HW/SW co-design for exposed-Datapath processors

An integrated review of the TCE/OpenASIP toolset
and its place among RISC-V customization flows

An open-source HW/SW co-design toolset from Tampere University—the **TTA-based Co-Design Environment (TCE)** and its successor ecosystem **OpenASIP**—lets engineers **design customized processors** and **program them in C/C++**, generating both **synthesizable RTL** and a **retargetable compiler** from the **same architecture description**. The approach centres on **exposed-Datapath** templates (Transport-Triggered Architecture, TTA) that make Datapath resources visible to the compiler, enabling fine-grained scheduling and rapid architectural specialization for domains such as **SDR**, **DSP**, **vision**, and **ML**. [trepo.tuni.fi], [github.com]

Why it matters

Customized processors offer a **middle ground** between fixed-function accelerators and general CPUs: you get **hardware tailoring** for target apps while keeping **post-manufacture programmability** and reducing verification effort compared with designing each accelerator from scratch. TCE/OpenASIP provides a mature, open flow (compiler, simulator, RTL generators) that has been used in **academic silicon** and **commercial-grade chips**, reducing engineering time and enabling quick design-space exploration. [trepo.tuni.fi]

How it works

- **Architecture template: TTA** exposes Datapath transfers; designers pick resources (register files, function units, interconnect), not just instruction encodings. [trepo.tuni.fi]
- **Tooling:** Retargetable **Clang/LLVM-based compiler**, instruction-set simulator, and **Verilog/VHDL** RTL generator in a unified flow; OpenASIP/TCE are actively developed open projects. [github.com]
- **Use cases:** Published design cases span **video/vision**, **audio**, **SDR**, and **CNN inference** (e.g., AivoTTA), demonstrating energy-efficient instruction streams and domain-specific accelerators programmable in C/OpenCL C. [Co-Design…Processors]

How it compares to RISC-V CI flows

Where CV-X-IF/RoCC attach **coprocessors** to an existing RISC-V core, TCE/OpenASIP generates an **ASIP** (a standalone programmable accelerator). Both aim for **retargetable compilers** and **rapid customization**, but TCE/OpenASIP is optimized for **exposed-Datapath design** and deep schedule control, while CV-X-IF/RoCC stress **modularity around existing cores**. **SCAIE-V** is an in-pipeline RISC-V extension framework; **Codasip Studio** and **Synopsys ASIP Designer** are commercial ASIP suites. [trepo.tuni.fi], [link.springer.com]

If you need a **bespoke programmable accelerator** with **fine scheduling freedom**, TCE/OpenASIP is a strong open option. If you want to **extend an existing RISC-V CPU** while keeping the core intact, consider **CV-X-IF/RoCC**; for **in-pipeline** ISA extensions,

evaluate **SCAIE-V**; for **enterprise-scale ASIP flows**, the commercial suites may fit best. [trepo.tuni.fi]

Introduction

The pdf **"HW/SW Co-Design Toolset for Customization of Exposed Datapath Processors"** outlines a methodology and open toolchain to co-design **application-specific instruction-set processors (ASIPs)** around an **exposed-Datapath** template, notably **Transport-Triggered Architecture (TTA)**. Rather than starting from opcode formats, designers configure **Datapath resources** (function units, register files, interconnect), while a **retargetable compiler** schedules transfers to exploit ILP/DLP as dictated by the application.

The authors report broad tool maturity: beyond academic examples, the toolset has supported designs **down to silicon layout** and **integrated in commercial-grade chips**. [trepo.tuni.fi] For readers following RISC-V customization, this differs from **socket-based CI** (e.g., CV-X-IF/RoCC) by **replacing** or **augmenting** the core with a programmable accelerator whose microarchitecture is **fully customized**, not just extended. OpenASIP subsequently generalized the stack and added RISC-V support and dynamic compiler retargeting (building on the same philosophy of **single-source descriptions** feeding both compiler and hardware). [riscv-europe.org]

Computation-resource-oriented design
Instead of focusing on instruction encodings, the toolset encourages designers to start from **resource selection**: choose **function units**, **register files**, and **buses**, then map app kernels to these resources. This aligns naturally with exposed Datapath's where **moves** (transfers) are the programmer's/ compiler's primitive, letting the compiler schedule **data movement and compute** with precision. [trepo.tuni.fi]

End-to-end, retargetable flow
The flow includes a **retargetable C/C++ compiler** (Clang/LLVM-based), **instruction-set simulator**, and automatic **RTL** generation (both **VHDL** and **Verilog**), delivering programmable accelerators whose **micro-ops** and **interconnect** match the domain's demands. The toolchain has been honed since the early 2000s, with public repos and documentation. [github.com]

Demonstrated in real designs
Case studies cited by the team include **SDR** and **vision** pipelines, and later presentations describe **CNN-oriented** TTAs (e.g., **AivoTTA**) showing strong performance-per-watt at 28 nm FDSOI, with domain-specific vector units and carefully pruned register files to minimize power. While performance numbers are case-specific, the message is that **resource-specific tailoring** + **compiler scheduling** yields **dense, energy-efficient** instruction streams. [trepo.tuni.fi], [Co-Design…Processors]

Where TCE/OpenASIP sits among RISC-V CI and ASIP ecosystems

Against CV-X-IF / RoCC (coprocessor sockets):

CV-X-IF and RoCC are **extension sockets** for RISC-V cores: you keep the core RTL and offload instructions to an attached accelerator via a standard interface (CV-X-IF adds out-of-order-capable **issue/register/commit/result** channels; RoCC uses **ready–valid**

command/response). TCE/OpenASIP differs by producing an **independent ASIP**—a programmable accelerator that may sit **alongside** a CPU or serve as a **domain processor**. Choose sockets if you prioritize **core certification/reuse**; choose TCE/OpenASIP if you want **microarchitectural freedom** and **compiler-driven scheduling** at the accelerator itself. [link.springer.com]

Against SCAIE-V (in-pipeline interface):
SCAIE-V integrates **inside the pipeline** of RISC-V cores and supports **combinational, multi-cycle, memory, and control-flow** ISAX, aiming for portable extension points across cores. That can deliver **lower latency** for small extensions, but implies **core modifications** and verification. TCE/OpenASIP sidesteps this by **owning the core** (ASIP), giving maximum latitude but requiring **software porting** to the ASIP. [link.springer.com]

Against commercial ASIP suites:
Codasip Studio and **Synopsys ASIP Designer** also generate **RTL + retargetable compilers** from a single description, with advanced simulators and verification aids. TCE/OpenASIP's advantage is **open licensing** and **exposed-datapath specialization**; the commercial tools offer **enterprise support**, **rich model libraries**, and broad **ISA design** options (VLIW, vector, heterogeneous RFs). Your choice depends on budget, required support, and whether you want **open** or **enterprise** flow. [link.springer.com]

Short Sidebar — What is an "exposed-datapath" TTA?

- **Classic CPU**: An instruction selects an operation, and the microarchitecture implicitly moves data between registers and function units.
- **Exposed datapath (TTA)**: **Moves** between registers and function units are **explicit** and **programmable**. The compiler decides when to **transfer operands** and **trigger** operations, enabling very fine control over **ILP**, **bypassing**, and **resource contention**.
- **Why it helps**: For domain kernels with regular dataflow (filters, MAC chains, tensor ops), explicit transport often allows **denser** schedules with fewer wasted cycles, and the hardware you build can omit unnecessary generalized structures. [trepo.tuni.fi]

Methods Appendix

Chapter scope and provenance
The chapter (Springer book chapter, 2017; repository record 2016-12-30) describes the **co-design methodology**, the **open toolset**, and several deployment notes; Tampere's repository entry and Springer PDF confirm authorship, context, and DOI. [trepo.tuni.fi], [link.springer.com]

Tooling highlights
- **Compiler**: tcecc retargetable **Clang/LLVM** front end targeting TTA; schedules explicit moves and FU triggers per architecture description. [github.com]
- **HW generation**: Verilog/VHDL **RTL generators** emit a synthesizable processor core matching the chosen register files, FUs, and interconnect; supports ASIC/FPGA flows. [github.com]
- **Workflow**: A single processor description drives **compiler–sim–RTL**, enabling **rapid iteration** over resource sets (e.g., add a custom multiply-accumulate FU, widen a

register file, or add a bypass bus) and immediate evaluation with the compiler. [trepo.tuni.fi]

Real-world deployments and case studies

The chapter asserts tool use "**down to silicon layout**" and "**commercial grade chips**," while later slides enumerate domain cases (video, SDR, CNN accelerators such as AivoTTA). Readers can consult the public GitHub and CPC group materials for additional designs and code. [trepo.tuni.fi], [Co-Design…Processors]

Comparison Table — Where this chapter's toolset fits

Option	What you build	Where it lives	Compiler story	Core RTL impact	Latency profile	Licensing	Best fit
TCE / OpenASIP (TTA ASIP)	**Programmable accelerator (ASIP)**, fully customized datapath	As a **peer** processor (domain accelerator)	**Retargetable** LLVM-based compiler, ISS, RTL	**N/A** (you own the ASIP core)	Fine-grain scheduling; low overhead inside the ASIP	**Open-source**	Domains needing tight **scheduling freedom** and bespoke FUs (SDR, vision, ML) [trepo.tuni.fi], [github.com]
CV-X-IF (OpenHW)	**Coprocessor** behind a standard socket	**Attached to RISC-V core** (e.g., CVA6 family)	Extend RISC-V backend / intrinsics	**None** to the host core (socket use)	Offload latency; flexible commit/kill	**Open-source**	Add accelerators while preserving **core verification** [link.springer.com]
RoCC (Rocket/Chipyard)	**Coprocessor** via ready–valid	**Attached to Rocket**	Extend backend / intrinsics	**None** to Rocket	Minimal wrapper latency; offload cost can matter for tiny ops	**Open-source**	Rapid add-on accelerators in Rocket-based SoCs [link.springer.com]
SCAIE-V	**In-pipeline** ISAX	**Inside** RISC-V core	External mapping/tooling	**Yes** (core modifications)	Often lowest for	**Open-source**	Fine-grained control/memory ops

Option	What you build	Where it lives	Compiler story	Core RTL impact	Latency profile	Licensing	Best fit
					small ISAX		tightly coupled to pipeline [link.springer.com]
Codasip Studio	**ASIP** (any ISA/VLIW/vector)	Standalone	Auto-retargeted LLVM + SDK	N/A	Depends on arch	**Commercial**	Enterprise ASIP with vendor support
Synopsys ASIP Designer	**ASIP** (nML-described)	Standalone	C/C++ compiler + **cycle/instr** ISS, RTL	N/A	Depends on arch	**Commercial**	Industrial ASIP design, verification & models

(CV-X-IF/RoCC/SCAIE-V are referenced here to position TCE/OpenASIP; their detailed specs are covered in standard documentation and literature.) [link.springer.com]

Discussion: when to choose an exposed-datapath ASIP vs. a RISC-V extension

- Choose **TCE/OpenASIP** when your kernels benefit from **explicit transport scheduling** and when you want to **co-optimize** microarchitecture and compiler together (e.g., design a vector MAC FU, trim register ports, and retune schedules). This shines in **DSP-like** workloads or **regular ML/vision** pipelines with predictable dataflows. [trepo.tuni.fi], [Co-Design…Processors]
- Choose **CV-X-IF/RoCC** when you must **keep the general-purpose core intact** (certification, IP reuse) and attach accelerators via **standard sockets** with a conventional RISC-V toolchain. This favors teams already invested in RISC-V cores who want to minimize core-level verification. [link.springer.com]
- Choose **SCAIE-V** if your extensions are **small, latency-sensitive** and must execute **in-pipeline**, accepting the extra verification effort within the CPU. [link.springer.com]
- Consider **commercial ASIP** flows for **large teams** needing **support**, **ISS sophistication**, and **IP libraries**, or where corporate process/tool qualification requires vendor ecosystems.

Limitations and future directions

The chapter's focus predates some newer RISC-V interface standards and vector trends, but the **methodology** (resource-first design + retargetable tools) remains relevant. Recent OpenASIP materials highlight **RISC-V support** (dynamic backend loading, intrinsics, DAG/HDL operation descriptions), indicating a path to **hybrid** strategies where TCE/OpenASIP co-exists with RISC-V ecosystems—e.g., ASIP as a **domain accelerator** beside a RISC-V control core. [riscv-europe.org]

The **exposed-Datapath** co-design approach documented in this chapter offers a **mature, open** way to obtain **compiler-programmable accelerators** that match a domain's dataflow.

It complements, rather than replaces, RISC-V **coprocessor sockets** and **in-pipeline** frameworks:

- Use **TCE/OpenASIP** when you want **maximum microarchitectural freedom** and **fine scheduling control**.
- Use **CV-X-IF/RoCC** when you want to **extend existing RISC-V cores** safely and quickly.
- Use **SCAIE-V** for **in-pipeline** latency-critical ISAX.
- Consider **commercial ASIP suites** for enterprise-scale projects with stringent verification needs.

In short, pick the lever—**exposed-Datapath ASIP** vs. **RISC-V extension**—that best aligns with your **verification constraints**, **software stack**, and **kernel structure**. [trepo.tuni.fi], [link.springer.com]

References

- Jääskeläinen, P., Viitanen, T., Takala, J., & Berg, H. (2017). *HW/SW Co-Design Toolset for Customization of Exposed Datapath Processors*. In **Computing Platforms for Software-Defined Radio** (Springer). (Open version: Tampere repository.) [trepo.tuni.fi], [link.springer.com]
- Tampere University Repository Entry for the Chapter (metadata, DOI, access). [trepo.tuni.fi]
- TTA-based Co-Design Environment (TCE) — Project overview and README. [github.com]
- Jääskeläinen, P. (2018). *Co-Design of Programmable Exposed Datapath Processors* (slides). Use cases and recent ASIC results (AivoTTA, etc.). [Co-Design…Processors]
- OpenASIP (2024): *Open-Source Development Platform for RISC-V ASIPs* (plenary slide deck). [riscv-europe.org]

Source: HW/SWCo-Design Toolset for Customization of Exposed Datapath Processors