TECHNISCHE
UNIVERSITÄT
DARMSTADT

TaPaSCo

Open-Source FPGA Tooling – Made in Germany

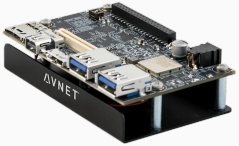
Torben Kalkhof David Volz Andreas Koch

2026-03-13

Embedded Systems and Applications, TU Darmstadt, Germany

Motivation

- FPGAs provide high flexibility, good performance, and high energy efficiency for custom hardware accelerators
- Integration into heterogeneous system required
- Lack of adequate tool and runtime support
  - AMD even discontinued XRT for new devices
  - System design from scratch
- FPGA designs often platform-specific
  - No "Write once, run everywhere"



Key Contributions

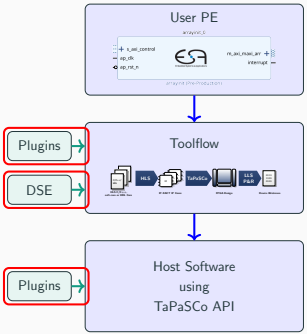
- Simple integration of FPGA-based accelerators into heterogeneous compute platforms
- Portability by hiding platform-specific details in hardware and software abstraction layers
- From embedded devices (e.g. ZYNQ) to data center acceleration (e.g. Alveo U280)
- From Virtex-7 (e.g. VC709) to Versal (e.g. VCK5000)

The TaPaSCo Open-Source Toolflow for the Automated Composition of Task-Based Parallel Reconfigurable Computing Systems

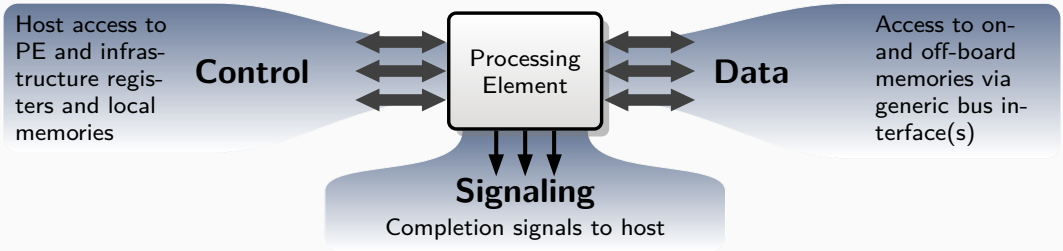
Jens Korinth, Jaco Hofmann, Carsten Heinz, and Andreas Koch

Task Parallel System Composer (TaPaSCo)

- User provides Processing Elements (PEs) in standardized format
- Toolflow to generate bitstreams for specified platform around user PEs
- Platform-independent software API for dispatching compute tasks from host applications to the FPGA (available in Rust/C++)
- Plugin system (provided or custom extensions)
- Design Space Exploration (DSE)

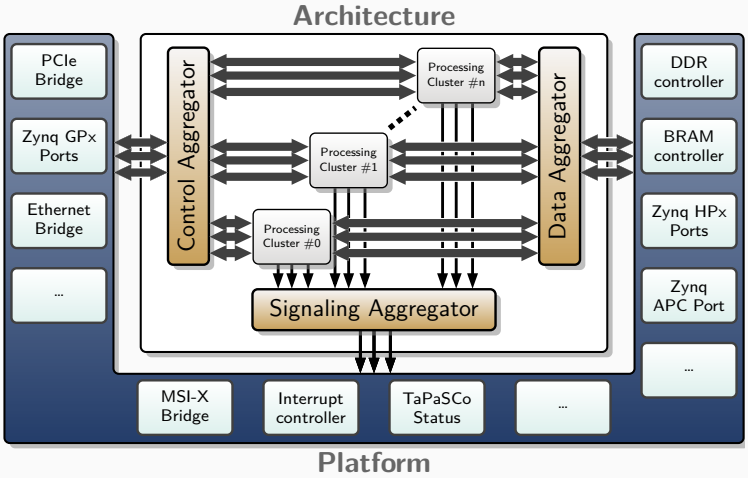


TaPaSCo - T-Shape of Processing Elements (PEs)



- All interfaces use AXI4 MM or AXI4 Lite (some plugins AXI4 Stream)
- Well-defined register layout on control bus
- Identical format for *all* platforms

TaPaSCo - Platform Architecture



TaPaSCo - Hardware Toolflow


```
tapasco compose [acc0 x 3, acc1 x 2] @ 400 MHz -p AU280
```

PE Count Frequency Platform

TaPaSCo - Hardware Toolflow

- `tapasco compose` commands can become rather complex
- Use JSON file to describe composition
- Still one command to build your design:

```
tapasco compose --jobsFile
↪ jobs.json
```

- More details in AMD HACC talk on YouTube [1]

[1] https://www.youtube.com/watch?v=7PdZ1SiObpA

```
[ {
  "Job": "Compose",
  "Design Frequency": 312.5,
  "SkipSynthesis": false,
  "DeleteProjects": false,
  "Platforms": [ "vck5000" ],
  "Architectures": [ "axi4mm" ],
  "Composition": {
    "Composition": [ {
        "Kernel": "DataStreamerVN",
        "Count": 1
    } ]
  },
  "Features": [  {
      "Feature": "DMA-Streaming",
      "Properties": {
        "slave_port": "S_AXIS_DMA",
        "master_port": "M_AXIS_DMA"
      },
      "Feature": "AI-Engine",
      "Properties": {
        "adf": "/path/to/my/libadf.a",
        "in_a": "M_AXIS_AIE_X",
        "in_b": "M_AXIS_AIE_Y",
        "out_x": "S_AXIS_AIE"
      }
  } ]
} ]
```

TaPaSCo - Software API Example

```
1  /* Initialize TaPaSCo FPGA device: */
2  Tapasco tapasco;
3  /* data buffer */
4  std::vector<int> v = {0, 1, ...};
5  auto buf = makeWrappedPointer(v.data(), v.size() * sizeof(int));
6  /* Launch a TaPaSCo task */
7  auto myTask = tapasco.launch(PE_TYPE, buf, 42);
8  /* Wait on myTask for completion */
9  myTask();
```

- Dynamic allocation of PEs (multi-threading supported)
- Automated data transfers with task launch and wait calls
- Manual memory management possible

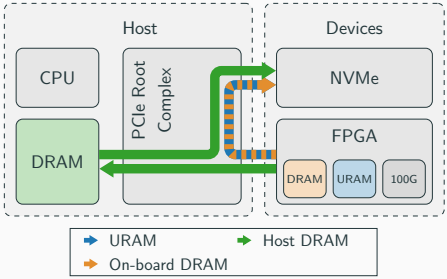
Features and Extensions

- Supported Platforms:
  - ZYNQ: ZCU102, Ultra96, ...
  - US+: Alveo U50, Alveo U280, ...
  - Versal: VCK5000, ProDesign HAWK
- Networking
  - Different protocols: Aurora, 10G, 100G
  - Simple interfacing with AXI4 Stream
  - TCP/IP stack available
- HBM support
- TaPaSCo IPEC
- Shared Virtual Memory (SVM)
- TaPaSCo NVMe
- AMD Versal Platform
- TaPaSCo RISCV
- HPX integration
  - Integrate FPGA tasks in an asynchronous many-task runtime for HPC
- Experiments:
  - Cascabel
  - TaPaSCo+CCIX
  - NVMulator

* More details on next slides

TaPaSCo NVMe (SNAcc)

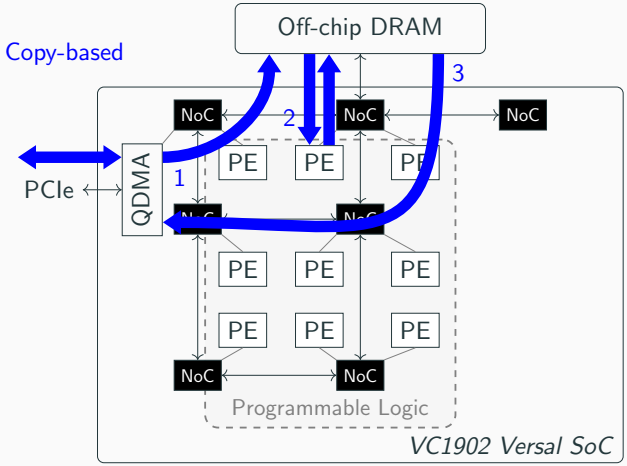
- Direct access to NVMe-based SSDs via PCIe P2P
- NVMe queue management on FPGA
- Streaming-based interfacing with user PE
- Three implementations of data transfers: URAM, on-board DRAM, and host DRAM
- Maximum achieved performance: 6.9 GB/s (read), 6.24 GB/s (write)
  - Best performance using host DRAM
  - Leveraging full bandwidth of SSD



Publication:
D. Volz, T. Kalkhof, and A. Koch. SNAcc: An Open-Source Framework for Streaming-based Network-to-Storage Accelerators. *H2RC 2025*.

TaPaSCo on AMD Versal ACAP

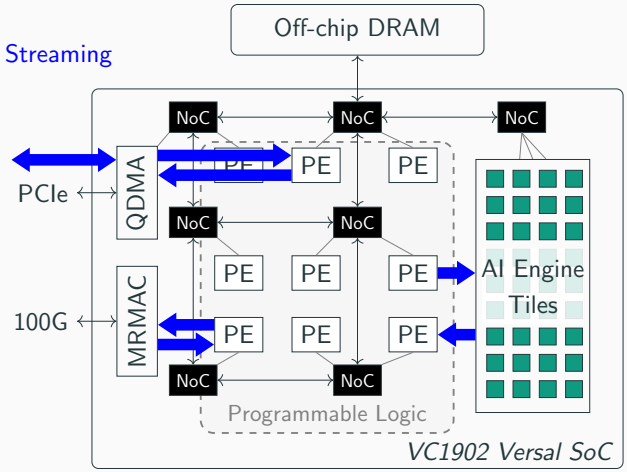
- Leverage hard-wired infrastructure IPs
- AXI NoC and memory controller
- PCIe and DMA engine
- Copy-based data management



Publication:
C. Heinz, T. Kalkhof, et- al. TaPaSCo-AIE: An Open-Source Framework for Streaming-based Heterogeneous Acceleration using AMD AI Engines. *RAW 2024.*

TaPaSCo on AMD Versal ACAP (Continued)

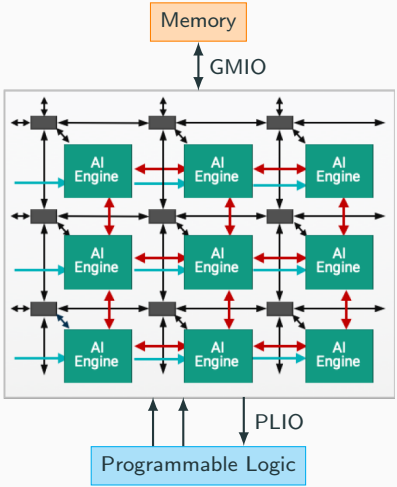
- Leverage hard-wired infrastructure IPs
- AXI NoC and memory controller
- PCIe and DMA engine
- QDMA Streaming
- 100G Networking
- AI Engines
- Supporting VCK5000 and ProDesign HAWK



Publication:
C. Heinz, T. Kalkhof, et. al. TaPaSCo-AIE: An Open-Source Framework for Streaming-based Heterogeneous Acceleration using AMD AI Engines. *RAW 2024*.

Versal AI Engines

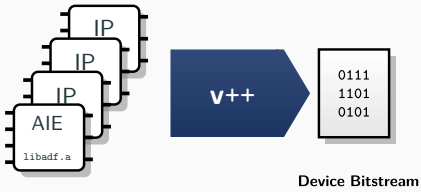
- Array of VLIW tiles with SIMD unit @ 1.25 GHz
- Streaming interconnect and connection to PL (PLIO)
- Programmed with AIE graph consisting of C++ kernels (SDK for Vitis)
- CGSim [H2RC'25]: fast software simulation and source-to-source translation
  - Embed AIE graph in existing host application
  - TaPaSCo integration planned



Source: https://www.xilinx.com/products/technology/ai-engine.html

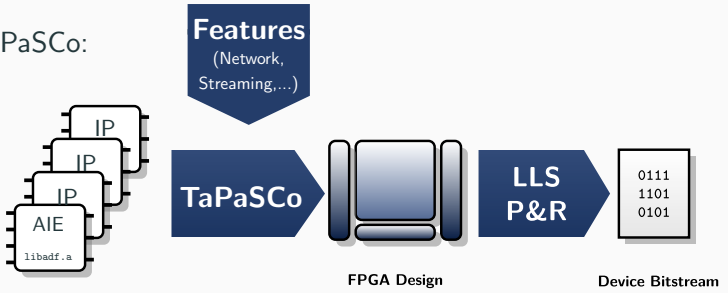
AI Engine Synthesis Flow (Vendor Flow)

Existing vendor flow:



Vitis lock-in

AI Engine Synthesis Flow (TaPaSCo Flow)

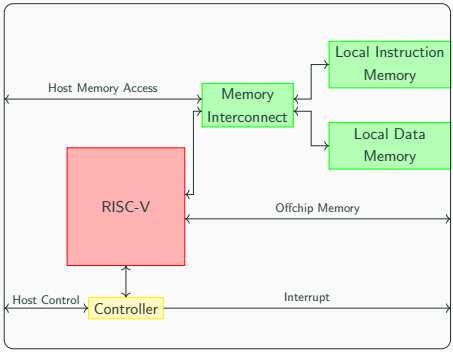


- Tapasco + Vivado (no Vitis)
- Higher flexibility
- More options of combining features
- Neural Network case study as PoC

```
tapasco compose [pe_0 x 1, pe_1 x 1, pe_2 x 1] @ 400 MHz -p vck5000        \
    --features 'AI-Engine { "adf": <path/to/>libadf.a,                      \
                            "AIE-stream-input-port0": pe_0-stream-output0, \
                            "AIE-stream-output-port0": pe_1-stream-input0, \
                  }, DMA-Streaming {...}, SFPPLUS { ... }'
```

Projects Using TaPaSCo - TaPaSCo RISC-V

- Motivation:
  - Fast prototyping of embedded RISC-V processors and hardware extensions
  - Quick evaluation of benchmarks (seconds on FPGA vs. hours in RTL simulation)
- Same PE architecture for each RISC-V core
- Reload instruction and data memory using TaPaSCo buffer transfers
- Support for nine different RISC-V cores
  - e.g. VexRiscv, CVA5, CVA6
- WIP: Support to Versal FPGAs



Publications:

C. Heinz, Y. Lavan, et al. A Catalog and In-Hardware Evaluation of Open-Source Drop-In Compatible RISC-V Softcore Processors. *ReConFig 2019*.

Outlook: TaPaSCo on Alveo V80

- High potential to become next de facto standard FPGA in academia
  - Doubled PCIe bandwidth (PCIe 5.0x8)
  - Doubled PL resources (compared to U280)
  - HBM attached to NoC
- First working TaPaSCo prototype (not published yet)
- Work in progress on:
  - Exploit increased PCIe bandwidth (rework of (Q)DMA subsystem required)
  - Use HBM instead of DDR as primary memory
  - Enable second 32 GB DDR DIMM, e.g., as scratch pad
  - Support 200G Ethernet (400G on AlphaData PA120)
- Port NVMe (and other) plugins to Versal

Challenges

- Easy adoption for TaPaSCo users does not mean easy adoption for us
- Every FPGA device is different, find working configuration
- Substitute vendor-provided IPs
- Restructuring of shell for new generation
- Keep up to date with vendor tools
- Benefit for community:
  - Increased initial porting effort
  - Easy re-usability for entire community

Embedded Systems & Applications

TECHNISCHE UNIVERSITÄT DARMSTADT

TaPaSCo is available on Github

github.com/esa-tu-darmstadt/tapasco

github.com/esa-tu-darmstadt/tapasco-examples

Get in touch: tapasco@esa.tu-darmstadt.de