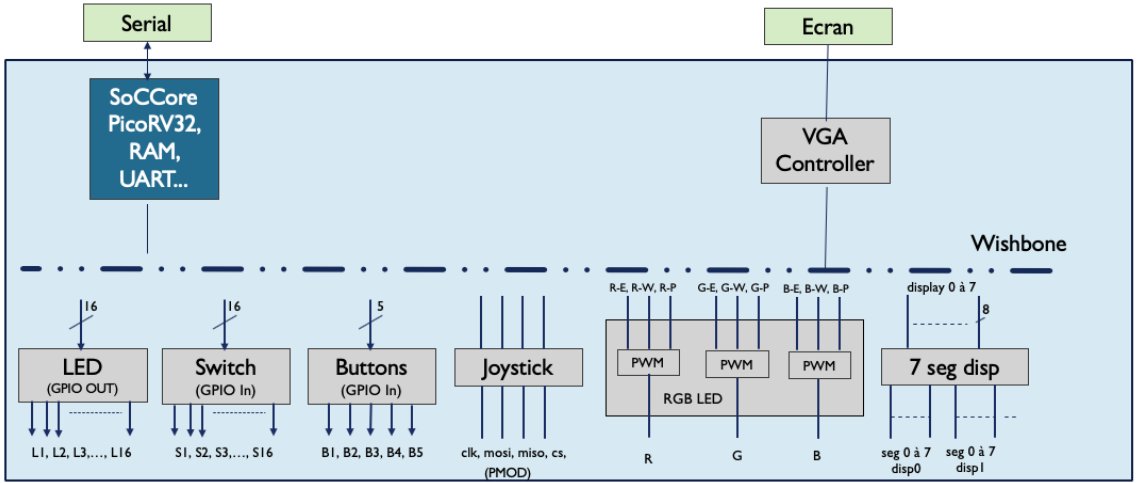


Figure: https://pcotret.github.io/ENSTAB-RISCV

LiteX example

```
class BaseSoC(SoCCore):
  def __init__(self):
    platform = nexys4ddr.Platform()
    # SoC with CPU
    SoCCore.__init__(self, platform,
    cpu_type="picorv32",
    clk_freq=100e6,
    integrated_rom_size=0x8000,
    integrated_main_ram_size=16*1024)
    # Adding a peripheral
    SoCCore.add_csr(self,"joystick")
    self.submodules.joystick =
    spijoystick.SpiJoystick(platform.request("joystick"))
```

Some LiteX features

- Several cores available (including the CV32E40P) : https://bit.ly/3CDs1DK
- Custom CPUs can be easily added (Python wrapper)
- Multi-CPU SoC may be available w/ minor changes: https://bit.ly/3FC8LbS

- Adding a peripheral: a few Python lines
- Example SoC to run a Buildroot Linux on VexriscV: https://bit.ly/3FxXLfB

Some cache mechanisms that may be interesting

RPCache[4]

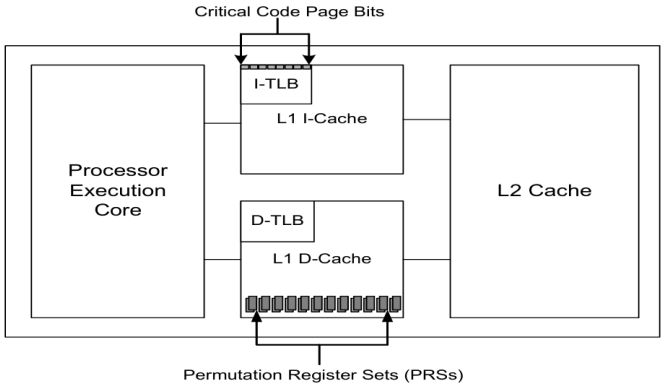


Figure 9. A processor with RPCache

[4] Zhenghong Wang and Ruby B. Lee. "Covert and Side Channels Due to Processor Architecture". In: *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*. 2006, pp. 473–482. DOI: 10.1109/ACSAC.2006.20.

Partitioned cache architecture[5]

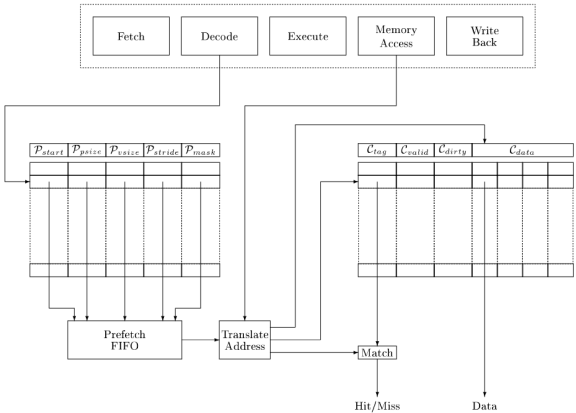


Fig. 2. A block diagram describing a partitioned cache.

[5] D. Page. *Partitioned Cache Architecture as a Side-Channel Defence Mechanism*. https://eprint.iacr.org/2005/280.

Cache-based SCAs[6]

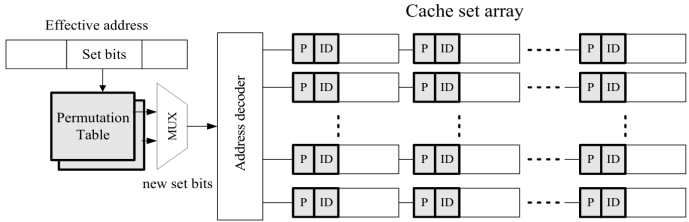


Figure 5. A logical view of the RPcache

This paper has some other interesting details on RPCache.

[6] Zhenghong Wang and Ruby B. Lee. "New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks". In: *SIGARCH Comput. Archit. News* 35.2 (2007), 494–505. ISSN: 0163-5964. DOI: 10.1145/1273440.1250723. URL: https://doi.org/10.1145/1273440.1250723.

References I

[1] Alexander Dörflinger et al. “A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations”. In: *Proceedings of the 18th ACM International Conference on Computing Frontiers*. CF ’21. Virtual Event, Italy: Association for Computing Machinery, 2021, 12–20. ISBN: 9781450384049. DOI: 10.1145/3457388.3458657. URL: https://doi.org/10.1145/3457388.3458657.

[2] Roland Höller et al. “Open-Source RISC-V Processor IP Cores for FPGAs — Overview and Evaluation”. In: *2019 8th Mediterranean Conference on Embedded Computing (MECO)*. 2019, pp. 1–6. DOI: 10.1109/MECO.2019.8760205.

[3] Tao Lu. *A Survey on RISC-V Security: Hardware and Architecture*. https://arxiv.org/pdf/2107.04175.pdf.

References II

[4] OpenHwgroup. *CVA6 RISC-V CPU*. https://github.com/openhwgroup/cva6.

[5] OpenHwgroup. *OpenHW Group CORE-V CV32E40P RISC-V IP*. https://github.com/openhwgroup/cv32e40p.

[6] D. Page. *Partitioned Cache Architecture as a Side-Channel Defence Mechanism*. https://eprint.iacr.org/2005/280.

[7] Benjamin Rohr and Moritz Waser. *Soft Cores and ARM/RISC-V Processors*. https://www.iaik.tugraz.at/wp-content/uploads/2020/07/softcores.pdf.

[8] Zhenghong Wang and Ruby B. Lee. "Covert and Side Channels Due to Processor Architecture". In: *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*. 2006, pp. 473–482. DOI: 10.1109/ACSAC.2006.20.

References III

[9] Zhenghong Wang and Ruby B. Lee. “New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks”. In: *SIGARCH Comput. Archit. News* 35.2 (2007), 494–505. ISSN: 0163-5964. DOI: 10.1145/1273440.1250723. URL: https://doi.org/10.1145/1273440.1250723.