CV32E4* cores family discussion

Manuel Pezzin

April 7, 2025

Current situation overview (personal point of view)

- 5 Official OHG projects / cores
  - cv32e40pv1
  - cv32e40pv2 (same repo as cv32e40pv1)
  - ~~cv32e41p (forked from cv32e40pv1)~~
  - cv32e40x (forked from cv32e40pv1)
  - cv32e40s (forked from cv32e40pv1 – sync with latest version of cv32e40x)
  - cv32e40px (forked from cv32e40pv2 – sync with latest version)
- Fragmented RTL/Verification code base
  - Separate verification meetings
  - Separate git repos for each core
  - Different verification environments, hosted on independent dev branches of core-v-verif that have diverged a lot from master and even in different git repos.
  - CAD tools used for design and verification ?
    - Same simulators / lint / formal / etc tools ?
    - Same version of Synopsys/Imperas DV ref model ?
    - What about Open source CAD tools support ?
    - What about SW / compiler support ?
- Low coordination between OHG partners
  - 3/4 separate companies led 4 projects (design and verification) that address primarily their own needs
  - Are these projects relevant for other end-users ?
- Low involvement of academic community
  - Some historical academic developers are OHG members but don't participate directly… Did we encourage / discourage them ?
  - Design/verif tools choice (commercial / signoff vs. open source ones) may be one of the reasons
- Too few contributors/volunteers left to maintain such a huge and fragmented code base !
  - OHG / EPFL
  - CEA
  - CIRRUS LOGIC
  - Pascal (as an individual volunteer member, during his spare time)
  - Others ?

RTL code status (As far as I understand it)

- cv32e40pv1
  - Credits (main ones) : UNIBO/ETHZ, SILABS, OHG
  - finished v1.0.0 - closed
- cv32e40pv2 (same repo as cv32e40pv1)
  - Credits (main ones) : DOLPHIN, OHG/EPFL, CIRRUS LOGIC, SILABS
  - almost finished v1.8.3 - Orphaned
  - Github public forks : 445 – 25 active over 1 year
- cv32e41p (forked from cv32e40pv1)
  - Credits (main ones) : OHG/EPFL/ETHZ, HUAWEI, SILABS
  - Proof of concept - Closed
  - Github public forks : 11 – none active
- cv32e40x (forked from cv32e40pv1)
  - Credits (main ones) : SILABS
  - on hold v0.10.0 - Orphaned
  - Github public forks : 53 - 11 active over 1 year
- cv32e40s (forked from cv32e40pv1 – sync with latest version of cv32e40x)
  - Credits (main ones) : SILABS
  - on hold v0.10.0 - Orphaned
  - Github public forks : 24 – 3 active over 1 year
- cv32e40px (forked from cv32e40pv2 – sync with latest version)
  - Credits (main ones) : OHG/EPFL, CEA
  - Development started
  - 2 Active forks
    - EPFL working on CV-X-IF
    - CEA working on RVP extensions support
  - Github public forks : 3 – 2 active over 1 year
- Non-OHG versions
  - Many forks maintained by academic community
  - Several closed-source forks / clones used in silicon products

More General thoughts

- Proposed CV32E4* projects are too big and too ambitious for our (not big enough) community
  - → we must (re-)build an active community
  - → we must attract more end-users (SW / system people) to help us scheduling work in accordance with their actual short / medium / long term needs
  - → we must attract more industrial partners to help us on stabilization, verification and release process
  - → we must be more welcoming to academic community to help us on innovative / disruptive stuff
- Partners come and leave
  - this is normal business life → we have to manage that properly
  - → We should make our projects more resilient to partners departure and make life easier for newcomers / contributors
- Huge feature sets means huge verification effort → delayed releases or even no release at all (when project lead leaves the boat)
- Initial project goals may not be relevant after 1 or 2 years
  - → we should consider a divide and conquer approach : “release less, release often” model
  - → focus on stabilizing features rather than introducing new ones (without sacrificing innovation)
  - → Review/revise regularly project plans regarding targeted end-user of our work
- We should think about a sustainable maintenance/support model for released cores
  - Maintaining old code can be time consuming, in particular if original designers are no longer here
  - Features are more important than the code itself → “upstream first” approach (make recent version backward-compatible and fix issues on recent version rather than old one)
- HW features are irrelevant without proper SW support → features implementation strategy must be coordinated between HW, SW and verification (ISS / models in particular)
- CVA6 development model is a source of inspiration
  - Single shared RTL code base
  - Proper, centralized CI(/CD?) environment
  - Integration of new features is non-regression driven
  - Cores are actually a set of supported configurations, verified properly

Proposed new development model: initial setup

- First objective: release CV32E40Pv2 (according to updated release criteria) to have a stable RTL+verif baseline
- Rebuild a single git tree for RTL (and verif ?)
  - Freeze existing CV32E40P/S/X/PX RTL and verif git trees
  - Rebuild a new single one (CV32E4 ?) based on most stable / verified code base : a cleaned-up version of CV32E40P tree (without unmerged PULP branches) seems a good candidate
  - Should we also include CV32E40X/S history (for later merge / cherry-pick of features) ?
- Update existing CV32E40Pv2 verification environment for broader usage / adoption:
  - Reference model: add possibility to use an alternative to Imperas-DV for broader adoption (envisioned option is spike for consistency with CVA6/CVE2 verif environments)
  - Restore Verilator simulation (at least for sanity check). Commercial simulator(s) remain main option for signoff / release
  - Setup a (public ?) CI/CD environment that can be duplicated locally
- Create maintenance branches for (fully?) verified configurations :
  - CV32E40Pv1 (should we rebrand it ?) : Basic RV32_IMC_Zicntr_Zicsr_Zifencei
    - v1.0.0 is current tag
    - Create maintenance branch / tag:
      - v1.8.3 of CV32E40Pv2 git tree is a good candidate since they are formally equivalent (and thus backward-compatible)
      - Wasn't verified with latest version of testbench, actual functional/code coverage figures not available → we may rerun it
  - CV32E40Pv2: (single brand or should we rebrand sub-configurations ?)
    - Upstream GCC Floating point only (F): RV32_IMCF_Zicntr_Zicsr_Zifencei (including F0/F1/F2 pipelining configuration ?) → not part of original verif plan, should not have any coverage issue
    - Upstream GCC Floating point only (FinX): RV32_IMC_Zicntr_Zicsr_Zifencei_Zfinx (including Z0/Z1/Z2 pipelining configuration ?) → not part of original verif plan, should not have any coverage issue
    - Upstream GCC Core-V extensions (aka PULP): RV32_IMC_Zicntr_Zicsr_Zifencei_xcvmac_xcvalu → not part of original verif plan, should not have any coverage issue (requires disabling unused PULP features in RTL code ?)
    - Full Core-V extensions (aka PULP), Core-V GCC ?: RV32_IMC_Zicntr_Zicsr_Zifencei_xcvhwlp_xcvmem_xcvmac_xcvbi_xcvalu_xcvsimd_xcvbitmanip → part of test plan (but coverage issue ?)
    - PULP+F/zfinx ? (verified configs as of v1.8.3)

Proposed new development model

- Put in place “per-feature” short design/verif/doc/release cycles (~like agile development sprints)
  - Regular meetings
    - Review and sort-out issue tracker content
    - (re-)schedule current and next release cycle(s)
  - HW, SW and verification environment update should be performed at the same time
    - Sync HW and SW schedules.
  - Addressing critical issues should have higher priority than integrating new features
- 2 kinds of releases:
  - Per-feature release tags
    - Baseline : RV32_I_Zicsr_Zifencei
    - Only one feature integrated (+dependencies)
    - → track big regression in an efficient way (shorter regression loops – idea is to trigger CI/CD only if pre-processed verilog code has changed since previous run)
  - Application-oriented release tags
    - Assembly of relevant feature sets for envisioned application classes → baseline for end-users
    - → track subtle regressions caused by interactions
- Challenges:
  - Find a good balance between feature inclusion and verification
  - Modular documentation generation for application-oriented releases
  - Probably others…

Proposed short/mid term integration plan (WIP)

- Split baseline CV32E40Pv1 baseline to make M and C features optional
- Split PULP design config / verification to match the 8 subsets defined by GCC support
- Re-integrate features from CV32E40PX/S. Priority based on increasing effort / complexity
  - 1/ Zmmul support → make HW divider optional
    - Easy to integrate (disable HW divider)
    - compiler / model / verif environment available
  - 2/ Initial Zc support (Zca, Zcf, Zcb) → clean replacement of C when no FPU + more base compressed instructions
    - Easy to integrate.
    - Integration by order of priority : 1/ Zca-Zcf split – 2/ Zcb
    - compiler / model / verif environment available
  - 1/ RV32B support (Zba, Zbb , Zbc, Zbs) :
    - compiler / model / verif environment available
    - Integration by order of priority : 1/ Zbb – 2/ Zba – 3/ Zbs – Zbc may wait until Zbkc
  - 2/ RV32K support
    - compiler / model / verif environment available
    - 1/ Zkt probably trivial on CV32E4 (only mulh and div may be data dependent)
    - 2/ Zbkb - 3/ Zbkx -4/ Zbkc
  - 3/ xcvhwlp support → to finalize CV32E40Pv2 original plan
    - Solving coverage issue may take time and require HW modifications to make verification easier (design for verification)
    - Upstream compiler support ?
  - 4/ CV-X-IF support → popular feature, requested by many end-users
    - Build verification environment based on existing CVA6/CVE2 existing work
    - Which implementation should we use : CV32E40X or CV32E40PX (EPFL fork) ?
    - Should we integrate existing 0.1.0 support of go directly to 1.0.0 ?
  - 5/ Optimized multiplier
    - Design done (CV32E40PX CEA fork)
    - Difficult to verify and debug
  - Next steps TBD

Envisioned Verification configurations (WIP)

- Feature verification configs (includes constant time execution check, a-posteriori advertised if OK):
  - Baseline : RV32_I_Zicsr_Zifencei
  - Multiplication: RV32_IM_Zicsr_Zifencei, RV32_I_Zmmul_Zicsr_Zifencei
  - Compressed integer: RV32_I_Zca_Zicsr_Zifencei, RV32_I_Zmmul_Zca_Zcb_Zicsr_Zifencei
  - Perf counters : RV32_I_Zicsr_Zicntr_Zifencei, RV32_I_Zicsr_Zicntr_Zihpm_Zifencei
  - Bitmanip : RV32_I_Zbb_Zicsr_Zifencei, RV32_I_Zba_Zicsr_Zifencei, RV32_I_Zbs_Zicsr_Zifencei
  - Scalar Crypto base: RV32_I_Zbkb_Zicsr_Zifencei, RV32_I_Zbkx_Zicsr_Zifencei, RV32_I_Zbkc_Zicsr_Zifencei
  - Single precision FPU : RV32_IF_Zicsr_Zifencei, RV32_I_Zfinx_Zicsr_Zifencei, RV32_IF_Zcf_Zicsr_Zifencei, RV32_I_Zfinx_Zcf_Zicsr_Zifencei
  - Core-V extensions (aka PULP):
    - RV32_I_Zmmul_Zicsr_Zifencei_xcvhwlp
    - RV32_I_Zmmul_Zicsr_Zifencei_xcvmem
    - RV32_I_Zmmul_Zicsr_Zifencei_xcvmac
    - RV32_I_Zmmul_Zicsr_Zifencei_xcvbi
    - RV32_I_Zmmul_Zicsr_Zifencei_xcvalu
    - RV32_I_Zmmul_Zicsr_Zifencei_xcvsimd
    - RV32_I_Zmmul_Zicsr_Zifencei_xcvbitmanip
  - …TBD
- Application-oriented verification configs (may be enriched as new features get integrated):
  - Size-optimized RT microcontroller (CV32E40Pv1 tiny) : RV32_I_Zmmul_Zca_Zcb_Zicsr_Zifencei
  - Profiling-enabled RT microcontroller (CV32E40Pv1) : Basic RV32_IMC_Zicntr_Zicsr_Zifencei (maybe replace C by Zca or Zca+Zcb ?)
  - Floating point enabled RT microcontroller (Upstream GCC Floating point F): RV32_IMCF_Zicntr_Zicsr_Zifencei (including F0/F1/F2 pipelining config ?)
  - Size-optimized Floating point enabled RT microcontroller (Upstream GCC Floating point FinX): RV32_IMC_Zfinx _Zicntr_Zicsr_Zifencei (including Z0/Z1/Z2 pipelining config ?)
  - Performance-optimized RT-DSP microcontroller (Upstream GCC Core-V extensions, aka PULP): RV32_IMC_Zicntr_Zicsr_Zifencei_xcvmac_xcvalu (TBC)
  - Performance-optimized RT-DSP microcontroller, fully featured (Core-V extensions, aka PULP, Core-V GCC ?): RV32_IMC_Zicntr_Zicsr_Zifencei_xcvhwlp_xcvmem_xcvmac_xcvbi_xcvalu_xcvsimd_xcvbitmanip
  - … TBD

Backup slides :
CV32E4* features tables

RISC-V Unprivileged spec support

Supported extensions	RV32I		RV32M		RV32C	Zc				RV32A	RV32B				RV32P				RV32K				RV32F	
supported subsets	RV32I	RV32E	RV32M	Zmmul (no HW divider)	RV32C	Zca: C with the FP load/stores removed.	Zcb: simple operations.	Zcmp: push/pop and double move (overlap with c.fsdsp)	Zcmt: table jump.	RV32A	Zba: Address calculation	Zbb: Base instructions	Zbc: Carry-Less Multiply	Zbs: Bit set, Bit clear, etc.	SIMD Subset ?	Bitmanip Subset ?	Q-Format / Saturated arithmetic Subset ?	Register pair Subset ?	Zkt: Data Independent Execution Latency	Zbkc: Constant time Carry-Less Multiply	Zbkb - Bitmanip instructions for Cryptography	Zbkx: Crossbar permutations	RV32F	Zfinx
CV32E40Pv1	2.0		2.0		2.0	?																	2.2*	v?*
CV32E40Pv2	2.0		2.0		2.0	?																	2.2*	1.0*
CV32E40X	2.1	2.0*	2.0	0.1*	2.0	1.0.0 RC5.6	1.0.0 RC5.6	1.0.0 RC5.6	1.0.0 RC5.6	2.1*	1.0.0*	1.0.0*	1.0.0*	1.0.0*					1.0.0*	1.0.0*				
CV32E40S	2.1	2.0*	2.0	0.1*	2.0	1.0.0 RC5.6	1.0.0 RC5.6	1.0.0 RC5.6	1.0.0 RC5.6	2.1*	1.0.0*	1.0.0*	1.0.0*	1.0.0*					1.0.0*	1.0.0*				
CV32E40PX	2.1		2.0		Zca 1.0.0 ?						1.0.0*	1.0.0*	1.0.0*	1.0.0*	0.x*	0.x*	0.x*	0.x*					2.2*	1.0*
CV32E4	2.1	2.0*	2.0*	1.0	Zca+Zcf ?	1.0.0*	1.0.0*	1.0.0*	1.0.0*	2.1*	1.0.0*	1.0.0*	1.0.0*	1.0.0*	0.x*	0.x*	0.x*	0.x*	TBD	TBD	1.0.1*	1.0.1*	TBD	TBD

RISC-V Privileged & non-ISA spec support

Supported extensions	Privieged spec								RV DEBUG	RV CLIC
supported subsets	Zicntr: Base Counters and Timers	Zihpm: Hardware Performance Counters	Zicsr (ctrl/status reg insts)	Zifencei (instruction fetch fence)	Smstateen: State Enable Extension	U-Mode	PMA	Smepmp: PMP Enhancements for memory access and execution prevention on Machine mode	RV DEBUG	RV CLIC
CV32E40Pv1	?	?	?	?						
CV32E40Pv2	2.0	?	2.0	2.0						
CV32E40X	2.0	2.0	2.0	2.0			Y	1.0	1.0	0.9
CV32E40S	2.0	2.0	2.0	2.0	0.6.3	Y	Y	1.0	1.0	
CV32E40PX	2.0	?	2.0	2.0					1.0	TBD
CV32E4	2.0	2.0*	2.0	2.0	1.0.0*	Y*	Y*	1.0.0*	1.0	TBD

CORE-V extensions/features support

Supported extensions	XCV: CORE-V PULP ISA Extensions								CV-X-IF	Xsecure	OBI
supported subsets	XCValu: miscellaneous ALU extension	XCVmac: multiply-accumulate	XCVelw: event load word extension	XCVbi: immediate branch	XCVmem: load/store	XCVsimd: SIMD	XCVbitmanip: Bit manipulations	XCVhwlp: Hardware loop	CV-X-IF	Xsecure: Security extensions	OBI
CV32E40Pv1	?*	?*	?*	?*	?*	?*	?*	?*			?
CV32E40Pv2	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*			1.2
CV32E40X									0.1*		1.6.0
CV32E40S										1.0	1.6.0
CV32E40PX	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	?*		
CV32E4	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.0*	1.6.0

Thank you!