CV32E40* FEATURES AND PARAMETERS

Revision History

Revision	Date	Author	Comment
v1.0_draft00	12/17/2019	A.Bink	• Initial Revision

Table of contents

1 References 5
2 Introduction 6
3 Features & parameters 7
4 Xpulp extensions 11
4.1 Post-incrementing load & store 11
4.2 Hardware loop 11
4.3 Bit manipulation 11
4.4 General ALU operations 12
4.5 Immediate branching instructions 12
4.6 Multiply-Accumulate 13
4.7 SIMD (renamed from ‘Vectorial’ to avoid confusion) 13

List of tables

Table 1 Proposed Feature List.....8
Table 2 Proposed Parameter List.....10

1 References

Reference ID	Reference Title	Comment/Location
[RV_UNPRIVILEGED_ISA]	The RISC-V Instruction Set Manual Volume I: Unprivileged ISA Document Version 20190608-Base-Ratified	https://riscv.org/specifications/
[RV_PRIVILEGED_ISA]	The RISC-V Instruction Set Manual Volume II: Privileged Architecture. Document Version 20190608-Priv-MSU-Ratified	https://riscv.org/specifications/privileged-isa/
[RV_DEBUG]	RISC-V External Debug Support Version 0.13.2	https://riscv.org/specifications/debug-specification/
[RI5CY_USER_MANUAL]	RI5CY: User Manual. August 2019. Revision 4.1	https://github.com/pulp-platform/riscv/blob/master/doc/user_manual.doc
[RVI]	RISC-V Interface specification	
[CLIC]	RISC-V Core-Local Interrupt Controller (CLIC) Version 0.9-draft-20191208	https://github.com/riscv/riscv-fast-interrupt/blob/master/clic.adoc

2 Introduction

This document describes the proposed (high-level) features and parameters of the CV32E40P and CV32E40 OpenHW RISC-V cores.

The goal of the document is to come to a common understanding of the supported and not-supported features as well as the parameter (combinations) that are required to be verified.

The (initially) proposed feature and parameter list for CV32E40P is a rather aggressive cut-down when compared to RI5CY. The primary reason for that is to align this CPU (and its verification) with the OpenHW BHAG, which has a proposed tape-out date of July 2020.

3 Features & parameters

Table 1 shows the features that are under discussion for CV32E40P and CV32E40. For each feature the following options are possible:

- Yes

  Feature will be unconditionally implemented and verified.

- No

  Feature will not be implemented.

- Optional

  Feature inclusion depends on a hardware/RTL parameter. If the feature involves compiler support, then the compiler should be configurable (via a command line option) to include/exclude the feature as well. Optional features are fully verified. Note that optional features expand the verification space as normally all option combinations need to be verified.

- Tentative

  Feature inclusion depends on a hardware/RTL parameter. If the feature involves compiler support, then the compiler should be configurable (via a command line option) to include/exclude the feature. Tentative features are considered low priority and are allowed to be partially verified (they are the first features to be skipped if so demanded by the BHGA schedule).

- TBD

  Not decided yet.

Table 1 Proposed Feature List

Feature	RI5CY	CV32E40P	CV32E40	Documentation
Base instruction set plus standard instruction extensions				
RV32I	Yes	Yes	Yes	v2.1 ([RV_PRIVILEGED_ISA])
Zifencei extension	Yes	Yes	Yes	v2.0 ([RV_PRIVILEGED_ISA])
Zicsr extension	Yes	Yes	Yes	v2.0 ([RV_PRIVILEGED_ISA])
M extension	Yes	Yes	Yes	v2.0 ([RV_PRIVILEGED_ISA])
F extension	Optional	Tentative	Optional	v2.2 ([RV_PRIVILEGED_ISA])
Zfinx extension	Optional	TBD	TBD	Not standardized yet
C extension	Yes	Yes	Yes	v2.0 ([RV_PRIVILEGED_ISA])
B extension	No	No	Yes	TBD ([RV_PRIVILEGED_ISA])
P extension	No	No	Yes	TBD ([RV_PRIVILEGED_ISA])
N extension	No	No	No	Not standardized yet
Counters extension	No	Yes[9]	Yes[9]	([RV_PRIVILEGED_ISA])
Privileged spec				
User mode	Optional	No	Yes	([RV_PRIVILEGED_ISA])
PMP	Optional[1,4]	No	Yes[5]	
Xpulp instruction extensions				
Post-increment load/store	Yes	Yes	Yes	[RI5CY_USER_MANUAL]
Hardware Loop	Yes	No[6]	No[6]	[RI5CY_USER_MANUAL]
Bit Manipulation	Yes	Yes	No[7]	[RI5CY_USER_MANUAL]
General ALU	Yes	Yes	Yes	[RI5CY_USER_MANUAL]
Immediate branching	Yes	Yes	Yes	[RI5CY_USER_MANUAL]
SIMD	Yes	TBD[13]	No[8]	[RI5CY_USER_MANUAL]
Custom circuitry				
RI5CY performance counters	Yes	No[10]	No[10]	[RI5CY_USER_MANUAL]
Advanced Processing Unit itf	Yes	No	Yes	
128-bit wide Instruction Bus itf	Optional	No	No	
RI5CY interrupt scheme[2]	Yes	No	No	
PULP cluster itf	Yes	No	No	
Sleep interface	No	Yes	Yes	https://github.com/pulp-platform/riscv/issues/131
Future extensions				
Stack overflow protection	No	No	Yes	https://github.com/pulp-platform/riscv/issues/183
Interrupts				
CLINT	No	Yes	No	
CLINT extension (MIP2, MIE2)	No	TBD	No	
CLIC	No	No	Yes	Version TBD ([CLIC])
Debug & Trace				
Debug	Yes[14]	Yes[14]	Yes[14]	Version 0.13.2 ([RV_DEBUG])
Trigger module[12]	No	Yes	Yes	Version 0.13.2 ([RV_DEBUG])
Trace	No	No	Yes	Version TBD ()
RVI-compliant interface[11]				
RVI Instruction Bus interface	No[3]	Yes	Yes	([RVI])
RVI Data Bus interface	No[3]	Yes	Yes	([RVI])

[1] In RI5CY the User mode and PMP options are tied together. It should however be possible to have a PMP on a CPU without User mode.
[2] RI5CY proprietary interrupt scheme including secure interrupt. Proposal is to replace this by standard CLINT or CLIC.
[3] RI5CY would be compliant if https://github.com/pulp-platform/riscv/issues/126, https://github.com/pulp-platform/riscv/issues/127, https://github.com/pulp-platform/riscv/issues/128 are solved.
[4] RI5CY PMP does not support MPRV and LOCK.
[5] PMP version is TBD. Support for MPRV and LOCK is TBD.

[6] The reasons to propose 'no' for Hardware Loops are the expected verification effort and the associated increased interrupt latency. It is furthermore not clear yet if/how Hardware Looping can work together with trace in the future. If OpenHW decides to include Hardware Loops, then our proposal would be to make it optional (both in RTL and compiler).
[7] To be replaced by RISC-V standard B extension.
[8] To be replaced by RISC-V standard P extension.
[9] Interface to and configurability of additional hardware performance counters (hpmcounter3(h)-hpmcounter31(h) is TBD.
[10] Will be replaced by RISC-V standard counters extension plus TBD interface.
[11] RVI stands for RISC-V (Bus) Interface. It is an interface with the same pinout as currently used on RI5CY, but with additional rules/constraints to make it a better fit for high-performance implementations and to allow efficient (external) adapters to AMBA protocols.
[12] The Trigger module is an optional part of the [RV_DEBUG] specification, but it is essential when debugging code from ROM. Most importantly it provides hardware breakpoint functionality which is considered a must-have.
[13] We are okay with excluding SIMD in order to ease the BHAG related verification schedule.
[14] We are proposing to only implement/verify the abstract access and not the (also optional) program buffer.

Table 2 shows whether specific RTL parameters are present or not. Note that if a parameter is not included it does not mean that the related feature is not present; it just means that the feature is not optional. For example, CV32E40 is proposed to unconditionally include a PMP, so USE_PMP is no longer needed as a parameter.

Table 2 Proposed Parameter List

Feature	RI5CY	CV32E40P	CV32E40	Legal Range
N_EXT_PERF_COUNTERS	Yes	No	No	
INSTR_RDATA_WIDTH	Yes	No (only 32-bit)	No (only 32-bit)	{32, 128}
PULP_SECURE	Yes	No (USER mode excluded)	No (USER mode included)	
N_PMP_ENTRIES	Yes	No	Yes	
USE_PMP	Yes	No (PMP excluded)	No (PMP included)	
PULP_CLUSTER	Yes	No	TBD	{0, 1}
FPU	Yes	Yes[2]	Yes[2]	{0, 1}
ZFINX[1]	Yes	TBD	Yes	
FP_DIVSQRT	Yes	No (controlled by FPU)[2]	No (controlled by FPU)[2]	{0, 1}
SHARED_FP	Yes	No	TBD	
SHARED_DSP_MULT	Yes	No	TBD	
SHARED_INT_MULT	Yes	No	TBD	
SHARED_INT_DIV	Yes	No	TBD	
SHARED_FP_DIVSQRT	Yes	No	TBD	
WAPUTYPE	Yes	No	TBD	
APU_NARGS_CPU	Yes	No	TBD	
APU_WOP_CPU	Yes	No	TBD	
APU_NDSFLAGS_CPU	Yes	No	TBD	
APU_NUSFLAGS_CPU	Yes	No	TBD	
DM_HALTADDRESS[1]	Yes	Yes	Yes	

[1] Parameters should be all capitals.
[2] The proposal is to redefine the FPU parameter to control both FPU and FP_DIVSQRT inclusion.

4 Xpulp extensions

This chapter list the Xpulp instructions from [RI5CY_USER_MANUAL]. The reason to repeat them here is to discuss their grouping. E.g. some of the *general ALU instructions* (e.g. *min*, *max*) should maybe be categorized as bit manipulation instructions. For CV32E40P the grouping is not really relevant; for CV32E40 the grouping becomes relevant as the current proposal is to exclude the Xpulp *bit manipulation* and *SIMD* extensions in favor of the standard RISC-V *B* and *P* extensions.

4.1 Post-incrementing load & store

- Register-Immediate Loads with Post-Increment
  - p.lb rD, Imm(rs1!)
  - p.lbu rD, Imm(rs1!)
  - p.lh rD, Imm(rs1!)
  - p.lhu rD, Imm(rs1!)
  - p.lw rD, Imm(rs1!)
- Register-Register Loads with Post-Increment
  - p.lb rD, rs2(rs1!)
  - p.lbu rD, rs2(rs1!)
  - p.lh rD, rs2(rs1!)
  - p.lhu rD, rs2(rs1!)
  - p.lw rD, rs2(rs1!)
- Register-Register Loads
  - p.lb rD, rs2(rs1)
  - p.lbu rD, rs2(rs1)
  - p.lh rD, rs2(rs1)
  - p.lhu rD, rs2(rs1)
  - p.lw rD, rs2(rs1)
- Register-Immediate Stores with Post-Increment
  - p.sb rs2, Imm(rs1!)
  - p.sh rs2, Imm(rs1!)
  - p.sw rs2, Imm(rs1!)
- Register-Register Stores with Post-Increment
  - p.sb rs2, rs3(rs1!)
  - p.sh rs2, rs3(rs1!)
  - p.sw rs2, rs3(rs1!)
- Register-Register Stores
  - p.sb rs2, rs3(rs1)
  - p.sh rs2 rs3(rs1)
  - p.sw rs2, rs3(rs1)

4.2 Hardware loop

- Long Hardware Loop Setup instructions
  - lp.starti L, uimmL
  - lp.endi L, uimmL
  - lp.count L, rs1
  - lp.counti L, uimmL
- Short Hardware Loop Setup Instructions
  - lp.setup L, rs1, uimmL
  - lp.setupi L, uimmL, uimmS

4.3 Bit manipulation

- Bit manipulation instructions
  - p.extract rD, rs1, Is3, Is2
  - p.extractu rD, rs1, Is3, Is2
  - p.extractr rD, rs1, rs2
  - p.extractur rD, rs1, rs2
  - p.insert rD, rs1, Is3, Is2
  - p.insertr rD, rs1, rs2
  - p.bclr rD, rs1, Is3, Is2
  - p.bclrr rD, rs1, rs2
  - p.bset rD, rs1, Is3, Is2
  - p.bsetr rD, rs1, rs2
  - p.ff1 rD, rs1
  - p.fl1 rD, rs1
  - p.clb rD, rs1
  - p.cnt rD, rs1
  - p.ror rD, rs1, rs2

4.4 General ALU operations

- General ALU operations
  - p.abs rD, rs1
  - p.slet rD, rs1, rs2
  - p.sletu rD, rs1, rs2
  - p.min rD, rs1, rs2 (TBD: move to Bit Manipulation group?)
  - p.minu rD, rs1, rs2 (TBD: move to Bit Manipulation group?)
  - p.max rD, rs1, rs2 (TBD: move to Bit Manipulation group?)
  - p.maxu rD, rs1, rs2 (TBD: move to Bit Manipulation group?)
  - p.exths rD, rs1
  - p.exthz rD, rs1
  - p.extbs rD, rs1
  - p.extbz rD, rs1
  - p.clip rD, rs1, Is2
  - p.clipr rD, rs1, rs2
  - p.clipu rD, rs1, Is2
  - p.clipur rD, rs1, rs2
  - p.addN rD, rs1, rs2, Is3
  - p.adduN rD, rs1, rs2, Is3
  - p.addRN rD, rs1, rs2, Is3
  - p.adduRN rD, rs1, rs2, Is3
  - p.addNr rD, rs1, rs2
  - p.adduNr rD, rs1, rs2
  - p.addRNr rD, rs1, rs2
  - p.adduRNr rD, rs1, rs2
  - p.subN rD, rs1, rs2, Is3
  - p.subuN rD, rs1, rs2, Is3
  - p.subRN rD, rs1, rs2, Is3
  - p.subuRN rD, rs1, rs2, Is3
  - p.subNr rD, rs1, rs2
  - p.subuNr rD, rs1, rs2
  - p.subRNr rD, rs1, rs2
  - p.subuRNr rD, rs1, rs2

4.5 Immediate branching instructions

- Immediate branching instructions
  - p.beqimm rs1, Imm5, Imm12
  - p.bneimm rs1, Imm5, Imm12

4.6 Multiply-Accumulate

- 32-Bit x 32-Bit Multiplication Operations
  - p.mac rD, rs1, rs2
  - p.msu rD, rs1, rs2
- 16-Bit x 16-Bit Multiplication
  - p.muls rD, rs1, rs2
  - p.mulhhs rD, rs1, rs2
  - p.mulsN rD, rs1, rs2, Is3
  - p.mulhhsN rD, rs1, rs2, Is3
  - p.mulsRN rD, rs1, rs2, Is3
  - p.mulhhsRN rD, rs1, rs2, Is3
  - p.mulu rD, rs1, rs2
  - p.mulhhu rD, rs1, rs2
  - p.muluN rD, rs1, rs2, Is3
  - p.mulhhuN rD, rs1, rs2, Is3
  - p.muluRN rD, rs1, rs2, Is3
  - p.mulhhuRN rD, rs1, rs2, Is3
- 16-Bit x 16-Bit Multiply-Accumulate
  - p.macsN rD, rs1, rs2, Is3
  - p.machhsN rD, rs1, rs2, Is3
  - p.macsRN rD, rs1, rs2, Is3
  - p.machhsRN rD, rs1, rs2, Is3
  - p.macuN rD, rs1, rs2, Is3
  - p.machhuN rD, rs1, rs2, Is3
  - p.macuRN rD, rs1, rs2, Is3
  - p.machhuRN rD, rs1, rs2, Is3

4.7 SIMD (renamed from ‘Vectorial’ to avoid confusion)

- General ALU Instructions
  - pv.add[.sc,.sci]{.h,.b}
  - pv.sub[.sc,.sci]{.h,.b}
  - pv.avg[.sc,.sci]{.h,.b}
  - pv.avgu[.sc,.sci]{.h,.b}
  - pv.min[.sc,.sci]{.h,.b}
  - pv.minu[.sc,.sci]{.h,.b}
  - pv.max[.sc,.sci]{.h,.b}
  - pv.maxu[.sc,.sci]{.h,.b}
  - pv.srl[.sc,.sci]{.h,.b}
  - pv.sra[.sc,.sci]{.h,.b}
  - pv.sll[.sc,.sci]{.h,.b}
  - pv.or[.sc,.sci]{.h,.b}
  - pv.xor[.sc,.sci]{.h,.b}
  - pv.and[.sc,.sci]{.h,.b}
  - pv.abs{.h,.b}
  - pv.extract.h
  - pv.extract.b
  - pv.extractu.h
  - pv.extractu.b

  - pv.insert.h
  - pv.insert,b
- Dot Product Instructions
  - pv.dotup[.sc,.sci].h
  - pv.dotup[.sc,.sci].b
  - pv.dotusp[.sc,.sci].h
  - pv.dotusp[.sc,.sci].b
  - pv.dotsp[.sc,.sci].h
  - pv.dotsp[.sc,.sci].b
  - pv.sdotup[.sc,.sci].h
  - pv.sdotup[.sc,.sci].b
  - pv.sdotusp[.sc,.sci].h
  - pv.sdotusp[.sc,.sci].b
  - pv.sdotsp[.sc,.sci].h
  - pv.sdotsp[.sc,.sci].b
- Shuffle and Pack Instructions
  - pv.shuffle.h
  - pv.shuffle.sci.h
  - pv.shuffle.b
  - pv.shuffleI0.sci.b
  - pv.shuffleI1.sci.b
  - pv.shuffleI2.sci.b
  - pv.shuffleI3.sci.b
  - pv.shuffle2.h
  - pv.shuffle2.b
  - pv.pack.h (TBD: move to Bit Manipulation group?)
  - pv.packhi.b
  - pv.packlo.b
- Vectorial Comparison Operations
  - pv.cmpeq[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmpne[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmpgt[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmpge[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmplt[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmple[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmpgtu[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmpgeu[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmpltu[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}
  - pv.cmpleu[.sc,.sci]{.h,.b} rD, rs1, {rs2, Imm6}