ARCHIVIO ISTITUZIONALE DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

A Heterogeneous RISC-V Based SoC for Secure Nano-UAV Navigation

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Valente, L., Nadalini, A., Veeran, A.H.C., Sinigaglia, M., Sá, B., Wistoff, N., et al. (2024). A Heterogeneous RISC-V Based SoC for Secure Nano-UAV Navigation. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. I, REGULAR PAPERS, 71(5), 1-14 [10.1109/tcsi.2024.3359044].

Availability:
This version is available at: https://hdl.handle.net/11585/959822 since: 2024-02-21

Published:

DOI: http://doi.org/10.1109/tcsi.2024.3359044

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

06 March 2026

A Heterogeneous RISC-V based SoC for Secure Nano-UAV Navigation

Luca Valente, Alessandro Nadalini, Asif Veeran, Mattia Sinigaglia, Bruno Sá, Nils Wistoff, *Student Member, IEEE,* Yvan Tortorella, Simone Benatti, Rafail Psiakis, Ari Kulmala, Baker Mohammad, *Sr. Member, IEEE,* Sandro Pinto, Daniele Palossi, Luca Benini, *Fellow, IEEE,* Davide Rossi, *Member, IEEE*

***Abstract*—The rapid advancement of energy-efficient parallel ultra-low-power (ULP) μcontrollers units (MCUs) is enabling the development of autonomous nano-sized unmanned aerial vehicles (nano-UAVs). These sub-10cm drones represent the next generation of unobtrusive robotic helpers and ubiquitous smart sensors. However, nano-UAVs face significant power and payload constraints while requiring advanced computing capabilities akin to standard drones, including real-time Machine Learning (ML) performance and the safe co-existence of general-purpose and real-time OSs. Although some advanced parallel ULP MCUs offer the necessary ML computing capabilities within the prescribed power limits, they rely on small main memories (<1MB) and μcontroller-class CPUs with no virtualization or security features, and hence only support simple bare-metal runtimes. In this work, we present Shaheen, a 9mm^2 200mW SoC implemented in 22nm FDX technology. Differently from state-of-the-art MCUs, Shaheen integrates a Linux-capable RV64 core, compliant with the v1.0 ratified Hypervisor extension and equipped with timing channel protection, along with a low-cost and low-power memory controller exposing up to 512MB of off-chip low-cost low-power HyperRAM directly to the CPU. At the same time, it integrates a fully programmable energy- and area-efficient multi-core cluster of RV32 cores optimized for general-purpose DSP as well as reduced- and mixed-precision ML. To the best of the authors' knowledge, it is the first silicon prototype of a ULP SoC coupling the RV64 and RV32 cores in a heterogeneous host+accelerator architecture fully based on the RISC-V ISA. We demonstrate the capabilities of the proposed SoC on a wide range of benchmarks relevant to nano-UAV applications including general-purpose DSP as well as inference and online learning of quantized DNNs. The cluster can deliver up to 90GOp/s and up to 1.8TOp/s/W on 2-bit integer kernels and up to 7.9GFLOp/s and up to 150GFLOp/s/W on 16-bit FP kernels.**

***Index Terms*—Heterogeneous, Linux, Low-Power, Autonomous Nano-UAVs, RISC-V**

Manuscript submitted July 30, 2023.

Luca Valente, Mattia Sinigaglia, Alessandro Nadalini, Yvan Tortorella, Simone Benatti, Luca Benini, and Davide Rossi are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy. Asif Veeran and Baker Mohammad are with the Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, UAE. Bruno Sá and Sandro Pinto are with Centro ALGORITMI, University of Minho, 4800-058 Guimarães, Portugal. Nils Wistoff, Daniele Palossi and Luca Benini are with the Integrated Systems Laboratory (IIS), ETH Zürich, 8092 Zürich, Switzerland. Ari Kulmala and Rafail Psiakis are with Technology Innovation Institute, Secure Systems Research Center, Abu Dhabi, UAE. Daniele Palossi is also with the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, 6900 Lugano, Switzerland.

This work was supported by Technology Innovation Institute, Secure Systems Research Center, Abu Dhabi, UAE, PO Box: 9639, by the Spoke 1 on Future HPC of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Mission 4 - Next Generation EU, and by KDT TRISTAN project (g.a. 101095947)

I. Introduction

THE number of Internet-of-Things (IoT) devices and the spectrum of IoT applications are rapidly growing: from home automation, robotics, industrial gateways, and building automation to smart cities, digital signage, medical equipment, and more [1]. In this context, nano-sized unmanned aerial vehicles (nano-UAVs) can be considered the "ultimate" IoT node, thanks to their ability to navigate, sense, analyse, and understand the surrounding environment. Nano-UAVs have a form factor of a few centimeters in diameter, and a weight of only tens of grams, which allows them to safely operate near humans and in narrow, cramped spaces [2], [3]. They have a total power envelope of a few Watts, of which only 5-15% for computation [4], and their small physical footprint and limited payload restrain the maximum battery, the printed circuit board size and exclude any form of active cooling. Nowadays, μcontroller units (MCUs) are the only computing platforms that meet the nano-UAV's power and form-factor constraints.

MCUs feature simple RISC host processors (e.g., ARM Cortex-M) with low computational capabilities and no virtualization support, to which they expose just a few hundred kBytes of on-chip SRAM scratchpad memory (SPM) [5]–[11]. To deliver more advanced computational capabilities, state-of-the-art (SoA) MCUs integrate accelerators with high data processing capabilities [5]–[11]. Usually, ultra-low-power (ULP) devices' accelerators are hardwired application-specific data-paths [9], [10] which achieve the best energy efficiency but are tailored to a single application domain, leading to poor programmability and a high nonrecurring engineering cost [12] while occupying a considerable part of the scarce area resources. To improve the overall versatility of the SoC, recent works replace ASIC accelerators with fully programmable parallel accelerators [7], [8] that achieve competitive energy efficiency while maintaining significant flexibility, and hence make the most out of the available power and area.

The increase in computing capabilities of SoA MCUs has enabled nano-UAVs to achieve autonomous flight while executing intelligent auxiliary tasks. For example, Quantized Neural Networks (QNNs) have been proposed to carry out obstacle avoidance [2] or human pose estimation and object detection [3]. At the same time, floating point (FP) digital-signal-processing (DSP) computation has been proposed for path planning or structural build monitoring [13], [14]. Also, a recent trend for edge devices is online learning, which enables

a small portion of the Neural Networks (NN) training to happen on the edge, increasing the accuracy and reliability directly on the field [15], [16]. Nevertheless, even the most advanced MCUs, supporting this new class of applications, lag behind in terms of software support. Due to the small amount of available memory and the simplicity of their host CPU, SoA MCUs only provide close-to-metal software environments, based on minimal real-time operating systems (RTOS) or simple bare-metal runtimes. However, enabling the execution of full-fledged OSs (like Linux), securely along with the real-time control applications, would allow nano-UAVs to leverage an existing, mature, and solid software stack and hence ease the software development [17]. In this context, this work presents a step forward in the current and future generation of autonomous nano-UAVs. We present Shaheen, a $9mm^2$ 200mW heterogeneous System-on-Chip (SoC) implemented in $22\,\mathrm{nm}$ FDX technology that couples an application-class RV64 host processor with a low-power HyperRAM memory controller and with a flexible cluster of eight RV32 cores, providing best-in-class energy efficiency and performance for IoT applications. The design is fine-tuned to accommodate the requirements of emerging nano-UAV applications.

The host includes hardware virtualization support [18]: to the best of our knowledge, it is the first silicon implementation fully compliant with the ratified RISC-V Instruction Set Architecture (ISA) Hypervisor extension[1], enabling the secure coexistence of an RTOS and a full-blown OS onto the same host core. In particular, the Hypervisor extension aims to provide confidentiality and integrity of virtual machines (VM) by enforcing isolation (via two-stage virtual memory) between multiple consolidated guest OSes, i.e., General Purpose OS (GPOS) and RTOS. To further isolate the execution of these coexisting software stacks (trusted and untrusted), prevent security threats, and ensure multi-domain operations, the host core features Physical Memory Protection (PMP) [19] and ISA and micro-architecture extensions for timing channel mitigation [20]. Namely, the PMP aims to provide confidentiality and integrity by limiting the physical addresses accessible by software running on CVA6. PMP enforces the separation between the bare-metal firmware (running in machine mode) and everything else through a set of additional registers, which specify the physical memory access privileges (read, write, execute) for each physical memory region. Lastly, timing channel mitigation aims to provide confidentiality by eliminating side-channel attacks.

Apart from a 1MB on-chip SRAM SPM, Shaheen connects to up to 512MB of off-chip low-power HyperRAMs [21] directly on the main interconnect, through a low-power, low-cost $0.27mm^2$ $1.6\,\mathrm{Gbps}$ fully-digital memory controller. Relying on HyperRAMs instead of high-end LPDDR4/5 memories, typically integrated into embedded Linux-capable systems, frees Shaheen from expensive and proprietary memory controllers with large mixed-signal PHYs, while still exposing hundreds of MB to the host processor and matching the tight power, form factor and cost requirements of nano-UAVs.

[1]SiFive, Ventana, and StarFive have announced RISC-V CPU designs with Hypervisor extension support, but we are not aware of any silicon available on the market yet.

TABLE I: UAVs taxonomy by vehicle class-size [2].

Vehicle class size	Diameter : Weight $[cm : kg]$	Power Budget $\frac{\text{Total[W]}}{\text{Compute[W]}}$	Onboard computer
standard [22]	$\sim 50 : \geq 1$	$\geq$ 100 / 5-15	Desktop/Emb.
micro [17]	$\sim 25 : \sim 0.5$	$\sim$ 50 / 2.5-7.5	Embedded
nano [2]	$\sim 10 : \sim 0.01$	$\sim$ 5 / 0.25-0.75	MCU
pico [4]	$\sim 2 : \leq 0.001$	$\sim$ 0.1 / 5-15E-3	ULP

The cluster integrates the so-called Flex-V cores. The Flex-V core extends the RISC-V ISA with custom instructions for reduced-precision single instruction multiple data (SIMD) FP-based computation and byte and sub-byte mixed-precision QNN inference, achieving State-of-the-Art (SoA) software power and energy efficiency. Thanks to the aggressive optimizations, the cluster achieves up to 22.5 GigaOperations per second (GOp/s) and $90\,\mathrm{GOp/s}$ on 8-bit and 2-bit integer kernels, enabling low-latency mixed-precision QNN-based autonomous navigation [2], [3]. Furthermore, the cluster achieves up to 4 GigaFloating-Point Operations per second (GFLOp/s) and $7.9\,\mathrm{GFLOp/s}$ on FP32 and FP16 kernels enabling DSP and online training of neural networks (NNs) [15], [16].

To sum up, compared to the State-of-the-Art MCUs for nano-UAVs, Shaheen is the first one coupling:

- an RV64 host with Hypervisor support and security features,
- a low-power memory controller exposing hundreds of MB at up to 1.6Gbps to the host core,
- a fully-programmable parallel RV32 cluster providing SoA software performance for IoT,

while keeping the overall power envelope within 200mW.

The structure of this manuscript is as follows: in Section II, we will present an overview of the State-of-the-Art SoC for UAVs. Following this, Sections III and IV will delve into an exhaustive discussion of Shaheen's architecture, its implementation, and the measurements obtained from the silicon prototype. Moving forward, Sections V and VI will address the software stack and provide a benchmark of the cluster's performance and energy efficiency on a relevant set of applications for nano-UAVs. In Section VII, we will compare Shaheen with similar silicon prototypes from both industry and academia. Finally, Section VIII will summarize our results and offer insights into potential future research directions.

II. Related work

Table I shows the four categories of UAV systems according to size, weight, power budget and onboard processing platform. The latter two characteristics are tightly coupled, as only around $\sim 5-15\%$ of the power budget is allocated to computation [2]. Across all the categories of UAVs, autonomous navigation is achieved by the combination of two components: mission control and flight control. Mission control is the high-level decisional part of the navigation algorithm, e.g., path planning [23], optimization-based control [24], etc. To carry out these types of tasks, SoA drones mostly rely on machine learning (ML) algorithms [2], [17]. Flight control, on the other hand, is the actuation of the output decisions of mission control: it collects data from the sensors to determine the

TABLE II: State-of-the-Art SoCs for UAVs.

SoC (Proxy price)	Target Platform	Task	CPU : Max freq [MHz]	Supported OS	HW Virt. support	Parallel accelerator	Accelerator FLOp/s	Power envelope : Technology	Main memory
NVIDIA Jetson TX2 [29] (100-150$)	Standard/ micro-size	Mission	4×Cortex-A57 : 2 GHz	Linux	✓	256x Pascal CUDA	1.33 TFLOp/s	7.5-15W : 16nm	8GB LPDDR4
Intel Atom x7-E3950 [30] (50$)	Micro-size	Mission	4xIntel Atom : 2GHz	Linux	✗	GPU Intel HD 505	230 GFLOp/s	∼10W : 14nm	8GB LPDDR3
Allwinnner H3 [17] (<10$)	Micro-size	Mission	4xCortex-A7 : 1.2GHz	Linux	✓	GPU Mali-400 MP2	10 GFLOp/s	>1W : 40nm	512MB LPDDR3
STM32-H7 [5] (15$)	Standard/ micro-size	Flight	Cortex M7 + Cortex M4 : 480 - 240	Linux	✗	-	<500 MFLOp/s	<200mW : 40nm	512kB SRAM
STM32-F4 [6] (15$)	Nano-size	Both	Cortex M4 : 180	RTOS	✗	-	-	<200mW : 90nm	1MB SRAM
GAP8 [7] (40$)	Nano-size	Both	Ri5cy : 250	RTOS	✗	8xRi5cy	-	<100mW : 55nm	1.5MB SRAM
GAP9 [7] (40$)	Nano-size	Both	Ri5cy : 450	RTOS	✗	9xRi5cy	<3GFLOp/s (FP32)	<50mW : 22nm	1.5MB SRAM
Kraken [8] (*Prototype*)	Nano-size	Both	Ri5cy-NN : 350	RTOS	✗	8xRi5cy-NN	<3GFLOp/s (FP32)	<300mW : 22nm	1MB SRAM
This Work (*Prototype*)	Nano-size	Both	RV-64 : 600	Linux +RTOS	✓	8xFlex-V	7.9GFLOp/s (FP16)	<200mW : 22nm	8-512MB HyperRAM

vehicle's state and generates the control law, which manages the actuators [25]. Flight control is often based on cascade PID control [26], especially in the case of nano-UAVs [27], [28], and it is not as computationally intensive as mission control, but it requires low-latency guarantees. As a consequence, also in the context of standard and micro-drones, flight control is usually carried out by simple MCUs with a predictable execution time like the STM32-H7 [5] integrated into the Pixhawk board [25]. Table II shows some mainstream SoCs successfully deployed on drones of standard, micro, and nano size. For each SoC, it highlights the different computational capabilities and power envelope, as well as the specific tasks and UAV platforms they are suited for, detailed in the three sections below. Sections II-A and II-B describe the state of the art of standards, micro, and nano UAVs SoCs.

A. SoCs for Standard and Micro-sized UAVs

As table I shows, micro-size drones integrate embedded computers, while standard-sized drones can even accommodate desktop processors. Nevertheless, embedded processors can nowadays deliver performance in the order of hundreds of TOp/s and hundreds of GFLOp/s, which has proven to be sufficient to support the full flight stack for mission control, both for micro [17] and standard-size UAVs [22].

Embedded computers integrate high-end SoCs with application-class cores, supporting virtualization and various privilege levels (and hence full-fledged OSs), embedded GPUs, and GBytes of high-performance off-chip LPDDR/DDR4/5 memories, connected through expensive, large and power-hungry mixed-signal DDR controllers, all within a power envelope of few watts [29], [31]–[33].

The NVIDIA Jetson TX2 is claimed to be *"the fastest, most power-efficient embedded artificial intelligence (AI) computing device"* by NVIDIA [29] and it is the board of choice for the Agilicious drone [17]. It features a Quad-core Cortex A57 running up to 2GHz and a Pascal CUDA GPU, which can deliver up to 1.33TFLOp/s resulting in an overall power consumption of more than 7.5 W. The Intel Atom x7 is the heart of the Intel UpBoard platforms, as big as a credit card. It features 4 Intel Atom processors running up to 2GHz and an Intel HD 505 GPU delivering up to 230GFLOp/s and roughly consuming 10W. Another compute hardware platform commonly used on autonomous UAVs is the NanoPi Neo Air, which integrates an Allwinner H7 SoC with a quad-core CortexA7 and a Mali-400 MP2 GPU, delivering up to 10GFLOp/s. All these SoCs offer a mainstream Ubuntu-ready software stack and virtualization capabilities and can handle very sophisticated and complex applications. However, due to their power envelope, size, and the necessity for high-end off-chip memories, these SoCs can only be integrated into standard and micro-UAVs.

Naturally, Shaheen can not compete with these architectures in terms of performance, but our approach borrows the best of their characteristic while targeting a much smaller power envelope. Firstly, to mimic high-end SoCs with their heterogeneous GPU-based architecture, Shaheen integrates an RV32-based parallel programmable cluster along with an RV64 CPU. Secondly, it exposes a significant amount of off-chip main memory to the CPU. However, instead of high-performance DRAMs (LPDDR3/4/5) that are connected through large, proprietary and expensive mixed-signal PHYs with a high pin count (>30), Shaheen leverages HyperRAMs, which are fully-digital low-power small-area DRAMs with less than 14 pins and feasible to be deployed on nano-UAVs. A similar approach is adopted in Cheshire [34], which is not optimized for nano-UAV applications. Cheshire revolves around CVA6 as Shaheen and exposes up to 1GB of Reduced Pin Count (RPC) DRAM memory, which uses a minimum number of signals to deliver DDR3-level in-system bandwidth at the cost of 22 switching signals for a 16-bit wide data bus [34], [35]. While RPC and

the related controller offer higher bandwidth than HyperBUS, the RPC protocol is more convoluted, leading to higher design complexity and a bigger area, mostly due to the four 8kB buffers [34]. More importantly, Cheshire's CVA6 does not feature hardware virtualization support and micro-architectural extensions for timing channel mitigations. Lastly, while being easily extensible through the AXI4 interface, Cheshire's silicon prototype does not integrate a parallel accelerator, heavily limiting the offered performance. To sum up, Shaheen is the first silicon implementation of a heterogeneous MCU coupling an RV64 core with a cluster of eight RV32 cores and tens of MB of main memory.

B. SoCs for Nano-UAVs

A state-of-the-art MCU for nano-UAVs platforms is the STM32-F4 [6]. The STM32-F4 is the computational unit of the Crazyflie [36] platform, integrating a Cortex-M4 core and 192kB of on-chip SRAM with 180MHz of maximum operating frequency. Its low performance and small memory capacity limit the autonomous navigation capabilities of the nano-drone when compared to embedded computers. To this extent, two kinds of approaches have been proposed: minimization of the workload [37] or offloading of the mission control computation to an external base station [38], limiting the MCU to flight control. The latter approach presents severe drawbacks, in the first instance, it introduces network-dependent latency, limiting the maximum distance from the workstation to a few tens of meters. Also, the data transmitted are subject to noise on the transmission channel, limiting reliability, and eavesdropping on confidential data [39].

To offer enhanced computational capabilities within a small power budget, recent works also propose SoCs featuring hardwired ASIC accelerators designed for specific UAV applications, like, for example, motion-control [9], visual-inertial odometry (VIO) [10], simultaneous-localization-and-mapping (SLAM) [40], or QNN inference [7], [8]. These accelerators achieve impressive energy efficiency, in the order of hundreds of TOp/s/W, by carefully mapping the target algorithm to the hardware. For example, many accelerators exploit the inherent parallelism of the target application, such as using a systolic array for motion control [9]. Another common approach exploits reduced-precision arithmetic, as in SLAM [40] and VIO [10], to reduce the memory footprint and the datapath size. Exploiting both parallelism and reduced-precision computation is also a well-established technique to accelerate QNN inference and training, due to the nature of such algorithm. For example, accelerators like the NE16 in GAP9 [7], the HWCE in GAP8 [7], and the ternary weight neural-network (so-called CUTIE) accelerator in Kraken [8], able to reach peaks of 11.6 TMAC/s, have been proposed to speed up QNN inference. However, due to poor flexibility and programmability, these accelerators have to anyway rely on general-purpose CPUs to achieve end-to-end flight. Furthermore, the high area cost per device makes them hard to adopt as they risk becoming obsolete due to the rapid evolution of the target nano-UAVs applications.

To overcome these limitations, recent MCUs integrate parallel fully-programmable and flexible accelerators, that have successfully proved to enable autonomous navigation [2], [3]. Namely, GAP8, GAP9 [7] and Kraken [8] are MCUs with enhanced computational capabilities, based on parallel programmable accelerators. GAP8 and GAP9 are commercial products by GreenWaves Technologies compliant with the so-called Crazyflie-AIdeck [41] board, which is meant as a companion of the Crazyflie to offload the mission control tasks [2]. GAP8 embeds the so-called Ri5cy [7] core as host CPU and 1.5MB of on-chip SRAM memory, accompanied by a parallel programmable cluster of other eight Ri5cy cores delivering up to 150 GOP/s on 8-bit data. Ri5cy is a 4 pipeline-stages core compliant with the so-called XpulpV2 ISA, a custom RISC-V ISA based on RV32 with extensions for DSP and ML applications, with support for 16/8-bit SIMD operations and hardware loops. GAP9 is an improved version of the GAP8 processor. It is fabricated in a more advanced node than GAP8, halving the power envelope and it features as well 2MB of non-volatile SVM memory. Also, differently from GAP8, GAP9's cluster includes 4 FPUs with FP16/32 support. Lastly, Kraken [8] is a research prototype based on the same heterogeneous architecture of GAP8 and GAP9, i.e., an RV32 CPU along with an eight RV32 core cluster, which delivers up to 90 GOp/s on 2-bit data. Kraken's RV32 cores are a more advanced version of Ri5cy, i.e., the Ri5cyNN cores [8] with support of sub-byte SIMD operations and fused *Mac&Load* instructions, which enable the concurrent execution of SIMD dot-product and memory accesses, increasing the computation efficiency up to 94%. Kraken embeds 1.5MB of on-chip SRAM memory and the CUTIE accelerator, able to achieve up to roughly 90k Ternary-MACs per cycle. Furthermore, it provides an event-based camera, tightly coupled with a Spiking Neural Engine accelerator. When compared to traditional cameras, event-based cameras offer high temporal resolution (in the order of μs), very high dynamic range (140 dB vs. 60 dB), low power consumption, and high pixel bandwidth (on the order of kHz) resulting in reduced motion blur [42].

Shaheen's approach leverages the best from these advanced AI IoT SoCs, integrating its own fully-programmable parallel 8-core RV32 cluster accelerator. Shaheen's RV32 cores stem from the Ri5cyNN cores and are further enhanced with mixed-precision support to eliminate the massive software overhead necessary for packing and unpacking data when executing mixed-precision sub-byte kernels, providing up to 8.5x speed-up over Kraken and less than 5.6% extra area resource over the baseline core without extensions. In addition, Shaheen addresses a major limitation of SoA MCUs: the software stack based on lightweight RTOSs or simple bare-metal runtimes. Programming applications on these stacks is hard, owing to (i) the lack of virtualization capabilities of the host CPUs and (ii) the small amount of memory directly accessible through loads and stores, which limits the maximum software memory footprint. In the context of MCUs, memory resources coincide with on-chip SRAMs and off-chip DRAMs. The first ones provide high bandwidth but are limited to a few hundred kB, due to the area and power constraints [21]. The latter ones offer one order of magnitude more capacity but are much slower and are typically accessed only through explicit input-output copy functions. Thus, beyond SoA, to support a richer software

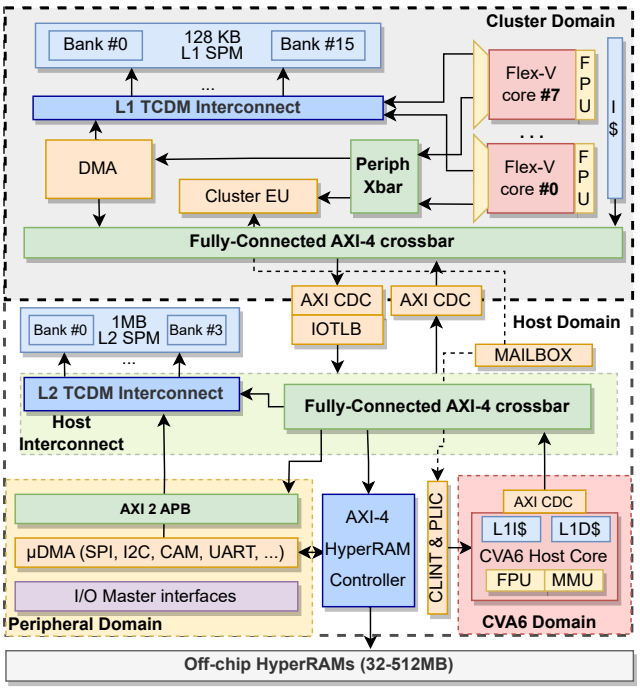

Fig. 1: Shaheen architecture block diagram.

stack while offering the advanced computing performance and energy efficiency of the RV32-based cluster, Shaheen integrates an RV64 core with advanced virtualization and security features, along with up to 512MB of main memory. This enables the secure coexistence of rich and mature general-purpose OS and bare-metal RTOS on the same platform and eases porting of feature-rich software stacks for robotics, such as ROS [43].

III. Shaheen Architecure

Shaheen consists of 4 clock domains, as illustrated in Fig. 1: (i) the CVA6 domain, where the host core is; (ii) the host domain, including the main interconnect and 4 256kB interleaved SRAM banks; (iii) the cluster domain, served by 16 16kB interleaved SRAM banks and 8 specialized RV32 cores; and (iv) the peripheral domain.

A. CVA6 host core

CVA6 [44] is the heart of Shaheen. It is an open-source 6-stages, single-issue, in-order, 64-bit Linux-capable RISC-V core, supporting the RV64GC ISA variant, SV39 virtual memory with a dedicated Memory Management Unit (MMU), three levels of privilege (Machine, Supervisor, User), and PMP [19]. In the context of this work, the baseline version of CVA6 has been enhanced with 2 extra features to provide high-assurance isolation between the different applications co-existing on the core:

1) hardware support for virtualization compliant with the ratified 1.0 version of the RISC-V Hypervisor specification [18].
2) temporal fence instruction, namely *fence.t* [20], which flushes μarchitectural state and enables the OS to close covert channels with a low increase in context switch costs and negligible hardware overhead.

Virtualization Mode (V)	Nominal Privilege	Name	2-stage translation
0	U	User (U) mode	Off
0	S	Hypervisor-extended Supervisor (HS) mode	Off
0	M	Machine (M) mode	Off
1	U	Virtual User (VU) mode	On
1	S	Virtual Supervisor (VS) mode	On

TABLE III: RISC-V ISA privilege modes with the Hypervisor extension.

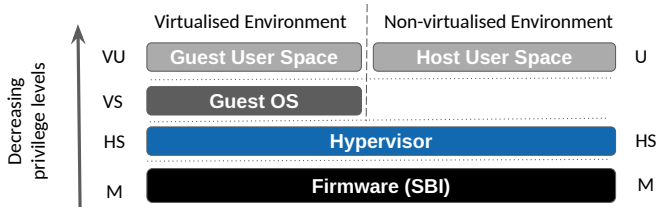

Fig. 2: RISC-V privilege levels.

Such new features are relevant to many UAV applications such as the co-existence of full-fledged and real-time OSes (both custom and legacy), as well as isolation for safety and security reasons.

1) H extension: Tab. III shows the different privilege modes when implementing the Hypervisor extension and Fig. 2 the resulting software stack. The *nominal privilege* modes are *machine* (M), *supervisor* (S), and *user* (U). The Hypervisor extension adds the *virtualization mode* (V), indicating whether the hart is currently executing in a guest (V=1) or not (V=0). When V=0, the S-mode is modified into the *hypervisor-extended supervisor* (HS) mode, well suited to host both type-1 and type-2 hypervisors. Other than in the HS-mode, when V=0, the hart can either be in M-mode or in U-mode atop an OS running in HS-mode. When V=1, two new privilege levels are added, namely the *virtual supervisor* (VS) mode and the *virtual user* (VU) mode. Also, the hypervisor extension defines a second stage of translation (the so-called "G-stage") to virtualize the guest memory by translating guest-physical addresses (GPA) into host-physical addresses (HPA).

To enable these new execution modes, the Control Status Register (CSR) and Decode modules have been modified. The CSR module was extended to implement the first three building blocks that comprise the hardware virtualization logic, specifically: (i) access logic and permission checks for VS-mode and HS-mode CSRs, (ii) delegation and triggering of exceptions and interrupts, and (iii) handling of trap entry and exit. The Decode module underwent changes to enable the decoding of hypervisor instructions (such as hypervisor load/store instructions and memory-management fence instructions), as well as the execution of all VS-mode-related instructions and the triggering of access exceptions.

The MMU's page table walker (PTW) and translation lookaside buffer (TLB) have been modified to support the second stage of translation. The PTW features a new control state to monitor the current stage of translation and facilitate

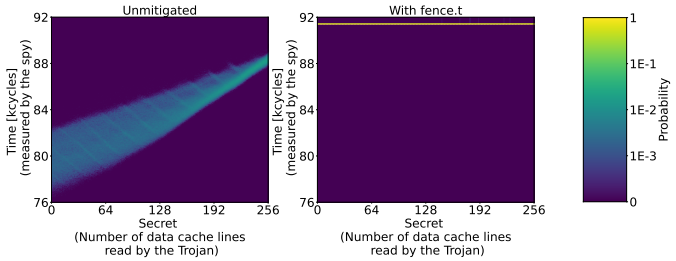

Fig. 3: Channel matrices on the *CHANNEL BENCH* test.

the switching of contexts between VS-Stage and G-Stage translations. Finally, the TLB entries have been extended to store both VS-Stage and G-Stage Page Table Entries (PTEs), as well as their corresponding permissions and virtual machine identifiers. Overall, all these modifications account for less than 6% extra area and hardware while enabling the safe co-existence of a full-fledge guest-OS (executed in VS-mode) together with a bare-metal RTOS (executed in U-mode).

2) fence.t: The CVA6 core also implements the *fence.t* instruction [20]. This instruction is added to CVA6's ISA to prevent *timing channels*: exploitable hardware resources holding state depending on the execution history (caches, TLBs, branch predictors, and prefetchers) can leak information if not properly reset during a context switch. Timing channels can be exposed by *prime-and-probe* attacks [20]. In these kinds of attacks, the spy first brings the target hardware resource into a known state (*prime*). In the following time slice, the OS switches to an application containing a Trojan, which accesses a subset of the hardware resource to encode a secret. Finally, when the execution switches back to the spy, it again probes (*probe*) the whole buffer and observes an execution time t, correlated with the encoded secret. For data caches, the spy traverses a large buffer of n lines so that the Trojan can then transmit a secret $s \leq n$ by touching s lines: in the last time slice, the spy decodes s from the measured execution time t.

In this context, the *fence.t* extends the control that the OS, or the Hypervisor, has over the hardware. Namely, it provides the capability of clearing vulnerable microarchitectural states to enable a history-independent context-switch latency by flushing the caches and the TLB and resetting the internal FSMs of the core. The *fence.t* has been validated against prime-and-probe attacks from the *MASTIK* toolkit [20], [45]. These attacks are implemented within Ge's *CHANNEL BENCH* [46], [47] suite, which provides a minimal OS and data collection infrastructure, running on an experimental version of seL4 supporting timing protection. To visualize the correlation between s and t, we use channel matrices. A channel matrix represents the conditional probability of getting an execution time t, having an input secret s. In Fig. 3, we represent the channel matrix as heatmaps: s (the secret encoded by the Trojan by touching $s \leq n$ data cache lines) varies horizontally, and t (the execution time measured by the spy) varies vertically, bright colours indicate a high probability and dark colours indicate a low probability of measuring such t, given a certain s.

Fig. 3 shows the channel matrices on the *CHANNEL BENCH* test for CVA6's write-through L1 data cache. On the left it is shown the matrix when not using the *fence.t*: the correlation between the Trojan's secret and the spy's probe time indicates a covert channel. On the right, when using the *fence.t*, there is no correlation. With less than 320 additional clock cycles to the context-switch latency (insignificant at typical switch rates of 1 kHz), the *fence.t* requires a low implementation effort and negligible hardware costs.

B. Host & Peripheral Domain

The host domain leverages the popular AXI4 protocol [48] for the main interconnect. Namely, it includes a 64-bit AXI4 crossbar delivering up to 32Gbps on each AXI4 port, respectively on read and write channels. It also includes 4 256kB SRAM banks, composing a 1MB L2 ScratchPad Memory (L2SPM) delivering up to 64Gbps, either for writing or reading. The L2SPM is meant to (i) store data to be shared with off-chip peripherals, (ii) store the cluster code, (iii) for fast communication between CVA6 and the cluster, and, more in general, (iv) for low-latency ($<$10 clock cycles) and predictable accesses.

To enable independent data transfer from peripherals to the SoC, Shaheen includes in the peripheral domain the so-called *"μDMA subsystem"* which is a controller intended to autonomously serve a set of I/O interfaces popular in critical applications. Such interfaces include for instance HyperBUS, I2C, (Q)SPI, CPI, SDIO, UART, CAN, PWM, and I2S. The μDMA exports two ports, one for receiving and one for sending data, to read/write data from/to the L2SPM SRAM memory to/from the off-chip peripherals [7]. Shaheen also features an open-source Linux-compliant Ethernet IP, to be fully compliant with the Pixhawk standard [25], popular open-source hardware specifications and guidelines for drone systems development.

1) HyperRAM memory controller: Fig. 4 depicts Shaheen's HyperRAM controller, which provides a configuration APB port and an AXI4 subordinate port. It connects the SoC with off-chip HyperRAMs, compliant with the HyperBUS protocol, which is a fully digital protocol counting $11 + n$ pins: 3 control pins, n Chip Select (CS), and 8 Double-Data-Rate pins used both for commands and data [21]. Depending on the off-chip memory models, the controller exposes between 32MB and 512MB to the interconnect, and it provides up to 1.6 Gbps. HyperRAMs are the main memory of choice for Shaheen because, differently from high-end DDR DRAM memories, they target a much lower power consumption and silicon footprint while guaranteeing enough bandwidth for advanced AI IoT applications and capacity to boot embedded SPM Linux [21].

There are two distinct modules within the HyperRAM controller, i.e., the PHY controller (back-end) and the front-end, operating in different frequency domains. The front-end module consists of an AXI4-to-PHY converter and a specialized μDMA engine channel accessible through APB to execute software-programmed DMA transfers. The AXI4 and μDMA transactions are multiplexed towards the PHY, which translates the incoming data packets into HyperRAM transactions and vice versa. The AXI4 front-end enqueues

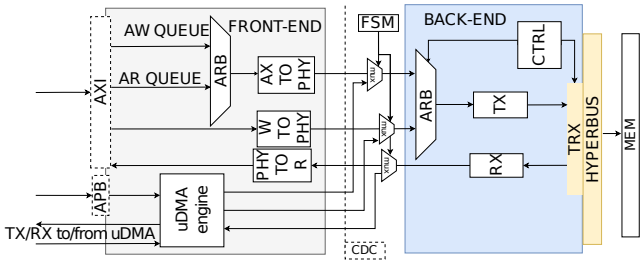

Fig. 4: HyperRAM memory controller architecture.

the AXI4 transactions individually and lets through only one read or one write at a time and converts it into a request for the PHY. At this point, the back-end translates the request into a command for the HyperRAMs and issues it over the HyperBUS. Following, in the case of a write, the W channel transactions get converted into multiple PHY data packets. For reads, the PHY back-end sends data packets to a converter that then populates the R channel. The μDMA engine directly connects the L2SPM and the back-end and can generate both 1D and 2D burst transactions. These features are highly valuable for the efficient execution of ML algorithms on the cluster, as it is achieved through explicit orchestration of the data movement between the off-chip memory, the L2SPM and the L1SPM [49].

To double the bandwidth and the capacity, Shaheen's back-end module controls 2 HyperBUS interfaces in parallel, and it controls 2 memories on each HyperBUS, with 2 dedicated CS. Each memory is seen as a memory block of 16 bits width and N rows, programmable at runtime according to the onboard memories available. The pair of memories on the same CS of the two different buses are mapped as interleaved, hence occupying the first $2 \cdot 2 \cdot N$ Bytes. The other pair of memory is placed contiguously on top.

C. Parallel Programmable Cluster

While the host core supports advanced virtualization, security and isolation features, it is not optimized for number crunching: when running computation-intensive kernels is needed, it invokes the cluster. The cluster domain is a programmable parallel accelerator connected to the main host interconnect through a controller and a subordinate AXI4 port. The cluster is composed of 8 70kGE 4-pipeline stages RV32 cores, optimized for general-purpose DSP and ML applications, described below. The cores share 16 16kB interleaved SRAM banks, composing a 256kB L1 SPM, accessible through a single clock cycle latency logarithmic interconnect, providing up to 256 Gbps at 500MHz. A hierarchical instruction cache, composed of 8 512 Bytes private caches and a 4kB of 2-cycle latency shared cache, assists the cores. It is implemented with latch-based SCM to improve energy efficiency over energy-expensive SRAM cuts. The cluster also includes a DMA with one 64-bit AXI4 port and 4 32-bit ports towards the L1SPM for high-bandwidth, low-latency transactions to/from the L1 SPM. Leveraging explicit memory DMA transfers and scratchpad memories, double-buffering

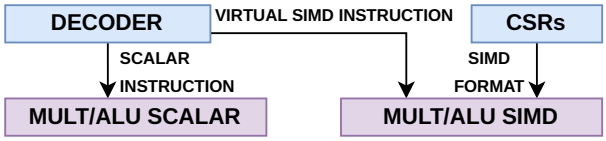

Fig. 5: Instruction decoding during the status-based execution.

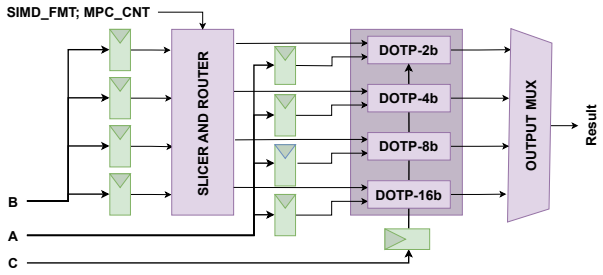

Fig. 6: Dotp unit Datapath.

and custom ISA extensions, the cluster avoids the hardware overhead of expensive data caches while maximizing the utilization of memory and computing resources [49].

1) Flex-V cores: The 8 RV32 cores are the so-called Flex-V cores [50]. Each core has a dedicated FPU unit supporting FP32, FP16 and bfloat16 types, supporting SIMD instructions on lower precision data. Also, all the cores share a single Floating Point division and square root operations unit (DIV/SQRT). The Flex-V core is an aggressively optimized version of the Ri5cy core [51], which support the XPulpV2 ISA extension, considered as the baseline. The Ri5cy core already provides custom instructions to accelerate the execution of ML and DSP workloads, namely, it supports post-increment LD/ST, hardware loops, and SIMD instructions down to 8-bit precision.

To enhance the performance of sub-byte uniform linear kernels, the XpulpNN ISA has been proposed [8], which extends XpulpV2 ISA with 4- and 2-bit SIMD operations. Additionally, it introduces fused *Mac&Load* instructions enabling simultaneous execution of SIMD dot-product operations alongside memory accesses, almost doubling the computation efficiency. More precisely, the *fused Mac&Load (mlsdotp)* instruction combines a SIMD *dot-product*-like operation with a load operation performed during the writeback stage. Doing so enables replacing the non-stationary data in a register to directly feed the next *Mac&Load* instruction with it. To decouple and simplify the *Mac&Load* execution, the XPulpNN core integrates six additional 32-bit registers, forming the so-called Neural Network Register File (NN-RF), enabling the Load operations (of weights and activations) during the *Mac&Load* write-back stage, which could not be performed otherwise on the general purpose register file (GP-RF). However, when dealing with mixed-precision inputs, the performance of XpulpNN degrades significantly because of the substantial software overhead required for packing and unpacking data.

To overcome this limitation and maximize the computational unit utilization, Flex-V further extends XPulpNN with mixed-precision operation support. To efficiently enable arbitrary mixed-precision operations while avoiding the prolifer-

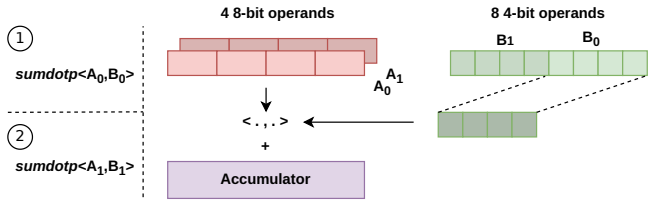

Fig. 7: Execution flow of a mixed-precision *sumdotp* instruction between 8-bit operand A and 4-bit operand B.

ation of extra instructions, Flex-V exploits the *dynamic bit-scalable execution mode*: the ISA instruction only encodes the type of the operation, while the format is specified by a CSR in the core. Figure 5 illustrates the relative decoding process: the decoder retrieves all the necessary information from the instruction and transmits it to the EX stage. If the received op-code corresponds to a *Virtual SIMD* instruction, such as a *(ml)sdotp*, the decoder activates the SIMD functional unit which will execute the instruction according to CSRs values and to signals from the dedicated *MAC&Load* and *Mixed-precision* controllers (MCD). Figure 6 shows the mixed-precision Dot Product (Dotp) unit. This unit integrates a *Slicer&Router*, responsible for the extraction of the 4- and 2-bit operands from a 32-bit input word, along with the two dedicated units for the sub-byte operations. For example, as shown in Fig. 7, the slicer is needed in the case of a *sum-of-dot-product (sdotp)* operation between an 8-bit operand A and a 4-bit operand B. Since the single instruction can consume just four elements of the eight 4-bit words inside the register, it selects either the 16 MSBs or the 16 LSBs, according to the value of the *MPC_CNT* signal from the MCD. Subsequently, the Router directs the desired elements to the Dotp units following the *SIMD_FMT* signal coming from the CSR, i.e. the DOTP-8b (operating on 8b inputs) for this example.

In the case of mixed-precision kernels, by re-arranging the data natively in hardware, Flex-V alleviates the substantial software overhead needed for pointer management and explicit data unpacking that would be needed otherwise. Table IV shows Flex-V's performance gain over XPulpV2 and XPulpNN on dense matrix multiplication kernels with weights and activations (operands A and B in Figure 7, respectively) of different bit widths. It expresses performance in terms of MAC/cycle, isolating the inner kernel and excluding the non-idealities that arise when running complex real-world applications. While on uniform kernels, Flex-V and XPulpNN achieve the same performance, on mixed-precision kernels, Flex-V outperforms XPulpNN by up to 6.8× and for only 5.6% extra area resources.

2) IOTLB: The cluster accesses towards the host interconnect are mediated by an IO TLB (IOTLB) unit [52]. Since the Flex-V cores cannot perform virtual-to-physical address translation, the IOTLB unit is meant to ease pointer sharing with CVA6 and further prevent cluster unauthorized accesses towards the shared memory. The latter is a fundamental feature for critical applications: without any control, malicious or buggy applications running on the cluster could potentially cause denial-of-service to the host core or break *confidentiality* (i.e., get unauthorised access to sensitive data).

TABLE IV: Flex-V's performance [MAC/cycle] on MatMul kernels, against XPulpNN and XPulpV2 [50].

Input widths [bits]		XpulpV2 [8]	XpulpNN [7]	Flex-V	Flex-V Speedup vs. XPulpV2 / XPulpNN
Act.	Weight	$\left[\frac{MAC}{cycle}\right]$	$\left[\frac{MAC}{cycle}\right]$	$\left[\frac{MAC}{cycle}\right]$	
2	**2**	-	90.8	91.5	-/≤ 1%
4	**2**	-	7.62	51.9	-/6.8x
4	**4**	-	49.5	50.6	-/≤ 1%
8	**2**	4.91	6.07	27.8	5.6x/4.5x
8	**4**	6.38	7.63	27.6	4.3x/3.6x
8	**8**	16.59	26.1	26.9	1.6x/3%

The IOTLB provides 32 entries. For each entry, CVA6 has to specify the starting and ending virtual addresses, the physical base address and the characteristics of the region: if the cluster can access it, and if it is readable or writeable. Before offloading a task to the cluster, the host statically reserves the portions of the main memory to be shared with the cluster and then it programs the entries. Then, once a transaction from the cluster arrives, its address is compared against the 32 virtual address ranges. If it is within one of the available ranges and the cluster has the right permissions, the address is translated through simple subtraction of the virtual base address and the addition of the physical base address. Otherwise, the IOTLB sends an interrupt to CVA6 to notify the cluster's attempt at accessing memory outside the expected regions. Then, the IOTLB behaves as a simple AXI4 subordinate to not break the AXI4 protocol: for write transactions, it accepts the incoming data on the write channel, without propagating them, while for read transactions, it serves as many read beats as needed, providing an arbitrary value set at design time. At this point, the cluster is not aware that the transaction was not allowed and continues the execution until it receives an interrupt from CVA6. If, in this scenario, the cluster's runtime has not been compromised by the malicious/buggy application, the cluster will gracefully interrupt its execution and resume from a known state. If this is not the case and it is not possible to shut down the cluster, the IOTLB will anyway prevent *denial-of-service* attacks and prevent unauthorised access to sensitive data.

IV. Implementation and Measurements

Fig. 8 shows the microphotograph of the Shaheen SoC, highlighting the main building blocks described in Section III. The SoC is implemented in Global Foundries 22nm CMOS FD-SOI technology. It was synthesized with Synopsys Design Compiler 2019.12, while Place & Route was performed with Cadence Innovus 19.10. Shaheen's 4 different clocks are generated by 4 Frequency Locked Loops (FLLs), taking a 32KHz clock in input from an off-chip ring oscillator. The FLLs' maximum achievable output frequency at 0.8V is 600MHz. The different peripheral PHYs (I2C, SPI, HyperBUS, ...) internally feature clock division to further scale down the input clock when needed.

Figure 9 shows the test board developed for the bring-up and measurements. It provides four 8MB HyperRAM chips and a socket to test different chips easily. It also exposes the

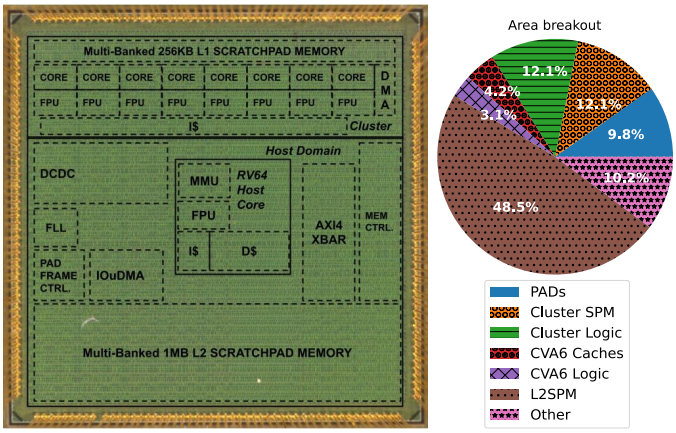

Fig. 8: Die micrograph (3mm x 3mm) and area breakout.

Technology	CMOS22nm FD-SOI (SLVt, LVt)
Chip Area	$9mm^2$
SRAM Memory	1280KB
VDD Range	0.625-0.8V
CVA6/Cluster Max Freq.	600/500MHz
Idle Power	9-19mW (0.625-0.8V)
Avg. Power (CVA6 Active)	45-130mW (0.625-0.8V)
Max. Power (Cluster Active)	75-200mW (0.625-0.8V)

TABLE V: Shaheen SoC features.

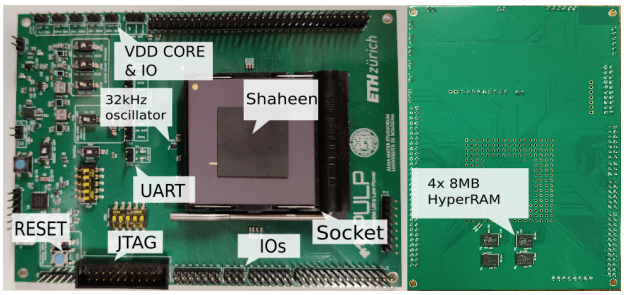

Fig. 9: Shaheen test-board, top and bottom.

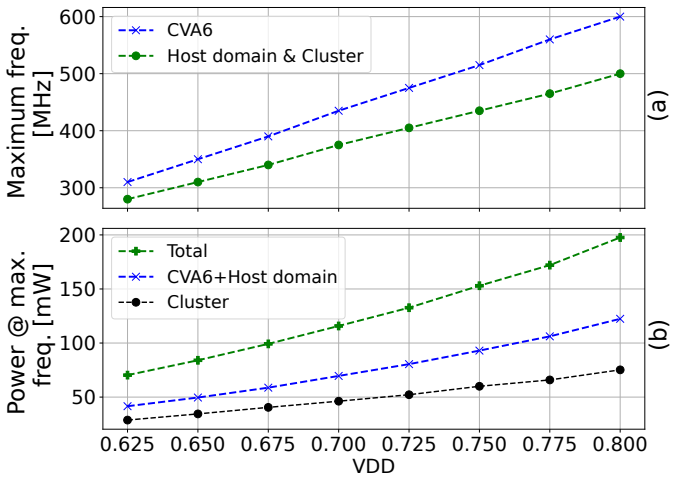

Fig. 10: Maximum frequency and power envelope varying VDD.

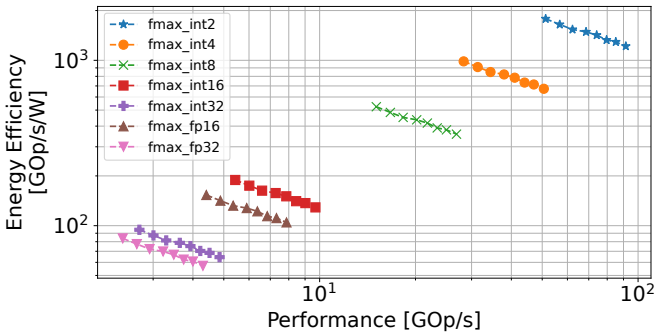

Fig. 11: Pulp cluster energy efficiency on dense matrix multiplication. 1 MAC = 2 Op.

interfaces required to debug the chip, such as JTAG and UART, as well as pin headers connected to all the other interfaces for testing purposes. Finally, it exposes the pin headers to regulate the voltage supply of the two power domains: (i) one for the core logic and the SRAM macros, which we vary between 0.625V and 0.8V, and (ii) one for the IOs, fixed at 1.8V. While the SoC fits all the requirements for Nano-UAV navigation, the board described above has not been designed for flying, but specifically for the testing and characterization of Shaheen.

First, we measure the idle power. To do so, we reduce the frequency of the SoC to 32kHz and clock gate the cluster while CVA6 is in a wait-for-interrupt state, i.e., a for loop of *nop* operations. As reported in Table V, idle power consumption is between 9mW and 19mW, depending on the supply voltage. Fig. 10 (a) shows the measured maximum frequency varying the voltage supply of the host domain, the cluster domain, and CVA6. The cluster and the host domain can run at up to 280MHz at 0.625V and up to 500MHz at 0.8V. Thanks to the more aggressive pipelining, the CVA6 core can reach up to 310MHz at 0.625V and up to 600MHz at 0.8V. Fig. 10 (b) also shows the measured maximum power consumption at the highest achievable frequency for the voltage supply. For these tests, CVA6 runs a dense FP64 matrix multiplication, and the cluster runs a dense INT32 matrix multiplication, both within an infinite loop. Then, we measure and sum the average currents consumed by the two power domains. To get the average power consumption, we measure the power consumption of Shaheen when the cluster is clock-gated, which matches with the power consumption of CVA6, the peripheral, and the host domain together. Then, we also enable the cluster and measure the resulting total power, which coincides with the maximum power consumed by Shaheen. Varying VDD and frequency, the power consumption of CVA6 and the host domain varies from 45mW to 130mW. On the other hand, the cluster consumes from 30mW to 70mW.

Fig. 11 shows the cluster domain energy efficiency varying frequency, VDD and data width. On 2-bit data, the cluster can achieve up to 90GOp/s and up to 1.8TOp/s/W. On 8-bit data, the cluster can achieve up to 26.9GOp/s and up to 540GOp/s/W. All the experiments were performed running on Shaheen the various n-bit matrix-multiplication kernels extracted from the PULPNN library on the software-programmable cores and extracting the MAC/cycle of the inner loops, excluding the initial data arrangement overhead [50].

Lastly, we performed post-layout, parasitic annotated sim-

ulations of the HyperRAM memory controller's netlist to characterize its power consumption. At 0.8V and a speed of 1.6Gbps, it consumes 1.25mW, 70% of which is consumed by the IOs. At 0.625V, delivering 1.1Gbps, the power consumption is 0.8mW, with 75% of the IOs. More details about the HyperRAM controller's performance and power characterization, as well as comparisons with traditional DDR controllers, can be found in [53].

V. Heterogeneous Software Stack

A. Software stack and programming model

Shaheen comes with a mature software stack for heterogeneous programming. On the cluster side, we provide a lightweight bare-metal runtime that allows low programming overhead, and fast hardware functionality validation and performance profiling. On the host side, CVA6 can either run full-ledged Buildroot-based Linux distribution (v5.16.9) on top of the Bao Hypervisor [18] or a bare-metal runtime, and both are equipped with a dedicated driver for the cluster management. The APIs provided by the cluster runtime and the CVA6's driver are already sufficient to run heterogeneous code on the platform. However, one must write two different codes for the host and cluster. To avoid this, Shaheen adapts the OpenMP 5 framework from HERO [52], allowing users to use a high-level, directive-based, intuitive programming interface to efficiently offload the computationally intensive part of a program to the cluster within one single heterogeneous source code. Also, to map the execution of QNNs on the cluster, we adopt the data and execution flow presented in Dory [49]. Dory is a tool that given the description of a QNN in input generates the corresponding C code to be executed on parallel programmable clusters. Dory calculates data tiling solutions fitting the available L1SPM (where it puts the data to be processed by the cluster) and it schedules the DMA data transfers from the main memory to the L1SPM and vice-versa. Thanks to the efficiency of tiling and double-buffering, when the execution is not memory bound, data movements overlap with computation for more than 95% of the execution time [49].

B. Offload mechanism & Performance

To perform the offload, CVA6 *lazily* (at first occurrence) loads the cluster code into the L2SPM to then communicate to the cluster where is the code to execute. Such a mechanism requires a few thousand clock cycles, depending on the length of the code. Hence, when the cluster execution time is very short (<100k cycles), the cluster's offload overhead (i.e., loading the code) dominates the total execution time and reduces the speedup. Based on our (empirical) experience, this is a very uncommon case.

Figure 12 shows the offload speedup and overhead over an FP matrix multiplication. It is important to notice how the cluster can run such a benchmark with reduced precision (down to FP16), exploiting the SIMD extensions otherwise unavailable on the CVA6 core.

The plot on the left in Fig. 12 shows CVA6 and cluster performance at the maximum frequency at 0.8V (600MHz for CVA6, 500MHz for the cluster), and it also highlights the speed up. The figure shows the acceleration when executing the accelerated kernel once or 1000 times on the cluster; the first case represents low code utilization, while the second represents high code utilization. In each execution, the cluster performs the computation on a different couple of input matrices. At the same time, it fetches the input matrices for the next execution and writes back the result of the previous computation. Data movement is performed through the DMA and overlaps computation. On a simple FP32 matrix multiplication, the cluster can deliver up to 4.3 GFLOp/s, which is roughly 27 times more than CVA6. Furthermore, the plot shows once again the benefit of scaling down the number precision, which is not a possibility on CVA6.

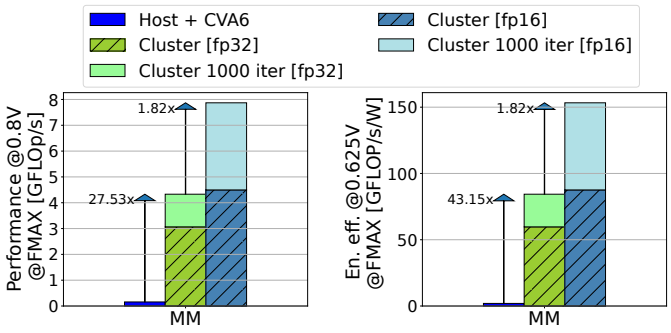

Fig. 12: Offload performance breakout on an FP MM.

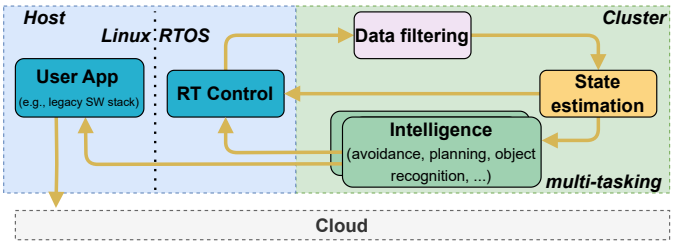

Fig. 13: Shaheen's autonomous mission & flight functional blocks.

The plot on the right in Figure 12 compares the energy efficiency achieved by CVA6 and the host domain against the cluster on the same benchmark, with the IPs working at the maximum frequency at 0.65V (280MHz for the cluster and 310MHz for CVA6). On the reduced-precision matrix multiplication, the cluster can reach up to 157 GOp/s/W, while CVA6 can only provide 2 GOp/s/W, $\approx 80\times$ less.

VI. Benchmarking

Figure 13 presents the loop that Shaheen executes to achieve autonomous flight while executing other auxiliary tasks, and how it maps on Shaheen's hardware. In the first instance, it collects and filters the sensors' input data, which are subsequently used to estimate the current state. Then, in the "intelligence" block it has to independently determine the next state (i.e., what to do next) and carry out the target auxiliary tasks such as object detection, recognition or monitoring [3], [14]. Once the next state is determined, the control part actuates the change. In Shaheen, the first three phases (filtering, state estimation,

and intelligence) are mapped on the cluster hardware, while the real-time control is left to the host core, also executing the general-purpose OS, to leverage its legacy software stack for non-real-time tasks (e.g. transmitting the classification of a detected object to the cloud through the legacy network stack). In this section, we focus on benchmarking the cluster on its target tasks as they are the most computing-intensive phases and potential bottlenecks of the autonomous flight and mission loop.

Filtering of the input data and state estimation as well as intelligence tasks like path planning or structural health monitoring usually rely on general-purpose DSP primitives [13], [14]. At the same time, QNN inference is widely adopted for tasks like classification or recognition for obstacle avoidance or object recognition and localization [2], [3], [17], [54], [55]. However, one limitation of the QNN inference at the edge is the mainstream adoption of the *train-once-deploy-everywhere* approach, which trains the networks offline and then deploys them later on the embedded devices, where no further modifications to the weights happen. This approach prevents the models to adapt in the deployment environment and possibly leads to accuracy degradation and unreliability [15]. On-device learning potentially overcomes this limitation by enabling small portions of the training to happen on the field, directly on the MCU [16]. Thus, we benchmark the proposed SoC on three sets of kernels representative of the different tasks described above, i.e., (i) general-purpose DSP, (ii) DNN, and (iii) online learning benchmarks.

A. General Purpose DSP

Figure 14 shows the cluster performance and energy efficiency over seven open-source FP benchmarks [56] representative of DSP applications for filtering, feature extraction classification, and basic linear algebra functions, relevant both for input data filtering, state estimation but also intelligence tasks such as path planning or structural health monitoring [13], [14]. To show the advantage given by parallelism and reduced-precision computation, such benchmarks are executed both at full precision (FP32) on a single Flex-V core, and then on 8 cores at full and lower precision (FP16 & bfloat16) to exploit the available packed-SIMD support.

Some of the benchmark kernels are representative of digital data acquisition and analysis, such as the Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. To characterize the cluster on frequency-domain applications, we run a decimation-in-frequency radix-2 variant of the Fast Fourier Transform (FFT) and a Discrete Wavelet Transform (DWT), a standard kernel used for feature extraction. As for more state-estimation-oriented kernels, we provide the performance results when executing a K-Means classifier kernel. Lastly, we also benchmark two classical basic linear algebra kernels such as a Matrix Multiplication and a 1D Convolution.

As the plots in Fig. 14 show, on all these benchmarks, thanks to the ISA extension not available on the host core, a single Flex-V running at 500MHz provides from 1× to 3× the performance delivered by CVA6 on a dense and regular matrix multiplication at 600MHz. Furthermore, the parallel execution

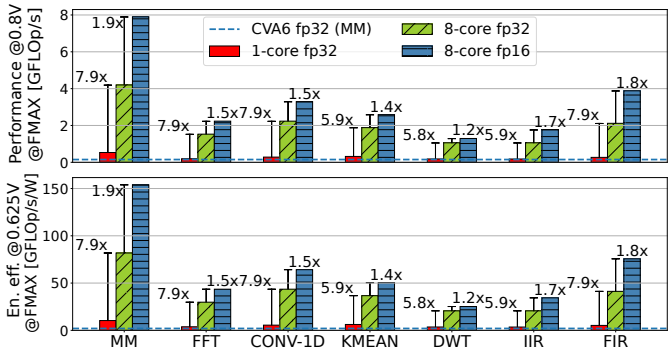

Fig. 14: Performance and en. eff. delivered by the cluster on general-purpose DSP benchmarks.

TABLE VI: Accuracy, memory footprint, perf. & en. of end-to-end networks.

Network	MobilNetV1 (8b)	MobileNetV1 (8b4b)	ResNet20 (4b2b)
Top-1 Acc.	69.3%	66.0%	90.2%
Deg. w.r.t. 8b	-	3.3%	0.15%
Model size	1.9 MB	997 kB	142 kB
MACs	325M	328M	10M
Mem. saved		47%	63%
Avg. Perf. : Latency : Energy [MAC/cycle : ms : mJ] @0.8V -run to sleep-			
XpulpV2 GAP9, 450MHz	5.6 : 126 : 5.87	3.2 : 227 : 10.23	4.8 : 4.9 : 0.22
XpulpNN Kraken, 380MHz	6.0 : 141 : 12.7	2.7 : 319 : 28.76	4.4 : 6.4 : 0.58
Flex-V Shaheen, 500MHz	6.0 : 108 : 8.55	5.8 : 119 : 8.83	11.2 : 1.9 : 0.15

of the benchmarks on 8 Flex-V cores can give an additional speed-up between 5.9 and 7.9 times when compared to single-core execution. Leveraging the reduced precision arithmetic can further provide almost a 2x speed-up, allowing the core to reach up to 7.9GFLOp/s and up to 157 GFLOp/s/W.

B. QNN inference

In this subsection, we focus on two real-world 8-bit QNNs fine-tuned for Nano-UAVs application scenarios, namely Tiny-PULP-Dronet [2] and FrontNet [3], as well as two aggressively quantized mixed-precision QNNs for object detection and classification. The Tiny-PULP-Dronet is a lightweigth QNN based on the ResNet architecture [55] and it enables autonomous navigation within tight spaces avoiding obstacle collision. FrontNet on the other hand is based on the MobileNet [54] architecture and it is used for Human-Robot Interaction (HRI): it allows the nano-drone to recognize a face and follow it. The cluster is able to achieve 320FPS on a Tiny-PULP-Dronet and 260FPS on an optimized 6.7MMAC FrontNet, which is well above the 20FPS needed to achieve autonomous flight [2], [3]. This means that more than 90% of the cluster's computational capabilities are actually available to carry out other activities.

Stemming the analysis from the QNNs mentioned below, we first benchmark the cluster on a relatively big (325MMAC) 8-bit MobileNetV1 [54] for object classification. Then, we extend the analysis to a mixed-precision MobileNetV1 with 8-bit activations and 4-bit (8b4b) weights and an aggressively

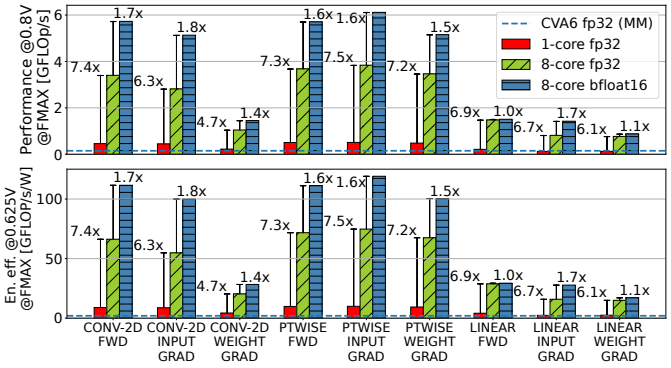

Fig. 15: Performance and En. Eff. delivered by the cluster on FP training benchmarks.

quantized 4b2b ResNet-20 [57] for object detection. The two MobileNetV1 networks have been trained on ImageNet while the 4b2b ResNet-20 targets CIFAR10. As table VI shows, reducing the operands' precision does not automatically jeopardize the accuracy: in the case of the MobileNetV1, there is a 47% memory footprint reduction for a negligible 3% accuracy loss, from 69% to 66% [50], while the ResNet-20 achieves 90.2% accuracy [55]. As shown in table VI, Flex-V is the only version of Ri5cy able to efficiently deal with mixed precision networks: in terms of MAC/cycles, on the 4b-2b mixed-precision ResNet-20, it achieves 2.3× and 2.5× of speedup with respect to XPulpNN and XPulpV2. Table VI also compares the latency and energy consumed by Shaheen over the three networks when running at the maximum frequency at 0.8V, compared to two other 8-core clusters respectively implementing the baseline XPulpV2 instructions or the XPulpNN, namely GAP9 [7] and Kraken [8]. On the uniform-precision MobileNetV1, thanks to the higher frequency and optimized ISA, Shaheen's cluster provides the smallest latency, but not the lowest energy, due to higher power consumption (70mW) when compared to GAP9 (50mW), being the latter tuned for energy-efficient operation. As soon as the mixed-precision extensions can be exploited, Shaheen's cluster emerges both as the fastest and most energy-efficient one.

C. Online training

In this subsection, we benchmark the cluster against a set of open-source kernels to enable online learning on MCU controllers [16]. In particular, we benchmark three very popular layers such as 2D Convolution, Pointwise and Fully-Connected which are the three building blocks of Convolutional NNs (CNNs), used to find patterns in images. Convolutional and pointwise layers are the core building blocks of CNN, where most of the computation happens, and are used to perform feature extraction. The Fully connected layer connects the information extracted from the previous steps (i.e., Convolution layer and Pooling layers) to the output layer and eventually classifies the input into the desired label. For each layer, we consider the three phases of training: (i) the forward pass, to compute the output result and hence the loss, (ii) the backward computation of the gradients with respect to the activations, and (iii) the backward computation of the gradients with respect to the weights. The kernels we leverage map each of these computation phases directly to one matrix multiplication containing all the matrix multiplications needed to obtain the output [16]. Depending on the matrices' shapes, the amount of parallelizable work changes, and hence the performance [58]. Figure 15 shows performance and energy efficiency over such benchmarks. As for the DSP benchmarks, the parallelization provides a significant speed-up for most of them. Except for the weight gradient computation on the convolution kernel, which achieves a 4.7× speedup, the parallelization provides between 6.1x and 7.5x faster execution. At the same time, leveraging the bfloat16 format (providing a wide dynamic range explicitly thought for ML training) and the dedicated SIMD extensions provides up to 1.8x more performance. Overall, the cluster is able to achieve up 6.2 GFLOp/s and 120 GFLOp/s/W on this class of benchmarks.

VII. Comparison with State-of-the-Art

Table VII shows Shaheen against 6 SoCs for UAVs, both from industry and academia. To have a thorough comparison, we extend it also with SoC not explicitly optimized for UAVs but with similar general-purpose software performance and functionalities that could fit the purpose, namely Cheshire [34], the work from Jia et.al. [31], the STM32-H7 [5] and the work by Ju et.al. [11]. From an architectural viewpoint, Ju et. al. [11] consists of a homogeneous systolic array of RV32 cores, while Jia et al. [31] instantiates a cluster of four RV64 cores along with a set of hardwired ASIC accelerators. More advanced nano-UAV SoCs, such as GAP9 [7] and Kraken [8], incorporate an RV32 CPU that can offload compute-intensive tasks to a parallel cluster of cores with the same ISA.

In this context, Shaheen is the first silicon demonstrator of a heterogeneous RV64/RV32 architecture. When offloading compute-intensive tasks to the fully-programmable parallel cluster of Flex-V cores, performance can be improved by up to 2 orders of magnitude achieving state-of-the-art performance with up to 90 GOp/s on heavily quantized integer tasks and up to 7.9 GFLOp/s/W on 16-bit floating point tasks. Shaheen stands out as the only nano-UAV SoC that provides Linux, hypervisor, and security capabilities to the host enabling the secure co-existence of user applications running on full-fledged OSes and control tasks running on real-time OSes while providing up to 512MB of low-cost and low-power off-chip memory within the power envelope of 200 mW.

VIII. Conclusion

We presented Shaheen: a heterogeneous and flexible SoC implemented in 22 nm FDX technology. Shaheen features a Linux-capable RV64 core, compliant with the v1.0 ratified Hypervisor extension. To the best of our knowledge, it is the first silicon implementation fully compliant with the ratified RISC-V ISA Hypervisor extension. It features support for timing channel protection to isolate concurrent execution of multiple software stacks (trusted and untrusted), preventing security threats and ensuring multi-domain operations. It provides up to

TABLE VII: Comparison with SoA SoCs.

	Neo - Cheshire [34]	*Jia et al. [31]*	*STM32-H7 [5]*	*Ju et al. [11]*	*GAP9 [7]*	*Kraken [8]*	*Shaheen (ours)*
Target	Prototype	Prototype	Product	Prototype	Product	Prototype	Prototype
Technology	65nm	12nm	40nm	65nm	22nm	22nm FDSOI	22nm FDSOI
Die Size	$6,4mm^2$	$21,6mm^2$	-	$4,47mm^2$	$12mm^2$	$9mm^2$	$9mm^2$
CPU / ISA	CVA6/ RV64GC	4xCVA6 / RV64GC	Cortex-M7 + CortexM4 / ARM32	10xRISC-V / RV32	10x RI5CY / RV32	9x RI5CY-XNN / RV32	CVA6 + 8x FLEX-V / RV64GC + RV32
Power Env.	300mW	1.83W	-	116mW-589mW	50mW	300mW	195mW
Max. Freq.	300MHz	1,5GHz	480-240MHz	400MHz	450MHz	330-390MHz	500-600MHz
Memory exposed to the CPU	1GB RPC	2GB SerDes	512kB On-chip SRAM	150kB On-chip SRAM	1.5MB On-chip SRAM + 8-64MB HyperBUS (XIP)	1.5MB On-chip SRAM	1MB On-chip SRAM + 32-512MB HyperBUS
Supported OS	Linux / RTOS	Linux / RTOS	RTOS	RTOS	RTOS	RTOS	Linux + RTOS
Security Features	-	-	Crypto/hash processor	-	AES128/256 acc., PUF	-	fence.t, IOTLB, PMP, Hypervisor
SW INT/FP arithmetic support	SP/DP-FP	SP/DP-FP	SP-FP	-	bfloat16, fp16/32, int32/16/8	bfloat16, fp16/32, int32/16/8/4/2	bfloat16, fp16/32, int32/16, int-mixed8/4/2
Peak SW INT Performance	0.5GOp/s (32b)	1.5GOp/s (32b)	390MOp/s (8b)	16GOp/s (8b)	15.6GOp/s (8b)	22GOp/s(8b-8b) 45GOp/s(4b-4b) 85GOp/s(2b-2b)	26GOp/s(8b-8/4/2b) 50GOp/s(4b-4/2b) 90 GOp/s(2b-2b)
Peak SW FP Performance	0.5GFLOp/s (32b)	1.5GFLOp/s (32b)	240MFLOp/s (32b)	-	3.3GFLOp/s (32b)	3.12GFLOp/s (32b)	4.0GFLOp/s(32b) 7.9GFLOp/s(16b)

512MB of main off-chip HyperRAM memory, large enough to host general-purpose OSs as well as RTOSs. Also, it is the first silicon implementation of a heterogeneous MCU coupling an RV64 host together with a multi-core RV32 cluster, achieving up to 90GOp/s and up to 1.8TOp/s/W on 2-bit integer kernel and up to 26.9GOp/s and up to 540GOp/s/W on 8-bit integer kernels.

After this thorough evaluation, we envision the miniaturization of the testing PCB (see Fig. 9) and development of *ad-hoc* control software, tightly coupled with the physical characteristics of the board, to achieve real-world nano-UAV flight, exploiting Shaheen's secure and scalable architecture with host/cluster decoupling and advanced virtualization. Overall, Shaheen is the first prototype SoC providing support for general-purpose OSs within a $200mW$ power envelope while offering state-of-the-art performance over a wide spectrum of applications, thanks to the programmable multi-core cluster. All the IPs integrated within Shaheen are released as open source[2] under a liberal license to foster future research in the area of AI-IoT computing devices.

REFERENCES

[1] M. O. Ojo, S. Giordano, G. Procissi, and I. N. Seitanidis, "A Review of Low-End, Middle-End, and High-End Iot Devices," *IEEE Access*, vol. 6, pp. 70 528–70 554, 2018.

[2] L. Lamberti *et al.*, "Tiny-PULP-Dronets: Squeezing Neural Networks for Faster and Lighter Inference on Multi-Tasking Autonomous Nano-Drones," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022, pp. 287–290.

[3] E. Cereda *et al.*, "Deep Neural Network Architecture Search for Accurate Visual Pose Estimation aboard Nano-UAVs," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 6065–6071.

[4] R. J. Wood *et al.*, *Progress on "Pico" Air Vehicles*. Cham: Springer International Publishing, 2017, pp. 3–19. [Online]. Available: https://doi.org/10.1007/978-3-319-29363-9_1

[5] STMicroeletronics, "STM32H7," 2020, Accessed 30 July 2023. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html

[6] STMicroeletronics, "STM32F4," 2020, Accessed 30 July 2023. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html

[7] GreenWavesTechnology, "GAP8/9," 2023, Accessed 30 July 2023. [Online]. Available: https://greenwaves-technologies.com/low-power-processor/

[8] A. Di Mauro, M. Scherer, D. Rossi, and L. Benini, "Kraken: A Direct Event/Frame-Based Multi-sensor Fusion SoC for Ultra-Efficient Visual Processing in Nano-UAVs," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–19.

[9] I.-T. Lin *et al.*, "2.5 A 28nm 142mW Motion-Control SoC for Autonomous Mobile Robots," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, 2023, pp. 1–3.

[10] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1106–1119, 2019.

[11] Y. Ju and J. Gu, "A Systolic Neural CPU Processor Combining Deep Learning and General-Purpose Computing With Enhanced Data Locality and End-to-End Performance," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 1, pp. 216–226, 2023.

[12] W. J. Dally, Y. Turakhia, and S. Han, "Domain-Specific Hardware Accelerators," *Commun. ACM*, vol. 63, no. 7, p. 48–57, jun 2020. [Online]. Available: https://doi.org/10.1145/3361682

[13] P. Tsiotras, D. Jung, and E. Bakolas, "Multiresolution hierarchical path-planning for small UAVs using wavelet decompositions," *Journal of Intelligent & Robotic Systems*, vol. 66, pp. 505–522, 2012.

[14] A. Khadka, B. Fick, A. Afshar, M. Tavakoli, and J. Baqersad, "Non-contact vibration monitoring of rotating wind turbines using a semi-autonomous UAV," *Mechanical Systems and Signal Processing*, vol. 138, p. 106446, 2020.

[15] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," *arXiv preprint arXiv:1606.06565*, 2016.

[16] D. Nadalini, M. Rusci, G. Tagliavini, L. Ravaglia, L. Benini, and F. Conti, "PULP-TrainLib: Enabling On-Device Training for RISC-V Multi-core MCUs Through Performance-Driven Autotuning," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Cham: Springer International Publishin, 2022, pp. 200–216.

[17] P. Foehn *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, p. eabl6259, 2022.

[2]https://github.com/pulp-platform

[18] B. Sá, L. Valente, J. Martins, D. Rossi, L. Benini, and S. Pinto, "CVA6 RISC-V Virtualization: Architecture, Microarchitecture, and Design Space Exploration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 11, pp. 1713–1726, 2023.

[19] M. Schneider, A. Dhar, I. Puddu, K. Kostiainen, and S. Čapkun, "Composite Enclaves: Towards Disaggregated Trusted Execution," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 1, p. 630–656, Nov. 2021.

[20] N. Wistoff, M. Schneider, F. K. Gürkaynak, G. Heiser, and L. Benini, "Systematic Prevention of On-Core Timing Channels by Full Temporal Partitioning," *IEEE Transactions on Computers*, 2022.

[21] B. John, "HyperRAM as a low pin-count expansion memory for embedded systems," 2020, Accessed 30 July 2023. [Online]. Available: https://www.infineon.com/

[22] A. Das, P. Kol, C. Lundberg, K. Doelling, H. E. Sevil, and F. Lewis, "A Rapid Situational Awareness Development Framework for Heterogeneous Manned-Unmanned Teams," in *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, 2018, pp. 417–424.

[23] B. Forsberg, D. Palossi, A. Marongiu, and L. Benini, "GPU-Accelerated Real-Time Path Planning and the Predictable Execution Model," *Procedia Computer Science*, vol. 108, pp. 2428–2432, 2017, international Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917308256

[24] S. A. Quintero and J. P. Hespanha, "Vision-based target tracking with a small UAV: Optimization-based control strategies," *Control Engineering Practice*, vol. 32, pp. 28–42, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066114001774

[25] Pixhawk, "PX4," 2023, Accessed 30 July 2023. [Online]. Available: https://pixhawk.org/standards/

[26] M. Idrissi, M. Salami, and F. Annaz, "A Review of Quadrotor Unmanned Aerial Vehicles: Applications, Architectural Design and Control Algorithms," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 2, p. 22, Jan 2022. [Online]. Available: https://doi.org/10.1007/s10846-021-01527-7

[27] C. Budaciu, N. Botezatu, M. Kloetzer, and A. Burlacu, "On the Evaluation of the Crazyflie Modular Quadcopter System," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1189–1195.

[28] O. H. Zekry, T. Attia, A. T. Hafez, and M. M. Ashry, "PID Trajectory Tracking Control of Crazyflie Nanoquadcopter Based on Genetic Algorithm," in *2023 IEEE Aerospace Conference*, 2023, pp. 1–8.

[29] NVIDIA, "NVIDIA Jetson TX2," 2023, Accessed 30 July 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2

[30] Intel, "Intel Atom x7-E3950," 2020, Accessed 30 July 2023. [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/96488/intel-atom-x7e3950-processor-2m-cache-up-to-2-00-ghz/specifications.html

[31] T. Jia *et al.*, "A 12nm Agile-Designed SoC for Swarm-Based Perception with Heterogeneous IP Blocks, a Reconfigurable Memory Hierarchy, and an 800MHz Multi-Plane NoC," in *ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, 2022, pp. 269–272.

[32] C.-H. Lin *et al.*, "7.1 A 3.4-to-13.3TOPS/W 3.6TOPS Dual-Core Deep-Learning Accelerator for Versatile AI Applications in 7nm 5G Smartphone SoC," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 134–136.

[33] C. Schmidt *et al.*, "An Eight-Core 1.44-GHz RISC-V Vector Processor in 16-nm FinFET," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 1, pp. 140–152, 2022.

[34] A. Ottaviano, T. Benz, P. Scheffler, and L. Benini, "Cheshire: A lightweight, linux-capable risc-v host platform for domain-specific accelerator plug-in," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 10, pp. 3777–3781, 2023.

[35] Etron, "256Mb High Bandwidth RPC DRAM," https://etronamerica.com/wp-content/uploads/2019/05/EM6GA16LGDABMACAEA-RPC-DRAM_Rev.-1.0.pdf, 2022.

[36] Bitcraze, "Crazyflie," 2023, Accessed 30 July 2023. [Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1/

[37] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-Swarm: Decentralized Close-Proximity Multirotor Control Using Learned Interactions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3241–3247.

[38] F. Candan, A. Beke, and T. Kumbasar, "Design and Deployment of Fuzzy PID Controllers to the nano quadcopter Crazyflie 2.0," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*, 2018, pp. 1–6.

[39] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici, "SoK: Security and Privacy in the Age of Commercial Drones," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1434–1451.

[40] J.-H. Yoon and A. Raychowdhury, "31.1 A 65nm 8.79TOPS/W 23.82mW Mixed-Signal Oscillator-Based NeuroSLAM Accelerator for Applications in Edge Robotics," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 478–480.

[41] Bitcraze, "AI-Deck," 2023, Accessed 30 July 2023. [Online]. Available: https://www.bitcraze.io/products/ai-deck/

[42] G. Gallego *et al.*, "Event-Based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 01, pp. 154–180, jan 2022.

[43] ROS, "Robot Operating System," 2022, Accessed 30 July 2023. [Online]. Available: https://www.ros.org/

[44] OpenHW, "CVA6," *Github repository*, 2023. [Online]. Available: https://github.com/openhwgroup/cva6

[45] Y. Yarom, "Mastik: A micro-architectural side-channel toolkit," 2016, Accessed 30 July 2023. [Online]. Available: https://cs.adelaide.edu.au/~yval/Mastik/Mastik.pdf

[46] Q. Ge, "Principled elimination of microarchitectural timing channels through operating-system enforced time protection," Ph.D. dissertation, UNSW Sydney, 2019.

[47] Sel4, "Timing channel benchmarking tool," *Github repository*, 2023. [Online]. Available: https://github.com/SEL4PROJ/channel-bench

[48] ARM, "AMBA AXI Protocol Specification," https://developer.arm.com/documentation/ihi0022/j/?lang=en, 2022.

[49] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "DORY: Automatic End-to-End Deployment of Real-World DNNs on Low-Cost IoT MCUs," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1253–1268, 2021.

[50] A. Nadalini *et al.*, "A 3 TOPS/W RISC-V Parallel Cluster for Inference of Fine-Grain Mixed-Precision Quantized Neural Networks," in *2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2023.

[51] OpenHW, "CV32E40P," 2023, Accessed 30 July 2023. [Online]. Available: https://github.com/openhwgroup/cv32e40p

[52] A. Kurth, B. Forsberg, and L. Benini, "HEROv2: Full-Stack Open-Source Research Platform for Heterogeneous Computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, p. 4368–4382, dec 2022. [Online]. Available: https://doi.org/10.1109/TPDS.2022.3189390

[53] L. Valente *et al.*, "HULK-V: a Heterogeneous Ultra-low-power Linux capable RISC-V SoC," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–6.

[54] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[55] Z. Dong *et al.*, "HAWQ: Hessian AWare Quantization of Neural Networks With Mixed-Precision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 293–302.

[56] P. Platform, "TransLib," 2023, Accessed 30 July 2023. [Online]. Available: https://github.com/ahmad-mirsalari/TransLib

[57] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015.

[58] G. M. Amdahl, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: Association for Computing Machinery, 1967, p. 483–485. [Online]. Available: https://doi.org/10.1145/1465482.1465560