GENOV: Streamlining the Design and Optimization of Heterogeneous RISC-V-based FPGA Overlays

Team number: *xohw23-150*
Team: Gianluca Bellocchi*, Alessandro Capotondi*
Supervisor: Andrea Marongiu*
**University of Modena and Reggio Emilia, Italy*
{*name.surname*}@unimore.it

***Abstract*—Heterogeneous SoCs (HeSoCs) designs are increasingly seen to couple general-purpose processors with application-specific accelerators to enable high performance and energy efficiency. High-Level Synthesis (HLS) eases the design and exploration of such complex architectures, but automated tools still lack the maturity to efficiently and easily tackle system-level integration of the many hardware and software blocks, hence motivating the need for innovative *system-level design (SLD)* approaches. On behalf of this competition, we propose GENOV, an open-source design automation tool to seamlessly generate and optimize heterogeneous FPGA overlays built around the RISC-V instruction set architecture (ISA). GENOV offers *plug-and-play* integration of HLS-based accelerators, given a customizable accelerator interface - or *accelerator wrapper* - which provides shared-memory communication to the overlay cores. Our FPGA overlays are an extension of HERO, an open-source research platform for heterogeneous multi-/many-core architectures based on the Parallel Ultra Low Power (PULP) Platform.**

***Index Terms*—FPGA Overlay; Hardware Accelerators; High-Level Synthesis; System-level design; RISC-V; Open-source**

I. INTRODUCTION

This report presents GENOV [1], an open-source design automation tool to seamlessly generate and optimize heterogeneous FPGA overlays built around the RISC-V instruction set architecture (ISA).

GENOV offers *plug-and-play* integration of High-Level Synthesis (HLS) hardware accelerators, given a customizable accelerator interface - or *accelerator wrapper* - which provides shared-memory communication to the overlay cores.

Our FPGA overlays are an extension of HERO, an open-source research platform for heterogeneous multi-/many-core architectures based on the Parallel Ultra Low Power (PULP) Platform [2], [3].

We instantiate our overlays on commercially off-the-shelf FPGA-based SoCs, like the AMD-Xilinx Zynq UltraScale+ ZU9EG MPSoC.

GENOV[1] and its associated components and repositories are released open-source under a permissive license.

II. OVERLAY ARCHITECTURE

The FPGA overlay architecture is shown in Figure 1.

[1]https://github.com/gbellocchi/genov

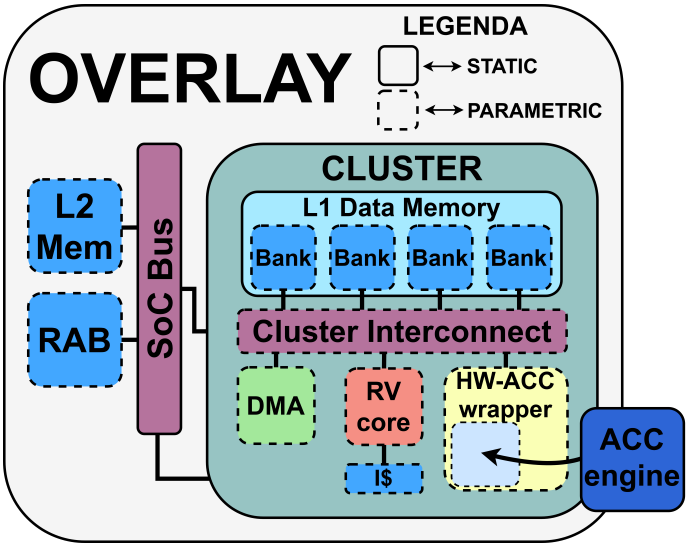


Figure 1: FPGA overlay architecture

The cluster is composed of general-purpose, 32-bit cores and an application-specific accelerator, or *engine*.

The core is an RV32 implementation, serving as an *orchestrator* of the many HW/SW tasks co-existing inside the cluster, thus enabling efficient runtime management of the available accelerator resources. Besides, a soft-core provides a suitable interface to the host application processor of the AMD-Xilinx Zynq UltraScale+ SoC.

L1 memory is a multi-banked, tightly-coupled data memory (TCDM) that is shared by accelerators and cores via a scalable, fully-connected crossbar. The cluster employs an SW-programmable DMA engine for efficient data transfers.

The application-specific accelerator communicates with the cluster components through a Hardware Processing Engine (HWPE) interface, with a loose coupling to the cores and a tight coupling to the cluster TCDM, which they all can access autonomously.

At the SoC-level, the overlay comprises a remapping address block (RAB) - a simplified IOMMU - and an L2 scratchpad memory.

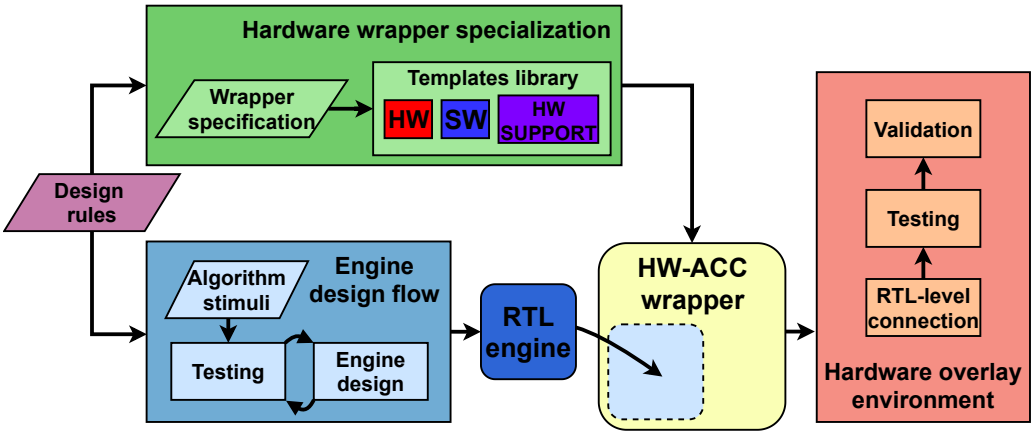


Figure 2: GENOV design flow

III. METHODOLOGY

The GENOV design flow is shown in Figure 2.

GENOV is implemented in Python and takes advantage of the Mako template library[2] as a back-end. The generation approach is *template-based* as it parameterizes a marked-up text - the template - using user-defined input parameter specifications.

In principle, the toolchain transparently generates the required HW/SW accelerator interfaces from *accelerator specifications* where users can annotate the main characteristics of their accelerator I/O data ports, register files, and optional optimizations, as shown in Listing 1. More accelerator inputs are grouped into an accelerator library.

Additionally, the user needs to provide an *overlay specification* with fundamental details about the FPGA overlay SoC architecture.

```
class acc_specs:
    def __init__(self):
        self.target              = 'mmult'
        self.design_type         = 'hls'
        self.list_sink_stream    = [ 'in1' , 'in2' ]
        self.list_source_stream  = [ 'out' ]
        self.n_sink              = len(self.list_sink_stream)
        self.n_source            = len(self.list_source_stream)
        self.std_reg_num         = 4
        self.custom_reg_name     = [ 'custom_reg' ]
        self.custom_reg_dim      = [ 32 ]
        self.custom_reg_num      = len(self.custom_reg_name)
```

Listing 1: Python interface for wrapper specialization.

IV. REUSABILITY

A template-based approach can be applied regardless of the rendered content, thus templates may consist of variegated nature. For instance, GENOV include HDL designs and test benches, C libraries and header files, HW management, simulation and build scripts, as well as documentation.

The approach can further scale up by committing different template designs - with varying source language - to different engineering teams. These would jointly employ GENOV-specific symbols and rules to automate the design flow and extend the template library with additional HW/SW components. Besides, input specifications can be added as well to widen the overlay specialization horizons.

[2]https://www.makotemplates.org/

V. EXPLORATION

In this section, we evaluate our overlay in terms of performance. The presented results have been published and presented at a conference by the members of the competing team [1]. Besides, benchmarks are released open-source under a permissive license[3].

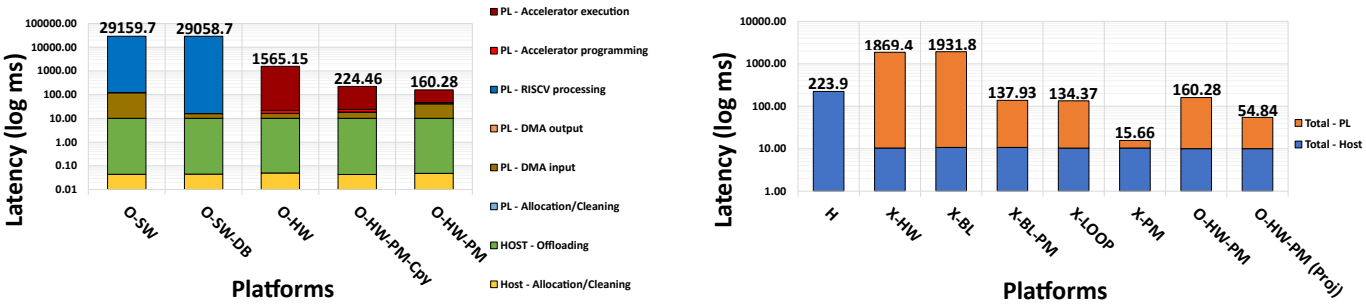
The proposed implementations have been synthesized and deployed on an AMD-Xilinx Zynq UltraScale+ ZU9EG MPSoC. We used AMD-Xilinx Vivado HLS v2018.2 to design accelerated *engine*, AMD-Xilinx Vivado v2019.2 to synthesize and implement RTL designs, AMD-Xilinx Petalinux v2019.2 to deploy the application on the target board.

We use a Matrix Multiplication (MatMul) kernel as a benchmark with a fixed matrix size of 512×512 and 32-bit unsigned data elements.

1) Overlay Designs: Figure 3a shows application breakdowns for five increasingly optimized overlay implementations runming at a target frequency ($f_{overlay} = 90MHz$). O-SW is a baseline pure-SW implementation of the MatMul kernel on the overlay cores. O-SW-DB is optimized by employing a double-buffered memory-hiding technique. In O-HW the algorithm kernel is no more executed by the soft-core, but, instead, an HLS MatMul accelerator is wrapped and integrated into the overlay. In O-HW-PM-Cpy, the overlay integrates an HLS engine containing a prefetching stage, a write stage and private BRAM buffers. As our overlay already comprises the required system IPs to support hardware acceleration, we believe most of O-HW-PM-Cpy components to be in excess. However, this example shows our flow to be flexible to particular user requirements. Finally, in O-HW-PM the interface of the accelerator is parallelized to better accommodate the optimized datapath with the required bandwidth. Compared to O-HW-PM-Cpy, no additional private memory or read/write stage is included.

2) Reference Designs: In Figure 3b, we compare with alternative deployment strategies. H is a baseline host implementation at $f_{HOST} = 1.2GHz$. Besides, are results from reference AMD-Xilinx (X-*) and overlay-based (O-*) designs,

[3]https://github.com/gbellocchi/xil_open_hw_23



(a) Overlay application breakdown

(b) Platform comparison

Figure 3: FPGA acceleration of a Matrix Multiplication (MatMul) kernel

running at an FPGA frequency of $f_{REF,PL} = 150MHz$ and $f_{OV,PL} = 100MHz$.

Investing in IP-centric optimizations allows for an X-based improvement for up to a factor of $119.3\times$ compared to H, thus justifying the efforts to deploy HW-accelerated algorithms. Still, X-PM is no representative of real-world applications because of its excessive implementation cost, so our O-based designs do compare to X-LOOP.

Hence, our O-HW-PM can better perform up to a factor of $2.5\times$ with additional programming and communication services through a programmable FPGA overlay.

VI. Conclusion

We have proposed GenOv, an open-source design automation tool to seamlessly generate and optimize heterogeneous RISC-V-based FPGA overlays. We have shown the advantages of our approach to better SLD and demonstrated various of our overlay implementations using AMD-Xilinx FPGA SoCs and tools for synthesis and system deployment.

References

[1] G. Bellocchi, A. Capotondi, F. Conti, and A. Marongiu, "A risc-v-based fpga overlay to simplify embedded accelerator deployment," in *2021 24th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2021, pp. 9–17.

[2] A. Kurth, A. Capotondi, P. Vogel, L. Benini, and A. Marongiu, "Hero: An open-source research platform for hw/sw exploration of heterogeneous manycore systems," in *Proceedings of the 2nd Workshop on AutotuniNg and aDaptivity AppRoaches for Energy efficient HPC Systems*, 2018, pp. 1–6.

[3] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, G. Tagliavini, A. Capotondi, P. Flatresse, and L. Benini, "Pulp: A parallel ultra low power platform for next generation iot applications," in *2015 IEEE Hot Chips 27 Symposium (HCS)*. IEEE Computer Society, 2015, pp. 1–39.