Diss. ETH No. 29489

Near-Memory Computing Architectures and Circuits for Ultra-Low Power Near-Sensor Processors

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES

(Dr. sc. ETH Zurich)

presented by

MANUEL CHRISTOPH EGGIMANN

MSc ETH EEIT, ETH Zurich

born on 01.02.1993

accepted on the recommendation of

Prof. Dr. Luca Benini, examiner

Prof. Dr. Riccardo Rovatti, co-examiner

2023

Acknowledgments

Ever since high school, I was fascinated by the seemingly naive question: “How do computers work?”. Never would I have imagined the unfathomable depth of complexity surrounding a complete answer to that question and that the search for an answer would eventually lead me to pursue a PhD. The last five years at IIS have been the most challenging, but at the same time, also the most rewarding journey of my life so far and taught me so much more than just pure knowledge. For all of this, I am incredibly indebted and grateful to the many people who accompanied me on this path in various forms.

First and foremost, I want to thank my PhD advisor, Prof. Luca Benini, not only for offering me the opportunity to become part of this fantastic group but also for providing me with the freedom and the trust to pursue my interests in the broader scope of electrical engineering. Your guidance, boundless experience, and expertise nourished the environment that inspired and pushed me to keep going the extra mile. Secondly, I would like to thank my co-examiner Prof. Riccardo Rovatti for his willingness to dive into my thesis topic and for providing thoughtful and reflective feedback and questions during my doctoral examination.

A huge role in my decision to pursue this path also played my two theses projects during my undergraduate studies. A big thank you thus goes to Michael Schaffner and Lukas Cavigelli whose demonstrated

technical prowess opened my eyes to the amount of skill and knowledge that could be obtained from this career path. I also want to thank Christelle Gloor, my group project partner, who accompanied me during this first hands-on deep dive into the VLSI world. Furthermore, I am very grateful and indebted to my master thesis supervisors Michele Magno and Stephan Mach, who, besides deepening the impression I got on the technical expertise within IIS, showed me the fun and social side of IIS and ultimately opened the opportunity and implanted my desire to pursue a PhD with this amazing group. A huge "thank you" also goes to Abbas Rahimi, who mentored me through the first months of my PhD and laid the scientific fundament on which I started my research.

Of course, the PhD experience is not just the joyful extension of student life by another couple of years and contained many stressful periods that made me question my decision. Without the many colleagues who accompanied me on this "rollercoaster ride" I am not sure I would be writing these words today. First and foremost, I thus want to thank my fellow J68 (in)mates, starting with Philipp for always lightening the mood with Austrian humor and his company during coffee (and other beverage) breaks, Xia for her contagious cheerfulness, and Georg and Vlad for occasionally dragging me out of tunnel vision with crazy videos, talks about car mechanics and the sound of drones flying through our office, followed by all my other coffee break companions like Moritz, Gianna or Robert. The time with all of you was what motivated me to keep going.

Furthermore, I want to express my gratitude to the "permanent residents", the support staff of IIS, including Frank, Beat, and Zerun, whose boundless experience in EDA tooling and digital design saved countless tapeouts from disaster, Hansjörg for his diligent support in whatever PCB assembly or parts ordering problem I was facing and to Christoph for putting up with the many "special requests" on software to be installed on my machine throughout the years.

Finally, I am forever thankful to my parents for always supporting me in my interests and decisions, my brother and friends for preventing me from completely neglecting my work-life balance, and lastly, my soon-to-be wife, Nina. Your unconditional love, patience, and understanding supported me through the most challenging times of this journey, and for this, I am eternally grateful.

Abstract

Artificial intelligence has started to permeate the entire technological fabric of our interconnected world and has long found its way to the network edge. Be it as part of novel biomedical devices that constantly monitor patient health, smart sensors in industrial settings where they drive the transformation from reactive- to pro-active maintenance in industry 4.0 or integrated into the next evolution of human-machine interfaces like extended reality (XR), ML-enacting near-sensor circuits are omnipresent. The tight power and latency constraints of these applications fuel a paradigm shift in the hardware architecture domain, away from conventional von-Neumann-based computing, which is bound by the memory bandwidth bottleneck of the data- and control path. The "new golden age of computer architecture", as the famous computer pioneer David Patterson calls it, is marked by innovative Near- and In-memory architectures around conventional CMOS as well as novel "beyond-CMOS" technologies like PCM or ReRAM. However, many of these techniques are explored with an isolated, device-level-focused view, whereas advances at the system-level demand a holistic multi-objective optimization approach that involves hardware-software co-design and the exploration of new computing paradigms.

This thesis investigates energy-efficient digital hardware architectures at both the circuit- and the system level and develops adequate strategies to enable *energy-proportionality* for general-purpose near-

sensor analytics, i.e. the proportionality of energy consumption to vastly varying dynamic changes in workload compute intensity. We follow a *multi-stage* architectural approach where highly energy-efficient circuits based on the compute framework of Vector-Symbolic Architectures make up the first, always-on stage of our architecture "stack". In the second part of this thesis, we shift focus to the next, more computationally performant stage around heterogeneous compute cluster architectures, with an emphasis on the memory hierarchy. Here we propose a novel architectural design pattern to tightly couple non-volatile memory to the hardware accelerator. Both aspects are demonstrated and evaluated in several silicon realizations in 65 nm bulk, 22 nm FDSOI and 16 nm FinFET technology.

Zusammenfassung

Künstliche Intelligenz hat begonnen, das gesamte technologische Gefüge unserer vernetzten Welt zu durchdringen und hat längst den Weg zum Netzwerkrand gefunden. Sei es als Teil neuartiger biomedizinischer Geräte, die ständig die Gesundheit von Patienten überwachen, intelligente Sensoren in industriellen Umgebungen, wo sie die Transformation von reaktiver zu proaktiver Wartung in der Industrie 4.0 vorantreiben, oder integriert in die nächste Evolution von Mensch-Maschine-Schnittstellen wie erweiterter Realität (XR), ML-gesteuerte Near-Sensor-Schaltkreise sind allgegenwärtig. Die engen Leistungs- und Latenzbeschränkungen dieser Anwendungen treiben einen Paradigmenwechsel im Bereich der Hardware-Architektur voran, weg von der herkömmlichen von-Neumann-basierten Rechenarchitektur, die durch das Speicherbandbreitenlimit des Daten- und Kontrollpfads begrenzt ist. Das neue "goldene Zeitalter" der Computerarchitektur, wie der berühmte Computer-Pionier David Patterson es nennt, ist geprägt von innovativen Near- und In-Memory-Architekturen rund um konventionelle CMOS-Technologie sowie auch neuartigen "beyond-CMOS"-Technologien wie PCM oder ReRAM. Viele dieser Techniken werden jedoch mit einem isolierten, auf Geräteebene fokussierten Ansatz erforscht, während Fortschritte auf Systemebene eine ganzheitliche, mehrzieloptimierte Entwurfsstrategie erfordern, die Hardware-Software-Co-Design und die Erforschung neuer Re-

chenparadigmen umfasst. Diese Arbeit untersucht energieeffiziente digitale Hardware-Architekturen sowohl auf Schaltkreis- als auch auf Systemebene und entwickelt geeignete Strategien, um Energieproportionalität für allgemeine Near-Sensor-Analytik zu ermöglichen, d.h. die Proportionalität des Energieverbrauchs zu stark variierenden dynamischen Änderungen in der Rechenintensität der Workload. Wir folgen einem mehrstufigen architektonischen Ansatz, bei dem hochenergieeffiziente Schaltkreise, die auf dem Rechenframework von Vector-Symbolic Architectures basieren, die erste, immer aktive Stufe unseres Architektur- "Stacks" bilden. Im zweiten Teil dieser Arbeit konzentrieren wir uns dann auf die nächste, rechenleistungsfähigere Stufe rund um heterogene Rechencluster-Architekturen, wobei der Schwerpunkt auf der Speicherhierarchie liegt. Hier schlagen wir ein neuartiges architektonisches Designmuster vor, um nichtflüchtigen Speicher eng an den Hardware-Beschleuniger zu koppeln. Beide Aspekte werden in mehreren Silizium-Realisierungen in 65 nm Bulk-, 22 nm FDSOI- und 16 nm FinFET-Technologie demonstriert und evaluiert. german

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Key Components of a Smart Sensing System 4
1.3 Objective Metrics of Near Sensor System Design . . . 7
1.3.1 Throughput 7
1.3.2 Latency 10
1.3.3 Energy Efficiency and Power Envelope 10
1.3.4 Error Resiliency and Reliability 12
1.3.5 Cost 13
1.4 A Systematic Breakdown of the Ultra-Low Power Design Space 15
1.4.1 On-chip Memory-to-Compute Proximity 15
1.4.2 Memory Technology 18
1.4.3 Physical Data Representation 27
1.4.4 Dataflow/Compute Coordination Paradigms . . 30
1.5 Outline 33
1.6 Contributions 35
1.6.1 List of Publications 36

2 Hardware friendly Algorithms for Smart Sensing Applications **39**
2.1 Introduction 39

2.2 Quantized Neural Networks 41
2.3 Hyper-Dimensional Computing 47
2.3.1 Hollistic Symbol Representation as Hyper Vectors 48
2.3.2 Fundamental Operations of Binary Spatter Codes 53
2.3.3 Compound HDC Data Structures 56
2.3.4 Applications 60
2.4 Summary 64

3 Energy Efficient Circuits for ULP Near-Sensor Processing 65
3.1 Introduction 65
3.2 Methodology 67
3.3 Building Blocks for Energy-Efficient HDC 68
3.3.1 Fully-Synthesizable Associative Memory 68
3.3.2 HDC Encoder Units and Item Materialization . 71
3.4 Hypnos: An Energy Proportional Standard Cell Memory based HDC Accelerator 80
3.4.1 Introduction 80
3.4.2 Related Work 81
3.4.3 Architecture Overview 85
3.4.4 An ISA for HD-Computing 85
3.4.5 Tuning for Maximum Energy Efficiency 94
3.4.6 Applications and Use Cases 97
3.4.7 Energy Efficiency Analysis and Comparison . . 103
3.5 Silicon Realization inside the TSMC 65nm Rosetta Testchip 106
3.5.1 SoC Architecture 106
3.5.2 Physical Implementation 110
3.5.3 Results 111
3.6 A Lightweight Data Preprocessing Sensor Interface for Hypnos 114
3.6.1 Architecture 114
3.6.2 Results 116
3.7 Summary 118

4 Architectural Considerations at the System Level 119
4.1 Introduction 119
4.2 The Vega 10-Core SoC for IoT End-Nodes 122

4.2.1 VEGA SoC Architecture 123
4.2.2 Chip Implementation and Measurements 136
4.2.3 Benchmarking 139
4.2.4 Comparison With SoA 147
4.2.5 Conclusion 148
4.3 A Non-Volatile First-Level Memory Subsystem: The Siracusa SoC 150
4.3.1 Introduction 150
4.3.2 Architecture 152
4.3.3 Results 156
4.4 Comparison with the State of the Art 162
4.5 Summary 164

5 Conclusions **165**
5.1 Main Results 167
5.2 Outlook 171

A Chip Gallery **175**
A.1 Treated in this Thesis 176
A.2 Other ASICs 179

B Notations and Acronyms **183**

Bibliography **189**

Curriculum Vitæ **217**

Chapter 1

Introduction

1.1 Motivation

"What is the energy cost of computation?" This a question for which we have already had an answer for quite a while. In 1961, Rolf Landauer tied the fundamental limits on energy dissipation of non-reversible computing, the basis of most modern signal processing algorithms, to the preservation of entropy i.e. the second law of thermodynamics [1]. While the switching energy of transistors in every new technology node is getting closer and closer to this so-called Landauer limit of conventional computing, system-level computer architecture has comparably remained rather static. More than 70 years after Landauer's findings, energy efficiency at the system level today is thus largely determined by architectural design choices rather than the efficiency of the compute fabric itself and there is still a gap of many orders of magnitudes ahead of us [2]. Yet at the same time, the transformation of society across the globe towards the post-digital age is rapidly accelerating and affects more and more aspects of our environment.

Sensor systems play a crucial role in this cyber-physical landscape. Being at the boundary between the realm of physical entities and abstract information, they make up the majority of electronic devices worldwide. In the past, sensors were part of simple measurement devices that operated in isolation from each other. With the recent advances in MEMS (Micro-Electro-Mechanical-Systems) technology [3], CMOS technology scaling as well as the resulting cost reduction of sensor nodes, they can now be deployed as parts of much larger networks of interconnected sensors, a trend commonly summarized as the Internet of Things (IoT) [4].

Applications for such systems are ubiquitous; wearable or even digestible sensors find tremendous interest in the medical domain, the next evolution in human-machine interfaces like augmented reality requires the seamless interplay from a multitude of camera sensors for visual feedback, and industry is in the midst of a paradigm shift from reactive machine maintenance towards sensor-network-enabled proactive maintenance as part of industry 4.0 [5].

With most of these applications requiring untethered, battery-powered or even energy autarkic operation, we can not solely rely on the stagnating progress in battery and energy harvesting technology, but the devices must drastically improve energy efficiency to keep up with the market expectations. As wired transmission is hardly an option for the massive deployment of low-maintenance sensor nodes, wireless transmission is the only viable option for data aggregation from sensor networks. As a result, the majority of today's wireless sensor device's energy consumption lies in data transmission [6]. Furthermore, applications like augmented reality (AR) have very tight constraints with respect to latency in the order of 2 ms to 20 ms [7]–[10] which is very challenging to achieve over-the-air.

The natural way to cope with these problems is to reduce the amount of data that needs to be sent by moving data processing closer to the sensor, thus tightening the sense-to-feedback loop. Instead of raw data transmission and centralized processing in the cloud, the data is processed directly on these so-called *smart sensor* devices [11], and only the relevant portion of information is transmitted (think of transmission of a single `imminent machine failure` message instead of the raw vibration and temperature data). At the heart of most of these application domains are machine learning (ML)-based

algorithms that gradually replaced domain-specific data processing and interpretation pipelines. The most predominantly used forms of these ML algorithms, i.e. *Deep Neural Network* are characterized by a very regular compute-heavy dataflow. Research and development efforts on Artificial Inteligence (AI) hardware acceleration for server-class devices and mobile devices has been fueled tremendously by the spotlight of public interest in AI technology, e.g. in the form of products like Dall-E and ChatGPT that have unfathomable compute demands for training. This so-called "new golden age of computer architecture" [12] has already progressed enough for mass-deployment of AI hardware in high-performance computing (HPC) and the mobile system on chip (SoC) market [13], [14]. However, solutions for battery-powered sensor nodes in the sub 10 mW range are still an open research topic and impose a departure from traditional von-Neumann compute architecture towards the near- and in-memory computing paradigm.

In this thesis, I analyze the key requirements and characteristics of integrated circuits for near-sensor processing and present a number of novel circuit architectures based on alternative ML paradigms and the concepts of in-memory computing. As part of several taped-out SoCs in modern technology nodes, I demonstrate these building blocks' capabilities to adequately address the challenges of compute-intensity-to-energy proportionality and limited IO and memory bandwidth advancing ultra-low power (ULP) smart sensing one step further on the very long way towards the fundamental limits of compute energy-efficiency.

1.2 Key Components of a Smart Sensing System

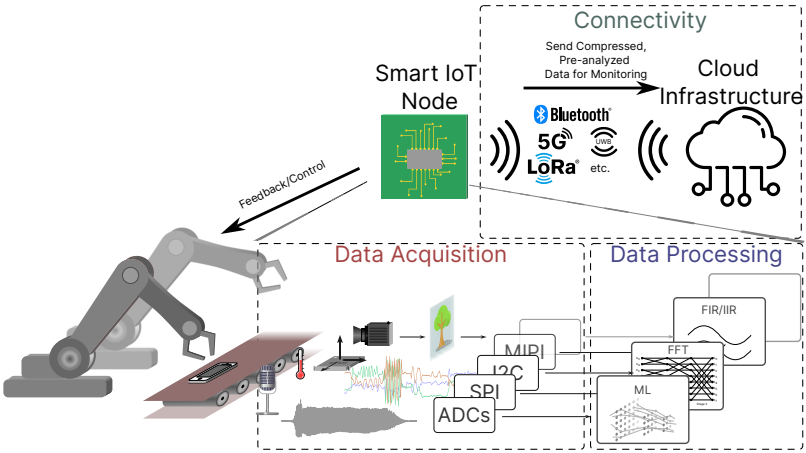


Figure 1.1 – Key components of an interent of things (IoT) Smart Sensing System

There are several key aspects in ULP sensor nodes, each of them significantly affecting overall energy efficiency as well as throughput and latency characteristics of the system. Figure 1.1 illustrates the interplay of these essential building blocks; They can roughly be categorized into:

1. Data Acquisition
2. Data Processing
3. Connectivity

Data Acquisition At the source of the processing pipeline, we have the *Data Acquisition* system, responsible of capturing physical properties of the environment and exposing the data in a processable format to the compute resources. Besides the actual sensor to be used for a particular physical quantity, a major design consideration lies

in the choice of the interface between the sensor and the compute resources. We can differentiate by two key aspects of the sensor interface: Sensor proximity to the compute resources and the mode of data exchange. In most current ULP sensor node architectures, sensor and compute resources are still physically separated devices requiring off-chip communication. The implied inter-chip communication has strong implications on the IO energy and the interface's minimal bandwidth. Even considering modern 3D integration techniques like hybrid bonding, the energy cost per bit of data leaving the monolithic silicon die is still in the order of ~200 fJ [15], two orders of magnitude above the average metal interconnect transmission energy per mm even in older technologies (e.g. ~4.1 $\mathrm{fJ\,mm^{-1}}$ to 38.4 $\mathrm{fJ\,mm^{-1}}$ in 65 nm technology [16]). It is thus to no surprise that the recent industry trend is to transition towards integrated smart sensing systems that combine Data Acquisition and the compute resources for data processing in a single system-in-package (SiP) or even a monolithic silicon die. Regarding data exchange modes, we can taxonomize the design space into four quadrants along the time and value representation domains. While completely analogous sensors operating in the continuous-time, continuous-data regime are still relatively common, the majority of modern sensors exchange data in the discrete-time discrete-data domain. That is, each sensor contains dedicated Analog-to-digital converters (ADCs), custom-tailored to the dynamic range of the analog sensor and exchange data using digital off-chip protocols like Inter-Integrated Circuit (I2C) or Serial Peripheral Interface (SPI). However, the current research focuses on a new category of so-called event-driven sensor interfaces that exchange data in discrete value and continuous time, thus saving power during low activity intervals [17]–[19].

Data Processing The next step in the processing pipeline after data acquisition is data processing. The first generations of sensor networks almost entirely offloaded the data processing to the cloud. However, this compute-offloading approach's energy consumption and latency are prohibitively high for applications with hard real-time constraints like, e.g. augmented reality applications[20]. Thus the current trend is to bring data processing closer to the sensor creating so-called near-sensor computing systems[21]–[23]. In contrast to cloud-

compute infrastructure with virtually unbounded power envelopes, near-sensor systems require orders of magnitude higher energy efficiency to enable reasonable battery lifetime or even fully energy-self-sufficient deployment in the field. An essential requirement for energy-efficient edge computing is hardware-software co-design. The choice of the right processing algorithm for a given application is closely tied to the target compute resource architecture during deployment. The design space of these architectures is very vast; Important dimensions in this design space are the choice of compute units, the memory hierarchy and the computing paradigm, i.e. the interaction between hardware accelerators and conventional software used, e.g. for post-processing.

Connectivity The final design aspect of an ULP sensor system is node connectivity. While near-sensor processing can drastically reduce bandwidth requirements, at some point data still needs to be exchanged with the environment. Here the design choice dimensions are just as vast as the processing architecture's and range from network topology (e.g. centralized start-topology in the form of classical cloud computing versus partially or fully decentralized mesh topologies like mobile edge or fog computing [23], [24]) to the choice of the wireless communication protocol and the design of the RF transceivers.

This thesis mainly focuses on the data processing aspect of near-sensor computing systems. Still, designing such an architecture requires awareness of the design space dimensions across the entire data processing pipeline from the sensor interface to the communication scheme with other network entities as well as an overview of the delicate interdependencies of algorithm development, hardware mappability and scalability.

1.3 Objective Metrics of Near Sensor System Design

Before diving deeper into the different macroscopic design options, we must define the objective and performance metrics. After all, a "narrow-eyed" focus on optimization towards just a single performance metric is most likely bound to fail the test of reality.

In the remainder of this introductory chapter, we provide an overview of the most important technological advances under the light of *five* key objectives:

- Throughput
- Latency
- Energy Efficiency
- Reliability
- Cost

Each of these abstract design objectives can be characterized by different concrete performance metrics. In the following subsections, we quickly introduce these five dimensions of the optimization target space and motivate their relevance in the context of ULP sensor architectures.

1.3.1 Throughput

Compute throughput is a key objective for near-sensor computing systems. In contrast to HPC environments, the objective for edge compute devices is often not "the more, the better" but a question about meeting the minimal requirements of the target applications. After all, higher throughput is almost always bought at the expense of one of the other design objectives outlined in this section.

There are various, at times incomparable, performance metrics commonly used to indicate a system's compute throughput. Table 1.1 outlines some of the most commonly used metrics. They can be roughly categorized into two kinds: *unit-operation-referenced* and *benchmark-referenced* throughput metrics. The first kind of metric measures

	Metric	Definition	Description
Unit Operation-referenced	Op/s	Operations per second	The definition of the unit *Operation* is not always very clear. Often it is used to denote a single arithmetic or logic operation like *add*, *multiply* or *divide*. Comparisons of different designs using this metric are quite difficult because the numbers have to be adjusted for different operand precision and different workload conditions, which are not always reported.
	MAC/s	Multiply-accumulate Ops. per second	Similar to Op/s but defines one operation as the combination of multiplication and subsequent addition as frequently encountered linear algebra (e.g. dot product). Commonly used in the ML domain. One often encounters the rather crude conversion formula 1 MAC/s = 2 Op/s applied in the wild as part of comparisons.
	FLOP/s	Floating-point Ops. per second	This metric is commonly used in the HPC domain, often dominated by floating-point compute heavy loads. However, given the emergence of many new reduced precision floating point formats in recent years (e.g. IEEE 754 half-float, BFLOAT, posit etc.) and the increasing demand for FPU heavy workloads on energy-constrained devices, the metric is increasingly becoming relevant also for edge computing architectures.
Benchmark-referenced	DMIPS	*Dhrystone*/s (1 DMIPS = 1757 it/s)	*Dhrystone* [25] is a synthetic integer-only benchmark that combines a number of, at the time of creation, representative general-purpose workloads with a slight over-emphasis on string operations. With modern compiler techniques and the fairly small program- and problem size, which easily fits into the instruction cache of modern Central Processing Units (CPUs) the benchmark is a rather poor indicator for performance on modern workloads, yet it still remains very frequently used.
	CoreMark	*CoreMark* it/s	*CoreMark* [26] is one of the many open performance benchmarks geared towards embedded CPUs developed by the Embedded Microprocessor Benchmark Consortium (EEMBC). It improves on *Dhrystone* by preventing modern compilers' ability to optimize away major parts of the benchmark. Furthermore, the benchmark standardizes the reporting format, which besides the actual it/s, must include the compiler version/flags as well as the degree of hardware thread parallelization used.
	MLPerf Benchmarks	Achieved latency and Energy/Inference for various ML tasks	MLPerf [27] is a collection of various real-life ML workloads for inference and reccently also for training. The benchmarks are developed and maintained by the MLCommons organization. The inference benchmarks are grouped by categories ranging from data centre workloads to mobile and ultra-low power devices. The *tiny* category for always-on sensor nodes measures inference energy and latency to achieve a given FP32-referenced target accuracy for keyword spotting, visual wake word detection, image classification and anomaly detection. Compared to other performance metrics, the benchmark provides a solid indication of ML performance under real workloads. However, one must be careful when projecting the results to new Deep Neural Network (DNN) workloads that employ different network structures and layer compositions.

Table 1.1 – A selection of commonly used throughput metrics in the ULP-computing domain

the number of reference operations executed per time interval. The choice of reference operation is highly target application dependent. E.g. while DNNs accelerator throughput in the AI domain is usually expressed in "multiply-accumulate operations per second" (MAC/s), a commonly used metric in HPC is "floating point operations per second" (FLOP/s). These kinds of metrics capture only a very small aspect of a system's capability, and relying on these mono-dimensional indicators as a performance predictor for real-world applications can be very dangerous. E.g. Carrington *et al.* expose very poor correlation of various synthetic performance metrics in the HPC domain with predicted runtime errors as high as 68% [28].

The other kind of compute throughput metric expresses system performance in terms of the runtime to complete a given reference benchmark. As indicated in Table 1.1, there is a vast number of these benchmarks, with, at times, very different correlation characteristics to real-life performance. While benchmark-referenced throughput metrics are able to capture more than a single performance indicative system aspect, generalization of the results to any given target application for architecture selection needs to be done with care [28], [29].

A second very substantial issue with common performance metrics is the lack of distinction between peak throughput and sustained performance, a problem especially pronounced in the HPC domain. For example, Kramer shows more than an order of magnitude difference between peak and sustained performance for the TOP500 system [30]. The sustained versus peak performance gap is less severe for von-Neumann based compute resources in smart sensing fabrics due to less pronounced cache hierarchy effects. Still, it is highly relevant when comparing ML workload targeted accelerators due to their vastly different and, e.g for many non-volatile in-memory compute technologies, imbalanced read/write characteristics at the accelerator boundary.

A rigorous restriction to one objective metric for throughput in the context of this thesis is not practical, nor is the assessment of different performance metrics within its scope. In the remainder of this work, we will put emphasis on providing a holistic, system-level view on *sustained* throughput performance under real-life workloads.

1.3.2 Latency

Latency indicates a system's promptness to react to its inputs, i.e. in the context of ULP sensor systems, the time delta between sensor data availability and the completion of the corresponding output calculation. It must not be confused with the almost orthogonal throughput objective. Different application domains for ULP sensor nodes have vastly different requirements for system latency. While some edge computing applications like predictive maintenance [31] can have extremely relaxed latency requirements, many key applications in Healthcare, next-generation Human-Computer Interfaces, and smart mobility have way more stringent demands on maximum latency. E.g. for the application of myoelectric prosthetics control, Farrell and Weir show that maximum control delays above 125ms result in a clinically observable decrease in the subject's prosthetics handling performance [32]. Tracking algorithms in augmented reality environments require even lower latency in the orders of 2-20ms to avoid human perceivable lag or even motion sickness [7]–[10].

Table 1.2 puts those latency requirements into perspective with achievable round-trip latencies with various commonly used wireless protocols for mobile edge computing (MEC)-based sense-control loops; Except for BLE 5[1] and Wi-Fi 6[2], which has very high power consumption, none of those protocols comes close to achieving sub 10 ms of round-trip latency required for the most latency demanding IoT applications. This comparison further motivates the need for autonomous near-sensor computing from a latency point of view and demonstrates the importance of latency besides the usual throughput and energy-efficiency-oriented comparison of competing system architectures.

1.3.3 Energy Efficiency and Power Envelope

Naturally, energy efficiency is of utmost importance for any energy-constrained device and has been in the research focus within the circuit design community for more than two decades[33]. Yet there are countless different metrics to measure energy efficiency;

[1] Bluetooth Low Energy Version 5
[2] IEEE 802.11ax standard

Protocol	Latency	Bandwidth	Range
BLE 5	15 ms – 100 ms	0.5 MBit/s – 2 MBit/s	50 m – 1 km
Wi-Fi 6	5 ms – 20 ms	0.5 GBit/s – 1 GBit/s	10 m – 100 m
LoRa	80 ms – 2 s	0.3 KBit/s – 50 KBit/s	5 km – 15 km
5G mmWave	5 ms – 20 ms	1 GBit/s – 1.5 GBit/s	500 m – 1 km
eMTC	50 ms – 100 ms	0.5 MBit/s – 1 MBit/s	5 km – 10 km
NB-IoT	1.6 s – 10 s	50 KBit/s – 200 KBit/s	5 km – 15 km

Table 1.2 – Typical round-trip latencies, maximum bandwidth and range of common wireless protocols for IoT devices.

One metric often compared in literature within the neural network accelerator domain is the *Energy per Operation* or its inverse, the Number of operations per unit energy, often expressed in unit "tera operations/s/W", (TOP/s/W or TOPS/W). This metric is often misleading due to incompatible definitions of the reference operation or different scopes for energy measurement; I.e. we frequently observe energy efficiency reports in the literature with either arbitrarily limited, if at all reported scope of the power measurement or unprecise formulation of the unit operation by which the energy is normalized to the performance [34]. When comparing competing architectural design choices with figures of merit reported for different technology nodes or at different design stages, we must be careful not to mix up architectural energy efficiency gains with technology advantage or the bias resulting from inaccurate energy modelling at the various design stages. I.e. post-synthesis power figures usually are significantly lower than at the post-layout due to the lack of interconnect parasitics and glitching activity modelling.

For a fair comparison, the total energy cost at the system level is essential since different circuit styles have different support circuit requirements. E.g. while the energy efficiency of some Analog mixed signal (AMS) circuits might look very favourable at first, the power consumption of possibly required voltage biases, ADCs or temperature compensated clock sources might impact the real word energy efficiency quite significantly.

Another problematic aspect of the *Energy per Operation* metric in real-world applications can be the lack of significance to predict the average system power envelope, the metric that ultimately determines

the runtime of battery-operated systems or the overall feasibility in self-sustainable energy harvesting systems. Especially in the context of smart sensing systems, workloads are usually very far from fully utilizing the available compute resources all the time. Compute-intensive execution periods with some form of hard-realtime constraints regarding maximum latency take turns with periods of low compute demand. If we model such a sensor system with a power state machine that transitions between a *Full Performance* state an *Active Idle State* and potentially various *Low Power States*, we not only require the power under full utilization but also,

a) Knowledge of all Power States of the System including *power consumption* and *state transition times*.

b) the probability distribution of long enough "idle-time" intervals, that is, time intervals of little to no compute requirements that are long enough to outweigh the energy cost of power state transitions

in order to assess the average system power consumption [35].

A fair comparison of competing design alternatives thus requires full knowledge of all power states and must not be limited to the energy efficiency at their respective peak performance points alone.

1.3.4 Error Resiliency and Reliability

Another crucial system objective is error resiliency and reliability. The requirements on this particular aspect of an ULP sensor node architecture are extremely diverse and range from quasi-non-existent (e.g. in smart dust applications [36], [37]) to extremely high for mission-critical deployment in the health, automotive and aerospace sector.

The objective of reliability can be captured in the form of e.g. *bit-error rate (BER)*, *mean time between failures (MTBF)* or *availability* in the case of spurious system failures, *mean time to failure (MTTF)* or its inverse the *failure rate* for capturing the long-term reliability of a given system or *operating temperature range* and *thermal budget* for temperature dependent reliability and manufacturability.

While these metrics are mostly influenced by the physical aspect of an ULP sensor architecture, the choice of algorithm for data processing can considerably impact overall error resiliency. As we will see in

chapter 2, certain data processing algorithms like e.g. DNNs or hyper-dimensional computing (HDC) have inherent resiliency with respect to noise in input- or intermediate data. Careful hardware software codesign can thus be essential to counterbalance reliability detrimental characteristics of a technology choice [38], [39].

In the end, reliability requirements are one of the most crucial objectives for a lower power system to meet; After all, many industry sectors know hard constraints on the minimum figure of merit regarding system reliability (e.g. AEC-Q100 for the automotive sector), which can render entire areas within the hardware-software design space infeasible.

1.3.5 Cost

Finally, we have the cost objective, which once again comes with a plethora of different metrics that ultimately influence the total procurement and the operating costs of the product. The following (non-exhaustive) list indicates some of the most important influencing factors on recurring and non-recurring procurement as well as the operating costs of a very large-scale integration (VLSI) product:

Technology Cost Impacts This includes the recurring and non-recurring costs of the technology node itself in the case of conventional CMOS technology as well as the CMOS compatibility or SiP packaging requirements for any non-conventional integrated circuit manufacturing technology. Also, technology maturity needs to be taken into consideration since yield and time to market can have impacts at a business strategic scale, as companies like, e.g. Intel demonstrate time and again [40].

Design & Verification Efforts and EDA Support Labor costs are a very significant portion of the overall non-recurring engineering cost (NRE) costs. Design and verification as well as Electronic Design Automation (EDA) software licensing cost make up more than half of the overall NRE investment and experienced an increase of 4x moving from the 22nm to the 7nm node [41]–[43]. The engineering cost factor is however not only

dictated by the technology but also substantially determined by the choice of circuit style (e.g. digital CMOS or AMS).

External Support Circuitry While seemingly not as significant as the engineering or recurring and non-recurring cost of the silicon manufacturing process, the requirement of additional off-chip circuitry can still be quite relevant for assembly and packaging considerations. Although the trend clearly goes towards SiP integration [44] to reduce the bill of material (BOM) count of printed circuit boards (PCBs) to the bare minimum, off-chip components like temperature-compensated oscillators and inductors for voltage regulation remain hard to integrate.

Cost of External Infrastructure and Maintenance Looking at the overall cost breakdown up to the system deployment level, differences in terms of maintenance and infrastructure requirements become major cost factors of the operating costs of a particular ULP sensing solution and, e.g. in the context of IoT sensor network deployment can reach as high as 80% of the overall system deployment cost[45]. The need for higher bandwidth wireless communication infrastructure, cloud compute infrastructure or edge servers (in the context of MEC [46]) and the difference between fully self-sufficient systems and systems requiring regular battery replacement are all essential for integrative analysis of a given solutions operating costs.

1.4 A Systematic Breakdown of the Ultra-Low Power Design Space

After motivating and introducing the different objective metrics of a ULP near sensor processing system, we use this section to outline some of the macroscopic design space dimensions of ULP system. Furthermore, we are going to provide some overview of the current state of the art in the respective design dimensions. The aim is to establish the framework and current state of research at a larger scope to assess better the further contributions of the thesis outlined in section 1.5.

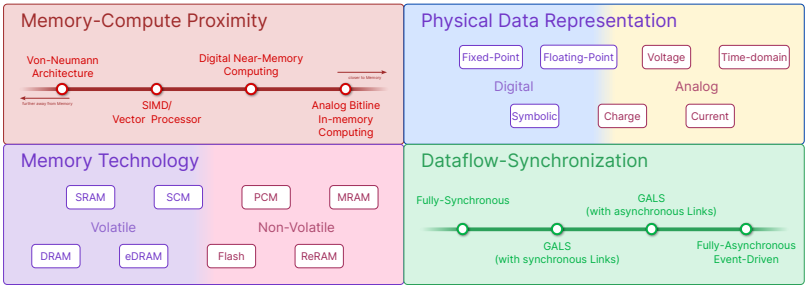


Figure 1.2 – Illustration of macroscopic ULP system architecture design space.

Given the vastness of the design space, we try to project it along the following four major dimensions illustrated in figure 1.2:

- On-chip Memory-To-Compute Proximity
- Memory Technology
- Physical Data Representation
- Dataflow Coordination Disciplines

1.4.1 On-chip Memory-to-Compute Proximity

The increasing gap between memory and traditional CPU core performance has severely hindered progress in addressing the challenges

of the big-data compute era[47]. This so-called *memory wall* or *von-Neumann bottleneck* is innate to the intrinsic separation of compute logic and memory in general-purpose cores that results in bandwidth limitations and increased energy consumption for the data transfers from and to the processing units [48], [49]. Whereas traditionally, the terms were used to symbolize the challenges of matching of-chip memory bandwidth with CPUs, the problem is deeper-rooted and fundamental to any form of data processing that separates information storage and processing [50]. With the increasing difficulty to gain grounds in terms of throughput and energy efficiency by means of technological advantage, one way to address this challenge is to depart from classical von-Neumann compute architectures and move computation closer to the memory or actually within the memory structures themselves, an architectural paradigm commonly denoted as *in-memory* computing. Yet, the term *in-memory* loosely denotes various degrees of data-compute proximity.

A very subtle form of increasing data-compute proximity is to have conventional instruction driven compute units operate on more than one data at a time, a class of computer architectures commonly labelled as single instruction multiple data (SIMD) [51]. I.e. the compute core has more internal, i.e. closer memory to operate on and processes several data items at a time. Vector processors are one subcategory of SIMD architectures that have recently experienced a reemergence of interest not only in the HPC domain for neural network training but also for more energy-constrained applications. For example ARM introduced the latest Armv9 instruction-set architecture (ISA) together with the second generation of their Scalable Vector Extension (SVE)[52] which finds application in e.g. their Neoverse N2 platform targeting edge learning applications but is also powering the currently second-most powerful supercomputer fugaku[53]–[55]. Also, the open-source RISC-V ISA is currently in the process of ratifying the new "V-extension", an ISA extension for vector processing which has already found adoption in open-source hardware research efforts like *ARA* and *SPATZ*[56]–[59].

An even closer interweaving of memory and compute fabric is achieved in so-called *systolic-arrays*, a computing architecture where data flows organically through an array of compute units. The units are spatially organized in a regular pattern and process data much like

a two- or multi-dimensional pipeline architecture; each processing unit contains internal memory and accepts inputs from its direct neighbours, updating its internal state and forwarding the results or input data to its neighbours. Systolic arrays can be extremely performant, as becomes evident when looking at commercial incarnations for ML workloads like e.g. Google's Tensor Processing Unit (TPU) [60], [61]. E.g. Wang *et al.* found that the Google TPU platform achieves throughput increases between 3×and 6.8×depending on the benchmark network when compared to the NVIDIA T100 GPU platform, an example of a vector processor architecture. Also, the platform achieves a 2.2×improvement in FPU utilization on average, a loose indicator of energy efficiency[62]. However, Systolic Arrays require an extremely regular data flow and not all algorithms can be mapped efficiently to a particular systolic array architecture.

While vector processors aim to increase the memory-to-compute proximity, their approach of interweaving small storage elements with compute logic is quite detrimental to area efficiency and cost. A completely different approach is to enhance a memory circuit with computing capabilities. Thus one form of in-memory computing that is currently in research focus is computational memories that execute GEMM kernels and other multiply-accumulate dominated operations directly on the bit lines. Figure 1.3 illustrates the concept; a classical volatile memory structure like static random access memory (SRAM) or dynamic random access memory (DRAM) is complemented with enhanced read-out and word-line activation circuitry to activate several memory rows at once and perform analog accumulation by means of Kirchhoff's current law [63] or in a bit serial manner[3]. In this particular case, the SRAM macro is able to compute a vector matrix multiplication in a single read operation in the analog domain. However, sticking to traditional 6T-SRAM bit cell architecture (as illustrated in figure 1.4) to maintain area efficiency comes with several key-challenges:

a) Even more so than in the regular memory-only mode of SRAM operation, dealing with the read-upset issue, i.e. the accidental overwrite of the bit-cell content during the read-process, becomes a major concern that mostly demands for

[3]See section 1.4.3 for a more detailed summary on the different realizations of computational memories.

the use of more complex bit-cell architectures.

b) Although the vector x can be represented with multi-bit precision, e.g. using pulse-width or pulse-count modulation, the matrix weights are essentially still single-bit. This can be dealt with by combining multiple bit-cells with logarithmically scaled pulse widths or by combining analog accumulation with sequential digital accumulation in a bit-serial manner.

An alternative approach to SRAM-based in-memory computing is thus to use novel non-volatile memory technology (as introduced in section 1.4.2) that is able to represent multiple bits in a single storage cell. However, these technologies come with their own share of problematic aspects in the form of noise and device reliability concerns.

1.4.2 Memory Technology

Not entirely orthogonal to the closeness of compute fabric and memory is the consideration of an architecture's memory technology. Even summarizing the entire technology stack around storage technology is completely out of the scope of this overview section. We thus limit ourselves to on-chip memory technologies only. Yet still, there are a plethora of different alternatives, each with its own strengths and weaknesses;

The memory technology landscape can be taxonomized into a number of different categories. However, the distinction between non-volatile and volatile memory technologies has the strongest implications for edge-compute node systems. Non-volatile memories (NVMs) have the fundamental, eponymous advantage of not requiring constant energy to "remember" their stored data. A characteristic that is of utmost importance for duty cycling and idle power considerations.

Volatile On-Chip Memory

SRAM The most common density-oriented on-chip memory technology belonging to the volatile category is SRAM. SRAM memories come in many different shapes and forms. Still, most SRAM macros used today are based on 6-T bit cells organized in a bit-cell array with horizontally shared word lines and vertically

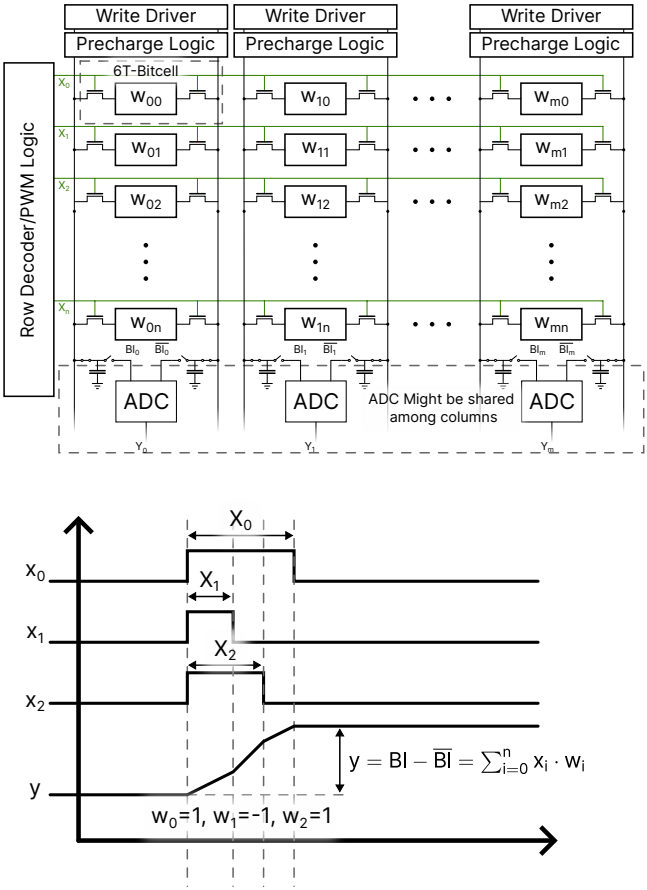


Figure 1.3 – A computational memory based on SRAM technology with analog accumulation for vector-matrix accumulation with bi-polar weights and multi-bit activations. In practice, the large area overhead of the ADC might demand a shared ADC for multiple bit lines with time-multiplexed read-out.

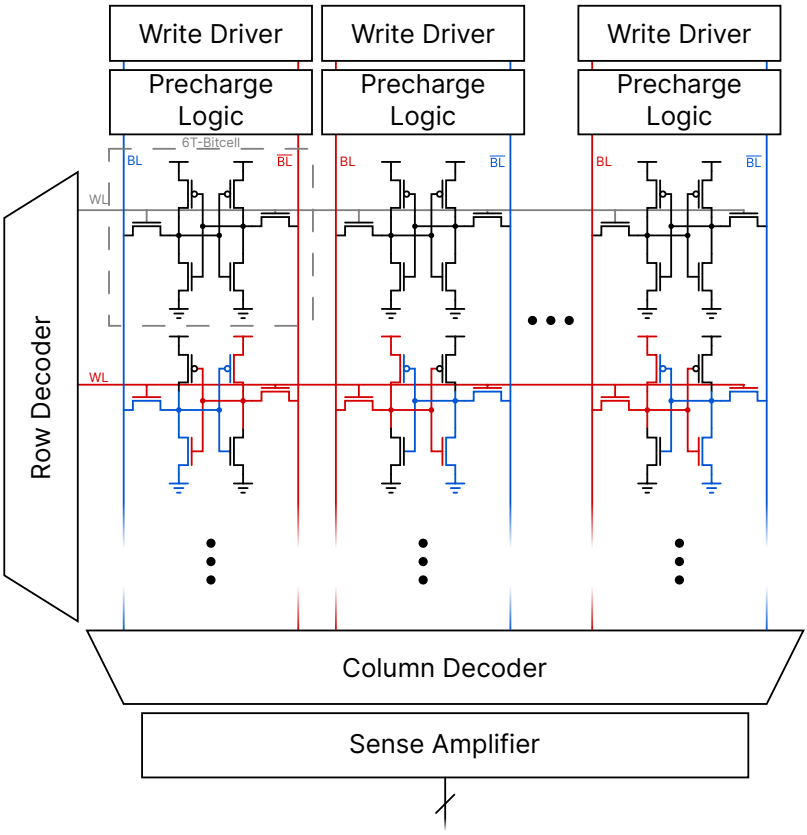


Figure 1.4 – General structure of a 6-T SRAM array.

shared differential bit lines, as indicated in figure 1.4. A row decoder is responsible for selecting the word of interest within the memory, enabling access transistors to propagate the voltage differential stored in the 4T feedback look onto the bit lines. Finally, a sense amplifier amplifies this voltage differential and measures its polarity to determine the stored binary value.

SRAM are readily available in almost any CMOS technology node in the form of memory macro generators that generate memories of the desired word width and word count. SRAM is highly optimized towards memory density, usually even benefiting from pushed foundry geometry rules due to their extremely regular structure.

eDRAM Another volatile on-chip memory technology that pushes even further with respect to the area efficiency metric is embedded dynamic random access memory (eDRAM). eDRAM applies traditional DRAM working principles to reduce the bit cell area by storing the value in a capacitor as opposed to a 4-T feedback loop. Yet, in contrast to conventional DRAM, the much smaller capacitor in eDRAM is manufacturable in regular digital CMOS technology. The most significant challenge of eDRAM is the need to counteract the loss of information due to the leakage-induced discharge of the bit cell capacitor. This results in the need for regular refresh cycles whose impact on energy efficiency and availability increases exponentially with respect to temperature due to an increase in leakage current [64], [65].

SCM The final volatile on-chip memory style we want to highlight are so-called standard cell memories (SCMs). As the name implies, they consist of sequential cells from normal standard cell libraries and can be synthesized in regular digital design flow tightly interwoven with the digital compute logic. SCMs excel in flexibility by supporting very unbalanced memory aspect ratios and maintain their area density even if scaled to very granular small-sized memory macros, a characteristic SRAMs struggle a lot more with[66]. Besides their inherent two-port access characteristic, which is quite beneficial for many accelerator architectures, SCM allow for more aggressive voltage scaling than

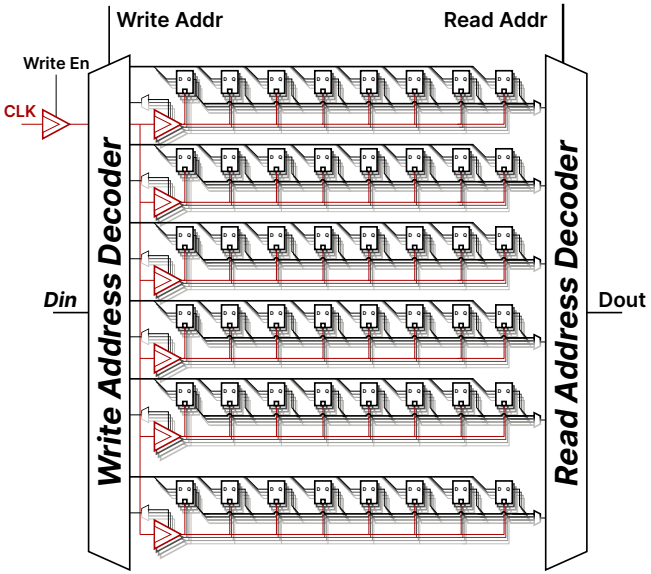


Figure 1.5 – Latch-based Standard Cell Architecture with clock-gate-based write access circuitry.

traditional SRAM macros [66]. I.e. Andersson *et al.* have shown that below capacities of 16 KiB SCM are highly competitive even in area density compared to specialized sub-V_{th} designs due to their fundamental area penalty for the sense-amplifier[67]. Techniques to achieve this competitive memory density involve the usage of latch-based memory architectures with clock gate controlled write-access logic, as illustrated in figure 1.5, and controlled regular cell placement techniques during the physical design flow as outlined by Teman *et al.* [68].

Non-Volatile On-Chip Memory

Even broader than the volatile memory landscape are the choices for non-volatile on-chip memory. Figure 1.6 illustrates a selection of the most important technologies for near- and in-memory computing applications. In this subsection, we will briefly introduce their working principles and their respective strengths and weaknesses.

Flash One of the most common non-volatile memory technology is flash. The traditional working principle of flash memory is modulating the threshold voltage in a floating-gate metal-oxide-semiconductor field-effect transistor (MOSFET) by injecting or removing electrons on an electrically isolated gate sandwiched between the control gate and the channel. The charge is deposited or removed via hot-electron injection, Fowler-Nordheim tunnelling or hot-hole injection. Traditionally, the floating gate was manufactured using conductive polysilicon material. However, more recent flash memories replaced the conductive floating gate with a thin layer of electrically insulating material like nitride. This so-called charge-trap flash bit cell architecture displays superior write-erase cycle endurance and allows for very area-efficient vertical integration in so-called 3D-V-NAND memories. During read-operation, the difference in trapped charge-dependent threshold voltage is measured and converted to a single (single-level cell (SLC)) or even multiple bits (multi-level cell (MLC))[69]. Due to the insulation of the floating gate/nitride layer, the electrical charge stays trapped for many years, which makes flash a non-volatile memory technology.

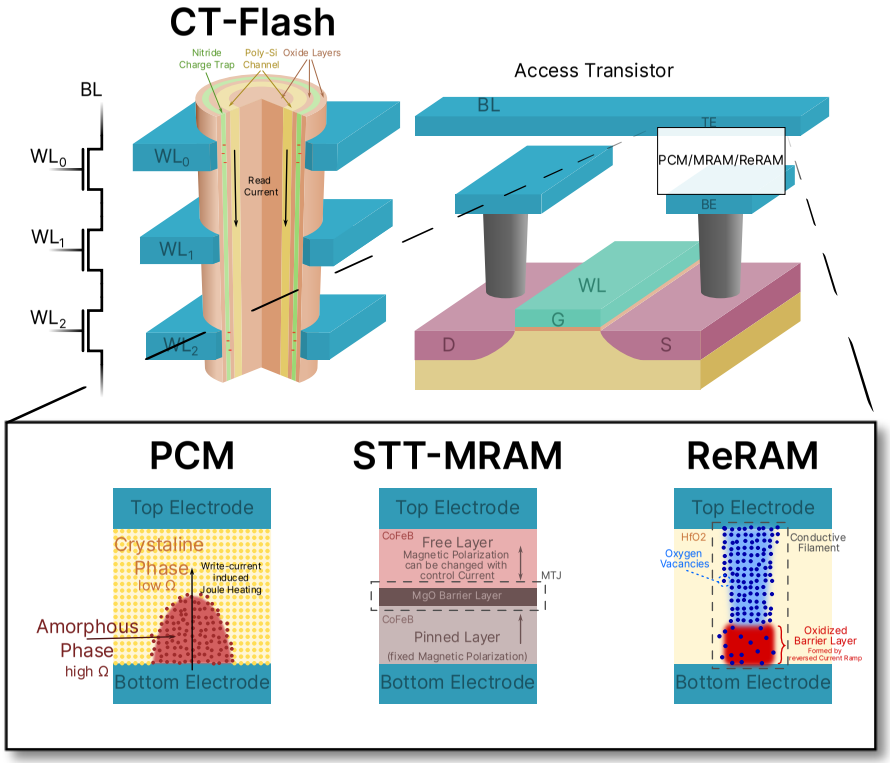


Figure 1.6 – An overview of various different NVM bit cells.

Flash memory technology is very mature and widely used in industry in two major flavours; NAND-flash is used for high-capacity/density applications due to its amenability for cost-effective 3D integration [70]. It features high sequential read/write bandwidth and currently is the most widely adopted technology for high-capacity memories like solid-state drives (SSDs)[71], [72]. NOR-flash, on the other hand, requires larger bit cells but exhibits, in comparison to NAND-flash, much higher random access read throughput, which makes it the predominant technology for embedded NVMs in microcontroller units (MCUs) for direct code execution of device firmware [73]. Besides the usage for data storage only, there are also numerous research projects to use flash memory as a computational memory [63], [74]–[78]. E.g. Guo *et al.* propose an embedded NOR-flash based neuromorphic classification circuit and Bavandpour *et al.* even leverage commercial 3D-NAND flash in their 3D-aCortex architecture for highly integrated inference acceleration. However, especially embedded NOR-flash faces numerous challenges in keeping up with the pace of CMOS technology scaling. At nodes below 28nm, flash memory becomes extremely costly to fabricate [79], and novel, non-charge-deposit-based technologies seem to be the future of embedded NVM[71], [80].

MRAM As an emerging alternative to flash memory for embedded NVM at advanced process nodes, spin-transfer-torque magnetoresistive RAM (STT-MRAM) and recently spin-orbit-torque magnetoresistive RAM (SOT-MRAM) promise superior geometry-size and improved write speed that rivals SRAM. The basis of magnetoresistive random access memory (MRAM) is to store information as magnetic polarization in a so-called magnetic tunnel junction (mtj), a device consisting of two ferromagnetic layers separated by an electrical insulator. During write operations, the magnetic polarization of one of the layers is altered by driving either a positive or negative write current through the layers. For read operations, the tunnel magnetoresistance (TMR) effect is leveraged which causes a difference in read current depending on the polar or anti-polar magnetization of the two layers [81], [82]. MRAM are used

widely for embedded memories at advanced process nodes and are sometimes even used as a direct non-volatile alternative to on-chip SRAM[83]–[85]. Like flash, MRAM-based computational memories are explored in various technology nodes and, due to their superior scalability, would allow interleaving of NVM and computation circuitry on the same die even below the 22nm node[86]. But in contrast to flash, STT-MRAM traditionally only allows for one bit per storage cell, which makes this technology less suitable for multi-bit computations in the analog domain. Furthermore, MRAM faces severe retention issues at elevated temperatures, which makes the adoption in mission-critical devices, e.g. in the automotive industry or for reflow soldering of pre-programmed devices, a challenge[71], [82] albeit there have been promising technological advances on that aspect with e.g. Gallagher *et al.* presenting an STT-MRAM at a 22nm technology node capable of 10+ years of data retention at 150 °C[87].

PCM Even more temperature sensitive is another type of emerging non-volatile memory called phase-change memory (PCM). Its working principle actually relies on very localized temperature changes to modify the resistivity of chalcogenide glass by switching between its crystalline and amorphous state [82], [88], [89]. In contrast to MRAM, PCM natively supports multi-bit storage using iterative programming and with projected cell topologies as introduced by Koelmans *et al.* to counteract strong drift, PCM has become a prime candidate for NVM-based neuromorphic computing[91]. However, besides the aforementioned averse temperature characteristics, the high programming current needed for resetting and the vastly asymmetric SET-RESET current-to-resistance characteristic makes it challenging to leverage PCM for online learning applications[63]. PCM has been commercialized more than a decade ago for embedded NVM applications and lately found adoption in the mass storage market in a 3D-integrated variant from Intel and Micron called 3D XPoint available under the brand Optane Memory [92]. However, with Intel only recently shutting down its Optane product line, the future of the technology, at least in the mass-storage market, is uncertain [93].

ReRAM Another memristor based memory technology is resistive random-access memory (ReRAM) sometimes also abbreviated with RRAM. In this technology, information is once again stored in the form of electrical resistance of a two-terminal device. By applying electrical pulses to the originally insulating resistive layer, low-resistance paths, so-called conductive filaments (CFs), are formed. The exact mechanism of CF formation and charge transport depends a lot on the particular material used as the insulating layer. One of the most popular and widely used materials for ReRAM based memories is hafnia (HfO_2), a high-k dielectric that already finds common use as a gate dielectric in the CMOS front-end-of-line (FEOL). HfO_2-based ReRAM is highly scalable down to the minimum metal feature size and very well compatible with existing CMOS back-end-of-line (BEOL) [88], [94]. Like PCM, the adoption of ReRAM for storage class memory at this time is wavering due to the largely improved scalability of charge-based 3D flash NAND memory [88]. Yet due to the excellent CMOS-compatibility and its high resolution of programmable resistance values, numerous ReRAM-based computational crossbars can be found in literature[95]–[99]. Although ReRAM requires way smaller write currents and is less susceptible to drift than PCM, ReRAM devices suffer from large device-to-device variability and random-telegraph noise [100], [101] which requires noise-aware training and network modelling strategies [102].

1.4.3 Physical Data Representation

The key characteristic of the digital era is the representation of information in the digital domain. This style of data representation for storage and computation maps information to a discrete value range and is predominantly implemented using binary logic. Today's implementation landscape of digital logic realization is less convoluted than it used to be a couple of years ago; the predominant logic style used in modern Integrated Circuits (ICs) is once again static-CMOS after a transient shift towards dynamic-CMOS variants like domino logic or differential cascode voltage switch logic (DVCSL) [103]. Nevertheless, there is a growing interest in non-digital compute styles. This interest is

mostly driven by the high demand for extremely-regular, linear algebra-dominated computing in matrix-matrix multiplication dominated AI training and inference applications since they map well to crossbar-style analog compute fabrics. The idea of performing computation in the analog domain is not new at all, and the first electrical analog computers date back to the 1930s [104]. However, its challenges in the form of non-linearity and scalability issues due to signal-to-noise ratio (SNR) decrease remain. The basis for most modern forms of analog-computing crossbars is to perform summation in the analog domain using either Kirchhoff's current law or capacitive charge accumulation and/or to use Ohm's voltage law for multiplication.

For the context of analog, NVM-based AI acceleration Xiao *et al.* provides a very comprehensive overview [63]; The weights are mostly encoded in the memory as analog resistance values, a digital multi-bit value or mixture of the two. The activations, on the other hand, can be encoded as voltage amplitude in the time domain using pulse-width encoding, using pulse-count modulation or bit-sequentially in the digital domain. Multiplication of activations with the weights happens implicitly by applying the encoded activations to the crossbar as indicated in figure 1.7. The resulting sum of currents on the crossbar's column lines is then converted to a voltage by means of either a trans-impedance amplifier or a capacitor acting as a charge integrator before it is converted back into the digital domain using an ADC. In the case of bit-serial activation encodings, the required sequential shift and accumulation can be implemented digitally using shift registers and digital accumulators or in the analog domain using switched capacitance network to perform divide-by-two operations by means of charge sharing between powers-of-two-sized capacitors. Although the computation in the analog domain can be very efficient, the conversion between the digital and analog domains is very costly. In fact, the overall energy consumption in analog in-memory compute crossbars is dominated by the ADCs [63], [105]. The choice of ADC is thus crucial for the overall performance in the throughput, latency and energy efficiency objective. Flash-ADCs have the benefit that they are fast enough to allow time-multiplexed operation, sharing one flash-ADC among multiple column lines, but they become prohibitively expensive for higher bit widths which limit the scalability of the crossbar's vertical dimension due to the limited output activation resolution. Delta-sigma

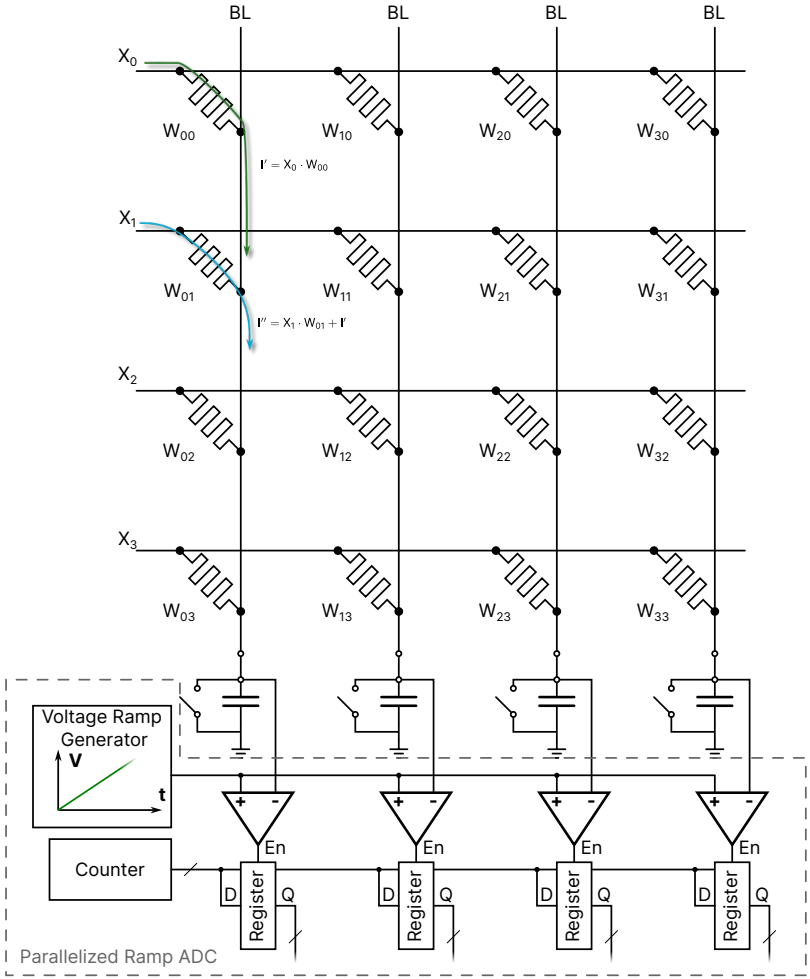


Figure 1.7 – Example of a memristive crossbar architecture for vector-matrix multiplication with switched-capacitance charge integrator and parallel ramp-ADCs.

ADCs, on the other hand, provide very high resolution at the cost of speed, which limits their usage in time-multiplexed architectures, while the slightly faster successive-approximation-register (SAR)-ADCs are fairly large. A popular pattern for crossbars to balance throughput and energy/area cost is thus to use ramp ADCs, which is also illustrated in figure 1.7. A single ramp generator provides a voltage ramp to one comparator per crossbar column. A digital counter per column then counts the number of cycles it takes for the voltage ramp to exceed the latched column voltages, thus digitizing it. This approach scales linearly with the number of columns, but exponentially with the required bit-resolution, i.e. for many rows, the approach can become rather slow.

1.4.4 Dataflow/Compute Coordination Paradigms

The final design space dimension we want to highlight is the various paradigms on coordinating dataflow and computing within a system. This aspect is sometimes denoted as the systems' clocking discipline[106]. The most predominantly used form of compute coordination is clock synchronous edge-triggered clocking. In such a system, one or several central clock sources coordinate the individual processing steps and data exchange in discrete time units, the clock period. Computation and data exchange is triggered only by the clock signal's rising- and/or falling edge. The resulting combinational propagation of signal changes is constrained to settle within one clock period not to cause any setup violations. This processing scheme is very well supported by modern EDA tools. Given a set of reasonable timing constraints, so-called static timing analyzers built into the various frontend and backend tools can automatically synthesize, place and route a design to comply with the constraints. However, clock synchronous design also has its demerits:

- The clock period of the design always has to adapt to the most critical, that is, the longest/slowest combinational path within the system. This limits the performance of less critical parts of the system that theoretically could operate at a higher frequency. This aspect impacts the throughput and energy efficiency because, for a given operating voltage, a circuit always operates most

energy efficiently at its maximum frequency.

- Since the clock signal has to reach all sequential elements within the circuit with as little as possible skew, the clock distribution is usually handled by a high-fanout network, the so-called clock-tree [4]. The clock tree, usually synthesized by a dedicated clock tree synthesizer, aims to balance the arrival time of the rising edges between its leave nodes, the sequential cells, by inserting buffers in a tree-like fashion. However, this balancing comes at a significant cost; since the clock signal has to toggle twice in every period regardless of the circuit's current activity, it consumes a very large amount of power. Although there are techniques to reduce the clock-tree activity like fine-grained and coarse-grained clock gating, it still is not uncommon for the clock tree to be responsible for more than 40% of the overall power consumption [107], [108].

For larger, scaled-out SoCs where the on-chip wire delays start to exceed the clock period, the first point can become the limiting factor for performance. Another point on our compute coordination paradigm axis is thus to compute clock-synchronously at the local scale but exchange data asynchronously between larger subblocks of the system. In this approach, also known as globally-asynchronous locally-synchronous (GALS), each subblock forms its own *clock-domain* with clock synchronous intra-block data exchange. At the macroscopic level however, data needs to be exchanged using clock-domain-crossing safe strategies like handshaking or dual-clock FIFOs which incurs overhead in terms of latency, area and energy [109], [110].

If we move even further away from fully synchronous designs, we get to the realm of asynchronous event-driven computing. In purely asynchronous systems, the circuit is designed to be robust under certain delay assumptions, like the commonly used quasi-delay insensitivity (QDI). This is achieved by introducing various forms of handshaking between isochronous circuit islands in the QDI system and the encoding of data validity within the data itself (e.g. by using differential signal

[4]There are alternative clock distribution schemes like the clock-mesh strategy. They share the same issues regarding high power consumption and difficulties in distributing the clock across larger spatial distances.

lines, invalid data is detected on the equality of both lines). The selling point of asynchronous computing is its built-in adaptability to process variations, its characteristic that the throughput naturally adapts to the maximum circuit speed achievable for a given subblock and that it consumes no dynamic power in the absence of data to process, hence the name event-driven computing. However, enforcing proper sequencing of the handshaking protocols involves heavy usage of non-standard circuits like muller-c elements [111] and sometimes even resorting to the use of dynamic logic in asynchronous pipeline primitives like the commonly used precharged half-buffer (PCHB). Providing an introduction on asynchronous computing principles is outside the scope of this thesis, and we refer the reader to the works from Martin and Nystrom for a good introduction to the topic [112], [113]. After all, asynchronous computing is not at all a new computing paradigm. The first computer based on asynchronous circuitry (ILLIAC) dates back more than 70 years, and complete asynchronous processors like the MiniMIPS [114] have been around for quite a while. But so far, the reasons mentioned above, coupled with very limited EDA tool support, have pushed asynchronous computing to lead a niche existence. However, recently asynchronous computing has been rediscovered for GALS-networks-on-chp (NoCs) [115], [116] and neuromorphic computing systems like Intels Loihi and Loihi2 [117], [118] platform or IBMs TrueNorth neuromorphic chip [119] developed as part of DARPAs SyNAPSE program [120].

1.5 Outline

As should have become clear with the introduction sections so far, the design space for energy-efficient systems is extremely vast and impossible to cover in its entirety with due diligence. For the remainder of this thesis, we thus disregard two out of the four defined design space dimensions and limit ourselves to research on energy-efficient *digital circuits* with mostly *clock synchronous* dataflow. Figure 1.8 illustrates the thesis outline; it is structured into three main parts:

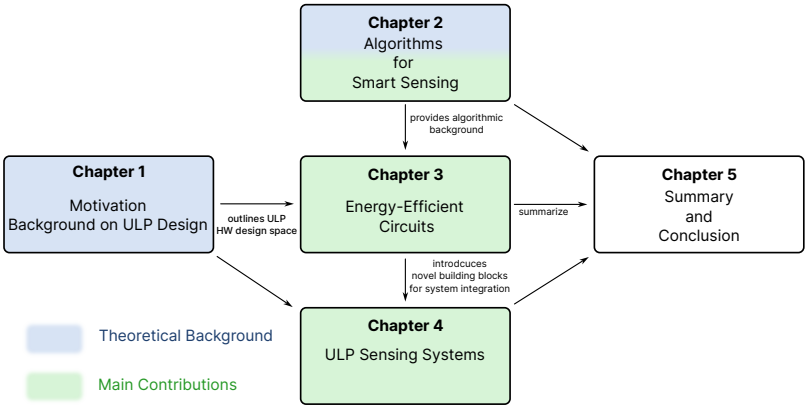


Figure 1.8 – Thesis structure and dependencies between chapters

Chapter 2 introduces the reader to the current low-power signal processing landscape like quantized neural network and the novel compute paradigm of hyper-dimensional computing with an emphasis on the latter.

Chapter 3 In Chapter 3, we shift the focus from applications and algorithms to their physical implementation in the form of energy-efficient circuits. We introduce some novel elementary building blocks for energy efficient HDC, before we shift focus to the next higher hierarchical level by presenting *Hypnos*, a highly efficient hardware accelerator for HDC, along with some first silicon results of the aforementioned circuitry as part of the

Rosetta prototype chip. Furthermore, we present lightweight data preprocessing units for energy-proportional wake-up circuitry.

Chapter 4 presents two SoC designs taped out in GF 22nm and TSMC 16nm that reuse the concepts and circuitry introduced in Chapter 3. This chapter focuses on the system-level architecture for dataflow bandwidth management, dataflow orchestration and the software development paradigm within those systems.

Chapter 5 concludes the thesis with a summary of our findings, drawing conclusions and providing the author's perspective on the future challenges and potential developments towards ever more energy-efficient circuitry at the edge.

1.6 Contributions

The contributions of this thesis within the scope of open-hardware energy-efficient circuit design are summarized in figure 1.9. For the sake of completeness, a quick summary of all contributions is provided below:

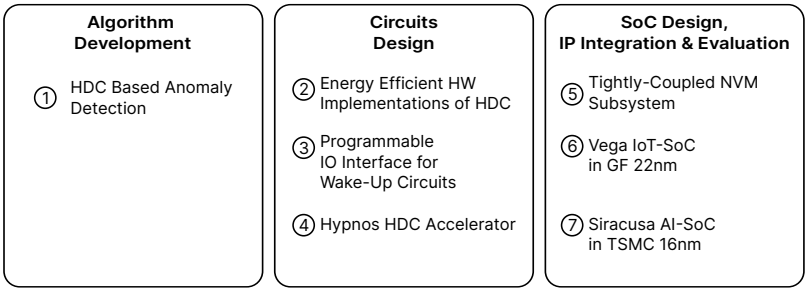


Figure 1.9 – Summary of this thesis' contributions categorized by topic.

1. Design and feasibility analysis of a novel HDC based algorithm for ball bearing anomaly detection. (Section 3.4.6)
2. Design and Verification of energy-efficient circuits for HDC operators. (Section 3.3)
3. Development of a lightweight SPI-based sensor data pre-processing unit with micro-codeable transaction flow and configurable multi-channel filter pipelines. (Section 3.6)
4. Design and silicon realization of Hypnos, an all-digital SCM based hyperdimensional computing accelerator taped-out in TSMC 65nm. (Section 3.4)
5. Integration of an energy-proportional smart wake-up pipeline in Vega an ULP IoT SoC taped out in 22 nm technology. (Section 4.2)
6. Development and Verification of an energy-efficient hybrid memory interconnect for digital hardware accelerators with support for hardware-assisted software-managed virtual memory. (Section 4.3.2)
7. Design, verification and implementation of Siracusa, a 16 nm

finFET heterogenous smart sensing SoC with an MRAM NVM-based first-level weight memory subsystem coupled with an ultra-low-latency high bandwidth interconnect. (Section 4.3)

While the topics covered in chapter 1 and 2 are the sole work of myself, the SoC architectures introduced in chapters 3 and 4 are a joint effort of many people. Thus unless I was the only contributor to the material covered in a section, I declare the exact degree of my own involvement in each relevant section's introduction.

1.6.1 List of Publications

The results and findings of this thesis have been published in a number of peer-reviewed publications. The following publications cover the above-specified key-contributions:

[121] M. Eggimann, A. Rahimi, L. Benini, "*A 5 μW Standard Cell Memory-based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing*", IEEE Trans. Circuits and Systems I: Regular Papers (TCAS I), 2021.

[64] O. Harel, E. Casarrubias, M. Eggimann, F. Gurkaynak, L. Benini, "*64-kB 65-nm GC-eDRAM With Half-Select Support and Parallel Refresh Technique*", IEEE Solid-State Circuits Letters (SSC-L), 2022.

[122] D. Rossi, F. Conti, M. Eggimann, S. Mach, A. Di Mauro, M. Guermandi, G. Tagliavini, A. Pullini, I. Loi, E. Flammand, L. Benini, "*A 1.3TOPS/W @ 32GOPS Fully Integrated 10-Core SoC for IoT End-Nodes with 1.7μW Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode*", in Proc. IEEE ISSCC, 2021.

[84] D. Rossi, F. Conti, M. Eggimann, S. Mach, A. Di Mauro, M. Guermandi, G. Tagliavini, A. Pullini, I. Loi, E. Flammand, L. Benini, "*Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode*", IEEE Journal of Solid State Circuits (JSSC), 2022.

The following publications are not directly included in the

thesis because they were outside the focus of this thesis research topic or because the contributions were limited to smaller technical contributions, ideas or project advisory.

[123] M. Eggimann, S. Mach, M. Magno, L. Benini, "*A RISC-V Based Open Hardware Platform for Always-On Wearable Smart Sensing*", in Proc. IEEE IWASI, 2019. **Best Paper Award**

[124] F. Conti, D. Rossi, G. Paulin, A. Garofalo, A. Di Mauro, G. Rutishauser, G. Ottavi, M. Eggimann, H. Okuhara, V. Huard, O. Montfort, L. Jure, N. Exibard, P. Gouedo, M. Louvat, E. Botte, L. Benini, "*A 12.4TOPS/W @ 136GOPS AI-IoT System-on-Chip with 16 RISC-V, 2-to-8b Precision-Scalable DNN Acceleration and 30%-Boost Adaptive Body Biasing*", in Proc. IEEE ISSCC, 2023.

[125] M. Scherer, M. Magno, J. Erb, P. Mayer, M. Eggimann, L. Benini, "*TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition With Short Range Radars*", IEEE Internet of Things Journal (IoT-J), 2021.

[126] M. Osta, A. Ibrahim, M. Magno, M. Eggimann, A. Pullini, P. Gastaldo, M. Valle, "*An Energy Efficient System for Touch Modality Classification in Electronic Skin Applications*", in Proc. IEEE ISCAS, 2019.

[127] X. Wang, L. Cavigelli, M. Eggimann, M. Magno, L. Benini, "*HR-SAR-Net: A Deep Neural Network for Urban Scene Segmentation from High-Resolution SAR Data*", in Proc. IEEE SAS, 2020.

[128] M. Magno, X. Wang, M. Eggimann, L. Cavigelli, L. Benini, "*InfiniWolf: Energy Efficient Smart Bracelet for Edge Computing with Dual Source Energy Harvesting*", in Proc. IEEE DATE, 2020.

[129] M. Eggimann, J. Erb, P. Mayer, M. Magno, L. Benini, "*Low Power Embedded Gesture Recognition Using Novel Short-Range Radar Sensors*", in Proc. IEEE SENSORS, 2019.

[130] K. Dheman, P. Mayer, M. Eggimann, S. Schuerle, M

Magno, "*ImpediSense:A Long Lasting Wireless Wearable Bio-Impedance Sensor Node*", Sustainable Computing: Informatics and Systems Journal, 2021.

[131] M. Eggimann, C. Gloor, L. Cavigelli, M. Schaffner, A. Smolic, L. Benini, "*Hydra: An Accelerator for Real-Time Edge-Aware Permeability Filtering in 65nm CMOS*", in Proc. IEEE ISCAS, 2018.

[132] K. Dheman, P. Mayer, M. Eggimann, M. Magno, S. Schuerle, "*Towards Artefact-Free Bio-Impedance Measurements: Evaluation, Identification and Suppression of Artefacts at Multiple Frequencies*", IEEE Sensors Journal, 2022.

Chapter 2

Hardware friendly Algorithms for Smart Sensing Applications

2.1 Introduction

Having introduced the design space of ULP architectures, this chapter focuses on the algorithm domain of smart sensing systems. After all, optimal energy efficiency is not achieved with general-purpose compute cores designed oblivious to the workload, but it rather is a matter of hardware-software co-optimization. In the light of this thesis, we exercise the developed circuits and systems with sample algorithms from the three application domains *predictive maintenance*, *smart-prostetics* and *augmented reality* applications. Like most edge computing applications, all of these workloads heavily rely on machine learning algorithms which in many domains replaced traditional signal processing algorithms based on expert knowledge models. Using machine learning even on energy-constrained embedded devices, an

approach often summarized under the term TinyML, has become an extremely effective general-purpose design pattern for all sorts of signal processing tasks [133]. Ultimately, all signal processing problems can be expressed as the task of letting a system execute a particular, most often non-linear mapping that, given some form of input data, so-called features in ML jargon, yields the desired output. E.g. being presented with human speech in the form of audio waves captured by a microphone produce the spoken words in text form corresponding to it. For applications like flight controllers for drones where the interaction between input and the desired output is well understood, this function can be expressed in terms of e.g. physics formulas based on our modelling approach to nature itself (in this case, classic control system theory). For other problems like the speech recognition example above, the mechanism of these interactions are either not obvious or very hard to express explicitly in a hand-crafted algorithm. What now contrasts machine learning methods from traditional experts-knowledge-based systems is that instead of encoding the non-linear function ourselves directly in the form of an algorithm custom-tailored to the problem, we let a machine learn the function by itself. All machine learning algorithms thus have in common that they aim to approximately represent arbitrary high-dimensional non-linear functions in the form of an algorithmic template that contains tunable components in the widest sense to steer the template towards the desired function. Where the various ML frameworks differ is in the algorithmic template they use as the basis to represent the non-linear target function and in the mechanism of how the template ”tuneables” are derived by the machine. We thus start this chapter with a brief introduction to two ML frameworks in the TinyML software ecosystem that will play a role in the remainder of this thesis. These are *Quantized Neural Networks* and *Hyperdimensional Computing*, with a strong emphasis on the latter since many of the circuits introduced in chapter 3 will rely on the fundamental concepts of HDC.

2.2 Quantized Neural Networks

The majority of machine learning algorithms deployed on ULP edge devices fall under the quite general category of *quantized neural networks (QNNs)*. In this section, we aim to provide a birdseye view on the topic to put this approach into context with competing ML paradigms. Given the vastness and pace of research in this domain, an in-depth introduction to the topic is entirely outside the scope of this thesis. For a more thorough introduction to neural networks, we warmly refer to the overview article published in nature by LeCun *et al.* [134], the books *Deep Learning* [135], *Python Machine Learning* [136] and *Probabilistic Machine Learning* [137] for a deep-dive into the topic.

With this disclaimer aside, let us jump onto the main subject; to understand how QNNs works, we first have to introduce the working principles of Neural Networks (NNs) in general.

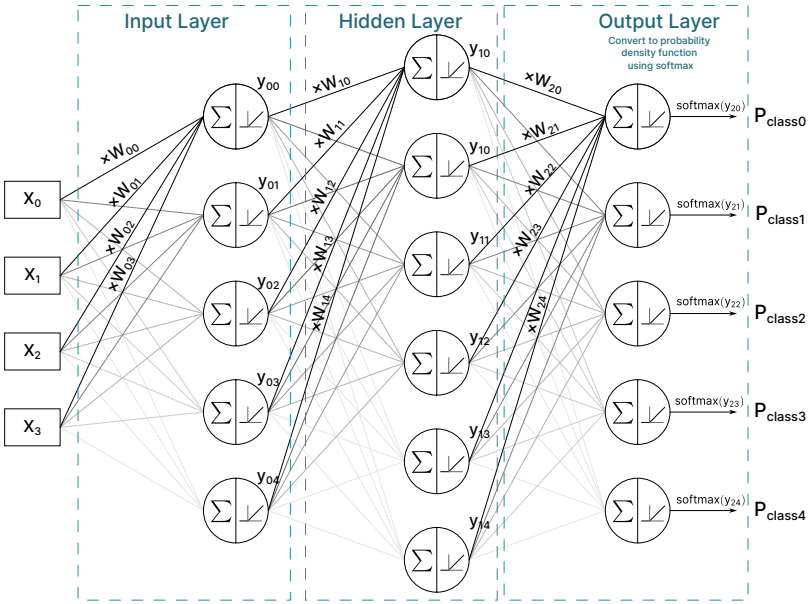


Figure 2.1 – Working principle of neural networks illustrated on a multi-layer perceptron.

A neural network is an algorithm often visualized in the form of a compute graph as depicted in figure 2.1. In such a compute graph, the inputs, usually called the *features* or activations, pass through the compute graph *forward* (here from left to right) to finally produce the desired output value. The basic building block in this compute graph is the neuron, a function unit that multiplies each of its inputs x with an input specific *weight* w (illustrated as edge labels), adds a scalar *bias* value b to it and passes this sum of the weighted inputs through a non-linear function σ. These neurons, inspired in their working principle by the human brain, are usually structured in several *layers* of neurons that process the preceding layer's outputs to produce the inputs of the following layer. The operation of weighted summation of inputs, basically a dot-product between the input vector and a weight vector, can be expressed as a single vector-matrix multiplication to represent an entire layer [1]:

$$\hat{\boldsymbol{Y}} = \sigma(\boldsymbol{W}^T \boldsymbol{X} + \boldsymbol{B}) \tag{2.1}$$

If we now stack many of these layers to approximate more complicated target mappings, we arrive at what is called a DNN. Of course, any linear combination of inputs, regardless of the used weights and the degree of layer stacking, can ultimately only lead to a linear function. This is where the aforementioned non-linear function σ of the neuron output, the *activation function* of the neuron itself, comes into the picture. Its purpose is to remove the linearity of the system at the layer boundary, which allows the NN to act as a universal function approximator [138]. There are many options of activation functions to choose from, and the choice influences the learning performance, the computational cost of inference as well as numerical stability [139], [140]. In practice, however, most network architectures settle for variants derived from the so-called *Rectified Linear Unit (ReLU)* function defined as follows:

$$\sigma_{ReLU}(x) = \begin{cases} x & \text{if } x >= 0 \\ 0 & \text{else} \end{cases} \tag{2.2}$$

[1] If we consider 1D inputs. For multiple input dimensions, the concept extends to the multiplication of multiple weight tensors with activation tensors, so-called *feature maps*.

The structure of the NN, the choice of activation function, loss function and other parameters affecting the network topology, also known as the *hyperparameters*, are chosen at the beginning of the training process, either by an experienced human or by dedicated network architecture search (NAS) algorithms. On the other hand, the weights are learned during the network training phase, where the initially random neuron weights are modified iteratively to approximate the desired mapping. The training process can be thought of as a multi-dimensional optimization problem where the aim is to minimize some form of a *loss function* derived from intrinsic[2] or extrinsic [3] feedback. During a training epoch, a mini-batch of sample inputs B is passed forward through the network, and for each sample i in the batch the resulting, initially very high, loss $\boldsymbol{Q_i}(\boldsymbol{W})$ is calculated. In a second step, the network is traversed *backwards* to calculate the loss average[4] function gradient $\overline{\nabla \boldsymbol{Q_B}(\boldsymbol{W})}$ at each layer (leveraging the chain rule). I.e. we calculate the vector towards the closest *local* minimum of the loss function. Finally, the weights are updated by moving one step along the gradient's direction with (dynamic) stepsize $\eta(k)$:

$$\boldsymbol{W_{k+1}} = \boldsymbol{W_k} - \eta(k) \cdot \overline{\nabla \boldsymbol{Q_B}(\boldsymbol{W_k})} \tag{2.3}$$

This process, known as mini-batch gradient descent, an improved variant of the traditional gradient descent algorithm, is repeated in each epoch using a different minibatch B until the network's accuracy converges. The result is a trained *model* that can be used for inference on previously unseen input. The number of epochs required for convergence depends largely on the *learning rate* $\eta(k)$ but is also a function of many other parameters. A standard technique to reduce the training time is batch-normalization, a technique to mitigate the so-called *internal covariate shift* of the network. Internal covariate shift describes the effect of ever-changing statistical distributions of the feature maps between the layers during the training process, which forces the following layers to adapt constantly. Batch normalization counteracts this effect by inserting an additional normalization step

[2] In the case of self-supervised training like e.g.generative adversarial networks (GANs) [141]

[3] Usually in the form of a training set of correct sample-label pairs for supervised learning scenarios.

[4] Averaged over the samples of the batch

between each layer; For every mini-batch of training data, the output feature maps of a layer are normalized to zero mean and standard deviation of 1 followed by a linear transformation with the per-layer learned scaling vector $\boldsymbol{\gamma}$ and offset vector $\boldsymbol{\beta}$ [142].

The illustrated network in figure 2.1, a so-called "multilayer perceptron" consists only of fully-connected layers; that is, every neuron is connected to every neuron output of its preceding layer. Especially in the image processing domain, a very successful alternative to fully connected layers is the use of *convolutional* kernels. Instead of multiplying the feature maps with weight tensors, we fold the feature maps with sets of convolutional kernels. These convolutional layers are combined with dimensionality reducing *pooling layers*[5] to form a convolutional neural network (CNN). The combination of convolutional and pooling layers significantly reduces the network size and makes CNN robust towards spatial translation and scaling of the inputs, an important characteristic for e.g. object detection tasks.

Besides CNNs, there are many other flavours of Neural Networks with different network topologies. Some NN topology variants aim to improve the network's ability to learn in general [6] like skip-connections in residual neural networks (ResNets) while other variants are custom tailored for certain problem classes, e.g. recurrent neural network (RNN), long short-term memory (LSTM) or recently *tranformer* networks for natural language processing applications. For a reasonably up-to-date overview of the various NN topologies, we refer to [143].

Traditionally, NN used to be trained and executed in floating point representation on general-purpose compute hardware using graphics processing unit (GPU) accelerated kernels. Yet, to enable deployment on energy-constrained edge devices, the computational throughput and memory requirements constraints have to be scaled down by several orders of magnitude to meet the hardware specifications. Training a shallower and thus smaller network is an obvious approach for model size reduction, but this usually implies sacrificing accuracy. Two commonly used techniques for model reduction that yield way less drastic decreases in classification accuracy are sparsification and

[5]These layers combine spatially correlated values in a feature map by keeping only the average value of the sub-tile (average pooling) or the maximum of all encountered values (max-pooling) in a tile.

[6]Counteracting an effect known as *vanishing gradients*

quantization. The weights in a neural network exhibit a lot of inherent sparsity; common techniques to counteract overfitting[7], like dropout layers and other weight pruning techniques usually result in 60% to 90% of the weights in CNN and fully-connected layers to be zero. While sparsity can be leveraged rather easily for memory reduction by using sparsity weight encoding formats, reduction in compute requirements are harder to obtain since skipping of obsolete computations usually implies departing from the regular data-flow NN hardware are designed for [144].

Conversely, Quantization directly translates not only to model size reduction but also a reduction of the energy cost per operation. Instead of computing at full floating point precision, the weights and/or the activations of the network are converted to integer representation using (usually) a step-wise approximation of linear mapping. The effect of quantization is significant; A bit-width reduction from 32-bit to 8-bit representation decreases the memory footprint of the model by a factor of 4 whereas the computational energy is quadratically scaled down by a factor of 16 [145] without any significant loss of accuracy. E.g. Nagel *et al.* report less than 1% accuracy loss on common image processing CNNs like MobilenetV2 or ResNet18 [146]. Their quantization technique belongs to the category of *post-training quantization* where quantization of full-precision model takes place *after* the training procedure. This approach works well with modest degrees of quantization. But with increasing quantization noise when reducing the data representation resolution, the network can no longer maintain the accuracy. The alternative category of quantization technique is *quantization-aware training* where quantization noise is already modelled and introduced during the training process or, alternatively, the post-quantized network is retrained in order to recover the accuracy drop [147]. This technique has been demonstrated quite successfully for very aggressive quantization regimes down to 3-bit ternary or even binary NNs [145]. Quantization not only finds application for inference deployment on ULP systems but is also regularly used during network training. Yet the dynamic range required to ensure numerical stability of the back-propagation algorithm disqualifies low-resolution integer

[7]The undesirable effect of a network memorizing the correct answers rather than actually "learning the concept"

computation. While the majority of the networks are still trained using traditional 32-bit floating point representation the current trend for AI-training accelerators is to use 16- or even 8-bit floating-point with various exponent/mantissa partitionings [148]. A famous example is Google's TPU SoCs that uses the 16-bit bfloat format [60], [61].

2.3 Hyper-Dimensional Computing

The drastically increased demand for computational throughput at high energy efficiency led to what we currently call the new golden age of compute hardware and architecture research. A plethora of novel circuits has emerged that is situated on a completely different corner of the efficiency/reliability-pareto curve. As we saw in chapter 1, many of these technologies trade improved energy efficiency for circuit reliability and accuracy. Deploying the traditional binary representation compute paradigm on these kinds of systems faces increasing challenges to keep up with their inherent error rates. The error resiliency of NN based algorithms can deal with this inherent noise to some degree. However, error rates of novel compute hardware like PCM-based in-memory computing can be orders of magnitude higher than on traditional systems. Furthermore, NN are not a silver bullet for all kinds of computation and e.g. struggles to be leveraged for symbolic reasoning or in the explainability of inference.

So-called vector symbolic architectures (VSAs), are a relatively new compute paradigm that have promising characteristics well matched for these emerging technologies [149], [150]; VSA also referred to as hyper-dimensional computing is a mathematical framework whose key characteristic is to operate with entities in a very high dimensional representation space [151]. It is well suited to operate both on symbolic and numerical data and has very favourable error resiliency properties making it an ideal candidate for deployment in both highly distributed computing environments or as the basis for approximate computing applications. Although the research on VSAs dates back to the 1990s with the work from Hinton on connectionist network mapping [152], the approach only recently gained more traction in terms of research efforts, notably with the work from IBM on PCM based hardware realization [153]. Today, besides other neuro-inspired systems like spiking neural networks HDC is listed as one of the most promising novel compute paradigms for the post-CMOS era in the 2022 IEEE International Roadmap for Devices and Systems [154].

Given the relevance for the circuits that will be introduced in chapter 3 this section introduces the concepts and mathematical foundation of HDC, compares its traits with the traditional digital

numeric and symbolic representation/computation framework and aims to instil an intuition about the primitive operations and data structures of VSAs.

2.3.1 Hollistic Symbol Representation as Hyper Vectors

At the heart of any computational framework we have:

(a) A mapping of conceptional entities like e.g. numbers or symbols like characters to patterns represented within a physical medium,

(b) A set of rules on how to manipulate, combine and interact with those patterns i.e. algorithms to form new patterns/representations like compound data structures.

For example, the conventional digital compute framework maps numerical values to representation formats like binary two's complement representation with a fixed number of bits or floating point formats like the ones defined in IEEE 754 [155]. These patterns are manipulated using familiar arithmetic and boolean operators to execute algorithms. Compound data structures are often represented as a collection of several primitive values where the spatial position in memory has semantic meaning e.g. consider an array data structure where the position in memory implies the ordering of values.

VSAs are different in two main aspects illustrated in figure 2.2:

1. The cardinality ratio between symbolic state space and representation space
2. The spatial concentration of information along the pattern dimensions

The traditional compute framework for symbolic reasoning and numeric calculations mostly use low- if not minimal pattern dimensionality to represent entities. E.g. the digital patterns for the representation of numbers like two's complement or ASCII for characters use just as many bits as necessary to represent every possible element of the abstract entity space (numbers or letters). These patterns are then manipulated using basic arithmetic or logic operations like addition, multiplication or bitwise operations. VSAs

Figure 2.2 – Comparison of low-dimensional and vector symbolic mapping of entities

instead draw random vectors, so-called *item vectors* from a very high-dimensional pattern space to represent a single entity. Commonly used are vectors between 1000 and 10,000 dimensions. Obviously, this results in the space of possible patterns being much larger than will ever be required to represent the possible entities in our compute framework. Therefore, the cardinality of the representation space in VSAs is much higher than necessary. This yields some interesting characteristics as we will see later.

The second aforementioned difference revolves around the significance of the individual pattern dimensions for determining the conceptual entity it represents. An example should make this characteristic easier to understand; Consider the individual dimensions, in this case the bits, of numerical values represented in two's complement format. In this compute framework the information is concentrated on the MSBs of the binary pattern. Expressed in a more abstract way, the similarity of a noisy value representation in the pattern space (e.g. measured in the hamming distance in this case) does not translate linearly to the similarity in the symbol space (the absolute error) and is *not* independent of the bit position.

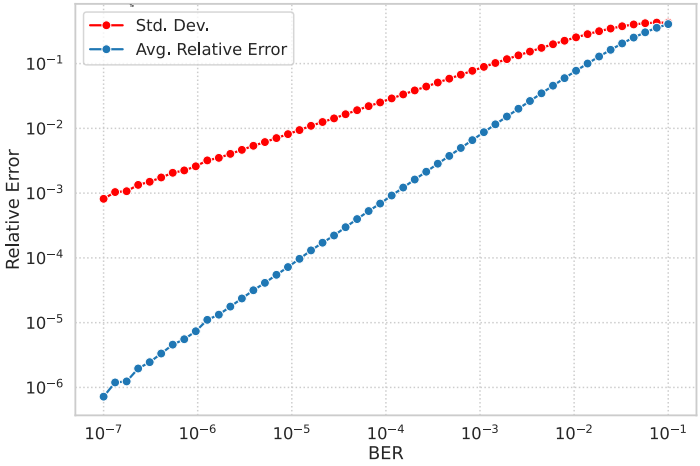


Figure 2.3 – The average relative amplitude error of 32-bit binary two's complement values under varying bit error rates.

This aspect of data representation can be quite problematic if we deal with high bit error rates. Figure 2.3 illustrates the problem. While the average amplitude error increases linearly with the relative amplitude error being about one order of magnitude higher than the corresponding bit error rate, the variance of the experienced errors is three orders of magnitudes higher and increases with the square root of the BER. The reason for this large variance of the errors is the exponentially larger impact if we flip an MSB which results in a relative error of 100%. This is under i.i.d. bit error distribution which is a rather conservative assumption. Many technologies exhibit a quite strong spatial correlation of bit errors due to on-chip variation which can result in even less favourable error statistics.

HDC on the other hand is much more error resilient. In stark contrast to traditional value representations HDC uses a *holistic*, i.e. dimension independent mapping of information to patterns. Since entities are mapped to random vectors, no particular bit of representation has more significance to information content than any

other bit. Also, the fact that we are dealing with very large dimensions results in some interesting properties regarding the probability of similarity and orthogonality in the vector space of a VSA; Given two HD vectors independently and randomly drawn from the vector space of dimensionality D, the probability of the two vectors having a hamming similarity of k follows the binomial distribution which can be approximated with the normal distribution for large N:

$$\lim_{D\to\infty} P(k) = \frac{\binom{D}{k}}{2^D} = \frac{1}{\sqrt{2\pi}} e^{\frac{-k^2}{D/4}} \tag{2.4}$$

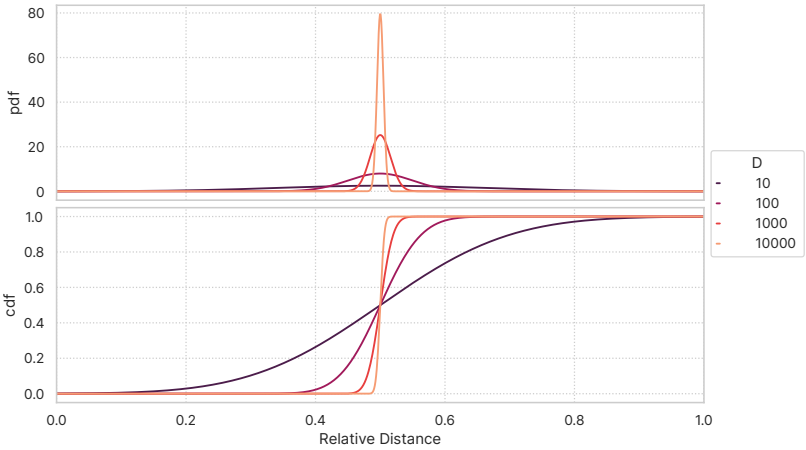


Figure 2.4 – Hamming Distance probability density function (top) and cumultative probability density function (bottom) for two randomly drawn vectors of various dimensionalities.

The expected value of their hamming similarity will always be $D/2$ regardless of D. However, as illustrated in figure 2.4 the probability of coincidental similarity, in other words, the relative variance of the similarity decreases with higher dimensions converging towards the normal distribution following the central limit theorem. E.g. the probability of two random vectors with $D = 10000$ having a similarity deviation from $D/2$ by only 2%, i.e. a similarity outside the range of 4800 to 5200 is only 60.5×10^{-6}. There are two important effects of this probability distribution:

1. We can generate a virtually unlimited amount of *quasi-orthogonal*, that is almost orthogonal vectors just by random sampling of the representation space.
2. Given a noisy version of any item vector, we can recover the original entity with high probability by looking up the most similar pristine entity vector in a so-called *associative memory*.

There are numerous flavors of VSAs with prominent examples being the holographic reduced representation (HRR) [156], multiply-add-permute (MAP) [157], binary spatter code (BSC) [158] or sparse binary distributed represenation (SBDR) [159]. These cognitive codes mainly differ in the format used for the individual vector dimensions, the sparsity of the vector entries, the distance metric used to asses pattern similarity and the vector manipulation operations. Yet they all share similar structural and statistical properties in the context of similarity preservation and the ability to represent compound data structures using the same atomic entities/vectors (we will highlight this property more thoroughly in subsection 2.3.3). For a comprehensive comparison of these VSAs we refer to the excellent work of Schlegel *et al.* [160]. In the remainder of this thesis we restrict ourselves to one particular, CMOS-friendly VSA named binary spatter code which operates on binary vectors of length D with i.i.d. probability of ones and zeroes.

2.3.2 Fundamental Operations of Binary Spatter Codes

The mathematical framework of binary spatter code is characterized by the nature of the vector constituents, the similarity metric to compare vectors with each other and the elementary operations to manipulate and transform vectors. In this section, we provide the mathematical fundament of BSCs. Given the thoroughness of Kleyko *et al.*'s survey on the mathematical constructs within the HDC framework [151], we intentionally stay close in notation style to their work. As already mentioned before, binary spatter codes operate on vectors consisting of binary values. Thus the *hamming distance*, the number of vector components where two vectors disagree whith each other, is a quite natural choice for the similarity metric. Throughout the remainder

of the thesis we use the notation $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle$ to refer to the hamming distance between the two vectors X and Y and whenever we use the approximate equals sign as in e.g. $\boldsymbol{X} \approx \boldsymbol{Y}$ we thus mean, the two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ have a small hamming distance that is statistically extremely unlikely to occur by chance i.e. $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle \ll D/2$.

Like most VSA, BSC defines four elementary operations, *bundling*, *binding*, *permutation* and *associative lookup*. We start by defining those three operators and highlighting their key properties. However, we will postpone the discussion of the applicability of these properties to section 2.3.3.

Bundling The *bundling*-, also referred to as the *superposition*-operator denoted with "+" is defined as the element-wise sum of the operand vectors followed by a normalization step.

$$\begin{aligned} \boldsymbol{S} &= (s_1, s_2, ..., s_D) \\ &= [\sum_k \boldsymbol{X}^{(k)}] \\ &= ([\sum_k x_1^{(k)}], [\sum_k x_2^{(k)}], ..., [\sum_k x_D^{(k)}]) \end{aligned} \tag{2.5}$$

The normalization step, in the previous equation denoted with [], is necessary in order to map the integer-valued output domain of the vector sum back to binary values. The thresholding function is defined as follows:

$$[x] := \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \\ 0 \text{ or } 1 \text{ with iid. probability} & \text{else} \end{cases} \tag{2.6}$$

In other words, for each dimension of the operand vectors, the bundling operator assigns a 1 to the output vector if the majority of the operand vectors had a 1 at this particular vector entry. It assigns a 0 if there are strictly more 0's at this vector dimension and ties (which are only possible if we bundle an even number of operand vectors) are broken at random. Breaking ties at random

is important to avoid any bias, i.e. similarity, towards the all "zeroes" or the all "ones" vector.

The bundling operation has the important property that it preserves the similarity of the input vectors. I.e. the output vectors will be very similar to every input vector and the operation is holistic since there is no flow of information between different vector dimensions.

Binding The *binding* operation, denoted with $\odot$, is defined as component-wise exclusive or (XOR) operation of each vector component. The operation has the property of producing an output vector that is dissimilar to both its input vectors, yet is an invertible function (in this case even self-invertible). Binding thus maps hypervectors to an entirely different vector space that is almost orthogonal to both operands. Like *superposition* the *bundling* operator is holistic.

Permuation Just as the name implies, *permutation* is the operation of applying an arbitrary D-dimensional permutation ρ to the elements x_i of the operand vector X,

$$\rho \in S_D, \rho(\boldsymbol{X}) := (\rho(x_1), \rho(x_2), ..., \rho(x_D)) \tag{2.7}$$

where S_D denotes the symmetric group of size D. In contrast to the previous two operators, permutation given a static permutation matrix ρ is a unary operation which yields another hypervector. Similar to the binding operation, permutation maps its operand to a quasi-orthogonal subspace i.e. produces very dissimilar output vectors.

Associative Lookup As we will see in the next section, an extremely crucial operation of HDC is the so-called *associative lookup* operation. Given a set of so-called *item vectors* $\boldsymbol{H_i} \in \mathbb{H}$ and a similarity threshold Θ the associative lookup operation on vector $\boldsymbol{X}$, denoted with $\mathbb{H}_\Theta \boldsymbol{X}$ determines the most similar vector H_i with threshold larger than Θ. I.e.,

$$\mathbb{H}_{\Theta}\boldsymbol{X} := \begin{cases} \text{argmax}\left(\langle \boldsymbol{X}, \boldsymbol{H_i} \rangle\right), \forall \boldsymbol{H_i} \in \mathbb{H} & \text{if} \max\left(\langle \boldsymbol{X}, \mathbb{H} \rangle\right) > \Theta \\ \text{None} & \text{else} \end{cases} \tag{2.8}$$

$\mathbb{H}$ is usually called the *item memory* of the HDC algorithm because it needs to keep a hold of all atomic entity vectors. An entity performing the above-defined operation of associative lookup is also known as an *auto-associative memory.*

Mathematical Properties

Given the previous definition of BSC the operations define a mathematical construct with almost field-like properties:

(1) Commutativity:
$\boldsymbol{X} + \boldsymbol{Y} = \boldsymbol{Y} + \boldsymbol{X}$ and $\boldsymbol{X} \odot \boldsymbol{Y} = \boldsymbol{Y} \odot \boldsymbol{X}$

(2) (approximate) Associativity[8]:
$\boldsymbol{X} + (\boldsymbol{Y} + \boldsymbol{Z}) \approx (\boldsymbol{X} + \boldsymbol{Y}) + \boldsymbol{Z}$ and $\boldsymbol{X} \odot (\boldsymbol{Y} \odot \boldsymbol{Z}) = (\boldsymbol{X} \odot \boldsymbol{Y}) \odot \boldsymbol{Z}$

(3) Self-inversion property of binding:
$\boldsymbol{X} \odot (\boldsymbol{X} \odot \boldsymbol{Y}) = \boldsymbol{Y}$

(4) Similarity preservation of binding:
$\langle \boldsymbol{Y}, \boldsymbol{Z} \rangle = \langle \boldsymbol{X} \odot \boldsymbol{Y}, \boldsymbol{X} \odot \boldsymbol{Z} \rangle$

(5) Similaritiy preservation of permutation:
$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \langle \rho \boldsymbol{X}, \rho \boldsymbol{Y} \rangle$

(6) Distributivity of binding over superposition:
$\boldsymbol{X} \odot (\boldsymbol{Y} + \boldsymbol{Z}) = \boldsymbol{X} \odot \boldsymbol{Y} + \boldsymbol{X} \odot \boldsymbol{Z}$

(7) Distributivity of permuation of binding and superposition:
$\rho(\boldsymbol{X} + \boldsymbol{Y}) = \rho \boldsymbol{X} + \rho \boldsymbol{Y}$ and $\rho(\boldsymbol{X} \odot \boldsymbol{Y}) = \rho \boldsymbol{X} \odot \rho \boldsymbol{Y}$

2.3.3 Compound HDC Data Structures

By now we already know that the conceptual entities of our symbol set, be it numbers, characters etc. are represented by vectors randomly

[8]The fact that we threshold the sum vectors during the superposition operation leads to the associativity being only approximate

drawn from the D-dimensional vector space. However, most algorithms require the use of more complex, compound data structures e.g. lists, sets and key-value pairs. Surprisingly, in HDC many of these compound data structures can still be represented with *single* vectors. This is in stark contrast to conventional compute frameworks where we usually combine multiple atomic patterns to a data record and thus assign semantic meaning not only to the content of each datum but also its spatial location within the entire record. E.g. in the case of a list stored in the form of an array in memory, the index of each entry is determined by the physical location of the corresponding binary pattern in memory.

In this section, we summarize the most commonly used holistic compound data structures in HDC: *sets*, *sequences* and *key-value records*. The literature on HDC also contains techniques to represent more advanced data structures like stacks or graphs [151]. But for the sake of providing sufficient background knowledge to understand the circuits introduced in chapter 3 of this thesis, sticking to the aforementioned basic data structures is quite sufficient.

Sets

Sets of hypervectors can easily be represented using the superposition/bundling operator. I.e. the set U with elements $\boldsymbol{S_0}, \boldsymbol{S_1}, ..., \boldsymbol{S_{N-1}}$ can be approximately represented with $\sum_{k=0}^{N-1} S_k$. We leverage the property of the bundling operation to produce vectors that are similar to each operand. Thus given a set vector $\boldsymbol{U}$ representing the set of elements $S \in \mathbb{H}$, we can iteratively recover each set element using the associative lookup operation. In each step, the lookup operation yields us the most similar item vector $\boldsymbol{S_k}$. In order to retrieve the other set elements, we add the inverse of the recovered set element $\overline{\boldsymbol{S_k}}$ to the set vector to shift the set vector further away from the just retrieved element. That way, the next lookup will yield us another, different set element until we have recovered all n elements. Being able to recover every item vector that constituents the set allows us to perform all the common set operations. Yet for many operations, there are faster approximate approaches; E.g. we can easily test set membership by calculating the similarity of a vector $\boldsymbol{X}$ and the set vector $\boldsymbol{U}$. Other common operations on sets like intersection and union can also be

mapped to the hyperdimensional domain using different thresholding levels during the associative lookup [151].

Of course, the *capacity* of a hypervector in storing a set is not limitless. Each element added to the set behaves like random noise added to the similarity lookup of all the other elements already in the set. In other words, the more vectors we add to the set, the more dissimilar each item vector within the set becomes to the set vector itself to the degree that eventually the similarities are no longer above the threshold level Θ. The storage capacity of a single vector is thus mostly determined by its dimensionality. For a more elaborate analysis of the bundling capacity, we refer to Schmuck *et al.* [161].

Sequences

Since the bundling operator is commutative, it cannot be directly used to encode the ordering of information required to represent sequences of items. One way of encoding the index information into a hypervector is to mark each element with repeated permutation. I.e. we store our sequence of vectors $\boldsymbol{X_0}, \boldsymbol{X_1}, ..., \boldsymbol{X_{N-1}}$ in a sequence vector $\boldsymbol{S}$,

$$\boldsymbol{S} := \sum_{k=0}^{N} \rho^k \boldsymbol{X_k} \tag{2.9}$$

where ρ^k denotes the repeated application of the same permutation ρ. Given the distributive property of the permutation operation, appending to the list is as easy as calculating $\boldsymbol{S'} = \rho\boldsymbol{S} + \boldsymbol{X_n}$. Random access to a list element at index k is also straightforward due to the similarity preservation of permutation:

$$\boldsymbol{X_k} \approx \rho^{-k}\boldsymbol{S} \implies \boldsymbol{X_k} = \mathbb{H}(\rho^{-k}\boldsymbol{S}) \tag{2.10}$$

Once again, the same limitations regarding *bundling capacity* also apply to sequences.

Dictionaries/Data Records

The third commonly used data structure we want to introduce at this point is key-value pairs, commonly referred to as dictionaries. For

a single key-value pair compound hypervector we have the following structural requirements:

- Querying the vector should be easy; i.e. given one of the two vectors, it has to be easy to recover the other.
- The compound hypervector must be *disimilar* to both, the key and the value item vector.

The reasoning behind the first property is obvious, given that querying is one of the fundamental operations performed on key-value pair structures. The need for the second property arises from the necessity to clearly distinguish data records using the same value but different keys. E.g. consider the two data records $(\boldsymbol{K_1}, \boldsymbol{V_1})$ and $(\boldsymbol{K_2}, \boldsymbol{V_1})$ represented by the compound record vectors $\boldsymbol{R_{K1}}$ and $\boldsymbol{R_{K2}}$ respectively. If $\boldsymbol{V_1} \approx \boldsymbol{R_{K1}}$ and $\boldsymbol{V_1} \approx \boldsymbol{R_{K2}}$ it would follow that $\boldsymbol{R_{K1}} \approx \boldsymbol{R_{K2}}$. This would lead to ambiguity in recovering the correct record vector given a noisy version of either $\boldsymbol{R_{K1}}$ or $\boldsymbol{R_{K2}}$, which would defeat the purpose.[9]

The *binding* operation fulfils both of the above requirements. The record vector $\boldsymbol{R} := \boldsymbol{K} \odot \boldsymbol{V}$ is dissimilar to both the key and the value vector. Yet leveraging the self-inverse property of binding, we can easily *unbind* i.e. recover the value item vector:

$$\boldsymbol{K} \odot \boldsymbol{R} = \boldsymbol{K} \odot (\boldsymbol{K} \odot \boldsymbol{V}) = \boldsymbol{V} \tag{2.11}$$

A dictionary obviously consists of more than a single key-value pair. Now that we have the means to represent single data records, we can represent dictionaries simply as *sets* of key-value pairs. That is, we construct our dictionary vector $\boldsymbol{D}$ as follows:

$$D = \sum_{i=0}^{N-1} R_i = \sum_{i=0}^{N-1} K_i \odot V_i \tag{2.12}$$

Looking up a value from this data structure can be accomplished once again by *unbinding* with the key vector $\boldsymbol{K_x}$ of interest which

[9]The similarity between the two record vectors would be lower than the similarity between each record vector and $\boldsymbol{V_1}$, but the ambiguity would still remain.

yields us a noisy version $\widetilde{\boldsymbol{V_x}}$ of the value vector which we feed into our associative memory (AM) for clean up.

An interesting property of this construction can be seen when we add another value V_y to our dictionary for a key K_x that already is present (bound to V_x) in our dictionary. Since binding distributes over addition and addition is approximately associative, it holds that

$$\begin{aligned}\boldsymbol{D'} &= \boldsymbol{D} + (\boldsymbol{K_x}, \boldsymbol{V_y}) \\ &= \boldsymbol{D} + \boldsymbol{K_x} \odot \boldsymbol{V_y} \\ &\approx (\boldsymbol{D''} + \boldsymbol{K_x} \odot \boldsymbol{V_x}) + \boldsymbol{K_x} \odot \boldsymbol{V_y} \\ &\approx \boldsymbol{D''} + \boldsymbol{K_x} \odot (\boldsymbol{V_x} + \boldsymbol{V_y})\end{aligned}$$

Simply speaking, our dictionary implicitly stored the set of $\boldsymbol{V_x}$ and $\boldsymbol{V_y}$ as the value bound to $\boldsymbol{K_x}$. This property of our compound hypervector makes it behave very similar to what software engineers would call a *hash table*.

2.3.4 Applications

Having introduced the mathematical framework of BSC it is time to discuss how these concepts are applied in practice. HDC has a wide range of applications ranging from classical inference tasks [162]–[166], self-supervised learning applications [167], over cognitive modelling [168], [169] to holistic execution of classical finite state machines [170], [171]. For the purpose of energy-efficient edge computing, we limit this introduction to inference applications yet similar concepts are used for self-supervised learning tasks.

Figure 2.5 illustrates the basic template of most VSA based inference algorithms found in the literature; At the beginning of our data processing pipeline, we start with low-dimensional input data, i.e., raw sensor data or preprocessed features thereof. The atomic elements, values, of this input data stream are mapped to item vectors in our D-dimensional BSC. The mapping to the high dimensional space of the input data should capture the desired relational properties of our input data. There are two main approaches to this mapping procedure:

Random Labeling Each possible input entity is assigned a random hyper vector. This is the approach we introduced in section

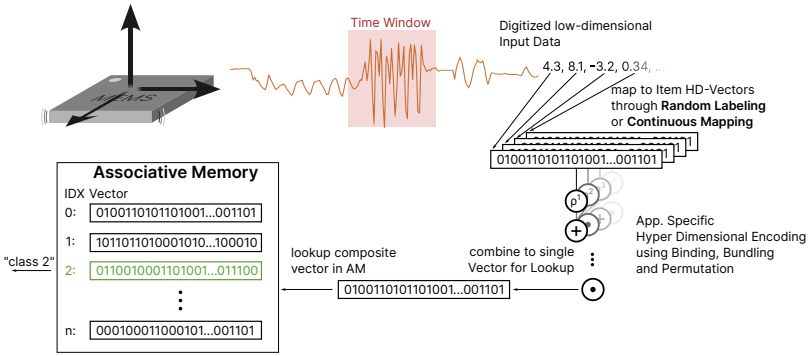


Figure 2.5 – Illustration of a VSA based inference pipeline.

2.3.1 and is mostly used to encode data labels. E.g. consider an application where we perform some form of inference on, among other features, the eye colour of people. In this case, we would assign each of the defined eye colour labels "blue", "green", "brown" and "grey" one random item vector. The same eye colour label is always mapped to the same item vector but vectors representing different eye colours do not share any similarities. In other words, the only *relational* property we are interested to capture in this example is discrimination, i.e. the ability to clearly distinguish between the eye colours.

Continuous Mapping Each possible number our sample could assume is assigned a random item vector. However, the magnitude of the difference between any two numbers in the input space correlates with the similarity of the corresponding item vectors in the high dimensional space. One way to achieve this property are so-called *thermometer codes* [165]. The idea is to first encode input values in the range $[min, max]$ to a thermometer-coded vector. That is, a vector with the number of '1's being proportional to the value ranging from all-zeroes to $D/2$ ones. E.g. consider the input being a 8-bit unsigned integer values that are supposed to be mapped to 1024-dimensional BSC. The corresponding thermometer code vectors would look as follows:

$$
\begin{aligned}
T_0 &= \overbrace{0000000000...0}^{1024} \\
T_1 &= 11\overbrace{00000000...0}^{1022} \\
T_2 &= 1111\overbrace{000000...0}^{1020} \\
&\vdots \\
T_{255} &= \overbrace{111...1}^{512}\overbrace{000...0}^{512}
\end{aligned}
$$

We then bind these thermometer codes with a random label vector $\boldsymbol{L}$ and scramble the bit locations with a static random permuation, meaning we initially choose a random permutation but once we have one, we keep using the same permutation for every encoding:

$$\boldsymbol{V_x} := \rho(\boldsymbol{L}\boldsymbol{T_x}) \tag{2.13}$$

The results are seemingly random item vectors but similar values in the input space are mapped to similar vectors in HD-space. The two ends of the value range are mapped to maximally uncorrelated vectors. There are techniques to extend this idea to circular data to encode e.g. angles, seasons of a year, positions or time of day where we want the two ends of our range to coincide in terms of similarity [172]. However the sample applications we will introduce in this work rely on *random labelling* and *continuous mapping* only.

The next step after item memory mapping is to combine several input items belonging to the same event of interest into a compound hyper-dimensional (HD)-data structure. Such an event of interest could be a time-window within a time series associated with a certain class e.g. vibration a pattern of a machine associated with mechanical failure. The crucial part of this step is to capture the essential characteristics like causality, time-translation invariance etc. in the compound vector.

This process is commonly called *hyperdimensional encoding* and relies on the experience and expertise of the algorithm designer to come up with a suitable encoding scheme. The result is a single HD-vector per labelled event.

During the training phase of the VSA-based classification algorithm, all compound vectors belonging to the same class are *bundled* together to create a so-called *prototype vector* that represents the set of the entire HD-encoded training data of that particular class. These prototype vectors are stored in an AM.

For inference, the same item mapping and encoding steps are repeated on the unseen input data. However, in contrast to the training phase, the compound vector is not used to update but to *lookup* the corresponding prototype vector in the AM.

Although quite simple, this basic HDC inference template achieves a quite competitive classification performance on small to moderately-sized problems like electromyography (EMG) gesture recognition, user activity recognition or language identification [173]. Three characteristics of the above algorithmic template are worth noting;

1. The classification algorithm depends crucially on the hyper-dimensional encoding algorithm used.
2. Training and Inference are almost symmetrical in terms of algorithm and computational complexity. The only difference is the very last step where we update the prototype vector during training using the bundling operation compared to an associative lookup operation used for inference
3. A single pass over the training data set is usually enough to achieve competitive results. This is in stark contrast to NN based algorithms that require numerous energy-consuming iterations over the dataset to converge to a local minimum in the loss function's gradient.

2.4 Summary

In this chapter, we introduced two alternative AI-driven data processing frameworks that are both suited for ultra-low power edge computing applications. QNN is the more established framework that sees more and more applications on energy-constrained devices in the wild. HDCs on the other hand has very interesting error resiliency and data parallelism properties which present it as a prime candidate for the next generation of *stochastic* computing platforms based on novel non-volatile memory technologies. However, we must emphasize once again that the problem classes which HDC is able to tackle with competitive performance are currently limited to smaller problems and the generalization capabilities of NN-based algorithms in scaled-up network topologies can still not be competed with. That being said it is worth noting that the research interest in HDC has only recently gained significant traction in the scientific community. There is still a lot of ground to cover whereas research on artificial neural networks is currently at an all-time peak with an orders of magnitude larger research community to fuel advancements.

Chapter 3

Energy Efficient Circuits for ULP Near-Sensor Processing

3.1 Introduction

After establishing the background on the hardware and software side for ULP always-on systems, this chapter introduces novel key building blocks for energy-proportional SoC architectures tailored to near-sensor computation that will be covered in chapter 4.

In section 3.2, we briefly introduce the common methodology used for our power, performance and area (PPA) analysis of the circuit blocks in isolation. We will then proceed with section 3.3 that proposes the elementary building blocks for energy-efficient hyper-dimensional computing using binary spatter code. In section 3.4, these building blocks are combined to form *Hypnos*, an all-digital ultra-low power autonomous HDC accelerator. After showcasing and comparing Hypnos' capabilities on a number of real-life benchmark applications, we culminate this section by introducing the silicon test vehicle *Rosetta* that combines Hypnos, a conventional microcontroller SoC architecture and various on-chip memory flavours on a single die.

Chapter 3 then proceeds with a lightweight data pre-processing circuit matched to Hypnos' low-dimensional data interface to enable truly autnomous end-to-end data processing. The circuit combines a micro-code configurable independent SPI sensor interface with a multi-channel data preprocessing pipeline.

Sections 3.3 and 3.4 have been published in IEEE Transactions on Circuits and Systems I [121][1], however subsection 3.3.2 has been extended significantly to contain a mathematical justification of the novel item vector mapping approach presented in the paper. Section 3.5 and 3.6 presents additional details and results on material that has been published in IEEE Solid-State Circuits and Letters [64] and the IEEE Journal of Solid-State Circuits respectively [84].

[1] ©2021 IEEE. Reprinted with permission, from M. Eggimann, „A 5 µW Standard Cell Memory-Based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing“, IEEE Transactions on Circuits and Systems I: Regular Papers, Bd. 68, Nr. 10, S. 4116–4128, Okt. 2021, doi: 10.1109/TCSI.2021.3100266.

3.2 Methodology

For the area and power analysis conducted on the basic circuit blocks in sections 3.3 to 3.6, unless otherwise indicated, we followed the subsequent methodology; the purely digital design written in SystemVerilog RTL was first synthesized with Synopsys Design Compiler using default settings for mapping effort. We evaluate the design's performance in two different target technologies: The first one is a 65 nm Low-Leakage Low-K process node using a high Vth (HVT) standard cell library to minimize cell leakage at low operating frequencies. If not denoted otherwise, all numbers were obtained with the typical case library characterization at 1.0 V, 25 °C. The second technology we targeted is a 22nm FDSOI node using a UHVT and SLVT library. The library characterization at 0.8 V, 25 °C without body biasing at the typical-typical corner was used. Using Cadence Innovus, we performed place and route with an eight-layer metal stack for the 65 nm node targeting a core area utilization of 80%. For the 22 nm node, a ten-layer metal stack with a target core area utilization of 70% was used. Post-layout power numbers were obtained with Cadence Voltus using switching activity for all internal nodes extracted from a timing back-annotated post-layout simulation of the benchmark algorithms using Siemens Questasim.

3.3 Building Blocks for Energy-Efficient HDC

3.3.1 Fully-Synthesizable Associative Memory

For a given search vector, the AM looks up the most similar vector currently stored within the memory. However, the obvious approach of combining traditional SRAMs to store the HD-vectors with digital logic yields suboptimal results. Although SRAMs are the go-to solution for fast and area-efficient volatile on-chip memory, conventional SRAM macro generators are not optimized for the extremely wide memory aspect ratios needed for parallel access to HD-vectors. Also, they are less energy-efficient under low V_{DD} conditions for low bandwidth applications [68], [174]. The nature of hyperdimensional computing with lots of simple, component-wise operations demands a non-von Neumann scheme of computation with computational logic intermixed with memory cells.

Using Latch Cells as Memory Primitives

Figure 3.1 shows the structure of the AM in our design; latch cells are used as primitive memory elements instead of flip-flops due to their lower area (-10%) and energy (-20%) footprint [174]. Each row of the memory consists of D/K latch cells and a single glitch-free clock gate. These row clock gates are activated by the one-hot encoded write address. A two-port design allows fetching a new HD-Vector from AM into the HD-encoder while simultaneously writing back the previous result without any stalls or energy-costly pipeline registers in the wide datapath. In addition, to read and write access of complete rows for normal HDC algorithm operation, each row is further divided into subwords of 32-bit to allow sequential read-out and programming of prototype vectors by an external device through a 32-bit APB configuration interface.

In most HDC-based classification schemes, the AM only keeps hold of the prototype vectors representing the individual classes. The proposed architecture differs in that regard by using rows of the AM to store the iterative encoding process's intermediate results. The AM

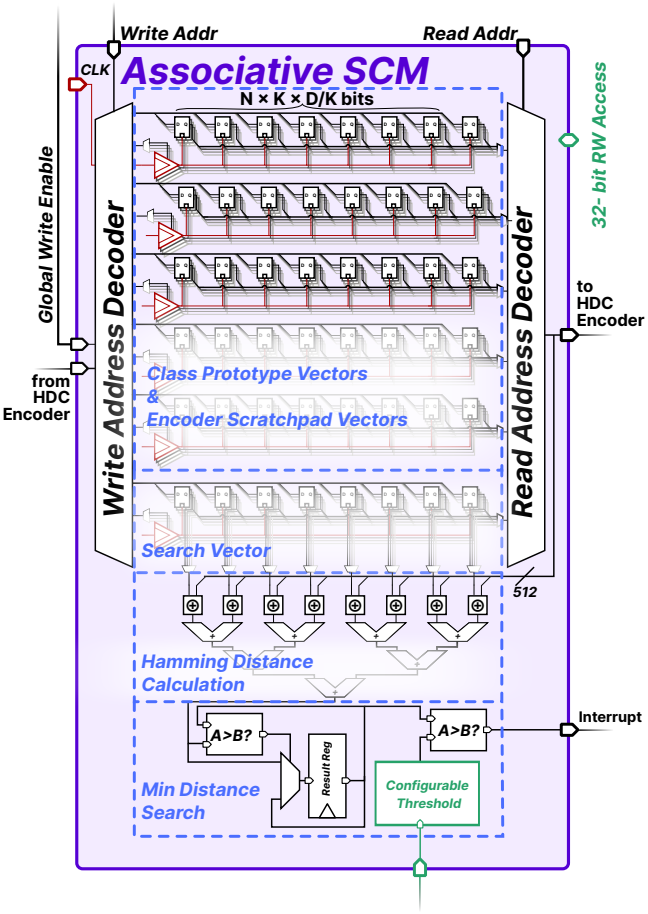


Figure 3.1 – Architecture of the latch-cell based all-digital AM. Vectors can be read and written simultaneously in sub rows of length D/K. The last vector within the memory acts as the search vector for the associative lookup logic. All memory rows share the D/K-bit adder tree for the pop count operation. The distance of the most similar entry is compared with a configurable threshold and conditionally raises an interrupt line to an external peripheral (e.g. power management unit in an SoC)

thus serves the double purpose of a register file for entire HD-vectors (or vector subparts in case *vector fold* $K > 1$).

Although latch cells drastically reduce the impact on area footprint compared to flip-flops, their usage can complicate static timing analysis (STA). Due to their transparent nature during write access, one must take care not to introduce combinational loops. While Teman *et al.* suggest decoupling the memory by using flip-flops at the IO boundary of the memory [68], we repurposed the output register in the encoder stage to break combinational loops. This approach, coupled with multi-cycle path constraints for STA [68], allows treating the AM like a regular flip-flop-based synchronous design during synthesis.

Associative Lookup Logic

There are several possibilities on how to design the digital logic for associative lookup. One option we considered was to have a fully parallel architecture with XOR-gates between the search vector and each memory row, combinational adder trees, and comparators to combinationally determine the entry of minimum Hamming distance. However, as we will see in Section 3.3.1, the overhead in area and leakage power ultimately lead us to go for an iterative approach. As shown in figure 3.1, the HD-vector slot acts as the search vector in the proposed architecture. The lookup logic iterates over each memory row, calculating the Hamming distance between one subpart of the search vector and a subpart of one of the stored HD vectors at a time. The control logic accumulates the Hamming distance between the subparts and iteratively determines the most similar entry's index and distance.

Energy and Area overhead Analysis of SCM-based AMs

Table 3.1 provides an evaluation of the area overhead and energy efficiency for a fully combinational and the row-sequential AM architecture described in Section 3.3.1. To get an accurate estimate of the delay and power consumption at sub-nominal voltages, the complete standard cell library was recharacterized with spice simulations using Cadence Liberate for a V_{DD} corner of 0.6 V. At this voltage, all standard cells within the library are still operational in spice simulation.

6T-bit cell-based SRAMs that are readily available in all commercial

	Area [kGE]	Throughput [MOPS/s]		Energy Efficiency [pJ/lookup]		Leakage Power [µW]	
		@1.2 V	@0.6 V	@1.2 V	@0.6 V	@1.2 V	@0.6 V
SRAM + Digital AM	17	2.56	—	3280	—	1.5	—
Sequential SCM AM	101	1.29	0.23	2353	556	7.5	1.7
Full parallel SCM AM	265	13.80	1.54	921	188	81.0	15.1

Table 3.1 – Area and Energy Efficiency comparison of SCM-based 128 by 128 bit AM- and SRAM-based AM-Architecture in 65nm technology using all three available VT flavors. The most energy efficient SRAM configuration generated by the available SRAM macro generator collection for the target technology was chosen.

technology nodes are no longer operational at such low voltages[174], [175]. Although there are specialized low-voltage SRAMs for sub-threshold operation [176], they are custom-tailored for a particular technology and not readily available for all technology nodes. Furthermore, experiments by Andersson *et al.* indicate that customized SCMs can still have an energy advantage over sub-threshold SRAMs for small memory sizes [67].

At the 0.6V operating corner, we see a 4× improvement in energy efficiency for the sequential architecture and almost 5× for the fully parallel version compared to operation at nominal voltage. The full-parallel implementation is 2.6x more energy efficient than the sequential one. However, for most HDC algorithms, the vast majority of the proposed HDC accelerator's compute time is spent on vector encoding, during which the AM lookup logic stays idle. For this reason, we focus on the row-sequential SCM AM architecture, which has a better trade-off between energy efficiency during lookup operation and static leakage power in the subsequent analysis.

3.3.2 HDC Encoder Units and Item Materialization

The first step of every HDC classification algorithm is mapping a dense input space to a high-dimensional holistic representation. Most algorithms encode the input data into a single high-dimensional search vector. The search vector is then compared with prototype vectors stored in the AM representing the different classes. The differences between the various HDC algorithms mainly lay in the

particular encoding algorithms. They are crafted to capture relevant characteristics from the raw data, e.g., amplitude distribution, spatial or temporal features, and are highly application dependent.

Figure 3.2 illustrates our proposed encoder architecture. It consists of three main components connected in a combinational pipeline. The input stage of the encoder multiplexes between 4 different input sources; the all-zeros vectors, a hardwired random seed vector, a vector addressed from AM, or the HD-encoder's own output. The IM materialization stage maps input data to item vectors using either quasi-orthogonal vectors (IM) or continuous item mapping (CIM). The encoder's bitwise encoder units perform binary or unary operations on the individual bits of the vectors.

There are no pipeline registers in the very wide datapath between the encoder stages. Although this design choice reduces throughput, it increases the energy efficiency of our architecture.

Encoder Units

The Encoder Unit processes one dimension of the input vector. Besides the combinational logic for the binary and unary bitwise operations, each unit contains an output Flip-Flop that stores the result after each encoding cycle.

Additionally, one saturating bidirectional 5-bit counter per unit performs the bundling operations. Analyses in [161] showed that for dimensions up to 10000, a 5-bit saturating counter implementation still achieves the same bundling capacity as a full precision model. To avoid any unintended similarity between the bundled vectors of disjunct input sets, ties during the bundling operation, i.e., the same number of ones and zeros were counted for the given vector component, must be broken without introducing a bias towards a constant HD-vector. Our architecture achieves this by always bundling an odd number of HD-vectors, if necessary adding one additional vector crafted from the xor-combination of the inputs.

A noteworthy detail of the saturating counter is its possibility to evict the current counter value to the AM in a bit-serial manner (i.e., one cycle for each of the five counter bits). Eviction and loading of the counter state allow the proposed design to execute multi-stage encoding algorithms with nested bundling operations.

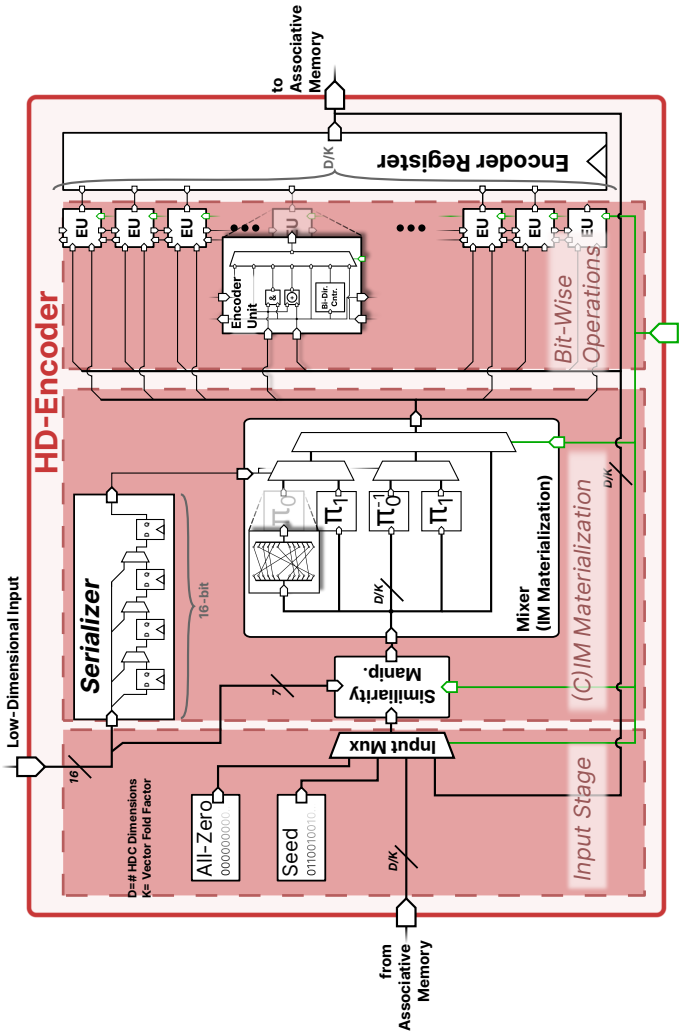


Figure 3.2 – Architecture of the HDC Encoder responsible for (Continouus) Item Memory materialization and search vector encoding. The width of the datapath is a function of the HDC dimensionality (D) and the design parameter K (discussed in Section 3.3.2).

Mixing Stage

The Mixer submodule visualized in figure 3.2 generates quasi-orthogonal pseudo-random HD vectors. The rematerialization, i.e., on-the-fly regeneration, of such vectors is an area-efficient alternative to explicit storage of large numbers of item vectors required for input to HD space mapping.

The mixer stage feeds the input vector selected by the encoder input stage through one of two hardwired random permutations π_0 and π_1. The encoder maps a given low-dimensional binary input datum w from the input domain $\mathbb{D}$ to the pseudo-random HD-vector V_w by iteratively applying one of the two permutations π_0 and π_1 to a hardwired seed vector S:

$$V_w = \prod_{k=0}^{n} \pi_i S, \text{ for } i = \begin{cases} 0 & \text{, if } w_k = 0 \\ 1 & \text{, if } w_k = 1 \end{cases} \tag{3.1}$$

where w_k denotes the k^{th} bit position in the input word w's binary representation and $n = \log_2 |\mathbb{D}|$.

Iteratively applying two random permutations as an alternative to storing actually random item vectors obviously requires some mathematical analysis of the quality of randomness produced by this approach. Let us assume for a moment that we are using a *different* random permutation π_w for every possible input value w. Intuitively, if we assume the seed vector S to be random with hamming weight W_H, a truly random permutation could yield us any possible D-bit vector with the same hamming weight H_S. Given the fact that we aim for i.i.d. distributed vector dimensions, in other words, vectors with a hamming weight of $H_S \approx D/2$ we are not too much concerned about the fact that the hamming weight cannot change. The target space $\mathbb{V}$ our truly random permutations can produce given our seed vector is still very vast with cardinality,

$$|\mathbb{V}| = \binom{D}{H_S} \tag{3.2}$$

However, our proposed mixing stage does not use a different random permutation for every w (the hardware overhead of this would be enormous) but instead uses two very cheap hardwired random

permutations iteratively. The mathematical question we are thus trying to answer is:

Given two random permutations π_0 and π_1 and a random input word w, how well does the algorithm outlined in equation (3.1) approximate the use of unique random permutation per item w?

To answer this question, it is useful to look at our problem from a different angle. We will thus conduct a brief excursion to the realm of abstract algebra:

A mathematical group is defined as a set of elements for which there is a binary operation, let's denote it with "$\cdot$", that fullfils the following properties, also known as the *group axioms*:

- The group is closed under the group operation:

$$\forall x, y \in G : x \cdot y \in G \tag{3.3}$$

- The group operation is associative i.e.

$$\forall x, y, z \in G : x \cdot (y \cdot z) = (x \cdot y) \cdot z \tag{3.4}$$

- The group contains an identity element:

$$\forall x \in G, \exists e \in G \text{ such that } e \cdot x = x \tag{3.5}$$

- There is a unique inverse element e for every group member:

$$\forall x \in G, \exists x^{-1} \in G \text{ such that } x \cdot x^{-1} = e \tag{3.6}$$

The set of all possible n-element permutation forms a mathematical *group* with the group operation being composition, i.e. chaining of permutations. This particular group is commonly called the *finite symmetric group* of size n, denoted S_n and is fundamental to numerous mathematics areas. An important aspect of any group is its so-called *generators*. A generator of a group G is any set of elements in G that, by arbitrarily combining its elements using the group operation, can form the entirety of G. In other words, given just the elements of

the generating set, we can form any possible element of G using only generator elements and the group operation.

Now that we are equipped with the bare minimum of the basic terminology around group theory for our cause let us reformulate the previous question in group theoretical terms:

Given two random elements π_0 and π_1 of the finite symmetric group S_n, what is the likelihood of them forming a generating set of S_n?

For this question, there is an answer known as Dixon's theorem from 1969, which states that for large n, this probability approaches 1, given we use at least one permutation with odd parity [177]. The parity of a given permutation is defined as the parity of the number of transpositions, i.e. swap operations. Although there are numerous possibilities to decompose a permutation into transpositions, the parity will always be the same. Babai provides us with a lower bound on this probability, improving on Dixon's work:

Theorem 1 (Babai [178]) *The probability that a pair of random permutations generate a primitive subgroup other than the alternating set A_n or the group S_n is less than $n^{\sqrt{n}}/n!$ for large n.*

If we choose one of the two permutations to have odd parity, we are guaranteed not to obtain the alternating set (the set of all even permutations in S_n). Therefore, our approach of approximating an arbitrary random permutation by iterative application of two fixed random permutations has a very high probability of producing the same quality of results[2]

The mixing stage not only serves the purpose of enabling *random labelling* item vector mapping but also provides the hardware primitive to perform the *permutation* operation of BSC.

The resulting HD-vectors V_w are all quasi-orthogonal with very high probability, given that π_0 and π_1 do not commute.

For algorithms that require random access to the item memory, the scheme presented in equation (3.1) rematerializes the item vector with time complexity $\mathcal{O}(\log_2 |\mathbb{D}|)$. However, many algorithms use

[2]Already for n = 100, the probability of this not being the case is less than 10^{-138}.

IM-mapping to bind a value vector V_{value} to a channel label $V_{chn[k]}$. In these scenarios, the channel label vectors are used with a fixed ordering assuming the raw data is fed to the accelerator using a fixed channel ordering. We can therefore reduce the time complexity to $\mathcal{O}(1)$ with the mapping:

$$V_{chn[k]} = \begin{cases} S & \text{if } k = 0 \\ \pi_0 V_{chn[k-1]} & \text{if } k > 0 \end{cases} \tag{3.7}$$

where we store the channel label from the previous iteration in an unused row of the associative memory.

Our proposed IM-mapping approach is more area-efficient than storing random vectors in a large ROM and scales well to large input domains whose cardinality is unknown in advance. From a hardware perspective, the mixer stage translates to N 4-input and N 2-input multiplexers, where N denotes the datapath width and some moderate wiring overhead caused by the random permutations.

Vector Folding

With default parameters, the proposed HD-Accelerator contains a datapath width equal to the size of a whole HD-Vector. However, as will be analyzed in more detail in section 3.4.5, going for a more parallel architecture does not always yield the most energy-efficient design for a given target technology. Thus, in addition to other design parameters, the RTL exposes the *Vector Fold* parameter; it allows tuning the design for the optimal amount of parallelism to improve energy efficiency. Increasing the value of *the vector fold* splits a single D-dimensional vector into K smaller subparts of equal size. The datapath of the accelerator shrinks accordingly and only processes one subpart at a time. While the throughput of the accelerator at constant frequency decreases by K, the area of the HD-Encoder, dominated by the saturating counters, reduces similarly by a factor of K.

An important detail is that by decreasing the datapath width, we also reduce the permutations' operand size within the similarity manipulator and the mixing stage. If we stick with the same IM-Mapping scheme described above, all subparts of a vector would be identical since they all pass through the same hardwired permutations

π_0 and π_1 of size $\frac{D}{K}$. The IM mapping scheme in equation (3.1) is thus modified as follows:

$$V_w^* = \pi_{\texttt{idx}} V_W \tag{3.8}$$

with,

$$\pi_{\texttt{idx}} = \prod_{k=0}^{log_2(j)} \pi_i, \text{ for } i = \begin{cases} 0 & \text{, if } h_k = 0 \\ 1 & \text{, if } h_k = 1 \end{cases} \tag{3.9}$$

where h is the value of the dedicated *part index counter*, which holds the index of the current vector subpart. This yields a unique set of permutations π_o^*, π_1^* per vector part at the expense of $\mathcal{O}(log((K))$ additional mixing cycles.

Similarity Manipulator

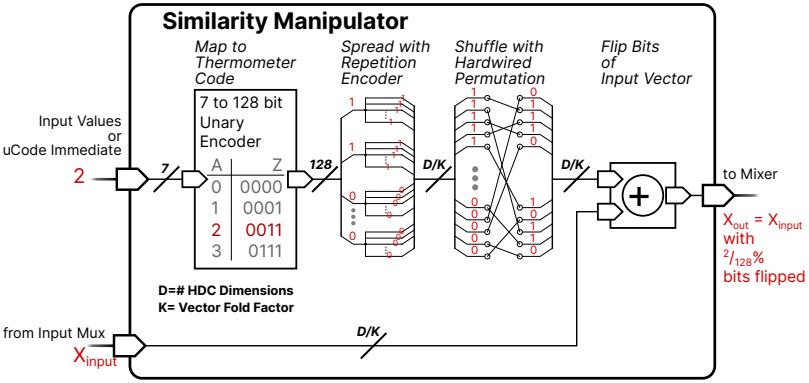


Figure 3.3 – Structure of the Similarity Manipulator stage

The Similarity Manipulator stage transforms the mixing stage's output vector by flipping a configurable number of its bits. This operation is a fundamental building block of various high-level operations like *binarized B2B bundling* [161], CIM mapping [165] and exponential forgetting. Figure 3.3 shows its internal structure; the 7-bit input word w is first mapped to a 128-bit unary representation w_{unary}. This unary representation is spread to the target HD-vector

dimensionality D/K by repeating each bit of w_{unary} $\frac{D}{K \times 128}$ times. The resulting vector passes through a hard-wired random permutation to distribute the 'ones' over all the vector dimensions. The result is XOR-ed with the input vector. A limitation of the proposed solution is that a uniform distribution of the input words does not yield equal distribution of the probabilities for a bit to be set across the HD-Vector's input dimensions. A multi-cycle approach can be used for operations where equal bit-flipping probability is a hard requirement; first, a bitmask with the desired bit-density is generated by passing the all-zero vector through the manipulator stage with the input word w. This mask is subsequently mixed in the Mixing stage using the same input word w to randomize the position of the 'ones' in the bitmask. The resulting bitmask is ultimately XOR-ed with the input HD-vector within the encoder units.

3.4 Hypnos: An Energy Proportional Standard Cell Memory based HDC Accelerator

3.4.1 Introduction

Energy boundedness is the key design metric and constraint in the development of internet-of-things (IoT) devices [179]–[181]. With more and more sensor modalities integrated into IoT end nodes, the amount of data to process and the complexity of the processing pipeline increases. Aiming for uninterrupted operation for years or even indefinitely within the tight power envelope of small batteries or environmental energy harvesting urges to drastically reduce the average power consumption of the sensor nodes themselves.

Among applications in the development of new consumer electronic devices in the form of wearable fitness trackers or Smart Home Systems, a considerable interest in the development of highly energy-efficient sensor nodes is seen in industry as part of the shift towards industry 4.0 e.g. in the form of asset tracking systems and machine performance and condition monitoring systems [5].

The demand for increasing functionality in small form-factor battery-operated, or even fully self-sustainable devices leads to a multitude of design challenges for engineers to tackle. Probably the most apparent one in that regard is the need for significant improvements in the energy efficiency of said devices. Observing that the majority of power consumption in today's wireless sensor devices is spent in data transmission [6] promotes moving data processing closer to the sensor. Instead of raw data transmission and centralized processing in the cloud, the data is processed continuously on these so-called *smart sensor* devices [11]. Only the analyzed portion of the information is transmitted (e.g., transmission of a single `imminent machine failure` message instead of the raw vibration and temperature data). This may not be easily achieved by application-specific integrated circuit (ASIC) designs because *general-purpose* always-on smart sensing systems operate in the µW range and also demand a programmable fabric. Therefore, the next evolution step

towards fully self-sustainable always-on smart sensors requires the exploration of new avenues of hardware-software co-design and outside the realm of traditional von Neumann-based computing [153], [182].

An energy proportional sensor data processing scheme, where a wake-up circuit (WuC) detects patterns of interest and aggressively duty cycles other circuitry, is a viable solution to drastically reduce average power consumption [183], [184]. While there are numerous WuCs, e.g., for biosignal anomaly detection, sound/keyword spotting, incoming radio transmissions in the µW range, all of these solutions are highly application-specific. Considering the cost of custom silicon development and the rapidly widening range of application targets, there is a need for configurable and application-agnostic WuCs with more flexible pattern extraction capabilities than the simple threshold-based solutions, which can suffer from high false-positive rate and thus energy losses of unnecessary wake-ups.

In this section, we present Hypnos, a novel flexible and highly energy-efficient all-digital HDC architecture for always-on smart sensing applications achieving up to 3× higher energy efficiency (191 nJ/inference) over the state-of-the-art (SoA). The architecture relies on the hardware-friendly embodiments of common HDC operators introduced in section 3.3 resulting in 3.3× technology scaled area reduction. Using an all-digital approach enables us to publicly release our architecture under the permissive Solderpad open-source license[3]. The section continues with a case study on our approach using three different HDC applications, including the first investigation (to the best of our knowledge) on the feasibility of HDC for the task of ball-bearing fault detection. Finally, we introduce *Rosetta*, a prototype implementation of the concepts introduced in this section taped out in TSMC 65nm technology.

3.4.2 Related Work

Tackling the power-consumption challenge of always-on sensing in a hierarchical manner using WuCs to apply aggressive duty cycling on more involved data processing modules is not a new idea. In the recent past, there have been several publications on low-power always-on

[3]Available under https://github.com/pulp-platform/hypnos

	Application Specific					*General-Purpose*		
	Giraldo *et al.* [185]	Shan *et al.* [186]	Zhao *et al.* [187]	Wang *et al.* [188]	Miro-Panades *et al.* [183]	Cho *et al.* [189]*	Rovere *et al.* [18]	**This Work**
Applications	Keyword Spott.	Keyword Spott.	EMG	Slope Matching	Wake-up Radio, Interrupts	General-Purpose	General-Purpose	General-Purpose
Technology	65nm	28nm	180nm	180nm	28nm	180nm	130nm	65nm / 22nm
Cross Tech.	Medium	High	Low	Low	Low	High	Low	High
Power Envelope	~10 µW	~500 nW	~13 µW	~75 nW	~33 µW	~14 µW	~2.2 µW	max. ~25 µW, typ. ~5 µW
Classification Scheme	MFCC, LSTM	DSCNN	ANN	Threshold, Slope	-	NN	Threshold Sequence	HDC
Configurability	App. Specific	App. Specific	App. Specific	Limited	App. Specific	High	Medium	High
Area	2.56 mm^2	0.23 mm^2	0.925 mm^2	2.5 mm^2	4.5 mm^2	15.6 mm^2	0.054 mm^2	1.43 mm^2 / 0.29 mm^2

Table 3.2 – Comparison of state-of-the-art WuCs with our proposed HDC-based WuC. Area numbers are reported 65nm and 22nm technology while power consumption is reported in 22nm for a compute-intensive language classification algorithm and a typical always-on classification algorithm for EMG data. For Cho *et al.* only the neural network processor without application specific VAD circuitry is considered.

wake-up circuitry in various domains. Table 3.2 gives an exemplary overview of current general-purpose wake-up circuitry research using selected publications in the recent past. The architectures are labeled as *application-specific* if they are designed to support only one particular kind of smart-sensing application and as *general-purpose* if their respective classification scheme is configurable to support various kinds of classification problems within the limits of their classification scheme.

Keyword spotting and voice activity detection (VAD) is a very actively researched target for always-on sensing; Giraldo *et al.* present a low power WuC for speech detection, speaker identification, and keyword spotting with integrated preprocessing blocks for MFCC generation and LSTM accelerator for classification [185]. Shan *et al.* proposed another implementation in the same application domain with state-of-the-art energy efficiency on the task of two-word keyword spotting using binarized depth-wise separable CNN's operating at near-threshold [186]. At the lower end of the power consumption spectrum Cho *et al.* present a 142 nW VAD circuitry with integrated analog-frontend that combines a configurable always-on time-interleaved mixer architecture with a heavily duty-cycled neural-network processor [189]. Although their analog input stage is highly specific to VAD only, the integrated 14 µW digital neural network processor could potentially be repurposed for other applications and we thus assume it to be general-purpose.

Monitoring life signals is another very active field; in the context of cardiac arrhythmia detection, Zhao *et al.* combine a level-crossing ADC with asynchronous QRS-complex detection circuitry with an artificial neural network accelerator to benefit from the energy advantage of non-Nyquist sampling [187]. Although these NN-based solutions achieve high accuracy at outstanding energy efficiency in their particular application domain, they are often hardwired for the respective task and do not support online training.

More in line with the target of a flexible and configurable smart sensing platform are Miro-Panades *et al.*; they present an asynchronous RISC processor with an integrated wake-up radio receiver for efficient low-latency wake-up from several external and internal triggers. While their architecture achieves outstanding reaction time to interrupts without the need for a high-frequency clock, the wake-up circuitry lacks the interface and compute capability to perform actual data

processing for data input pattern dependent wake-up [183]. Wang *et al.* present a configurable WuC resembling the work in [187] that combines an LC-ADC with a set of asynchronous detector blocks to extract low-level signal properties like peak amplitude, slope, or time interval between peaks. Each detector can be configured with a threshold, and the individual detector output can be logically fused to a single wake up signal. Although their architecture uses minimal power, continuous detection of more complex patterns is entirely outside the capabilities of a detector-set approach [188].

To the authors' knowledge, the only low power WuC with slightly more sophisticated pattern matching capabilities was introduced by Rovere *et al.* Instead of analyzing the delta-encoded signal from the LC-ADC with hardwired detectors, they continuously match the input signal against a sequence of upper and lower amplitude thresholds with up to 16 threshold segments. This scheme equates to matching the input signal's approximate amplitude slope against a configurable pre-trained prototypical signal slope of interest [18]. Their approach proved successful for pathological ECG classification and binary hand gesture recognition (finger-snap or hand clapping). Still, detecting more complex patterns in the spatial or time dimension remains outside their proposed architecture's scope.

Hyperdimensional computing (HDC) is an energy-efficient and flexible computing paradigm for near-sensor classification that gracefully degrades in the presence of bit errors, and noise [149], [165], [173]. Various works showcased HDC's few-shot learning properties and energy efficiency in multiple domains like biosignal processing [163], language recognition [190], DNA sequencing [167], or vehicle type classification [191].

In emerging hardware implementations, the HDC's inherent error-resiliency is leveraged for novel non-volatile memory (NVM) based in-memory computing architectures [38], [153], [192]. Targeting FPGAs, efficient mappings of binary and bipolar HDC operations are proposed [161], [193], [194]. However, the only complete digital CMOS-based HDC accelerator was recently introduced by Datta *et al.* They propose a data processing unit (DPU) based encoder design that interconnects with a ROM-based item memory, and a fully parallel associative memory [195]. While their implementation indeed excels in throughput, its' configurability as well as area- and energy efficiency

are limited; their encoder architecture is restricted to what they call *generic* multi-stage HDC algorithms with a hardwired encoder depth in feedforward configuration imposing hard limits on the supported encoding schemes. From an energy efficiency and area standpoint, their design suffers a lot from using a large read-only-memory (ROM) for item memory (IM) and pipeline registers in the very wide datapath of every encoding layer.

Our proposed architecture targets the sub 25µW power envelope (resulting in a lifetime of about four years from a small lithium-thionyl chloride coin cell battery). The always-on smart sensing circuitry leverages the flexibility of HDC to perform energy-efficient end-to-end classification on a diverse set of input signals. We achieve higher configurability, a reduction of 3.1× in area and up to 3.3× improvement in energy efficiency than the current SoA in HDC acceleration and present a first-in-class flexible and technology agnostic digital CMOS architecture for near sensor smart sensing wake-up circuitry.

3.4.3 Architecture Overview

Figure 3.4 illustrates the three major components of the accelerator, which we describe in detail in the following subsections; the **associative memory** (AM) stores the prototype vectors and performs the associative lookup operations, the final step of most HDC algorithms. The previously introduced *hyperdimensional encoder* (HD-Encoder, section 3.3) is responsible for mapping low-dimensional input values to HD-vectors. It operates on HD vectors from the AM or its own output in an iterative manner. The AM and HD-encoder are managed by a small controller circuit that sequentially consumes a stream of compact microcode instructions and accordingly reconfigures the datapath. A tiny user-programmable configuration memory supplies this microcode stream.

3.4.4 An ISA for HD-Computing

Previously proposed HDC accelerator designs hardwired large portions of their datapath to execute HD-algorithms of a particular structure [165]. HD-algorithms that do not fit into this rigid algorithmic skeleton can not be executed by existing accelerators like [153], [195]. On

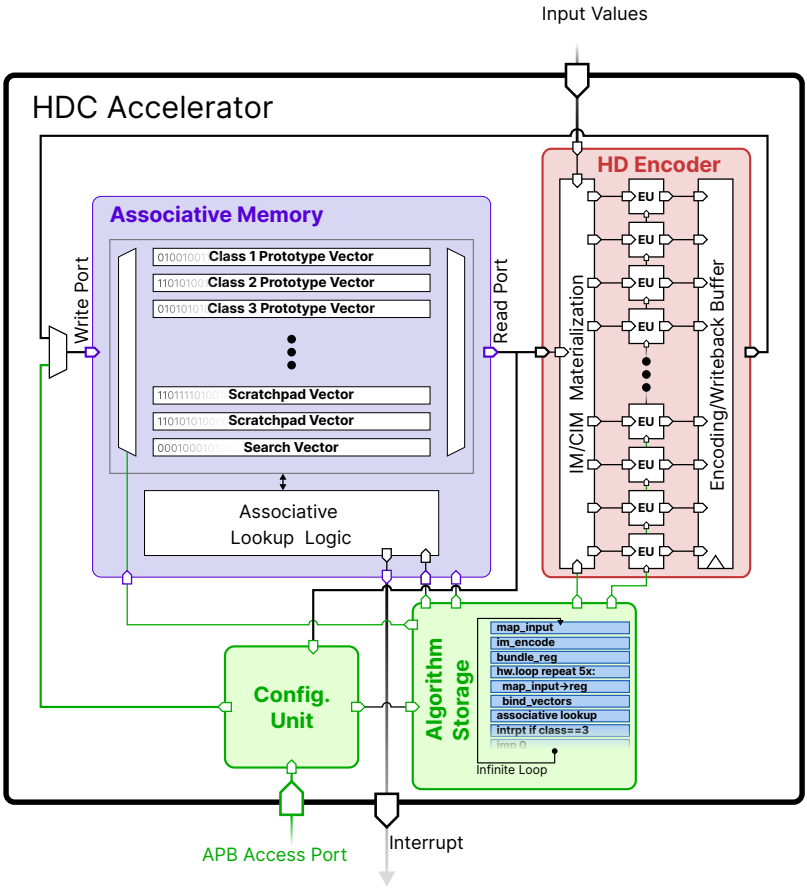


Figure 3.4 – High-level structure of the proposed HDC accelerator. The associative memory (blue) is responsible for storage and associative lookup of prototype vectors and serves as a scratchpad memory for the HD Encoder (red). Encoder and associative memory are orchestrated by user-programmable algorithm storage (green).

the other hand, the architecture we are proposing is not bound to execute only one specific class of algorithms. A control unit continuously reconfigures the datapath according to a stream of microcode instructions fetched from a tiny embedded configuration memory. This allows the accelerator to be reconfigured at runtime to execute algorithms of a much larger variety by altering the microcode stored in the configuration memory. After configuration, the algorithm is executed autonomously without any further interaction with a host processor. We propose a 26-bit instruction set architecture (ISA) with the encoding space split into 25-bit low-level datapath configuration instructions and 25-bit complex high-level instructions.

Although this structure bears some resemblance with a conventional processor, there are some key differences: The amount of area and energy overhead induced by the control path is less than 1% (16kBit CAM, 65nm) and thus much lower compared to a conventional core. Also, with only 18 available instructions in total, the microcode does not aim to be Turing complete. We introduce just enough configurability to support the execution of HDC operations in arbitrary order.

Low-level Instructions

The Low-level instructions directly encode the select signals of the multiplexers within the HD-encoder and the address lines of the HD-memory. They thus control the transformation and data flow within the HD-encoder unit during input data encoding. Figure 3.5 summarizes the function of the bitfields with a single 25-bit low-level instruction. They provide fine-grained control over the datapath with the RIDX and WIDX fields acting like source and destination register operands in a conventional ISA. However, since the Encoder unit contains an output Flip-Flop, many vector transformation operations can be performed without AM access using feedback.

If we synthesize the architecture with a *Vector Fold* parameter larger than 1, all instructions only process a smaller subpart of the complete HD-vector. The control unit does not transparently iterate over all subparts of the vector but leaves control to the user through the *part index counter*. The counter's value is automatically appended to the read- and write-port address lines of the AM and thus controls which subpart of the HD-vector is affected by the current instruction.

The counter can be cleared, increased, and decreased with dedicated instructions.

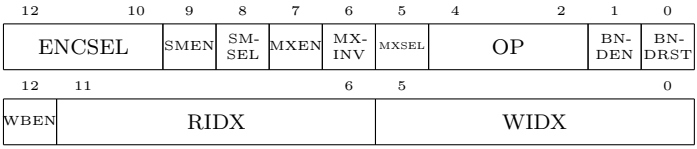
The rationale behind leaving control over the subpart iteration scheme to the user is that we also want to support iteration over the vector parts in the outermost loop of an HD-algorithm instead of only iterating in the innermost loop. That is, instead of first applying a transformation on all subparts of a vector before switching to the next transformation, we want the possibility to apply all operations of an HD-encoding algorithm on the first subpart and repeat the whole algorithm for subsequent subparts. For the first iteration scheme, we would have to swap the bundling counters' state after every bundling operation since we do not have individual counters for each vector part. The second iteration scheme does not require state eviction but requires multiple iterations over the input stream. The fine-grained control over the accessing scheme exposed by the HDC ISA also allows to adopt a hybrid accessing scheme; E.g., in an algorithm consuming a time series of multi-channelled data, the channel values could be iterated multiple times to improve the performance while complete time samples are accessed a single time only to remove the requirement for large external buffer memories.

High-level Instructions

The high-level instructions encode multi-cycle HDC operations and instructions for code size reduction and host interaction.

High-level HDC Operations For several HDC transformations, there are dedicated multi-cycle instructions; The *AM_SEARCH* instruction starts the associative lookup procedure within the AM. The vector currently stored at the highest index is used as the search vector. As its only operand, the instruction takes an immediate that limits the search space to a maximum index. Only vectors stored at an index smaller than the given maximum are considered during the lookup operation. The immediate value thus allows partitioning the AM dynamically into scratchpad and prototype memory.

The *MIX* instruction applies multiple mixing cycles to the current content of the encoder register and hence is the basis of IM-mapping. The mixing value is either an immediate or externally supplied input



ENCSEL	Select between the all-zero vector, a vector from AM and the current HD-encoder output as input for the encoder stage
SMEN	Enable/Bypass Similarity Manipulator Stage
SMSEL	Select between external input data and internal register as input for similarity manipulator stage
MXEN	Enable/Bypass Mixing Stage
MXINV	Select Inverse Permutation set in Mixing Stage
MXSEL	Select between permutation π_0 and π_1 or if MXINV is set between π_0^{-1} and π_1^{-1}.
OP	Select operations to be performed in Encoder Units.
BNDEN	Enable bundle counter thus bundling the current encoder output.
BNDRST	Reset the bundle counter to its initial value
WBEN	Enable write back of the encoder output to AM at index WIDX. If disabled HD-encoder is only stored in output buffer.
RIDX	Read index in case vector from AM is used as encoder input.
WIDX	Write index if the result of current iteration is written back to AM (WBEN = 1).

Figure 3.5 – Low-level Instruction format

data, e.g. from an external sensor. The instruction comes in three flavours that use different input values for the mixing procedure (see equation (3.1)).

- The input value can be supplied from the external 16-bit input signal. In this case, the instruction accepts the desired number of mixing rounds as immediate, starting with the LSB of the input to select the permutation of the first round.
- Alternatively, the mixing value can also be supplied as a second immediate operand. Again, the number of cycles a single *MIX* operation requires to complete directly correlates to the number of desired mixing rounds.
- The third possible source for the mixing value is the *part index counter*. This solves the problem explained in Section 3.3.2. Applying this instruction after every IM-mapping instruction has the effect of producing different subpart vectors given the same input value.

Host interaction and Code Size Reduction An autonomous WuC requires to conditionally signal a target system about the result of the classification algorithm. The proposed design uses a dedicated interrupt instruction to conditionally (or unconditionally) assert an interrupt signal line. The instruction has two operands:

- *Similarity Threshold* - The interrupt is not raised if the last associative lookup operation yielded a result with a Hamming distance higher than the given value.
- *Index Threshold* - The interrupt signal is not raised if the index of the most similar vector found in the last associative lookup operation is higher than the given threshold.

One use case of these thresholds is to wake up the target system only if the HDC classification algorithm detects one particular class with a certainty above a specific threshold.

For the architecture to be autonomous and energy-efficient, the amount of memory required to map a given HD algorithm to the proposed ISA must be kept small. Since most HDC algorithms only

consist of nested loops of constant iteration count (e.g., iterating over channels and time samples within a time window), a significant amount of code size reduction can be achieved with the introduction of hardware loops and unconditional jumps. Hardware loops, also known as "Zero-overhead hardware loops" are a commonly encountered hardware feature in digital signal processors (DSPs) that allow the execution of a small loop body for a constant number of iterations without the overhead of reevaluating the loop condition after every iteration.

As demonstrated by Gautschi *et al.*, they provide a significant performance boost with only a marginal increase in area given the requirement that the instructions of the loop body can be fetched in a single cycle [196]. This requirement is already fulfilled by the instruction memory we use in the proposed architecture. Thus the algorithm storage in our design supports up to 3 nested hardware loops. Each loop is initiated with a single instruction containing a 10-bit immediate for the number of iterations and a 10-bit immediate for the instruction address that marks the end of the loop body.

The combination of dedicated instructions for commonly used HDC algorithmic primitives and code size-reducing features like hardware loops results in a high expressiveness of the ISA. All examined HDC algorithms (see Section 3.4.6) can be mapped with less than 64 instructions.

An Example Configuration for Language Recognition

Language Recognition is a commonly used example application in the field of HDC [38], [153], [173], [190], [192], [197], [198]. The task is to determine the language given a sentence in the form of a character string. For a text corpus with 21 European languages, HDC achieves accuracies of up to 96.7% [197]. The algorithm consists of four main steps; in the preprocessing step, the test sentence is split into so-called n-grams, substrings of the test sentence, obtained when applying a sliding window of size n over the character string. In the next step, the individual n-grams of the sentence are each mapped to an HD-vector according to

$V_{\texttt{n-gram}} = \pi^{n-1}(V_{\texttt{n-1}}) \oplus \pi^{n-2}(V_{\texttt{n-2}}) \oplus \ldots \oplus V_{\texttt{0}}$, with $V_{\texttt{k}}$ denoting the HD-vector corresponding to the character at index k within the n-gram. This vector is obtained through IM mapping using 27

random HD-vectors (26 characters in the Latin alphabet plus one for whitespaces). π^k denotes the repeated application of a bit permutation (most commonly a binary shift operation), and $\oplus$ is the bind operator (XOR for BSC). The n-gram vectors $V_{\texttt{n-gram}}$ for the test sentence are then bundled together to a single search vector $V_{\texttt{sentence}}$ and in the final step, compared with prototype vectors for each language in the AM. The model of the described algorithm, thus the prototype vectors, are obtained by bundling together all sentence vectors $V_{\texttt{sentence}}$ of the training dataset of a language.

In practice, an n-gram size of 4 proved to yield the best performance in terms of accuracy [197].

Listing 3.1 shows the above algorithm for n=4 in Pseudocode:

```
1:  i ← 0
2:  char_vec_{i−[0,1,2,3]} ← 2048′b0
3:  ngram_{i−[0,1]} ← 2048′b0
4:  for char in sentence do
5:      char_vec_i ← im_map(char)
6:      ngram_i ← π(ngram_{i−1}) ⊕ char_vec_i ⊕ π^4(char_vec_{i−4})
7:      i ← i + 1
8:  end for
9:
10: search_vec ← bundle(ngram_0, ngram_1, ...)
11: idx ← 0
12: min_distance ← ∞
13: class_idx ← 0
14: for p in prototype vectors do
15:     distance ← popcount(search_vec ⊕ p)
16:     if distance < min_distance then
17:         min_distance ← distance
18:         class_idx ← idx
19:     end if
20: end for
```

Listing 3.1 – Pseudocode of an HDC algorithm for language recognition.

Instead of recalculating the same character vectors repeatedly when sliding over the sentence, we recursively compute the n-gram using a FIFO structure [190]. Mapping the above algorithm to the proposed ISA with an AM size of 16 vectors and vector fold of one results in the code in listing 3.2.

We omitted the initialization steps that would correspond to lines 1-3 in the pseudo-code listing for simplicity. The body of the algorithm (lines 4-8, listing 3.1) maps to the 12 instructions (lines 1-22) in listing

```
1 start:
2   hw.loop0 nr_characters_in_sentence, end_loop
3   # ENCSEL='output register', MXEN=1
4   enc_reg → mix → enc_reg
5   # ENCSEL='memory', RIDX=12, OP=bind, WBEN=1, WIDX=11
6   mem[12] → mix → bind → mem[11]
7   # ENCSEL='memory', RIDX=13, MXEN=1, WBEN=1, WIDX=12
8   mem[13] → mix → mem[12]
9   # ENCSEL='memory', RIDX=14, MXEN=1, WBEN=1, WIDX=13
10  mem[14] → mix → mem[13]
11  # ENCSEL='memory', RIDX=15, MXEN=1, WBEN=1, WIDX=14
12  mem[15] → mix → mem[14]
13  # Generate seed for char vector
14  # ENCSEL='zero', SMEN=1, SMSEL='50%', OP='passthrough'
15  zero_vec → man 50% → enc_reg
16  #IM-Map char represented with 5-bits
17  MIX_EXT 5 #5+2 cycles
18  # ENCSEL='output register', WBEN=1, WIDX=15
19  enc_reg → mem[15]
20  # ENCSEL='memory', RIDX=11, OP='bind', BNDEN=1
21  mem[11] → bind → bundle
22 end_loop:
23  # OP='threshold bundling counters', WBEN=1, WIDX=15
24  threshold_bndl_cntrs → mem[15]
25  am_search nr_classes #nr_classes+2 cycles
26  intr 400, 2
27  jmp start
```

Listing 3.2 – Microcode mapping of the language classification algorithm in pseudo code. Arrows indicate that operations happen in a combinational pipeline configured according to a low-level instruction (see figure 3.5). The relevant config bitfields for these instructions are indicated in the comments starting with '#' before the relevant line.

3.2. After the hardware loop (line 2 - 22) the search vector is extracted from the MSBs of the bundling counter (line 24) and the search vector is compared with the prototype vectors (line 25), which corresponds to lines 10-20 in listing 3.1. The instruction on line 26 triggers an interrupt if the processed sentence belongs to the classes represented by prototype 1 or 2 with a Hamming distance of less or equal to 400 bits. The final unconditional jump causes the algorithm to start over again, either immediately if the interrupt conditions are not met or after the host processor clears the pending interrupt.

3.4.5 Tuning for Maximum Energy Efficiency

As will be further elaborated in Section 3.4.6, the high amount of parallelism in the datapath and the efficiency of the proposed ISA in executing common HD-algorithms allows the architecture to be clocked at fairly low frequencies while still achieving real-time processing capabilities for many target applications. Figure 3.6 shows the power breakdown of the proposed architecture synthesized with an AM size of 16kBit (16x 1024 bits) while processing an EMG gesture recognition classification algorithm for different degrees of vector folding. Since higher vector fold values result in less datapath parallelism, we adjusted the frequency for each different vector fold configuration to achieve identical throughput for all configurations. In other words, although the different configurations run at different frequencies, they perform the same amount of useful work per time interval with different degrees of sequentiality.

We see entirely orthogonal tendencies for the two different technology nodes in energy efficiency versus Vector Fold. For 65 nm, the overall energy efficiency increases with lower vector folds, thus a higher degree of parallelism, while we see the opposite effect in 22 nm.

The reason behind this effect becomes evident when we have a closer look at the area breakdown in figure 3.7. For a Vector Fold value of one, almost 60% of the accelerator area is occupied by the HD-Encoder. In a technology node like 22nm with SLVT cells, the design is dominated by leakage power. Increasing the vector fold that directly affects the encoder's datapath width has a large effect on the overall area and thus static current draw of the accelerator.

Although the fully synthesizable architecture's technology independence would make it easy to switch to a different technology node with lower leakage, this is not always a possibility, especially when the device is integrated into a larger system. For these situations, the vector fold feature, in addition to its function as a control knob to trade off area for maximum throughput, provides the means to tune the design for maximum energy efficiency depending on the target technologies' leakage behaviour.

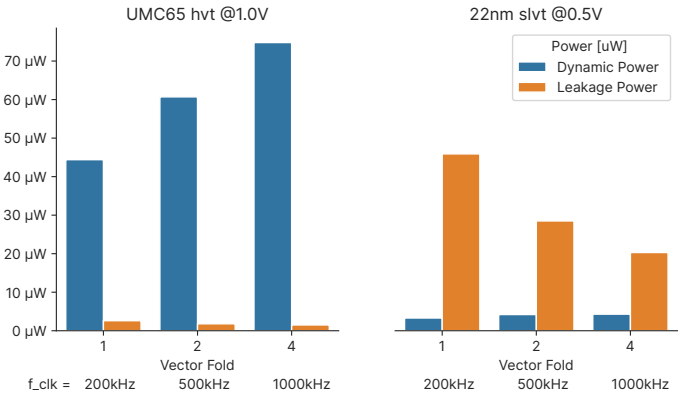


Figure 3.6 – Post-layout-simulated power consumption of the HDC accelerator (16 vectors, 1024 bits each) when executing a real-time HDC algorithm for different vector folds in 65nm and 22nm technology.

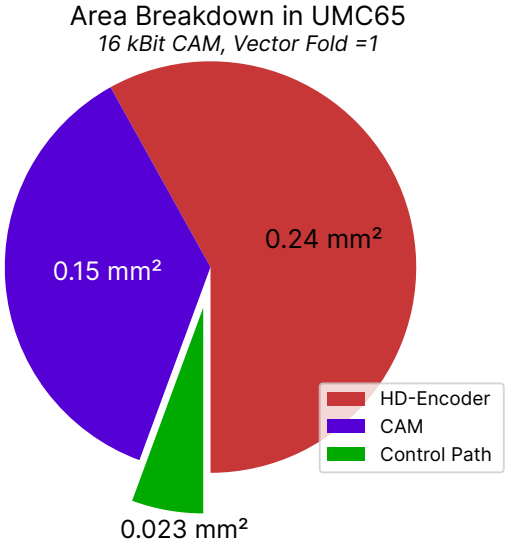


Figure 3.7 – Area breakdown of the HDC algorithm for a vector fold of 1, placed and routed in a commercial 65 nm technology node.

3.4.6 Applications and Use Cases

In this section, we take a closer look at the achieved accuracy of the proposed architecture when configured to execute different classification problems using state-of-the-art HDC algorithms. Both, to validate the soundness of the algorithmic transformations and to compare the energy efficiency with other fully digital HDC accelerators.

Training and accuracy evaluation was performed using a bit-true model of the primitive accelerator operations written in python that was verified against the hardware in RTL simulation. Energy numbers were obtained from simulation of the algorithm on the post-layout netlist using a representative subset of cycles.

Accuracy Analysis on Text classification and EMG Gesture Recognition

As mentioned earlier, the language classification of textual data is a prime example for classification with HDC. While this application does not fit the context of always-on smart sensing, it serves the purpose of validating the accuracy implications of the permutation-based item memory materialization described in Section 3.3.2. We tackle the same classification task to classify the text samples into 21 Indo-European languages [197]. We use the same HDC algorithm described in Section 3.4.4 with an n-gram size of five, which is identical to the algorithm used by Rahimi *et al.* Figure 3.8 indicates the achieved accuracy using a vector fold factor of 1 for different dimensionalities; for 8192-bit HD vectors, the modified HDC operators achieve an accuracy of 94.52%. This accuracy is almost identical to the results reported by Datta *et al.* on their accelerator (95.2%) [195]. The algorithm maps to only 14 HDC ISA instructions and has a memory requirement of five vector slots in the AM, in addition to the 21 language prototype vectors, for intermediate results during the encoding process. For a vector fold of 1, the algorithm executes at 14 cycles per processed input character, which results in 1400 cycles to classify a single sentence in the dataset [197].

The second application we evaluate is hand gesture recognition on electromyography (EMG) data recorded on the subject's forearm. We used the dataset and preprocessing pipeline from [163]; the data

consists of recordings from the subject performing five different hand gestures captured by a 64-channel EMG sensor array with a sampling rate of 1kSPS. The actual HDC classification algorithm works as follows; for each time sample, the 64 channel values are continuously mapped to HD-vectors using the similarity manipulator module described in Section 3.3.2 and bound to a per-channel label vector, generated in the mixer stage. Bundling the resulting 64 channel vectors together yields a single HD-vector that represents the state of all channels for a given instance in time. Five of these vectors are combined to a 5-gram analog to the language classification algorithm to form the search vector for associative lookup against the prototype vector. Training of the prototype vectors works like classification, but many search vectors corresponding to the same gesture are bundled together to form the prototype vector.

The whole algorithm maps very well to HDC ISA, requiring only 12 instructions and two memory slots for intermediate results. The inner loop over the 64 channels in the algorithm is executed in only two cycles for a folding factor of 1, which results in a total of **678 cycles** to classify a single 500ms window of data. Consequently, realtime classification of 64 EMG channels implies an accelerator clock frequency of only **1356 Hz**.

While the data preprocessing flow we used in our experiments was identical to [163], the HDC algorithm, although identical in general structure, differs in a few crucial aspects from the baseline implementation. Moin *et al.* perform CIM mapping of the individual samples to HDC vectors using scalar multiplication of the sample value with a per-channel bipolar label vector, effectively leaving the binary domain [163]. Moreover, the bundling operation to form a time sample vector is implemented as a scalar addition of the integer-valued vectors before thresholding the result back to a bipolar representation with positive values mapped to +1 and negative values to -1. Even though the proposed algorithm modification stays strictly in the binary domain, there is only a small drop in accuracy; with 8192 dimensions, the proposed architecture achieves 96.31% accuracy while Moin *et al.* report an accuracy of 99.44% accuracy using 10'000-bit vectors and arbitrary precision bundling [163].

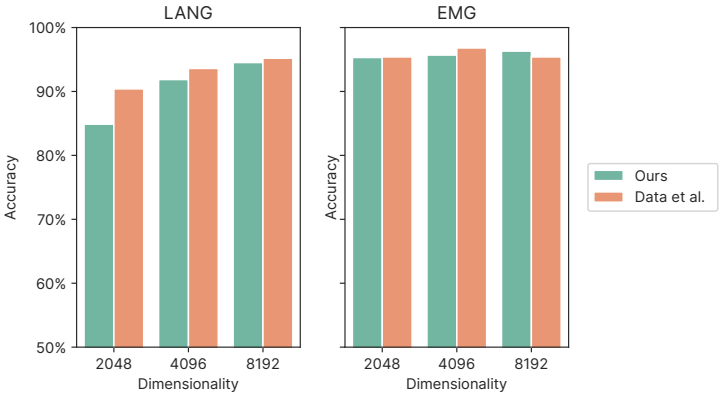


Figure 3.8 – Achieved accuracies for the target applications using different HD-Vector sizes.

Ball Bearing anomaly detection

Predictive maintenance, also known as condition-based maintenance, is a term for the process of estimating the current condition of in-service equipment to anticipate component failure. The goal is to switch to a maintenance scheme where components are replaced once they approach their end-of-life instead of fixed maintenance intervals based on preventive replacement according to the statistically expected lifetime [199]. As part of our algorithmic investigations, we investigate the feasibility of HDC for the task of ball bearing fault prediction using vibration data from low power accelerometer sensors. However, the process of estimating the condition of an in-operation piece of the machine is not trivial; most of the time, we have to draw conclusions from side-channel information like vibration, noise, or temperature captured by sensors attached to the machinery. A considerable amount of aggregate data needs to be analyzed, which, in the case of wireless and energy constrained sensors makes centralized processing infeasible due to the high energy cost of wireless communication. Decentralized processing close to the sensor is thus the key to enable energy autonomous sensors to be retrofitted on existing equipment.

However, the computational cost of preprocessing and actual

classification for all of these features render them infeasible for **continuous** surveillance on low power sensor nodes operating with a power budget in the μW range. We thus present a simple end-to-end HDC classification scheme that can be used as a first element to trigger more involved analysis, e.g., on a cluster of general-purpose cores, in an energy-proportional multistage classification system.

For our analysis, we use the IMS Bearing Dataset provided by the University of Cincinnati [200]. They recorded vibration data at a sampling rate of 20kHz from 4 different ball bearings on a loaded shaft rotating at a constant 2000rpm. We concentrated on the first of the three recording sets, which contains 1 second data records obtained with an interval of 10 minutes in a run-to-failure experiment that lasted 35 days with an accumulated operating time of about 15 days.

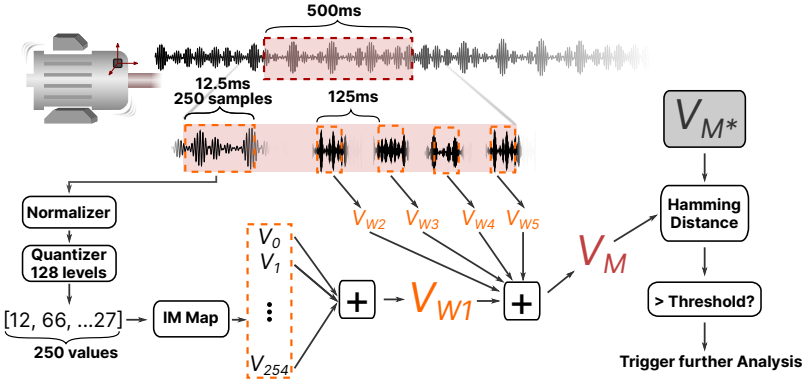


Figure 3.9 – Illustration of the proposed HDC-based ball bearing anomaly detection algorithm. V_{M*} denotes the online trained calibration vector from the first 24 operating hours of the ball bearing.

Figure 3.9 illustrates the basic classification procedure. The algorithm requires an initial calibration phase where a prototype vector representing the ball bearing's normal operating condition is generated. With the inherent feature of HDC that classification and training are of almost equivalent computational complexity, online

training with HDC imposes negligible additional energy costs. The current control path of the proposed HDC accelerator allows for online training algorithms to be encoded in the algorithm storage but requires an external control entity, e.g., a general-purpose core that provides the labels during algorithm execution.

The algorithm's basis is the encoding of small time windows from the raw vibration data to *measurement vectors* V_M. Each time window consists of 250 samples (12.5ms). The sample values are first normalized using a pre-trained normalization factor and quantized to 7 bits. Each sample value is then mapped to an HD-vector using IM mapping, and the whole window of 250 samples is bundled together to a *window vector* V_W. Five of these window vectors with an interval of 125ms are again bundled together to form a single *measurement vector* V_M. The resulting vector thus approximates the amplitude distribution over a 0.5-second time frame.

The general idea behind the proposed analysis scheme is to generate a prototype vector V_{M*} using the first couple of *measurement vectors* after commissioning. We then track the evolution of Hamming distance over time for subsequent measurement vectors. We calibrated the prototype vector using 100 random measurement vectors from the first 24 operating hours of the respective ball bearing in our experiments. Similarly, the normalization factor is generated using the 99% quantile of the amplitude within the same 24 hours after commissioning. The proposed algorithm can be mapped to **9 HDC ISA instructions** and requires two vector slots, one for the calibration vector and one for intermediate results.

Figure 3.10 shows the evolution of Hamming distance over time with an exponential moving average filter with a halflife of five hours. This feature can be computed very efficiently without the need for a large ring buffer. The line color indicates the labels proposed by experts on the manual analysis of the dataset [201].

By the end of the IMS ball bearing experiment, bearings 3 and 4 failed, while bearings 1 and 2 were severely worn down but did not fail yet. We see a sharp increase in Hamming distance for all four ball bearings several hours before the actual failure, in the case of ball bearing 3, even several days before the actual inner race failure.

While the proposed algorithm certainly does not replace more involved analysis on time and frequency domain features, the results

Algorithm	# of HDC instr.	Vector Memory	Cycles/classif.	Realtime Freq. [kHz]	Power [µW]		Min. Energy/classif. [nJ] @100 kHz	
					65nm	22nm	65nm	22nm
LANG	14	5 + 21	1400	100	86.5	23.7	1205	**332**
EMG	12	2 + 5	678	1.4	10.7	2.9	703	**191**
BEARING	9	1 + 1	12513	25	29.1	7.9	10913	**2966**

Table 3.3 – Memory requirements and post-layout energy numbers of selected HDC algorithm on the proposed architecture with an AM size of 32 x 2048 bit, vector fold 1

suggest that it can act as a first filtering stage for aggressive duty cycling of more power-intensive analysis schemes when combined with simple thresholding. However, more experiments on larger datasets and possibly with more complex HDC encoding schemes will be required to quantify the benefits of an HDC-based ball bearing fault predictor.

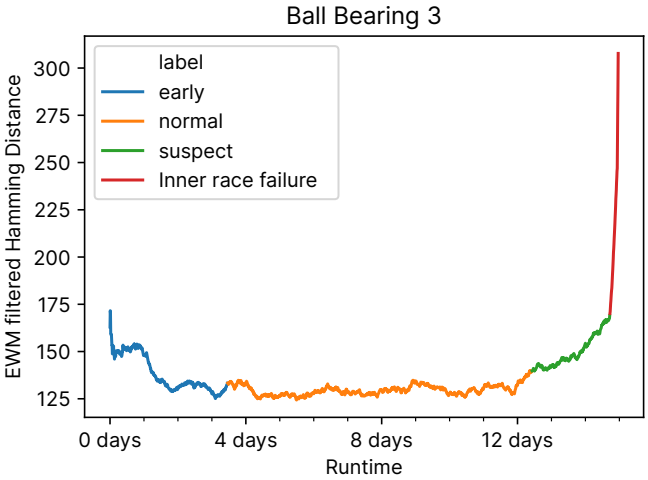


Figure 3.10 – Hamming distance evolution over time for ball bearings 3 in the IMS dataset. The Hamming distance was post-processed with an exponential moving average filter with a halflife of 5 hours. The other ball bearings in the dataset show a similar behavior.

	Technology	Area [kGE]	Architecture Type	IM / CIM resolution [bit]	Energy eff.[nJ/inference]	
					LANG	EMG
Datta *et al.*	TSMC28	3618	generic	10 or 10	250	610
Our Work	**22 nm-FDSOI**	**1094**	**general-purpose**	**arbitrary and 7**	**332**	**191**

Table 3.4 – Area and Energy efficiency comparison with the current state-of-the-art HDC accelerator architecture. The terms *generic* and *general-purpose* were introduced by Datta *et al.* in [195].

3.4.7 Energy Efficiency Analysis and Comparison

Table 3.3 summarizes the performance of the three introduced HDC algorithms, language classification (LANG), EMG gesture recognition (EMG), and ball bearing anomaly detection (BEARING). While EMG and BEARING represent typical workloads for streaming wake-up applications on life sensor data, we picked LANG for its common use in HDC literature as a benchmark application.[38], [153], [173], [190], [192], [197], [198] Columns 2 & 3 report the number of HDC instructions and the total number of required HD vector memory to map the algorithm to the architecture. Column 4 shows the required minimum frequency for realtime execution of the algorithm (not applicable for LANG since there is no realtime constraint for this application). The last two columns indicate the power when operating at the aforementioned minimum frequency and the corresponding energy efficiency per classification. For LANG, we consider a single classification to be the processing of a 100-character string, the average sentence length in the Wortschatz corpora. For EMG and BEARING, a single classification is defined as the analysis of a 500ms window as described in the algorithm sections 3.4.6 and 3.4.6.

In Table 3.4 we compare the energy efficiency of our solution to the current SoA HDC accelerator architecture from Datta *et al.* [195] which differs in several aspects from our work; While we use the rematerialization approach introduced in section 3.3.2 for IM and CIM mapping, their design uses a large ROM to explicitly store the input to HD vector mapping. Also, they combine a heavily pipelined encoder with a fully combinational, flip-flop-based associative memory whereas our architecture opts for a pipeline-free sequential encoder with a vector sequential, latch-based AM design. Finally, the architecture proposed by Datta *et al.* is only *generic* (a subset of *general-purpose*

architectures) according to their taxonomy on HDC algorithm classes established in [195]. This imposes a set of additional constraints on the structure of HDC algorithms that can be executed. In contrast, the microcode-based approach that our architecture follows allows for arbitrary HDC algorithm computation (including the *generic* ones) within the limits of the available AM and instruction memory resources.

Among other algorithms, Datta *et al.* report the energy numbers for EMG and LANG executed on a 32 by 2048-bit accelerator in TSMC28. We achieve a technology-scaled area reduction by **3.3**×. This can be explained by massive area reductions in all major components of the accelerator. The most considerable effect has the on-the-fly pseudo-random materialization of the item vectors used in our design, which removes the necessity to incorporate a large ROM to store all possible item vectors. In fact, 62% of the overall area in Datta *et al.* is occupied by a large 1024 by 2048 bit ROM. Besides the area and energy implications, the ROM-based solution has the added drawback of having a hardwired partitioning of the memory; one for the item memory, containing quasi-orthogonal vectors, and one for continuous item memory vectors, where the pair-wise Hamming distance between the vectors correlates to the difference of the corresponding input values. Another large reduction in area is achieved in the AM, where our solution uses latch cells and sequentially calculates the Hamming distance in contrast to the baseline, which uses a flip-flop-based fully parallel implementation.

In fairness, one has to notice that [195], with a maximum clock frequency of 434MHz, has a much higher peak throughput than our solution due to its parallel and heavily pipelined architecture. However, the results in Table 3.3 suggest that algorithms used for always-on sensing do not benefit from such a high throughput, and energy efficiency is the key metric by which we should judge the performance of the different approaches.

As we can see in Table 3.4, the energy efficiency differences between the two architectures depend a lot on the algorithm at hand. For LANG, the achieved energy efficiency is slightly worse (+31%) than the baseline, which is still impressive considering the 3.3× reduction in area. For EMG, on the other hand, we achieve a 3.1× improvement in energy efficiency. This is in contrast to Datta *et al.*'s architecture, where LANG exhibits a higher energy efficiency than EMG. This

can be explained by the difference in the computational complexity of orthogonal and continuous item mapping in our architecture. In LANG, input values are mapped to quasi-orthogonal vectors using the mixing stage (3.3.2), which requires $log_2(N)$ cycles, where N denotes the cardinality of the input set. The overhead of this iterative approach considerably lowers the energy advantage of not using a large ROM for item memory generation. For EMG, on the other hand, the input values are mapped continuously using the similarity manipulator, which can be performed in a single cycle and can even be combined with a bundling or bin operation in the subsequent encoder units. Hence, for this algorithm, the effect of not requiring a ROM comes to display and causes the EMG task to execute more efficiently than the LANG task.

In general, we can say that for very high input value resolutions, the overhead of iterative item vector generation starts to dominate the overall energy consumption of our architecture. Thus, for an application-specific accelerator with a small fixed input resolution using IM-based encoding, a ROM-based IM might be more energy efficient than our approach. Still, the fact that the computational complexity of the rematerialization approach grows with the logarithm of the input space instead of linear ROM area scaling suggests an advantage of our architecture for larger input space cardinality. In any case, the proposed architecture excels in its energy proportionality to the desired HDC algorithm. The ROM-based approach in [195] has an almost fixed cost for item memory mapping with an upper limit on the supported resolution. For example, in LANG, only 13% (27 out of 1024 item vectors) of all ROM entries are required to map the input values.

3.5 Silicon Realization inside the TSMC 65nm Rosetta Testchip

In this section we are introducing *Rosetta*, a silicon test vehicle SoC for near-memory processing circuit and on-chip memory evaluation taped out in TSMC 65nm technology. The chip features the first proof of concept silicon realization of Hypnos, a novel gain-cell eDRAM macro as well as other IPs from other project partners[4] In this section, we will only provide a summary of the eDRAM characteristics and instead focus on the macro architecture of the system and initial on-silicon evaluation of Hypnos.

Contributions This project has been a joint effort with the telecommunication circuits laboratory (TCL) at EPFL who developed the analog GC-eDRAM IP and me with ideas and inputs from Luca Benini and Andreas Burg. My responsibility within the project was the initial design of the SoC, the system integration of Hypnos and the other 3rd party IPs. The physical design and silicon measurements were done by myself with significant support of Beat Muheim and Frank Gurkaynak from the micro-electronics design centre at ETH. The write-up of this chapter was done by myself with guidance and feedback of Luca Benini. The results on the GC-eDRAM have been published separately in IEEE Solid-State Circuits Letter [64] where I contributed the SoC architecture section.

3.5.1 SoC Architecture

Figure 3.11 illustrates the SoC architecture; The system is based on PULPissimo [202], a single-core microcontroller system built around a 32-bit RISC-V core named RI5CY [196].

RI5CY is a 5-stage single-issue in-order 32-bit RISC-V core. The processor is coupled with a 32-bit hardware floating point unit and supports the standard RISC-V ISA extensions M, F and C thus making it an RV32IMFC-compatible core. On top of the standard ISA extensions, the RI5CY core in Rosetta supports several custom

[4]Not considered in detail within the scope of this thesis.

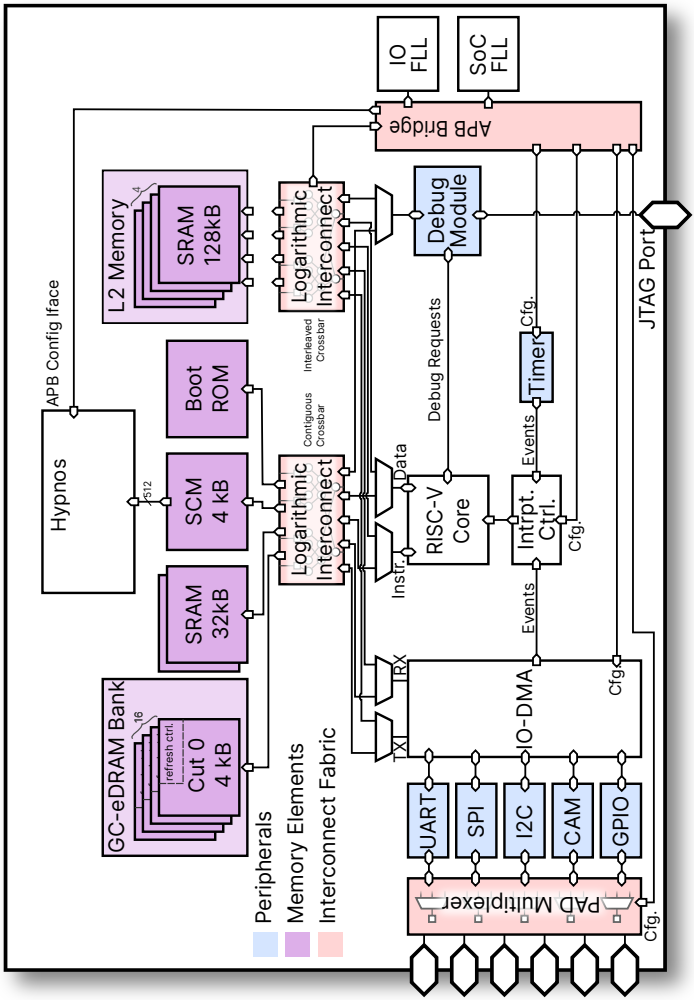


Figure 3.11 – Architecture of the Rosetta SoC.

instructions bundled in the custom ISA extension *xPULPV3*. Most notable for runtime efficiency improvements is the support for hardware loops, sometimes also referred to as zero-overhead loops and SIMD instructions.

The cores instruction and data fetch ports connect to the *tcdm*, a global system interconnect that enables high-bandwidth access with single-cycle transaction latency to Rosetta's various memory components. This so-called *logarithmic interconnect* is based on a forest of arbitration tree circuits that ensures fair, combinational arbitration in the fully connected crossbar [203].

The interconnect branches into three memory regions; The interleaved crossbar connects to the bulk of the system's main memory consisting of 4× 32 KiB SRAM banks each internally structured into two 16 KiB memory cuts. These memory banks are word interleavingly mapped to Rosetta's global address space such that contiguous address space access would evenly spread the transactions across the 4 banks. This mapping scheme reduces the average bank-conflict-induced latency for the common scenario where several bus masters simultaneously try to access contiguous (in the address space) blocks of memory; E.g. consider the scenario where two bus masters try to sequentially access a different block of memory. If the start address of the blocks maps to different memory banks, the two bus masters will never cause a conflict assuming that they access the memory at the same rate. If their blocks' start address however is mapped to the same memory bank, one of the two masters will be stalled by the arbitration tree for one cycle. After that initial stall, their access pattern is out of phase again and the remaining memory transaction will not interfere with each other anymore.

The second branch of the interconnect is what we call the *contiguous crossbar*. In contrast to the interleaved crossbar, the memory elements of slaves attached to the contiguous crossbar are mapped to contiguous address regions within the address space. Within Rosetta the following memory entities are connected to it:

- The *boot ROM* (8 KiB)
- Two smaller SRAM banks with a size of 32 KiB each.
- A 4 KiB dual-port latch-based SCM

- 64 KiB of GC-eDRAM

The boot ROM is a read-only memory storing the firmware to enable different boot sequences. The two private banks serve the purpose of providing conflict-free access to instructions and the core's call stack. Note however that these roles are only weakly enforced by the linker script settings of the software toolchain. The core can in fact access instruction and data from arbitrary memory locations within the address space.

The 4 KiB SCM serves the dual purpose of acting as a low-voltage tolerant general-purpose memory through its 32-bit read/write port. The second 512-bit wide read/write port connects the SCM to *Hypnos* to act as the memory backend for HD-vector storage and associative lookup. That is the latch memory AM previously described in section 3.3.1 has been externalized from the HDC accelerator to make it usable as a regular main memory. The HDC accelerator within Rosetta has been parametrized for a maximum HD-vector width of 2048-bit folded onto an internal datapath width of 512-bit with 5-bit bundling counters per folded vector dimensions. For HDC algorithm storage an internal 64 by 24 bit SCM stores the uCode instructions.

Finally, Rosetta contains a 64 KiB 3T-1C gain-cell eDRAM, internally split into 16× 4 KiB memory cuts with embedded refresh controller to research the PPA trade-offs of eDRAM versus SRAM based on-chip memory. The eDRAM in Rosetta is based on a 3T-1C gain-cell structure as the fundamental storage element.

Besides the cores, the HDC accelerator and different memory flavours, Rosetta contains various IO peripherals for off-chip communication with commercial sensors through an autonomous IO-DMA. This so-called *uDMA* subsystem not only allows peripherals to directly read and write to the system's main memory but also provides them with a stream of ucode instructions to encode the IO interactions. Therefore once setup, the IO peripherals like SPI or I2C can operate without any additional core intervention in an entirely autonomous manner.

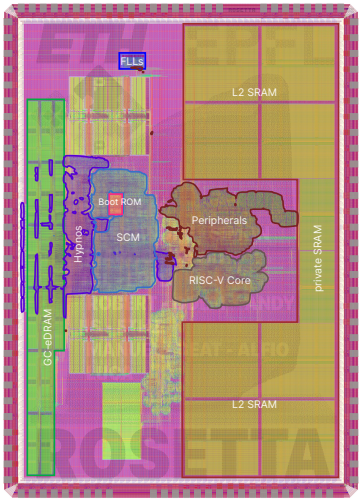
The entire SoC is split into two clock domains sourced by two bypassable frequency-locked loops (FLLs). One drives the IO peripherals while the other one drives the rest of the system. Coupled with a RISC-V debug specification-compliant JTAG debug unit that exposes the entirety of the internal address space, this provides us

with sufficient control over all internal aspects of the system for benchmarking and functional tests.

3.5.2 Physical Implementation

The entire system's digital frontend is described in RTL using the SystemVerilog hardware description language (HDL). Using Synopsys Design Compiler® 2019.3, the design was synthesized using 8 track standard cells coming in three different V_{th} flavours (high, regular, low) with all gate-length options enabled. The SRAM were generated using a commercial memory macro generator tuned towards area efficiency, while the eDRAM is a full-custom design developed at EPFL. During synthesis the design was constrained with a target SoC clock frequency of 200 MHz (1.2 V nominal V_{DD}, TT, 25 °C) and a clock uncertainty of 0.5 ns. The backend implementation of the physical design was conducted using Cadence Innovus® 19.10 for place and route and Siemens Calibre 2019.2 for signoff design rule check (DRC) and layout-versus-schematic (LVS) targeting a 9-metal layer BEOL.

Figure 3.12a shows the physical layout of the chip. The chip contains individual power domains for the IO pads, GC-eDRAM and the digital logic (including the SRAM macros). It thus allows to characterize the power consumption of the eDRAM in isolation.



(a)

Physical	Technology / Package	TSMC 65nm
	Package	QFN64
	Dimensions	4100 µm×3000 µm
	Gates	6 MGE
	Voltage	1.2 V
	Max. Freq.	190 MHz
Memory	shared SRAM	128 KiB
	private SRAM	64 KiB
	SCM	4 KiB
	GC-eDRAM	64 KiB
HDC	VSA	BSC
	HDC Vector Count	16
	Vector Width	2048 bit
	Bundling Width	5 bit
	Algo. Instr. Mem.	64 instructions

(b)

Figure 3.12 – (a) Full layout of the Rosetta SoC with annotated main components and (b) chip specifications

.

3.5.3 Results

Figure 3.13 shows the area breakdown of the SoC. Almost 60% of the overall die area is occupied by the various kinds of memories, with the large 128 KiB taking up the majority of the area. With a total area of 492 000 µm^2 Hypnos including its 4 KiB large latch-based SCM makes up about 7% of the overall chip area. Besides the SCM area-dominated associative memory (5%) the majority of Hypnos' area is required for the 512-bit wide hyperdimensional encoding unit (Hypnos in Rosetta is parametrized to fold the 2048-bit hypervectors into 512-bit parts to save area). Upon closer inspection of the area report, we see that the encoder area is dominated by the 5-bit flip-flop based saturating bundling counters instantiated for every folded dimension. The control path including the 64×26 bit HDC-ISA instruction memory consumes only 17 420 µm^2 i.e. 3.5% of Hypnos' area.

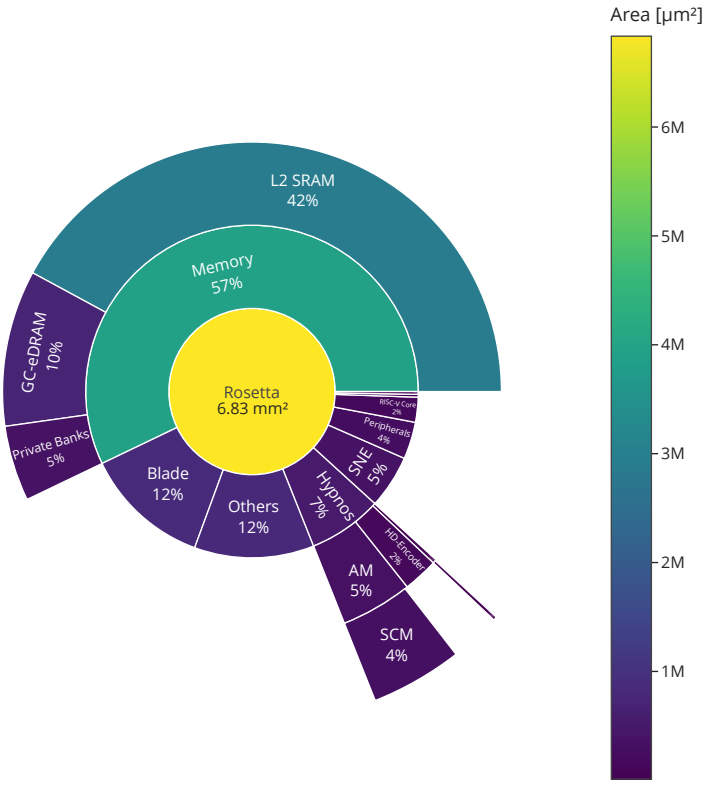


Figure 3.13 – Post-layout area breakdown of the Rosetta prototype SoC

The main goal of the Rosetta prototype chip was to verify the functionality of Hypnos in real silicon and to provide a test platform for alternative memory technologies like the eDRAM. The version of Hypnos in Rosetta has not been trimmed for energy efficiency during physical design. Still, we assessed Rosetta's overall energy efficiency in order to validate the results obtained in post-layout simulation. We thus executed the EMG benchmark application introduced in section 3.4.6 in an infinte loop while keeping the RISC-V core in a clock-gated wait for interrupt state and measured the current draw on all power domains. For this experiment, we aggressively lowered the supply

voltage from a nominal 1.2 V to 0.76 V, the lowest measured voltage where Rosetta remained operational. At a core frequency of 500 kHz the entire chip consumes **813 µW** while continuously running the EMG inference algorithm with EMG dummy input data fetched from the internal L2 SRAM. This amounts to an overall energy efficiency of 8.7 nJ/Inference at an inference rate of around 100 Hz. Given that we measure system-level energy efficiency, including all support circuits like FLL, SRAM leakage etc. and the fact that Hypnos in Rosetta is parametrized with a vector fold of 4 rather than 1, which we showed to be around 2× less energy efficient (see section 3.4.5), this number is in the expected ballpark.

3.6 A Lightweight Data Preprocessing Sensor Interface for Hypnos

Although the power envelope of Hypnos is very competitive within the setting of always-on smart sensing, it does not include the sensing frontend and expects the digital input data in a particular format. While the design of an analog frontend is outside the scope of this thesis we propose a lightweight data preprocessing unit that combines a flexible digital sensor interface based on the SPI protocol with an ultra-small multi-channel digital processing block. The aim of this IP is to bring the digital input data into a suitable format for further processing in Hypnos. More involved digital signal processing algorithms like FIR-filtering or filtering in the spectral domain were not considered in this work due to their prohibitively high-power consumption in comparison to Hypnos.

3.6.1 Architecture

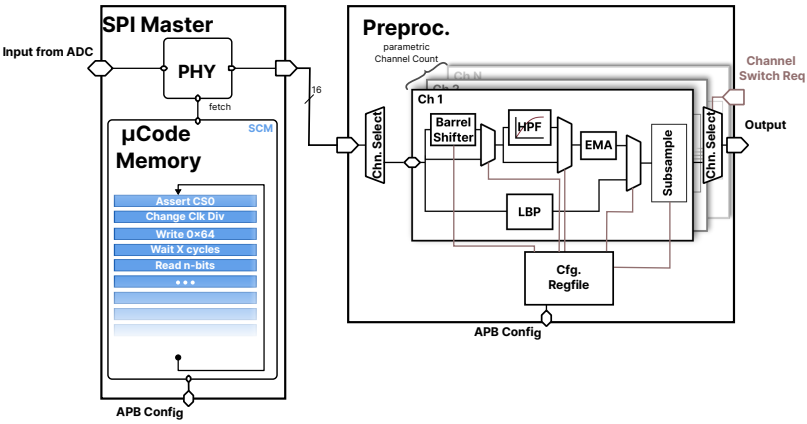


Figure 3.14 – Schematic overview of the proposed data preprocessing unit. The IP consists of an autonomous SPI peripheral (left) and a multi-channel preprocessing unit (right)

Figure 3.14 illustrates the overall architecture of the IP; An autonomous SPI peripheral communicates with conventional low-power off-chip sensors. When integrated into a larger system-on-chip a configuration interface enables setup and at-runtime configuration of the preprocessor using regular memory transactions by conventional processor cores.

Autonomous SPI Peripheral The autonomous SPI peripheral originally based on the work of Pullini *et al.* [204], most similar to the control path of Hypnos, operates on a small stream of microcode instructions that control the peripheral read from a small embedded SCM.

The instructions encode SPI configuration like polarity and phase, the operating sampling frequency, asserting one of the four available chip-select signals as well as the SPI read and write transactions themselves. The SPI peripheral executes the "io-programm" stored in the SCM in an infinite loop thus enabling the module to autonomously interact with one or several external sensors. Since write transactions with literal values can also be encoded as microcode instructions, even dynamic reconfiguration of a sensor's settings is possible.

Multi-channel Preprocessing Unit The read data of the SPI peripheral are forwarded to the preprocessing block using a simple read-valid handshaking protocol. This module contains a parametric number of parallel processing pipelines to provide support for multi-channel data. The data received from the SPI peripheral is multiplexed on the currently enabled channels in a round-robin fashion. Each of the 16-bit channel processing pipelines consists of several bypassable simple filtering blocks:

- A barrel shifter for shift alignment of the received SPI data
- An offset removal block for constant bias removal
- An exponential moving average (EMA) filter for dynamic offset removal i.e. IIR-based high-pass filtering
- An IIR low-pass filter block, again based on an EMA filter
- A decimation block to downsample the input to the desired output sampling rate at which to process the values in Hypnos

Besides the outlined processing pipeline, as an alternative to the first four processing stages a local binary pattern [205] filter block can be selected. Local binary pattern is an effective input symbolization technique that in combination with HDC showed promising results in epileptic seizure detection on intracranial electroencephalography [166], [206]. It operates according to the following formula:

$$LBP[n] := \begin{cases} 1 & \text{if } x[n] > x[n-1] \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Where $x[n]$ denotes the digital input stream and $LBP[n]$ is the bit serial output stream. This bit stream is then grouped in symbols of fixed length, in this particular case 6 bit.

3.6.2 Results

We synthesized the preprocessing sensor interface with 8 channels, 16-bit datapath and an SCM with 32 microcode instruction slots to encode the SPI interaction in a commercial 22 nm technology. The figures of merit for this parametrization are summarized in table 3.5; The entire processing block occupies a total area of **6179 µm²** or about 31 kGE, which is only about 1.3% the size of the Hypnos configuration we taped out in Rosetta.

To evaluate the preprocessor's power characteristics, we ran a post-layout simulation of the same EMG benchmark algorithm outlined in section 3.4.6 at 1 kSP/s/Channel. We configured the Preprocessing Sensor Interface to capture simulated SPI sensor data from multiple channels with an SPI clock frequency of 200 kHz capturing 16-bit values and enabled every preprocessing block for each channel of the 8-channels. With an overall power consumption of only 1.12 µW for this particular scenario, the preprocessing unit power consumption adds very little energy overhead for always-on classification. Due to the serial nature of the SPI protocol, only one channel is ever actively toggling at a time. Therefore the power consumption when scaling to more than 8 channels depends only on the clock frequency, which directly correlates with the required sampling rate per channel. Leakage power due to the very small size of the unit is mostly insignificant[5].

[5]Around 3.5% of the total power consumption in this particular technology

Technology	22 FDSOI, 8T-UHVT standard cell library
Corner	0.8 V, 25 °C, typical-typical
Datawidth	16 bit
Num. Channels	8
Microcode SCM Size	32×32 bit
Area	6179 µm^2
SPI Clock Freq.	200 kHz
Preprocessor Clock Freq.	200 kHz
Total Power	1.12 µW

Table 3.5 – Figures of Merit of the Preprocessing Unit in 22 nm FDSOI. Numbers obtained in timing back-anotated post-layout simulation with all preprocessing blocks active.

3.7 Summary

In this chapter, we have introduced a number of building blocks for energy-efficient near-sensor classification leveraging the ML paradigm of hyperdimensional computing:

- We started this chapter by presenting a standard-cell memory-based implementation of an all-digital associative memory.
- We then introduced Hypnos, an open-source, fully-configurable hardware accelerator for binary spatter code based HDC and demonstrated its 3.3× area density and up to 3× energy efficiency advantage over the previous state of the art.
- These pre-silicon results are then put to the reality test in the *Rosetta* SoC, a 65nm prototype chip that combines a complete microcontroller architecture with embedded eDRAM and Hypnos.
- Finally, we introduced an area-efficient digital sensor frontend with lightweight filtering capabilities for the purpose of signal conditioning external sensor data for subsequent processing by Hypnos.

In summary, we showcased a number of circuits for highly energy-efficient near-sensor processing without giving up on the intrinsic scalability, reliability and cost advantages of fully-digital designs.

Chapter 4

Architectural Considerations at the System Level

4.1 Introduction

In chapter 3 we have introduced a couple of building blocks for ultra-low power on-device smart sensing. In this chapter, we combine these circuits around hyper-dimensional computing with novel designs around the QNN data processing paradigm to design a complete end-to-end system able to withstand the reality-check of actual post-silicon evaluation. As emphasized in chapter 1, the transition from individual circuits to the system level in the form of a low-power SoC usually comes with many challenges that are less apparent at the individual IP level like synchronization, power distribution or cooling. However, in the scope of this thesis, we put particular emphasis on two key aspects;

Energy-Proportionality Besides the pure ML aspect of edge-computing algorithms, IoT applications usually contain various additional abstract application *components* like interaction with external sensors, signal conditioning, post-processing, user

interaction and wireless communication with edge infrastructure. Each of these abstract components comes with drastically varying compute- and IO-intensity demands. Mapping these components to an efficient hardware-software partitioning in the form of an integrated system architecture that behaves energy-proportional to the compute load of the current task is not straightforward to realize.

Memory Bandwidth and IO-Energy In contrast to what the name might imply, the so-called von-Neumann bottleneck is far from being limited to conventional computing on general-purpose cores only. A universal trait of all compute-intensive NN based applications is a high demand in memory bandwidth for fetching weights and storage of intermediate results (i.e. feature maps). While dealing with this demand on memory bandwidth in HPC is challenging enough, realizing systems that meet the application requirements on real-time behaviour under the way more stringent energy-efficiency constraints is even harder. Communication with high-bandwidth off-chip memory usually implies orders of magnitude higher power consumption than on-chip memory access. Conventional SRAM-based on-chip memory being volatile in nature, conversely faces the challenge of maintaining energy proportionality in highly duty-cycled applications due to their static current draw.

In this chapter, we introduce two taped-out systems that specifically address those two points: The Vega SoC puts emphasis on the first of the above challenges where we present an energy-proportional approach to always-on sensing using a heterogenous architecture with hierarchical wake-up.

In section 4.3 we present Siracusa that demonstrates an architecture to cope with the Memory Bandwidth and IO-energy of tinyML applications in the IoT domain. The SoC taped out in 16nm FinFET technology incorporates non-volatile MRAM and tightly couples it to a highly flexible digital QNN accelerator.

Contributions Section 4.2 on the Vega SoC has been first presented at the International Solid-State Circuit Conference (ISSCC) in 2021

[122] and subsequently published in the Journal of Solid-State Circuits (JSSC) in 2022 [84][1]. As is natural for such a large system, this was not the work of a single person; My responsibility within this joint project was the development of Hypnos, the preprocessing unit and the autonomous SPI peripheral which is jointly referred to as the *Cognitive Wakeup Unit (CWU)*. I was also responsible for system-level integration and verification of the CWU. Finally, I conducted the chip-bring-up and circuit characterisation of the SoC domain, the CWU and the IO-pads. The overall system architecture was developed by Davide Rossi with help from Antonio Pullini, Igor Loi, Jie chen and Eric Flammand. The QNN accelerator has been developed by Francesco Conti, while the FPUs used in the cluster where designed by Stefan Mach. On the software side, Giuseppe Tagliavini was responsible for compiler optimization. The whole project was supervised and advised by Luca Benini.

[1] ©2022 IEEE. Reprinted with permission, from D. Rossi et al., „Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode“, IEEE Journal of Solid-State Circuits, Bd. 57, Nr. 1, S. 127–139, Jan. 2022, doi: 10.1109/JSSC.2021.3114881.

4.2 The Vega 10-Core SoC for IoT End-Nodes

An increasing amount of near-sensor data analytics applications require inexpensive battery-operated micro-systems able to sense the environment and transmit meaningful, highly semantic compressed data to the cloud wirelessly. The tight constraints in terms of low-power in sleep mode, coupled with extreme performance and energy efficiency in active mode, calls for a new class of ultra-low-power microcontrollers (MCUs), namely, IoT processors. These devices require a large state-retentive memory to autonomously wake up when always-on ultra-low-power sensors detect a specific condition. Following wake-up events, more capable sensors and computing units can be activated to perform fully programmable complex near-sensor analytics, including modern Deep Neural Networks (DNNs) models. This approach enables the extraction of meaningful information from the sensor data locally before transmitting it, avoiding data deluge in the cloud.

Recent research on ULP MCUs design focused on main building blocks, such as memories [207], standard cells [208] and embedded power management [209]. On the other hand, while more traditional low-end IoT end-nodes are targeted to low-bandwidth sensors (e.g., temperature and pressure) and require limited compute capabilities, an increasing number of applications rely on embedding much more intelligence at the edge. Dedicated solutions explored in the last few years deliver high performance and efficiency, mainly focusing on inference [210], [211], and training [212] of DNNs, exploiting low-precision and tunable-precision arithmetic to adapt to the requirements of applications while minimizing energy consumption [213]. Although the efficiency of dedicated hardware accelerators is orders of magnitude larger than that of previously mentioned MCUs, the large variety and fast evolution of near-sensor data analytics algorithms running on IoT end-nodes cannot be satisfied by specialized and inflexible accelerators.

In this context, this work provides a significant step forward in high-performance, Parallel Ultra-Low-Power (PULP) IoT processors, presenting Vega. Vega introduces key contributions in two areas: always-on cognitive operation augmented by non-volatile memory

support and highly dynamic digital signal processing. First, the proposed SoC features 4 MB of non-volatile MRAM coupled with a fully programmable cognitive wake-up unit based on the Hyper Dimensional Computing (HDC) paradigm [149]. This non-volatile cognitive wake-up architecture enables the probing of ultra-low-power sensor data with power consumption as low as 1.7 μW and wakes up the system from a full memory retentive state. Second, Vega enables the highly dynamic exploitation of multiple data formats, from a few bit-width integer to full precision floating-point (FP). Thus, application developers can seamlessly tune the precision and dynamic range of portions of algorithms, matching them with the rich set of data formats natively available on the hardware. We demonstrate the capabilities of the Vega SoC on a wide range of Near-Sensor Analytic Applications (NSAA) in the bio-signal, audio/vibration, and imaging domains as well as inference of DNN, showing significant improvement in flexibility, performance, and efficiency over the state of the art.

4.2.1 VEGA SoC Architecture

As shown in Fig. 4.1, Vega consists of four main switchable power domains: the SoC domain includes a single-core MCU served by 1.7 MB of SRAM memory and several peripherals, the cluster domain includes the programmable parallel accelerator, a domain including 4 MB of non-volatile MRAM, and a fourth domain including the Cognitive Wake-Up (CWU) unit.

Non-Volatile SoC Subsystem

The SoC Domain is an advanced MCU featuring a RISCV processor named Fabric Controller (FC), served by 1.7 MB of embedded SRAM and several peripherals described in Fig. 4.1, including a 1.6 Gbit/s HyperBus/OCTA SPI/SDIO Double Data Rate (DDR) interface supporting external DRAM and Flash memories. For instance, the interface supports Cypress Semiconductor's HyperRAM and HyperFlash memories [214], APMemory IoT RAMs [215], and external Secure Digital Input Output (SDIO) cards.

To allow peripherals to transfer data independently from FC, Vega implements an I/O subsystem in which each peripheral has a dedicated

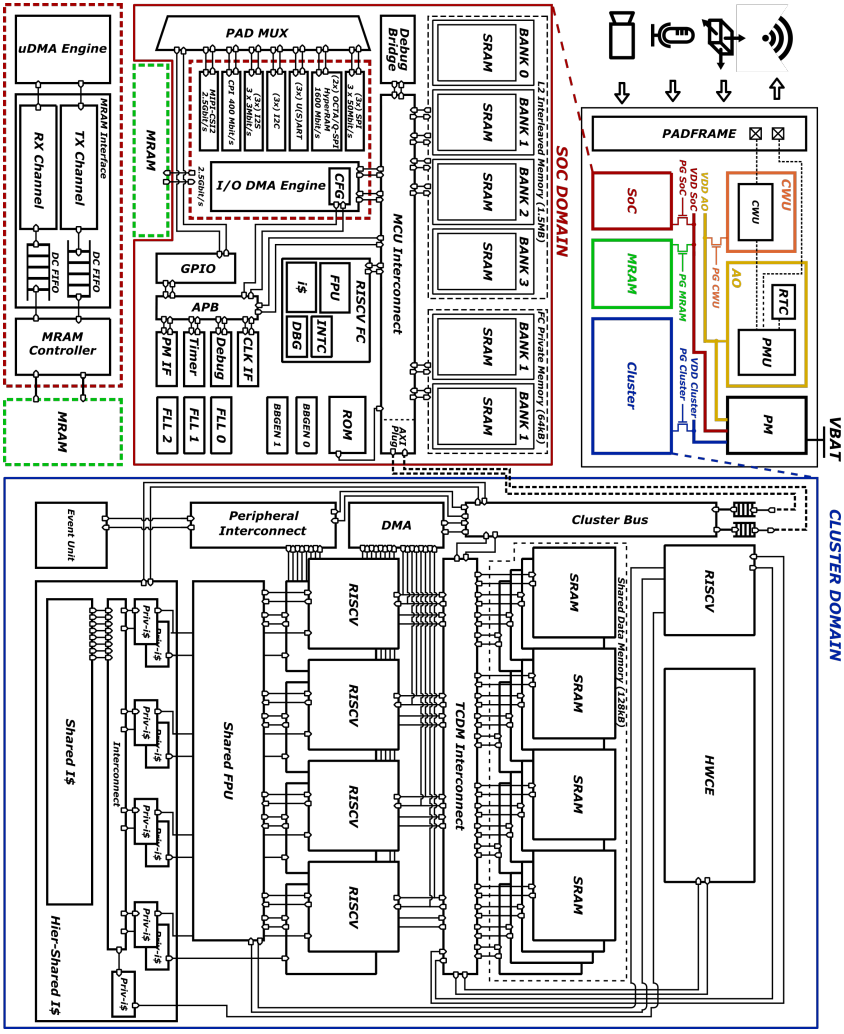


Figure 4.1 – Vega SoC Architecture

DMA channel enabling direct, autonomous data transfer to/from the L2 memory [216]. A 4 MB non-volatile Magnetoresistive Random Access Memory (MRAM) resides in an independent switchable power domain. The MRAM is connected to the L2 memory through an auxiliary IO DMA channel and managed just like a peripheral. A dedicated controller manages the specific protocol conversion operations required to access the MRAM in write and read mode, completely abstracting to the end-user the complexity of the specific protocol. During the read operation, the MRAM can operate at a frequency up to 40 MHz, delivering bandwidth of 2.5 Gbit/s through its 78-bit interface (including 14-bit ECC). The two main applications of MRAM within Vega SoC are the storage of read-only parameters of machine learning applications and program code. Except for the I/O DMA, the master resources of the SoC can only access MRAM data after being moved to the L2 memory.

The L2 memory memory consists of 4 word-level interleaved banks for a total of 1.5MB, plus 64 kB of private memory for the core. This architecture provides an overall bandwidth of 6.7 GBytes/s to the peripherals and accelerators in the system. To retain the SoC program and data in sleeping mode, the physical SRAM banks can selectively be configured in retentive mode, leading to retention power ranging from 1.2 to 112 μW for 16 kB to 1.6 MB of state-retentive L2 SRAM. Hence, once the SoC is woken-up from sleep mode, a *warm boot* can either be performed from L2 SRAM or from the MRAM. In the former case, some power consumption is required for preserving state-retention of SRAMs; in the latter case, sleep power for data retention is zero, but the program must be restored into L2 after wake-up. Hence, depending on the duty cycle and wake-up latency requirement of the target IoT application, one or the other approach can be selected.

Cognitive Wake-Up Unit

While Vega provides highly energy-efficient compute performance in active mode using its programmable cluster, the TinyML [186], [187], [189] power envelope for self-sustainable always-on signal processing applications cannot be met in full active mode. These applications require aggressive duty cycling and intelligent wake-up logic to detect events of interest. However, the threshold-based wake-up systems used

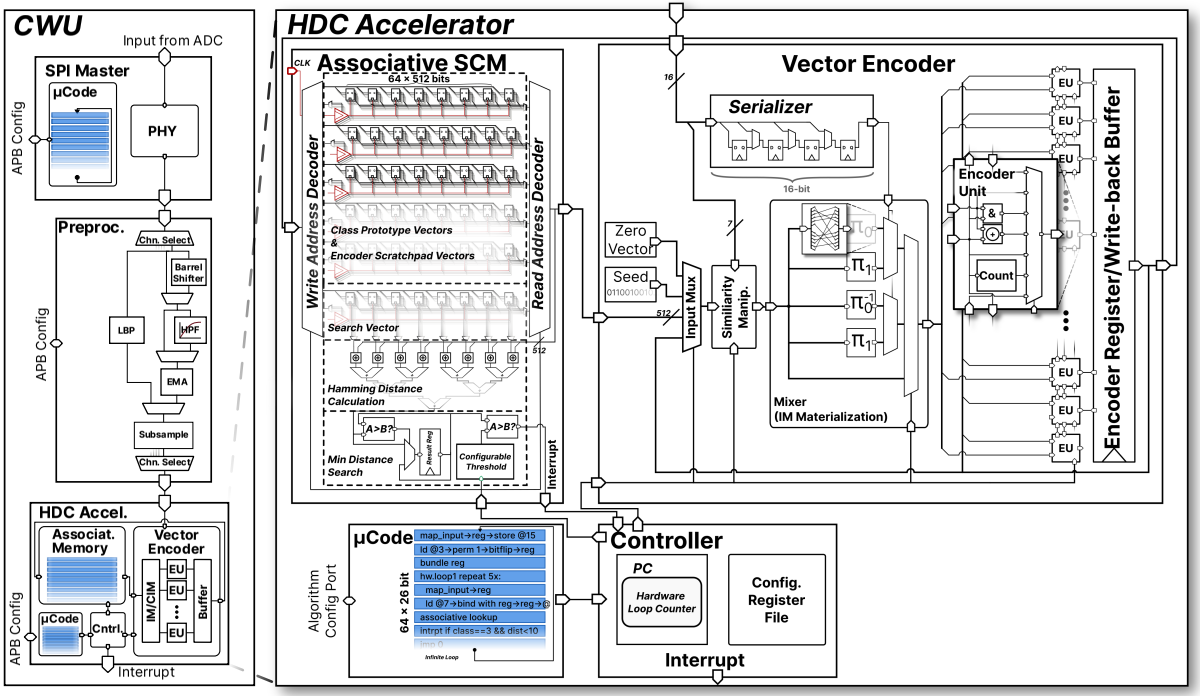


Figure 4.2 – Architectural overview of the Cognitive Wake-up Unit consisting of autonomous SPI master module, configurable preprocessor and HDC Accelerator.

by most applications do not provide a small enough false-positive rate at an acceptable false-negative rate for effective power saving [18]. To deal with those kinds of applications, Vega contains a programmable cognitive wake-up unit (CWU) that performs end-to-end machine learning on external sensor data and triggers the embedded power management unit (PMU) to power up the cluster for more advanced data analytics. The CWU is designed to operate entirely autonomously: after its initial configuration, the CWU continuously processes and classifies external sensor data without any further interaction of the cores.

Figure 4.2 gives a hierarchical overview of the CWU. It consists of three main components previously introduced in chapter 3:

- Programmable SPI master peripheral
- Low-power preprocessor module (Lightweight Data Preprocessing Interface, section 3.6)
- Configurable HDC accelerator (Hypnos, section 3.4)

The dedicated SPI master peripheral acts as the IO interface to interact with external sensors. It supports all four SPI phase and polarity configurations and controls up to four chip-select signals. Complex transaction patterns involving wait cycles and arbitrary read and write transactions with multiple external devices can be programmed utilizing an integrated micro-instruction memory that executes the configured access pattern in an endless loop. The preprocessor module optionally performs lightweight data preprocessing and data format conversion of the digital sensor data on up to eight independent channels. It supports data width conversion, offset removal, low-pass filtering, subsampling, and local-binary-pattern (LBP) filtering [217]. The offset removal and low-pass filters are based on an exponential moving average filter with a configurable decay rate to save area and power.

The core of the CWU is *Hypnos*, a programmable hardware accelerator for HDC. HDC is a brain-inspired computing paradigm for machine learning that operates on high-dimensional holistic representations of the input data [149]. HDC has been proven to achieve competitive accuracy performance in various domains

like biosignals processing [218], DNA sequencing [167], language classification [190], and vehicle classification [191]. With its few-shot learning capability [164] and inherent error-resiliency in the presence of random bit flips [219] HDC is an ideal candidate for an online-trainable wakeup circuit operating at low voltage.

In the context of *Hypnos*, HDC is used to encode a time series from one or several digital sensor channels to a high-dimensional binary *search vector* using a small set of bit-wise and thus well parallelizable operations. Then, this search vector is compared with so-called *prototype vectors* representing the individual classes of interest in associative memory (AM).

In Vega, *Hypnos* operates on 512, 1024, 1536, or 2048-bit HD vectors with a 512-bit wide datapath. The Vector Encoder module is responsible for encoding low-dimensional input data to high-dimensional vectors (HD-vectors). It performs so-called *item memory* (IM) mapping and the operation primitives of HDC like bundling and binding in an iterative manner [149]. Instead of employing a ROM-based IM that stores the mapping from low-dimensional input to pseudo-random HD-vectors, *Hypnos* uses IM "rematerialization" by using a set of four hardwired random permutations. Item memory mapping is thus performed by iteratively applying random permutations on a hardwired pseudo-random seed vector with the bits from the serialized input word acting as select signals to switch between the different permutations. In this manner, *Hypnos* can materialize an IM HD-vector in D cycles, where D denotes the configurable input data width from the pre-processor. While IM maps values from the low-dimensional input space to quasi-orthogonal HD-vectors, continuous item memory mapping (CIM) encodes input values so that low euclidean distance in the input space is mapped to low hamming distance in HD space [165]. In most HDC algorithms, IM mapping is used to encode channel labels and CIM to encode the channel values to preserve the similarity of the values. To support CIM mapping, the vector encoder contains the similarity manipulator module that allows the flipping of a configurable number of bits from an input HD-vector.

The bitwise HDC operations are realized in the Encoder Units (EU), of which *Hypnos* contains 512 instances (one for each bit). Each EU contains the logic for XOR/AND/NOT operation and a saturating bidirectional 8-bit counter that counts the number of encountered

ones and zeros for bundling. A 32 kbit standard-cell based associative memory (AM) can hold up to 16 HD-vectors and acts as both a scratchpad memory to store intermediate HD-vectors from the Vector Encoder and store the final prototype- and search-vector during the associative lookup operation. The vector encoder can fetch HD-vectors from the AM and stores the result of each encoding round in a 512-bit wide register that can be fed back as the source for the next encoding cycle or written back to the AM.

The AM uses latches as storage primitives with a single integrated clock gate (ICG) per ROW acting as a write-enable line. For associative lookup (search for entry with minimal Hamming distance to the search word), the AM sequentially compares each memory row with the search vector and combinationally calculates the Hamming distance. This operation is usually invoked last in most HDC based classification algorithms to compare an encoded search vector against a set of prototype vectors that represent the individual classes of interest [165]. Within our AM architecture, the index and Hamming distance of the most similar item are compared against a configurable threshold and target index. If the result of the lookup is of sufficient similarity to the target index, an interrupt is raised to trigger the wake-up sequence within Vega's PMU.

Since the optimal HDC encoding algorithm, that is, the sequence of encoding operations within the Vector Encoder, is application-dependent, the CWU contains another 64×26-bit SCM to encode the HDC algorithm in a sequence of compact micro-code instructions. The lightweight controller fetches these instructions in an infinite loop and reconfigures AM and Vector Encoder accordingly in each cycle. Thus, the CWU can be configured to run classification algorithms on the preprocessed sensor data and trigger wake-up sequences in Vega's PMU in a fully autonomous manner. The design is intended to operate at very low frequencies in the order of dozens of kHz. Thus the module was realized in UHVT logic to minimize leakage power. The design occupies a total area of 0.147 mm^2 and operates at 0.6V.

Table 4.1 summarizes the power consumption of the CWU within Vega; we measured the CWU's power operating at 32kHz when performing a real-time inference HDC algorithm on data received from 3 SPI peripherals (16 bit, 150 SPS/channel) and at 200 kHz (1 kSPS/channel). At 32 kHz the CWU consumes 2.97 µW of which

	$f_{clk} = 32\,\text{kHz}$	$f_{clk} = 200\,\text{kHz}$
Max. Samp. Rate	150 SPS/Channel	1 kSPS/Channel
$P_{\text{dynamic, datapath}}$	0.99 µW	6.21 µW
$P_{\text{dynamic, SPI pads}}$	1.28 µW	8.00 µW
$P_{\text{leakage, datapath}}$	0.70 µW	0.70 µW
P_{total}	2.97 µW	14.9 µW

Table 4.1 – Implementation details and power consumption of the Cognitive Wakeup Unit

	Cho *et al.* [189] [2, 3]	Giraldo *et al.* [185] [2, 3]	Wang *et al.* [188] [2, 3]	Rovere *et al.* [18] [3]	**Vega CWU**[1]
Applications	VAD	Keyword Spott.	Slope Matching	General Purpose	General Purpose
Technology	180nm	65nm	180nm	130nm	22nm
Power Envelope	14 µW	2 µW	17 nW	2.2 µW	2.97 µW
Classification Scheme	NN	LSTM, GMM	Threshold, Slope	Threshold Sequence	HDC
Area	~3.7 mm^2 [4]	~0.4 mm^2 [4]	~1.8 mm^2 [4]	0.011 mm^2	0.147 mm^2

[1] Power consumption is reported for a compute intensive language classification algorithm and a typical always-on classification algorithm for EMG data.
[2] Although these designs contain an application specific analog frontend (AFE) their digital NN accelerators could potentially be used for other applications in a smart wakeup scenario.
[3] For fair comparison with our digital-only CWU, only the power consumption of the included general-purpose classification logic is considered.
[4] Exact area breakdown is not available thus we estimated the area of the classification logic from the chip micrograph.

Table 4.2 – Comparison of state-of-the-art smart wake-up units

77% is dynamic and 23% leakage. At 200 kHz, overall power consumption increases to 14.9 µW. It is worth noticing that the dynamic power consumption of the CWUs datapath (0.99 µW and 6.21 µW respectively) is about 20% lower than the dynamic power of CWU's SPI pads.

Table 4.2 provides an overview of recently published smart wake-up units. Most of the state-of-the-art units implement application-specific wake-up algorithms such as KWS, VAD, EMG, although the designs mentioned in the table all contain classification circuitry that might be used for other applications [185], [188], [189]. To the best of the author's knowledge, *Hypnos* is, among entirely general-purpose solutions, the one providing the highest level of configurability thanks to its flexible digital implementation together with the versatility of HDC and the preprocessing chain, still featuring a similar power consumption with respect to the only other general-purpose solution implementing a less flexible threshold-sequence based wake-up [18].

Parallel Compute Cluster

The programmable parallel accelerator of the system resides in a dedicated power and clock domain, communicating with the SoC subsystems with two AXI 4 ports (one master and one slave) connected to the SoC interconnect through dual-clock FIFOs. The cluster, built around 9 70kGE 4-pipeline stages RISCV cores, is turned on and adjusted to the required frequency when applications running on the FC offload computation-intensive kernels. The cores share data on a 128 kB shared multi-banked L1 memory, implemented with 16 8kB SRAM cuts, through a 1-cycle latency logarithmic interconnect [203]. Similar to the L2 SoC interconnect, the L1 interconnect implements a word-level interleaving scheme to evenly distribute the requests, minimizing access contentions towards the SRAM banks. The cluster L1 memory can serve 16 parallel memory requests with less than 10% contention rate even on data-intensive kernels, delivering up to 28.8GB/s at 450MHz. The program cache, implemented with latch-based SCM to improve energy efficiency over energy-expensive SRAM cuts for high-intensity activity, is hierarchical and includes 8 512 B private per-core plus 4 kB of 2-cycle latency shared cache to maximize efficiency with data-parallel code. In the common usage scenario.

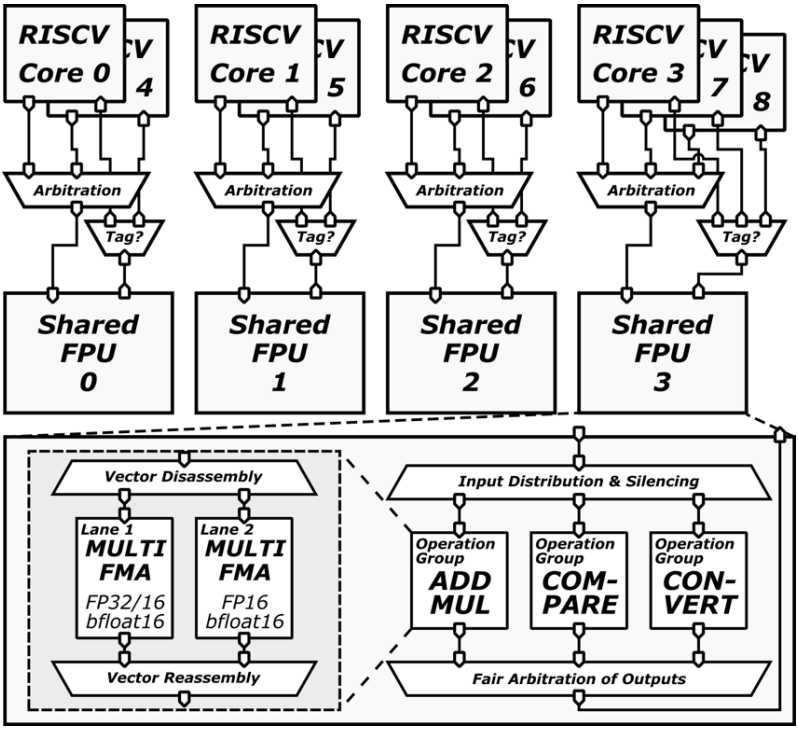


Figure 4.3 – Architecture of the shared multi-precision FPU and its integration in the 9-cores cluster.

the ninth core manages DMA transfers leaving the computational workloads to the other 8 cores. It also includes a larger 1 kB L1 instruction cache. The refill path from L1 to L1.5 can be bypassed to avoid polluting the shared L1.5 cache. Fine-grain parallel thread dispatching is accelerated by a dedicated hardware event unit, which also manages clock gating of idle cores waiting for synchronization and enables resuming execution in 2 cycles.

The RISCV cores used in Vega feature extensions (RVC32IMF-Xpulp) for NSAAs [196], such as hardware loops, post-incremented LD/ST, Single Instruction Multiple Data (SIMD) operations such as dot products operating on narrow 16- and 8-bit data types. The cores share 4 Floating-Point Units (FPUs) supporting FP32, FP16, and bfloat operations (a.k.a. SmallFloat extensions), as well as conversion operations among the different supported formats, including cast-and-pack instructions required to efficiently support packed SIMD vector operations [220]. Some of the key operations present in many NSAA accumulating data in a higher-precision format to avoid loss of precision, such as Multiplication and Fused-Multiply-Add (FMA) can be performed as multi-format instruction, taking the product of two 16-bit operands and returning a 32-bit single-precision result. Division and square root operations are also supported in a stand-alone shared unit (DIV-SQRT).

The FPUs are shared through a low-latency interconnect managing access contention in hardware, allowing to share one FPU among multiple cores in a fully transparent way from a software perspective. As opposed to [216], in Vega we employ a partial interconnect with a static mapping of FPUs to cores, where units 0, 1, 2, 3 are shared among cores 0 & 4, 1 & 5, 2 & 6, and 3 & 7 & 8 such that a core always accesses the same physical FPU instance, as shown in Figure 4.3. This choice limits the flexibility in sharing the FPUs across processors but reduces the complexity of the interconnect towards the FPUs which are on the critical path of the cluster, guaranteeing high compute efficiency thanks to the single-cycle latency behaviour of FP instructions, as demonstrated in section 4.2.3.

CNN inference efficiency is boosted by a cluster-coupled machine learning hardware accelerator (HW Convolution Engine - HWCE) for multi-precision (4b/8b/16b) 3×3 convolution, with 27 MACs in total. The microarchitecture of the HWCE is shown in Figure 4.4.

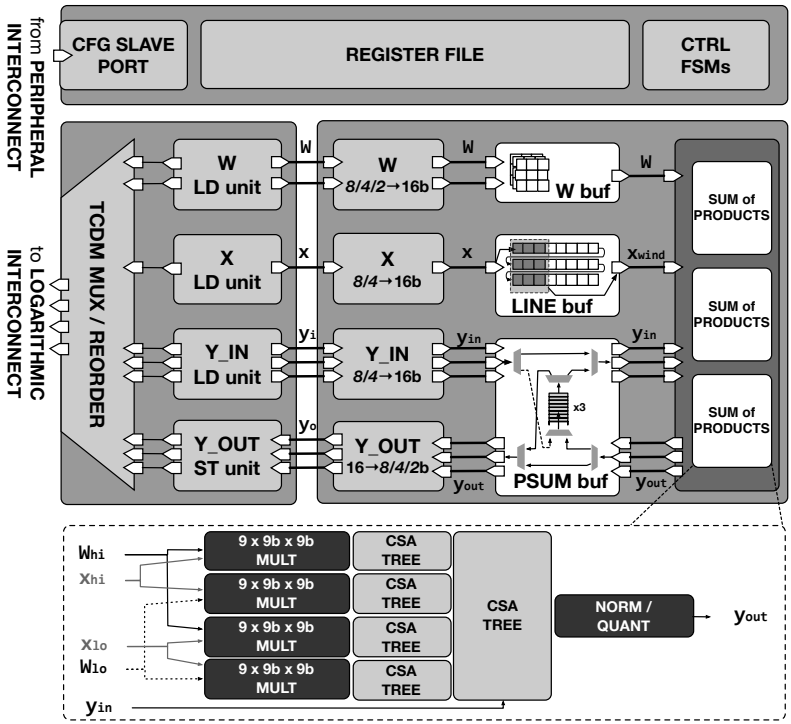


Figure 4.4 – Hardware Convolution Engine (HWCE) microarchitecture.

The HWCE accesses the shared L1 TCDM memory by means of four 32-bit ports on the TCDM logarithmic interconnect. The HWCE load/store units, shown in Fig. 4 on the left side, convert between the TCDM memory protocol and a lightweight streaming protocol using ready/valid handshaking to manage stalls caused by memory contention. Bubbles in the data streams result in additional latency, but do not disrupt the functionality of the accelerator. The HWCE can be programmed via a set of memory-mapped registers via the peripheral interconnect. A register shadowing mechanism enables to offload the next job without colliding with the currently running one. In each accelerator job, the accelerator loads a set of up to three 3×3 filters in an internal weight buffer; then, it starts streaming a continuous stream of input pixels from L1, which – using an embedded line buffer – are used to build a sliding window.

The stationary weights and the input sliding windows are upscaled to 16-bit and combined using three sum-of-product datapaths shared between all precision combinations. In each sum-of-products unit, 16-bits inputs and weights are first split into 9-bits sub-words (adding 1 bit for sign extension). The 9-bit sub-words are first combined along the 3×3 spatial filter dimension, using four carry-save array (CSA) reduction trees. Then, they are combined using another CSA tree. In this way, even if the original inputs were smaller than the 16-bits upscaled version, fine-grain data and clock gating can be employed to disable leaves and branches of the reduction tree, minimizing activity and dynamic power. Similarly, the datapath can also be reconfigured to implement 5×5 convolutions by combining the three sum-of-products units and clock-gating unused units. To further generalize to layers with partial or full input channel reuse, the accelerator includes a partial results FIFO buffer to accumulate convolution outputs from previous input channel contributions, either streamed-in from the L1 ports or from one of three internal partial sum buffers, acting as FIFOs. Symmetrically, the output of the three dot-product units can be either streamed out to L1 (possibly, after undergoing normalization and right-shift) or saved into the partial sum buffers to be re-used in the future. Focusing on filter reuse, this design is particularly effective on popular VGG-style convolutional networks dominated by 3×3 Conv layers [221] – achieving up to 19 MAC/cycle on a 3×3 convolutional layer.

Technology	CMOS 22nm FD-SOI
Chip Area	12mm^2
SRAM Memory	1728 kB
MRAM Memory	4 MB
Equivalent Gates (NAND2)	1.8 Mgates
Voltage Range	0.6 V – 0.8 V
Frequency Range	32 kHz – 450 MHz
Power Range	1.2 μW – 49.4mW

Table 4.3 – Vega SoC Features.

4.2.2 Chip Implementation and Measurements

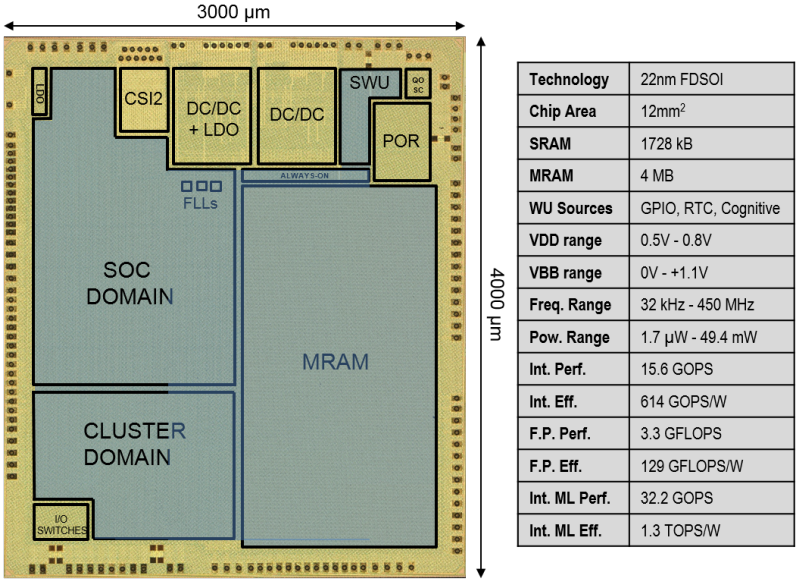


Figure 4.5 – Vega SoC Die Micrograph.

Fig. 4.5 shows the microphotograph of the Vega SoC, along with its four main power domains described in Section 4.2.1. These power domains are managed by an additional always-on power domain operating from 0.6 V to 0.8 V, including commercial off-the-shelf

Instance	Area [mm^2]	Percentage [%]
MRAM	3.59	29.9
SoC Domain	2.69	22.4
Cluster Domain	1.48	12.3
CWU	0.14	1.2
CSI2	0.15	1.2
DCDC1	0.36	3.0
DCDC2	0.36	3.0
POR	0.14	1.1
QOSC	0.03	0.2
LDO	0.03	0.2

Table 4.4 – Vega SoC Area Breakdown.

components: two on-chip DC-DC converters, an LDOs operating at 3.6 V (VBAT), a power management unit used to switch on and off the other domains, a real-time clock (RTC) clocked by 1 MHz oscillator. The wake-up sources for the PMU are an external pad, the RTC, and the Cognitive Wake-up Unit (CWU). Three Frequency Locked Loops (FLLs) reside within the SoC domain and multiply the input clock generated by a kHz crystal oscillator (QOSC) to adjust the cluster domain and SoC domain frequency to the desired values. A third FLL generates the peripheral clock, which is then further divided and adjusted to serve the requirements of the different peripherals. Tab. 4.3 summarizes the main features of the Vega SoC, while Tab. 4.4 summarizes the area breakdown of its main components. The largest blocks in the design are the 4 MB MRAM, the 1.6 MB of L2 memory within the SoC domain, and the power management IPs. On the other hand, the two programmable accelerators occupy less than 15% of the overall SoC area.

Fig. 4.7 shows the power consumption of the components of the Vega SoC highlighted in Fig. 4.5 in the different power modes supported by the SoC, while Fig. 4.6 shows the matrix multiplication performance of the system in the active modes (FC active and cluster active) for all the supported data formats, from 8-bit integer to single-precision FP. In cognitive sleep mode, the CWU consumes from 1.7μW when operating at 32 kHz to 20.9μW when considering 128 kB of L2 memory to be turned into state-retentive mode. Once the SoC is

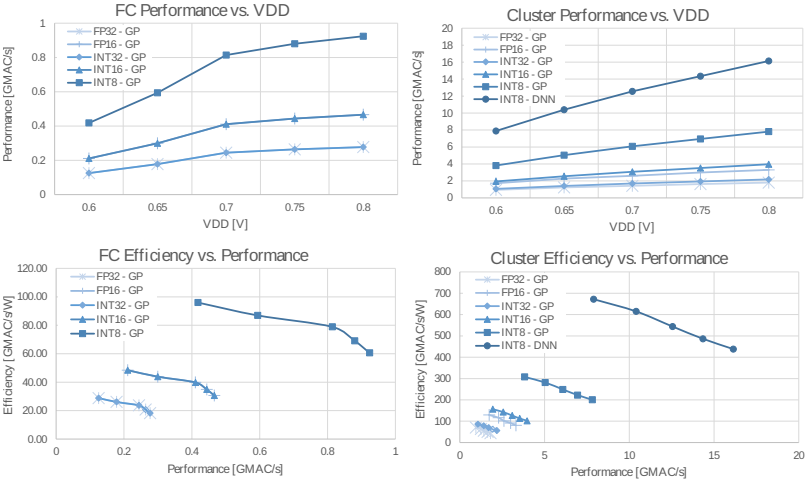


Figure 4.6 – Vega SoC Performance and Efficiency.

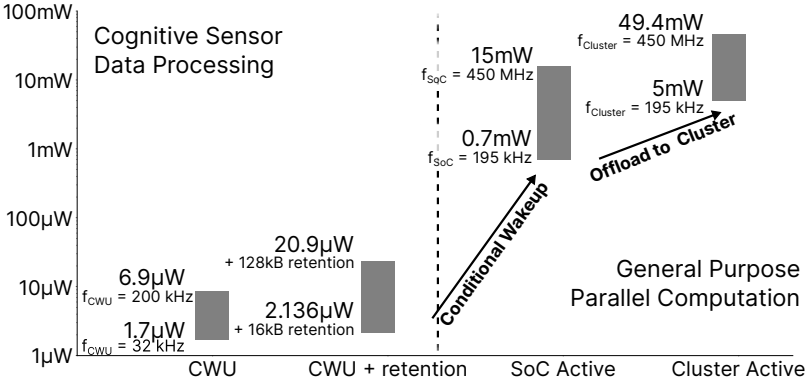


Figure 4.7 – Vega SoC Power Modes and Consumption.

Application	Description	FP intensity
MATMUL	Matrix multiplication (ExG, audio, image)	57%
CONV	Convolution kernel (ExG, audio, image)	55%
DWT	Discrete Wavelet Transform (ExG)	28%
FFT	Fast Fourier Transform (ExG, audio)	63%
FIR	Finite impulse response filter (ExG)	64%
IIR	Infinite impulse response filter (ExG)	46%
KMEANS	Unsupervised algorithm for data clustering (audio, image)	83%
SVM	Supervised algorithm for data classification (audio, image)	35%
Average		***53%***

Table 4.5 – Benchmark suite. For each kernel we report its description and its FP intensity. Main application fields are listed in parentheses.

turned on after a wake-up event, its power consumption goes from 0.7 to 15 mW, delivering up to 0.95 GMAC/s with an efficiency up to 100 GMAC/s/W (8-bit integer). Finally, the SoC can offload highly compute-intensive tasks to the parallel cluster, delivering a much higher performance up to 7.8 GMAC/s with an efficiency of 307 GMAC/s/W on general-purpose processors (8-bit integer), and a performance of 16.1 GMAC/s with an efficiency up to 650 GMAC/s/W on convolutional workloads when HWCE is activated to accelerate the available software programmable processors, all within a power envelope of 49.4 mW. All the experiments were performed running on Vega an 8-bit matrix-multiplication kernel extracted from the PULP-NN library [222] on the software-programmable cores and an 8-bit 3x3 convolution on the HWCE.

4.2.3 Benchmarking

This section presents an extensive benchmarking of the Vega SoC, including both general-purpose NSAA exploiting 32-bit and 16-bit FP arithmetic as well as DNN workloads.

Floating-Point NSAA Workloads

To assess the performance of diverse FP workloads, we considered a benchmark set including digital signal processing, machine learning algorithms, and basic linear algebra subroutines widely used in NSAA fields such as audio, image and ExG processing. To fully exploit the

multi-precision support provided by the shared FPUs, we have extended the C/C++ frontend with two additional data types, namely *float16* and *bfloat16*. The compiler backend lowers scalar operations involving these additional types into the corresponding assembly instruction introduced by the smallFloat extensions (Section 4.2.1). Packed-SIMD operations can be expressed through standard C/C++ operators on GCC vector types. Vectorization of 16-bit data types reduces by a factor of two the execution time of the related FP operations. In addition, there is also a beneficial effect on memory bandwidth utilization since two 16-bit operands are read/written at the same time for each 32-bit memory access.

Table 4.5 reports description and FP intensity each kernel. FP intensity is the percentage of FP operations over the total number of instructions, computed at ISA level (i.e., on the kernel assembly code). Fig. 4.8 illustrates performance and efficiency of each kernel. The performance metric reports how many millions of operations are completed per time unit considering FP32 and FP16 arithmetic for two operating points: 220 MHz, 0.6 V (LV) and 450 MHz, 0.8V (HV). These results demonstrate that the design choice of exploiting shared FPUs is not detrimental to the performance of FP workloads since programs include a mix of FP, ALU, control, and memory operations. The adoption of vectorial FP operations leads to an improvement of 1.46× over scalar ones. This value is lower than the theoretical speedup for the same effect discussed before: FP and memory operations take advantage of vectorization, while ALU and control operations are not affected. Some kernels (MATMUL, FFT, and FIR) are characterized by performance and efficiency gains higher than average values thanks to the use of fused multiply and accumulate instruction allowing to perform 2 FP operations per cycle. Thanks to the architecture design mapping integer and FP registers on a single register file, this gain is maintained for vectorial FP16 versions since programmers can manually optimize the code, including intrinsics for data packing and shuffling of vectors elements, with the final result to reduce the pressure on memory and shared FPUs.

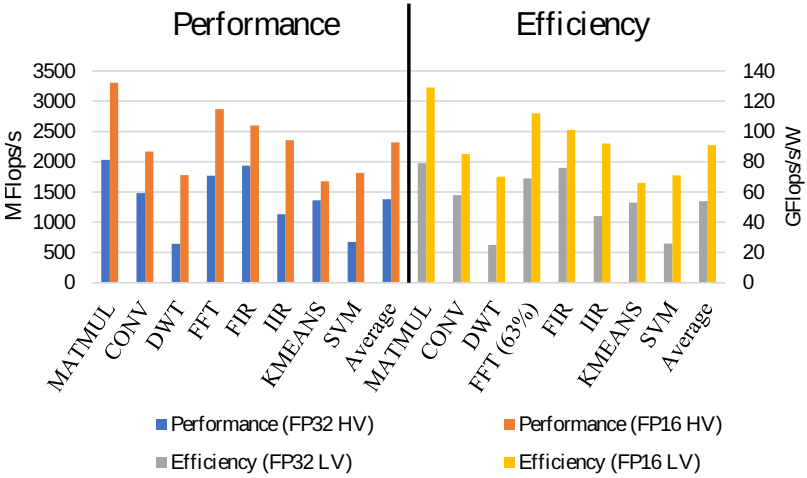


Figure 4.8 – Performance and efficiency of FP NSAA.

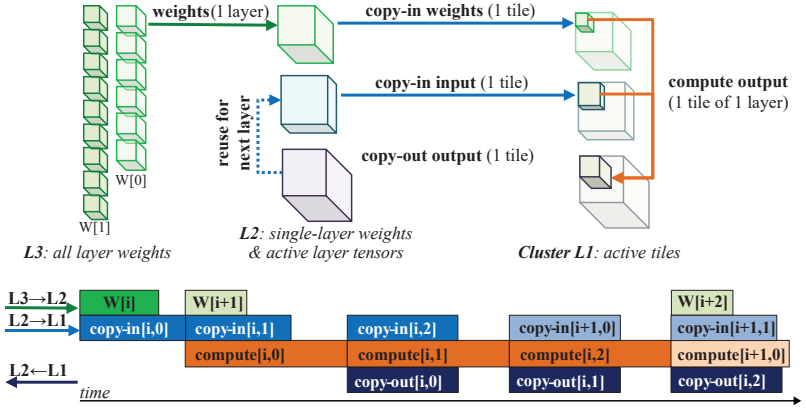


Figure 4.9 – Example of DNN tiling software pipeline, showing the concurrent execution of weight data copies L3→L2 (green); activation data copies L2↔L1 (blue); and compuation (orange) and the typical data flow in case of DNN tiling. Indeces distinguish layers i, $i+1$, etc. and tiles 0, 1, etc..

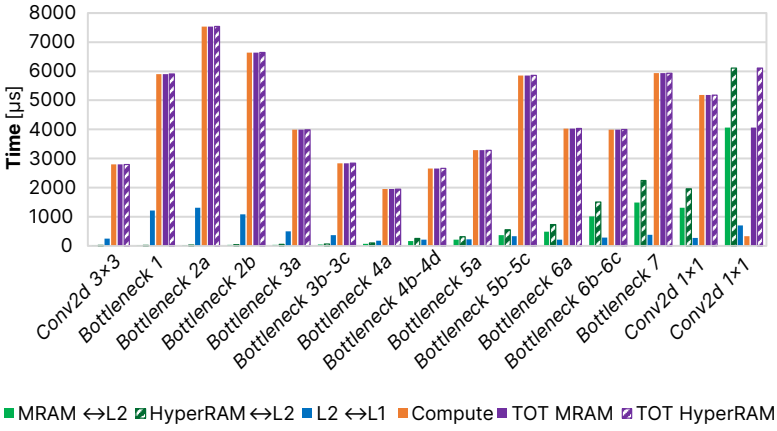


Figure 4.10 – Layer-by-layer compute (orange), L2↔L1 (blue), and L3↔L2 (green) memory transfer latency in a *MobileNetV2* use case. Total latency is shown in violet, highlighting MRAM (solid fill) and HyperRAM (dashed fill) as alternative choices. All layers except for the last one are compute-bound. Data reported in the nominal operating point with $f_{SOC} = 250\,\text{MHz}$, $f_{CL} = 250\,\text{MHz}$.

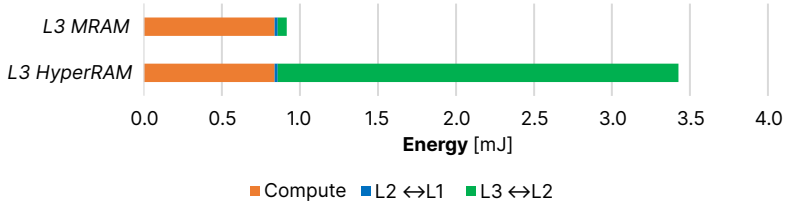


Figure 4.11 – Comparison of total *MobileNetV2* inference energy using weights allocated on external HyperRAM versus on-chip MRAM.

	Bandwidth [MB/s]	*Access Energy [pJ/B]*
HyperRAM ↔ L2	300	20
MRAM ↔ L2	200	880
L2 ↔ L1	1900	1.4
L1 access	8000	0.9

Table 4.6 – Comparison between the various data transfer channels used within a typical DNN execution in terms of available bandwidth and energy per byte.

DNN NSAA Workloads and Data Flow

The architecture of the Vega SoC is designed to be able to deploy realistically sized DNN for full on-chip inference, taking advantage of the MRAM for weight storage. The general data flow for DNN inference is shown in Fig. 4.9, focusing on a purely software execution pipeline based on the PULP-NN library, using 8-bit integers for all tensors (weights and activations). Weights for all layers in the network are stored either in the on-chip MRAM (4 MB) or in an off-chip HyperRAM module – both accessible via the I/O DMA, while intermediate activation tensors are allocated in the L2 shared memory (1.5 MB) and immediately deallocated after they are consumed by the following layer. To enable computation on the cluster, both weights and input activation have to be divided into tiles that fit within the 128 KB of cluster L1 shared memory [223]; 8 cores of the cluster are employed for actual computation, while the ninth is used as an *orchestrator core* to manage data tiling and schedule data movement using the cluster DMA. Computation is organized in a software pipeline with four stages, as shown in Fig. 4.9:

1. *copy weights MRAM/HyperRAM→L2*: weights for a full layer of the network are moved from MRAM/HyperRAM to L2 using the I/O DMA, programmed by the RISC-V FC core.
2. *copy-in input activations, weight L2→L1*: the cluster orchestrator core schedules the copy of a weight and input tile from L2 to L1 using the cluster DMA.
3. *compute*: the 8 cluster compute cores, working entirely on 1-cycle latency L1, consume an input tile and a weight tile to produce a weight tile. We employ the PULP-NN library

	ImageNet	Latency [ms]		Energy [mJ]		MMAC	Parameters [KB]	MRAM up to layer
	Top-1 Acc. [%]	*SW*	*HWCE* (speedup)	*SW*	*HWCE* (eff. gain)			
RepVGG-A0	72.41	358	118 (3.03×)	8.5	4.4 (+93%)	1389	8116	stage 4, layer 12
RepVGG-A1	74.46	610	200 (3.05×)	13.0	7.4 (+76%)	2364	12484	stage 4, layer 6
RepVGG-A2	76.48	1320	433 (3.05×)	25.7	15.8 (+63%)	5117	24769	stage 4, layer 3

Table 4.7 – Vega performance and energy efficiency on RepVGG-A using SW or HWCE-based acceleration

[222] capable of achieving up to 15.5 MAC/cycle on 8 cores. 3×3 convolutional layers can alternatively employ the HWCE accelerator, achieving up to 27 MAC/cycle.

4. *copy-out output activations L1→L2*: the cluster orchestrator core schedules the copy of the output tile from L1 to L2 using the cluster DMA.

As shown in Fig. 4.9, these four stages employ double-buffering and are fully overlapped so that throughput is dominated by the slowest stage. DORY [223] is used both to calculate data tiling solutions fitting the memory constraints at all stages and to generate the orchestrator code.

To provide further insight into the data tiling scheme, we measured the bandwidth and the energy per byte for each of the data transfers described above. MRAM and HyperRAM results are measured on the silicon prototype, while L2/L1 accesses are estimated with power analysis using the final silicon netlist. The results are shown in Table 4.6, in the nominal operating point ($Vdd_{SOC} = 0.8\,\mathrm{V}$, $f_{SOC} = 250\,\mathrm{MHz}$, $f_{CL} = 250\,\mathrm{MHz}$). MRAM and HyperRAM provide similar bandwidth, but thanks to on-chip integration, MRAM provides over 40× better energy efficiency. SRAMs (L2/L1) provide much higher bandwidth at a very low energy/byte but at a steep area cost and without state retention.

As a complete case study for DNN inference on the Vega SoC, we selected *MobileNetV2* [140]: a widely used network topology for computer vision on mobile devices, used both for classification and object detection. The MobileNetV2 template is very flexible, and it is also often employed as a template for tasks unrelated to vision [224]–[226]. The central block of MobileNetV2 is called a *BottleNeck* and it consists of a sequence of three layers: a 1×1 Convolutional *expansion* layer, a 3×3 Depthwise Convolution layer, and a 1×1 Convolutional *projection* layer. Additionally, the input of the expansion layer may be connected to the output of the projection

layer by means of an additive residual connection. For our experiments, we employed the standard MobileNetV2 with depth multiplier 1.0 and input size 224×224, which employes a total of 16 bottleneck layers with 7 different parameter combinations, plus 3 other layers at the front and back end of the network. Fig. 4.10 reports the layer-wise execution time in microseconds when running on Vega with the data flow explained previously, without using the HWCE. Layers and BottleNecks in the front of the network tend to have more intensive transfers for activations (because their size is larger than the others), resulting in more L2↔L1 traffic. Conversely, in the back end of the network MRAM transfers, caused by larger weight sizes, are more relevant. Nevertheless, all layers except for the final one are compute-bound by a considerable margin.

In Fig. 4.11, we consider the full inference compute time and energy for MobileNetV2, comparing fully on-chip execution with weights on MRAM and the "legacy" flow using HyperRAM for weights. We observe that the time per inference is essentially the same, and it is compatible with real-time computation at more than 10 frames per second. The small difference of 3 ms is related exclusively to the final layer. This is because, as shown in Fig. 4.10, all layers apart from the final 1×1 convolution are compute-bound by a significant margin: the 50% bandwidth improvement enabled by the MRAM, therefore, applies only to this layer. The substantial difference, however, is related to the much lower energy cost for memory access. Whereas in the legacy flow, HyperRAM accesses account for almost 25% of the overall energy, the capability to store full-network weights on MRAM reduces this cost by a factor of 40×, making it almost negligible compared to computing energy. As a consequence, the total energy per inference drops by 3.5× – from 4.16 mJ to 1.19 mJ.

The HWCE engine included in Vega is not designed to operate efficiently on networks dominated by 1×1 convolutions, such as MobileNet-V2, where the combined effect of parallel execution and RI5CY extensions deliver very high software throughput. In that use case, employing the HWCE on 3×3 depth-wise convolutions would improve their speed by a factor of ∼3×, but lead to a modest ∼5% speedup on the overall network. On the other hand, VGG-style networks dominated by 3×3 Conv layers are ubiquitous in real-world applications and are currently experiencing a resurgence of popularity in the DL community [221]. On such networks, the HWCE can

	RISC-V VP Schmidt et al. ISSCC 2021 [227]	*SleepRunner Bol et al. JSSC 2021 [228]*	*SamurAI Miro-Panades et al. VLSI 2020 [183]*	*Mr. Wolf Pullini et al. JSSC 2019 [216]*	*GAP8 Flamand et al. ASAP 2018 [229]*	***Vega (this work)***
Technology	FinFET 16nm	CMOS 28nm FD-SOI	CMOS 28nm FD-SOI	CMOS 40nm	CMOS 55nm	CMOS 22nm FD-SOI
Die Area	24 mm^2	0.68 mm^2	4.5 mm^2	10 mm^2	10 mm^2	12 mm^2
Type	Vector Processor	MCU	Heter. MCU	Parallel MCU	Parallel + Heter. MCU	Parallel + Heter. MCU
Applications	DSP	IoT GP	IoT GP + NSAA +DNN	IoT GP + NSAA	IoT GP + NSAA + DNN	IoT GP + NSAA + DNN
CPU/ISA	RV64GC	CM0DS Thumb-2 subset	1x RI5CY RVC32IMFXpulp	9x RI5CY RVC32IMXpulp	9x RI5CY RVC32IMXpulp	10x RI5CY RVC32IMFXpulp + SF
Embedded SRAM (State Retentive)	4.5 MB	64 kB s.r.	464 kB 40 kB s.r.	64 kB (L1) 512 kB s.r. (L2)	64 kB (L1) 512 kB (L2)	128 kB (L1) 1600 kB s.r. (L2)
Embedded NVM	-	-	-	-	-	4 MB MRAM
Wake-up Sources	-	WiC	WuR, RTC, Int, GPIO	GPIO, RTC	GPIO, RTC	GPIO, RTC, Cognitive
Sleep Power SRAM Ret. Slp. Power (St. Ret. SRAM)	-	5.4 μW 9.4 μW (64 kB s.r.)	- 6.4 μW (40 kB s.r.)	72 μW 76.5 - 108 μW (32 kB - 512 kB s.r.)	3.6 μW 30 μW (512 kB s.r.)	1.7 μW (CWU) 2.8 - 123.7 μW (16 kB - 1.6 MB s.r.)
INT Precision *FP Precision*	64 FP64, FP32, FP8	32 -	8, 16, 32 -	8, 16, 32 FP32	8, 16, 32 -	8, 16, 32 FP32, FP16, bfloat
Supply Voltage	0.55 - 1V	0.4 - 0.8V	0.45 - 0.9V	0.8 - 1.1V	1 - 1.2V	0.5 - 0.8V
Max Frequency	1.44 GHz	80 MHz	350 MHz	450 MHz	250 MHz	450 MHz
Power Range	n.a. - 4 W	5.4 - 320 μW	6.4 μW - 96 mW	72 μW - 153 mW	3.6 μW 75 mW	1.7 μW - 49.4 mW
[1,4] *Best Int Perf* [1,4] *Best Int Eff* [1,4] *@ Perf*	-	31 MOPS (32b) 97 MOPS/mW (32b) @ 18.6 MOPS (32b)	1.5 GOPS 230 GOPS/W @ 110 MOPS	12.1 GOPS 190 GOPS/W @ 3.8 GOPS	6 GOPS 79 GOPS/W @ 3.5 GOPS	15.6 GOPS 614 GOPS/W @ 7.6 GOPS
[2,4] *Best FP32 Perf* [2,4] *Best FP32 Eff* [2,4] *@ Perf*	n.a 92.3 GFLOPS/W n.a	-	-	1 GFLOPS 18 GFLOPS/W @ 350 MFLOPS	-	2 GFLOPS 79 GFLOPS/W @ 1 GFLOPS
[2,4] *Best FP16 Perf* [2,4] *Best FP16 Eff* [2,4] *@ Perf*	368.4 GFLOPS 209.5 GFLOPS/W @ 73 GFLOPS	-	-	-	-	3.3 GFLOPS 129 GFLOPS/W @ 1.7 GFLOPS
[3,4] *Best ML Perf* [3,4] *Best ML Eff* [3,4] *@ Perf*	-	-	36 GOPS 1.3 TOPS/W @ 2.8 GOPS	-	12 GOPS 200 GOPS/W @ 7 GOPS	32.2 GOPS 1.3 TOPS/W @ 15.6 GOPS

[1] 2 OPs = 1 8-bit MAC on MatMul benchmark unless differently specified.
[2] 2 FLOPSs = 1 FMAC on MatMul benchmark unless differently specified.
[3] 8-bit ML Workloads.
[4] Execution from SRAM.

Table 4.8 – Comparison With State Of The Art

provide a substantial performance boost with respect to software-based execution, thanks to its exploitation of filter data reuse.

To better showcase this point, in Table 4.7 we present results in terms of energy and latency for three networks of the recently presented RepVGG-A0 [221] family, which have been demonstrated to be highly competitive with the State-of-the-Art in terms of trainability, speed, and accuracy. They are divided into 5 stages composed of 1, 2, 4, 14, and 1 layers, respectively – all implemented as 3×3 convolutions, plus a final fully connected layer. All three networks are too big to fit entirely within the on-chip MRAM, so we revert to greedy allocation – we keep early layer weights in MRAM until they fit (as reported in the rightmost column of Table 4.7), to exploit its higher energy efficiency, and then we allocate back-end layers in HyperRAM.

The results are presented for both HWCE-based and SW-based computation, using the same PULP-NN layers of the MobileNetV2 case study in the latter case. Almost all layers are compute-dominated, except for the final fully connected layer. In such conditions, HWCE-

based execution delivers a 3× speedup over SW-based; and a 60-90% boost in system-level energy efficiency, depending on how much is the energy impact of HyperRAM traffic.

4.2.4 Comparison With SoA

Table 4.8 shows a comparison with a wide range of programmable embedded computing platforms, including RISC-V based vector processors for transprecision FP computations [227] and low-power IoT computing systems [228] exploiting either parallelism [216], heterogeneity [183], or both [229] to address the high computing requirements of emerging NSAA applications and DNNs.

The work presented in [227] proposes a RISC-V vector processor composed of 8 clusters, each one including a 64-bit scalar core coupled with a vector accelerator supporting double-, single-, and half-precision FP operations, with a maximum of 8, 16, and 32 operations per cycle, respectively. While the absolute performance of [227] is much higher than Vega, thanks to the significantly larger area and higher operating frequency, its peak energy efficiency on a matrix multiplication benchmark is only 1.62x and 1.16 better than the one of Vega for FP16 and FP32 operation, respectively, despite the more scaled technology node. Moreover, the efficiency of a vector processor is well known to significantly degrade when dealing with small datasets and irregular patterns typical of NSAA. In this work, we demonstrate leading-edge FP efficiency on a wide range of 32-bit and 16-bit FP NSAA thanks to the flexibility of the proposed software-programmable cluster, as shown in Section 4.5.

With respect to traditional fully programmable IoT endnodes such as [228], which is representative for a wide range of MCUs based on CortexM0 or similar low-cost processors, Vega delivers orders of magnitude better performance and efficiency in active mode, enabling the execution of complex NSAA not feasible on such tiny systems. However, it should be noted that the main focus of most of research work on IoT MCUs is on optimization of low-power states such as state-retentive SRAMs or minimization of deep-sleep power and wake-up time from deep sleep, addressed in Vega with commercial off the shelf IPs exploiting standard techniques not being a key contribution of this work.

With respect to the more closely related works, namely multi-core IoT end-nodes [216], [229], the proposed SoC delivers more than 1.3× better peak performance and more than 3.2× better peak efficiency on non-DNN NSAA workloads. The main improvements for these workloads come, on top of the more scaled technology node, from the more optimized integration between the prefetch buffer of the the cores and the hierarchical instruction cache. This cuts the critical path of the design with no loss in functional performance, lowering the power consumption and improving energy efficiency. For what concerns FP support, Vega is much more flexible and efficient than Mr.Wolf, delivering 2× better peak performance 4.3x better peak efficiency on 32-bit single-precision FP workloads, thanks to the highly optimized, shared FP unit offering support for single-cycle NSAA operations, such as fused multiply and accumulate. Moreover, Vega offers more flexibility in terms of support for low-precision FP formats (i.e., 16-bit and bfloat) further gaining performance and efficiency. With respect to the most efficient hardware-accelerated IoT end-nodes [183], Vega achieves similar energy efficiency on DNN inference workloads at 5.5× better performance. On non-DNN, NSAA workloads, Vega achieves 10× and 2.5× higher performance and energy-efficiency despite the 2 platforms employing the same RI5CY core [196]. This gain is achieved thanks to the architectural efficiency of the parallel computing cluster over sequential solutions, also demonstrated in Fig. 4.6.

Finally, to the best of the authors' knowledge, the proposed SoC is the only IoT end-node featuring a configurable 1.7 μW cognitive wake-up unit capable of fully on-chip execution of state-of-the-art mobile DNNs, such as MobileNetV2 and RepVGG, from non-volatile memory support.

4.2.5 Conclusion

We presented Vega, an always-on IoT end-node SoC featuring a 1.7 μW fully retentive cognitive wake-up unit coupled with a power/performance/precision scalable SoC. The proposed SoC can achieve up to 32.2 GOPS (@ 49.4 mW) peak performance on NSAAs, including mobile DNN inference, exploiting 1.6 MB of state-retentive SRAM, and 4 MB of non-volatile MRAM. To meet the performance and flexibility requirements of NSAAs, the SoC features 10 RISC-V cores:

one core for SoC and IO management and a 9-cores cluster supporting multi-precision SIMD integer and FP computation. Two programmable ML accelerators boost energy efficiency in sleep and active state, respectively. The proposed SoC can deliver a peak performance of 32 GOPS with an efficiency up to 1.3TOPS/W. The proposed SoC is capable of fully on-chip execution of state-of-the-art mobile DNNs such as MobileNetV2 and RepVGG-A0 at 1.19 mJ/Inference and 4.4 mJ/Inference, respectively.

4.3 A Non-Volatile First-Level Memory Subsystem: The Siracusa SoC

4.3.1 Introduction

Extended Reality (XR) has become increasingly popular in recent years, with applications in entertainment, education, healthcare, and more. However, mass adoption of XR technology still faces several challenges in meeting stringent latency and power consumption requirements.

On-sensor computing is a promising approach to overcome a key challenge posed by modern XR devices: minimizing power consumption for low-latency, human-in-the-loop processing of sensor streams and feedback control [7]. In on-sensor computing, data acquired from the sensor is processed on a tightly integrated compute layer rather than being sent off-chip to a downstream computing device. For XR applications, on-sensor data processing allows extracting semantically richer information from raw sensor data using machine learning, reducing the data transfer latency and energy while increasing user privacy.

However, the amount of on-chip memory (up to a few MiB), limitied by the area of constraints, and the tightly bound power envelope (a few hundred mW) of on-sensor computing platforms impose severe limitations on the machine learning-based workloads, and require low/mixed-precision neural networks to ensure adequate quality of results and optimal performance [230].

Moreover, while information extraction is nowadays primarily achieved with machine-learning-based processing pipelines, conventional signal processing algorithms, e.g., interpolation and compression, remain crucial functional modules of visual processing pipelines. In XR applications, such algorithms are often used to compute the ROIs that are transmitted between the sensor and an external aggregator [231]: their suboptimal execution limits the overall end-to-end application performance.

In this work, we present *Siracusa*, a RISC-V SoC for on-sensor visual processing targeting XR workloads fabricated in TSMC 16 nm technology. Siracusa introduces a low-power heterogeneous architecture for complex visual processing tightly coupled with the image sensor,

to enable scalable multiple-smart sensors integration in XR glasses. Siracusa overcomes the challenges posed by the requirements of XR applications for on-sensor computing platforms by tightly integrating a state-of-the-art fine-grained mixed-precision neural network accelerator, N-EUREKA, with an octa-core RISC-V compute cluster optimized for signal processing applications with extensive support for FP multi-precision computation. As a key innovation, Siracusa integrates a configurable weight memory subsystem with a virtual paging extension that enables efficient prefetching of weight data and minimizes copy overheads in low-latency DNN inference, as required for human-in-the-loop XR applications.

In detail, the main contributions of this work are as follows:

- We present the integration of an 8-core 120 GOp/s @ 530 MHz RISC-V cluster with instruction extensions enabling efficient image and signal processing with N-EUREKA, a weight-precision-tunable cooperative CNN accelerator achieving a peak energy efficiency of 9.9 TOp/J @ 1280 GOp/s.
- We introduce a novel memory hierarchy that combines high-bandwidth on-chip storage for neural network weights with a configurable virtual paging system, enabling seamless prefetching of weights for machine learning applications. We demonstrate that this design enables efficient tiled processing of complex neural networks, increasing inference efficiency of individual weight-intensive layers by 3.2 × compared to a shared-bandwidth configuration for weights and activations.
- We evaluate Siracusa in an on-sensor compute scenario by deploying a state-of-the-art hand detection workload. We demonstrate that leveraging the novel memory hierarchy increases the average inference throughput and energy efficiency by 2.5× and 1.8×, compared with a baseline shared-bandwidth architecture, achieving an 8-bit inference energy efficiency of 0.76 TOp/J.

Contributions The Siracusa SoC is a joint effort between ETH Zurich, Meta Reality Labs and Numem Inc. As such many people have been involved in the realization of this project. Project conception and planning were led by Davide Rossi, Francesco Conti, Barbara de Salvo

and Luca Benini. The N-Eureka hardware accelerator was developed by Arpan Prassad and Francesco Conti based on the original IP developed by Gianna Paulin. The STT-MRAM IP has been contributed by NuMem Inc., while I performed the system integration and the design of the tightly-coupled NVM subsystem around the memory macros. The SoC architecture and system integration was jointly developed by Alfio Di Mauro and me. The chip-bring up, firmware development and silicon measurements were executed by myself. Evaluation and write-up were done by myself, Moritz Scherrer, Francesco Conti, Arpan Prassad and Luca Benini. At the time of submitting this thesis, the content of this section has been accepted for presentation and subsequent publication at the 49th IEEE European Solid-State Circuits Conference 2023 in Lisbon[2].

4.3.2 Architecture

The Siracusa SoC integrates two domains: *I/O* and *Heterogeneous Cluster*, shown in Figure 4.12. The I/O domain consists of a 32-bit RISC-V fabric controller (FC) core, responsible for management tasks, alongside 2 MiB of L2 SRAM, and a rich set of standard peripherals including QSPI, I^3C, Hyperram, and UART, as well as a dedicated camera interface with the image sensor to be co-packaged for in-sensor computing [7], [232]. Located in its own power and clock domain, the Heterogeneous Cluster domain consists of a 8-core RISC-V cluster, the N-EUREKA accelerator, and the tightly-coupled Weight Memory subsystem.

RISC-V Cluster

The cluster features eight RISC-V cores implementing the *RV32IMCFXpulpnn* ISA tightly coupled with a dedicated 256 kB L1 scratchpad SRAM. The *Xpulpnn* ISA extensions include DSP instructions such as multiply-accumulate and hardware loop instructions, as well as SIMD operations such as dot-products with implicit load semantics for power-of-two bit widths (2-bit, 4-bit, 8-bit, 16-bit). To accelerate a wide range of visual processing applications beyond neural network inference, the cores feature dedicated multi-precision FPUs with FP8,

[2] ©2023 IEEE. Reprinted with permission.

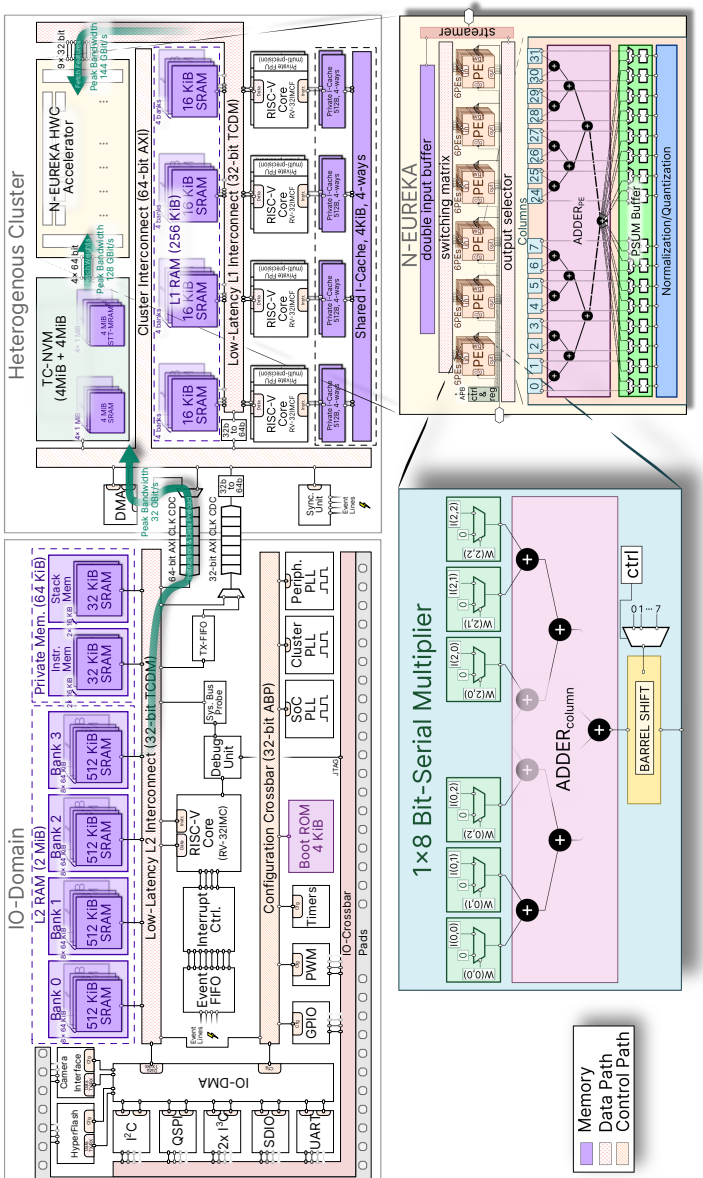


Figure 4.12 – Architectural overview of the Siracusa SoC consisting of IO-Domain (upper left), heterogenous cluster of 8-RISC-V cores (upper right) which includes the N-EUREKA bit serial hardware accelerator (bottom) which is attached to the tightly-coupled NVM subsystem consisting of 4 MiB SRAM and 4 MiB of non-volate STT-MRAM

FP16, FP16Brain, and FP32 support. High instruction and data memory energy efficiency are achieved thanks to a 2-level (4 KiB core-private, 4 KiB shared) hierarchical instruction cache with integrated prefetcher and a single-cycle latency interconnect towards tightly-coupled data memory.

N-EUREKA

Besides its compute cluster, Siracusa features a highly flexible mixed-precision ML accelerator called N-EUREKA. N-EUREKA is an 8-bit activation, 2 – 8-bit weight precision-scalable accelerator targeted at building blocks of modern CNNs, particularly 3×3, 1×1 and depthwise 3×3 layers. The architecture of N-EUREKA, shown in Fig. 1 (bottom), is built on a stationary double-buffered input buffer and an array of 6×6 Processing Elements (PEs), each responsible for the calculation of a distinct output activation. Each PE contains 32 columns, and each column combines 9 1×8-bit serial multipliers, receiving inputs dispatched from the input buffer according to the operating mode, and weights streamed from the system weight memory. By decomposing each layer in a combination of 1×8-bit dot-products, N-EUREKA maximizes performance and energy efficiency in each mode. Contributions from each column are summed up (in 3×3 and 1×1 modes) into a single output-channel-wise 32-bit accumulator or marshaled to separate accumulators (in depthwise 3×3 mode). N-EUREKA also supports per-channel activation scaling, biasing and 8-bit requantization, and automatic reloading to combine multiple jobs into a single complex task.

Thighly-coupled non-volatile memory

To sustain high-throughput computation in N-EUREKA, Siracusa closely couples the accelerator directly to the L1 memory, thus avoiding the need for superfluous memory transfers of activations between cluster cores and the accelerator. For network weights, the SoC features a large, power-optimized weight memory, consisting of 4 MiB of SRAM as well as 4 MiB of NVM, which is not characterized in this work. While networks weights may also be allocated in the L1 memory, storing them in the weight memory subsystem increases bandwidth

considerably, as N-EUREKA is tightly coupled to it with uncontended parallel access to all 4×64-bit read-write ports of each domain, whereas L1 bandwidth is used for activations and is shared with the RV32-cores. To achieve a maximum sustained bandwidth of 128 Gbit/s from either the non-volatile memory block or the SRAM, the data path has been carefully pipelined with just enough stages to completely hide any access latency towards the NVM given the regular weight access pattern of N-EUREKA. Access from the cluster side and N-EUREKA is arbitrated by an at-runtime configurable arbitration logic that provides per-transaction exclusive access to either of the two memory domains on either a hard priority policy or fair bandwidth arbitration.

Besides direct memory mapping of the combined 8 MiB address space, the weight memory subsystem can operate in a lightweight software-assisted virtual memory mode where N-EUREKA operates on virtual 4 MiB pages. A small page handling circuitry maps N-EUREKA's transaction to either of the two physical memory pages (residing in SRAM and non-volatile memory domain) by comparing the address prefix with the two live page index registers that are exposed via the weight memory subsystem's configuration interface. In virtual memory mode, if the page index associated with an accelerator transaction matches either of those live page index registers, the transaction is forwarded to the corresponding memory domain. In case of a page miss, i.e. neither of the two-page index registers are matching, the transaction is stalled and an interrupt is raised towards the FC. Firmware running on the fabric controller can instrument the IO-DMA to perform a page swap through the 32-bit AXI CDC concurrent to the L2-L1 DMA transfers of activations via the separate 64-bit CDC port. Page swapping can thus be performed without limiting the L2-L1 bandwidth required for tiling-based inference of large networks. Once finished, the FC updates the page index register which will unblock the stalled transaction, which is completely transparent to N-EUREKA. The system also supports proactive page swapping leveraging the typically deterministic weight access pattern in CNN workloads; Apart from the page-miss interrupt, the fabric controller is also asynchronously notified of page "switches", when N-EUREKA switches from accessing one live page to the other. Depending on the compute intensity of the workload, this allows for partial or even complete overlap of the page swap procedure with N-EUREKA's

computation. The software-assisted virtual memory feature thus allows for transparent network reconfiguration with negligible increase in overall circuit area.

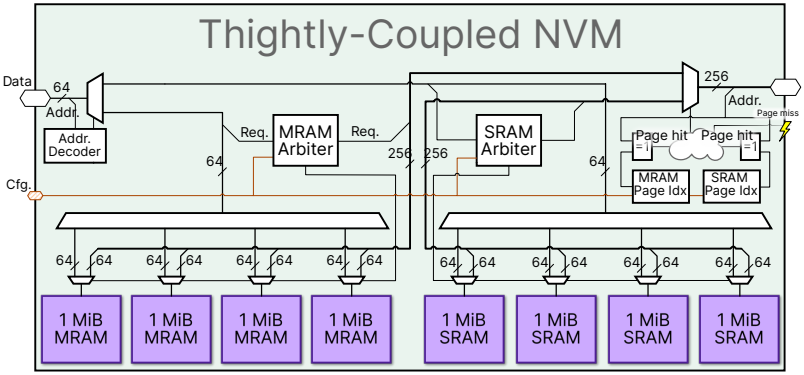


Figure 4.13 – Schematic of the tightly-coupled non-volatile memory (TC-NVM) subsystem consisisting of 4 MiB of STT-MRAM and 4 MiB SRAM exposed with a smart-arbitrated cluster- and accelerator acces port.

4.3.3 Results

Siracusa was taped out in 16 nm FinFET technology with a total die area of 16 mm^2, a micrograph of Siracusa is shown in Figure 4.14. At a nominal voltage of 0.8 V, the chip operates at 360 MHz to 530 MHz. At its most efficient operating point at 0.6 V, the cluster boasts a peak energy efficiency of 1.46 TOp/J (2-bit SIMD) while N-EUREKA pushes this number to 9.9 TOp/J (2×8-bit), as shown in Figures 4.15 and 4.16.

To quantify the performance improvements enabled by the additional memory bandwidth afforded by the weight memory subsystem, we deploy individual depthwise and dense convolution layers and measure the gain in performance and energy efficiency. The configurations used in these experiments are a 3×3 depthwise convolution with 64 channels and a pointwise and 3×3 dense convolution with 256 input and output channels. All convolutions operate on a 6×6 input feature map. For each convolution, we implemented three tiling strategies. In the reference strategy, we

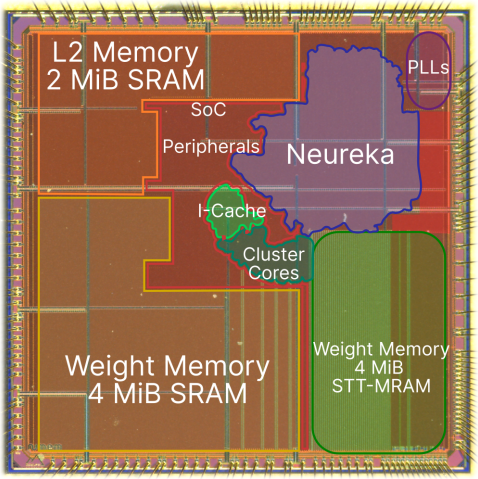


Figure 4.14 – Annotated chip micrograph of Siracusa.

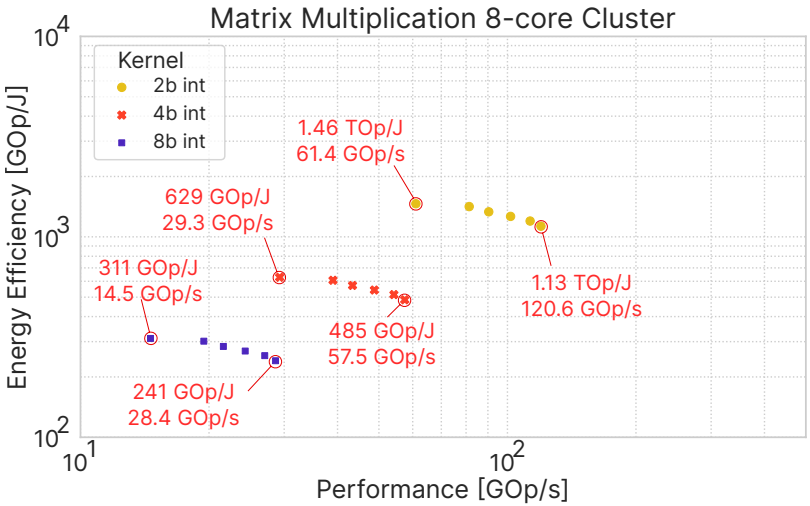


Figure 4.15 – Peak energy efficiency versus throughput along the voltage/frequency Pareto frontier of the cluster, executing matrix multiplication in different precisions. The data was measured at room temperature and operating voltages ranging between 0.6 V to 0.8 V and maximal frequency.

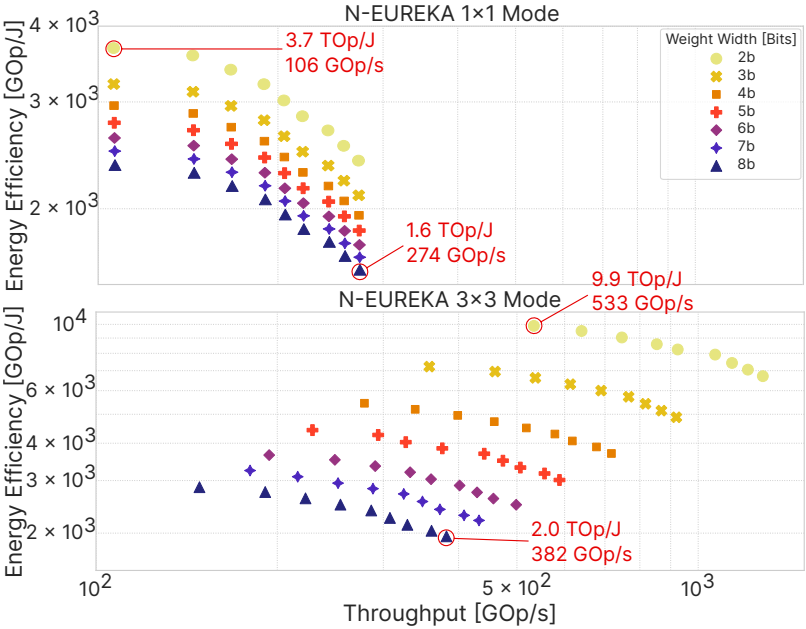


Figure 4.16 – Peak energy efficiency versus throughput along the voltage/frequency Pareto frontier of Neureka in 1×1 and 3×3 mode respectively. The data was measured at room temperature and operating voltages ranging between 0.6 V to 0.8 V and maximal frequency.

store the weights and activations of each layer in the L2 memory and execute optimized code to tile and transfer the inputs and weights, followed by N-EUREKA kernels. This corresponds to the expected performance of a version of Siracusa without our paged weight memory subsystem. The second implementation uses Siracusa's weight memory to store the weights of each test layer but still performs tiling to transfer input and output activations between L2 and L1. Finally, we implement a version of each layer that does not perform tiling and accesses inputs directly from L1, a theoretical ideal implementation.

To evaluate the end-to-end neural network inference performance of Siracusa, we deploy a complete hand detection backbone in 8-bit precision consisting of a MobileNet-v2-inspired feature extraction pipeline, introduced in Han et al. [233], shown in Figure 4.17. VGA (640 × 480 grayscale) resolution camera frames are processed by a CNN-based hand detection algorithm, which infers the position and size of hands in the picture. We use a layer fusion implementation for IRB layers and achieve an average throughput of 168.5 MAC/cycle, at an energy cost per inference of 107.3 µJ and a latency of 692 µs, equivalent to an average energy efficiency of 0.76 TOp/J.

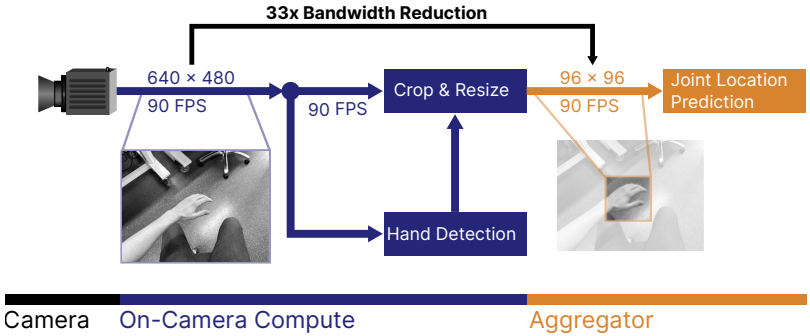


Figure 4.17 – Overview of the hand tracking pipeline. This work implements the hand detection algorithm on Siracusa.

As shown in Figure 4.18, leveraging the weight memory increases energy efficiency by 1.1, 3.2, and 2.1 × for individual depthwise, pointwise, and dense 3×3 convolutions, respectively, which pushes

the performance of dense convolutional layers to within 22 % of their theoretical maximum. Similarly, the throughput and energy efficiency of the end-to-end network are improved by a factor of 2.5 × and 1.8 × compared to an implementation without our paged weight memory subsystem.

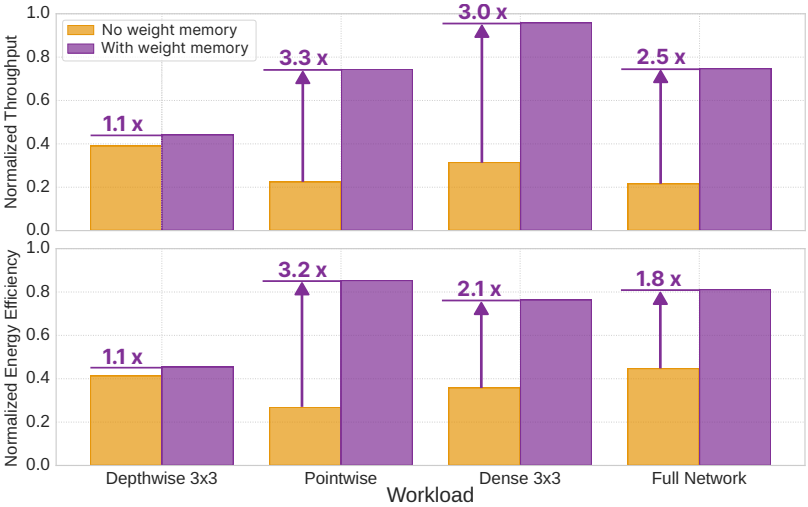


Figure 4.18 – Comparison of the energy efficiency of different tiling strategies of individual layers and the full network. The data was measured at room temperature at an operating voltage of 0.8 V and normalized to an ideal implementation without tiling.

While the depthwise convolutions used in the backbone only make up around 12.5 % of all operations, their relatively lower execution performance limits the overall energy efficiency of Siracusa in a layerwise execution schedule, achieving an energy cost of 159.9 µJ.

To address the bottleneck in the execution of single depthwise layers, we propose fusing the execution of IRB layers for each tile. Layer fusion avoids moving the inputs and outputs of depthwise layers between L2 and L1, which eliminates the underutilization of N-EUREKA due to stalling. We estimate the performance increase of applying layer fusion, by measuring the execution latency of IRB tiles in a layer fusion schedule. By boosting the throughput of depthwise operators by

2.2 × without sacrificing pointwise operator performance, this schedule increases the average throughput to 168.5 MAC/cycle and reduces the energy cost per inference to 107.3 µJ, an improvement of 1.48 ×, equivalent to an average energy efficiency of 0.76 TOp/J.

4.4 Comparison with the State of the Art

A comparison of our design with similar edge computing and on-sensor computing architectures is shown in Table 4.9. The peak energy efficiency of 9.9 TOp/J for dense 3×3 convolutions achieved in this work is 2 × higher than the result reported in [234], and 1.2 × higher than the result reported in [124]. Further, Siracusa meets the stringent XR footprint requirements, while the design proposed in [234] requires an area of 62 mm^2. In terms of general compute performance, Siracusa's optimized RISC-V cores improve on the state-of-the-art by 25 % in terms of integer SIMD performance, and 9 % in terms of fp32 performance. Siracusa's power consumption is below 250 mW, even in the highest performance setting, fully compatible with the in-sensor computing's tight power budget within XRs' glasses.

		SamurAI[183]	Vega[84]	Kraken[235]	Marsellus[124]	Eki et al.[234]	**This work**
Technology		28 nm	22 nm	22 nm	22 nm	22 nm	**16 nm**
Voltage		0.45 V to 0.9 V	0.5 V to 0.8 V	0.5 V to 0.8 V	0.5 V to 0.8 V	0.8 V	**0.6 V to 0.8 V**
Freq. Range		350 MHz	450 MHz	330 MHz	420 MHz	262 MHz	**360 MHz to 530 MHz**
Power Envelope		96 mW	49.4 mW	300 mW	12.8 mW to 123 mW	379 mW	**221 mW**
Area		4.5 mm^2	12 mm^2	9 mm^2	18.7 mm^2	62 mm^2	**16 mm^2**
ISA		1× RV32-IMFC + SIMD	9× RV32-IMFCxPULPv3 + SIMD	8× RV32-IMFCxPULPnnv2 + 1× RV32-IMC	16× RV32-IMFCxPULPnn + RV32-IMFCxPULP	-	8× RV32-IMFCxPULPnnv2 + RV32-IMFC
Application		GP IoT, QNN	GP IoT, QNN	GP IoT, QNN, SNN	GP IoT & AI-IoT	On-sensor Compute	On-sensor Compute, GP IoT
Memory		392 KiB SRAM	1.7 MiB SRAM, 4 MiB STT-MRAM	1.1 MiB SRAM	1.1 MiB SRAM	9 MiB SRAM	6.25 MiB SRAM 4 MiB NVM
General Purpose HW Accelerator		✓	✓	✗	✓	✓	✓
Peak Efficiency Performance/Core	General Purpose Cores int8 / int8	-	1.95 GOp/s	2.75 GOp/s	2.65 GOp/s	-	3.55 GOp/s
	General Purpose Cores int4 / int4	✗	✗	5.75 GOp/s	5.0 GOp/s	-	7.2 GOp/s
	General Purpose Cores int2 / int2	✗	✗	11.3 GOp/s	10.0 GOp/s	-	15.0 GOp/s
	General Purpose Cores fp32 / fp32	-	0.46 GFlop/s	0.33 GFlop/s	0.43 GFlop/s	-	0.47 GFlop/s
	Hardware Accelerator int8 / int8	36 GOp/s	32.2 GOp/s	✗	91 GOp/s	1.2 TOp/s	**381 GOp/s**
	Hardware Accelerator int8 / int4	✗	-	✗	186 GOp/s	✗	717 GOp/s
	Hardware Accelerator int8 / int2	✗	-	✗	373 GOp/s	✗	1280 GOp/s
	General Purpose Cores int8 / int8	-	0.61 TOp/J	0.46 TOp/J	0.88 TOp/J	-	0.31 TOp/J
	General Purpose Cores int4 / int4	✗	✗	0.9 TOp/J	1.66 TOp/J	-	**0.63 TOp/J**
	General Purpose Cores int2 / int2	✗	✗	1.8 TOp/J	3.33 TOp/J	-	1.43 TOp/J
	General Purpose Cores fp32 / fp32	-	79 GFlop/J	60 GFlop/J	207 GFlop/J	-	18.5 GFlop/J
	Hardware Accelerator int8 / int8	1.1 TOp/J	1.3 TOp/J	✗	1.6 TOp/J	5 TOp/J	2.9 TOp/J
	Hardware Accelerator int8 / int4	✗	-	✗	3.6 TOp/J	✗	5.4 TOp/J
	Hardware Accelerator int8 / int2	✗	-	✗	**7.4 TOp/J**	✗	9.9 TOp/J

✗: Not supported -: No data available

Table 4.9 – State of the Art Comparison of Edge and On-sensor Computing Systems. Best values are highlighted.

4.5 Summary

In this chapter, we widened our focus from circuit-level design to entire integrated systems with a focus on *energy-proportionality*, *memory bandwidth* and *IO power*. With the design and post-silicon evaluation on two different heterogenous multi-core SoC architectures in advanced technology nodes, we showcased a couple of different strategies to address those three focus points:

- In the *Vega* SoC we combined the circuits and findings from chapter 3 to design a system that can scale from an always-on 1.7 µW HDC-based cognitive wakeup to an 32.2 GOPS of on-demand ML performance at a mere 49.4 mW.
- With the *Siracusa* system, we presented a novel approach to tightly couple NVM to a bandwidth demanding QNN accelerator to drastically reduce the off-chip memory access induced IO-energy and bandwidth bottleneck with a peak-energy efficiency of 9.9 TOp/J for dense 3×3 convolutions.

Chapter 5

Conclusions

Smart Sensing nodes will be at the epi-centre of a larger technological paradigm shift in industry, transportation and human-computer interaction. Key "enablers" on this path towards pervasive energy-autonomous edge computation devices are continuous improvements in energy efficiency, performance and latency while at the same time balancing circuit reliability and cost. Energy-proportionality of the compute fabric is a crucial characteristic for meeting those requirements and demands to leave the beaten paths of von-Neumann-based computing and explore novel compute- and hardware-architectural concepts like near- and in-memory computing.

This thesis started with an outline of the current boundaries of the ultra-low power hardware design space, emphasising those near- and in-memory computing concepts. It continued with an introduction to the principles of vector-symbolic computation on the application side. We then proceeded with several concrete digital circuit architectures to efficiently realize the HDC operators in a hardware-friendly manner. We combined these building blocks to form Hypnos, the first fully-configurable hyperdimensional computing accelerator. The effectiveness of Hypnos was then demonstrated in

a complete SoC, where Hypnos was complemented with additional support circuitry to form a *cognitive wake-up unit*, the first stage of an energy-proportional system. Finally, we proposed a novel architectural template for the "second stage" of an energy-proportional system based on the combination of NVM tightly coupled to a QNN hardware accelerator in a heterogenous SoC.

In this final part of the thesis, we summarize the key results and findings of these contributions and provide a researcher's outlook on the path forward.

5.1 Main Results

Energy-Efficient Open-Source Implementation of Common HDC Operators

Hyperdimensional Computing is a novel compute framework well suited for stochastic computing on inherently noisy compute fabric. We proposed energy-efficient implementations for the fundamental operators of binary-spatter-code based HDC;

The presented HD-encoder unit allows for flexible implementations of the binding, bundling and shift-permutation operation. Based on the analysis of Schmuck *et al.*, we developed an encoder architecture with optimal saturating counter bit-width for full-precision bundling that allows evicting the counter state to external memory. This provides the flexibility to match the nested-loop iteration style to the problem at hand rather than being compelled by the hardware design to perform bundling operations in the innermost loop.

The similarity manipulator circuit is an elegant hardware-friendly solution to implement continuous item memory mapping and binarized-back-to-back bundling using a unary encoder (thermometer code) and hardwired random permutation.

With the proposed mixing stage, we introduce a radically different take on item memory mapping based on re-materialization. Contrary to previous re-materialization techniques based on cellular automata, our solution improves the time-complexity of random access to any label vector from $\mathcal{O}(N)$ to $\mathcal{O}(log(N))$ time steps. We furthermore mathematically prove the equivalence of our iterative permutation approach to using unique random permutations per label.

Finally, we present a latch-based auto-associative memory for vector lookup and cleanup operations. We analyzed the energy-efficiency behavior of this SCM-based AM under sub-nominal voltage conditions. We found that for 65 nm scaling from 1.2 V nominal voltage down to 0.6 V, the circuit not only remains functional but also yields a 4× to 5× improvement in overall energy efficiency depending on the degree of parallelization in the hamming distance computation logic.

Hypnos: The first all-digital near-memory HDC Accelerator

Although VSAs have interesting properties concerning error-resiliency, their extremely wide datapath makes HDC a poor match for conventional general-purpose cores. To fully leverage the potential of high-dimensional symbolic representations, we thus combined the HDC circuit building blocks to form *Hypnos*, the first fully configurable all-digital SCM-based hardware accelerator for binary spatter code. With its micro-codable datapath, Hypnos excels in configurability, requiring less than 15 micro-code instruction slots for all analyzed HD algorithms. At the same time, Hypnos's circuit architecture improves area- and energy-efficiency over the previous SoA by up to 3.1× and 3.3× respectively. We showed that these improvements mainly originate from the area advantage of our novel item memory mapping scheme based on the mixing circuit and the ability to repurpose part of the associative memory as a vector buffer during the HD-encoding step. To tune the architecture for the leakage characteristics of the target technology, the proposed architecture exposed the *vector fold* design parameter. We demonstrated that proper tuning of this parameter could reduce the average system power consumption by 40%-55% depending on the target technology and V_{th} cell flavor mix. With a target power envelope in the sub 25 µW range in 22 nm technology, we clearly positioned the design as a candidate for ultra-low power sensing.

Feasability Analysis of HDC-based Anomaly Detection for Predictive Maintenance Applications

As part of demonstrating Hypnos' microcode expressibility, we performed the first initial feasibility analysis of HDC for predictive maintenance applications. Our algorithm performs online learning of the prototypical healthy-system state HD-vector and, after the initial calibration phase, tracks the hamming distance deviation of the 0.5 s time windows from the calibration vector. We demonstrated a visually evident correlation of the HD distance metric to the labeled healthiness of the monitored ball bearings in our dataset and estimated Hypnos' average power consumption below 8 µW when continuously executing the proposed HD-algorithm.

Silicon Realization of Hypnos as first-stage Always-On Cognitive Wakeup Unit

A key requirement for optimal energy efficiency in flexible always-on sensing platforms is their ability to adapt to rapid dynamic changes in the compute intensity of the workload, a characteristic we call *energy-proportionality*. As part of the full system silicon realization in 22 nm FDSOI technology, called *Vega*, we demonstrate the effectiveness of a *multi-staged*, heterogeneous architecture to deliver on this particular aspect. The proposed *cognitive wakeup unit*, a combination of Hypnos and an autonomous digital sensor interface with lightweight preprocessing capabilities, forms the first stage in this architecture, enabling the system power management unit to form "smart" decisions on appropriate time instances to request additional processing power by the RISC-V cluster or the hardware convolutional engine. With a power consumption of as little as 1.7 µW, the CWU in Vega is not only competitive to the current SoA of application-specific full custom wake up designs but is also a significant step-up in terms of configurability and application agnosticism. Our proposed CWU thus forms, to the author's knowledge, the very first ML-powered, *general-purpose* wakeup circuit in the µW range. It thus enables Vega to cover the entire range of sub 2 µW cognitive sensor analytics up to 1.3 TOPS/W 8-bit DNN inference performance.

Design and Realization of novel NVM-centered System Level Architectures for Energy-Efficient DNN Execution

While VSAs are a powerful framework for highly error-resilient computing, their prowess in tackling more complex sensor analytics tasks like computer vision is still a heavily debated research question. Without a doubt, the current SoA in this application domain is dominated by energy-optimized QNN-architectures and are the way to go for the most latency and efficiency-demanding applications like e.g. eye- or hand-tracking algorithms deployed in extended reality. In the final contribution of my thesis, I thus concentrated on how to improve energy efficiency and the IO-bottleneck of the next compute *stage* of an energy-proportional smart sensing system; the Siracusa

SoC taped out in 16 nm Fin-FET technology, provides state-of-the-art 9.9 TOp/J (8-bit activations, 2-bit weight, dense-layers) *peak* energy-efficiency by directly coupling STT-MRAM as first-level memory to the multi-precision hardware accelerator N-EUREKA. We found that the bandwidth-optimized memory-subsystem especially boosts the efficiency of low-input reuse layers like pointwise layers, where the *thightly-coupled NVM* improves energy-efficiency by 3.2×. The optimized architecture of N-EUREKA combined with its aggregated bandwidth of 272 Gbit/s, not only excels in peak performance but also delivers solid 0.76 TOp/J sustained performance at 692 µs latency on a MobileNetv2 derived hand-tracking algorithm. Similar to *Vega*, the proposed architecture in Siracusa not only delivers ample raw tensor multiplication performance for CNNs but also SoA 3.55 GOp/s (8-bit integer operations) per general-purpose RISC-V core performance for non-ML pre- and post-processing workloads in near-sensor analytics pipelines.

5.2 Outlook

This thesis brought a number of key insights on holistic ULP system design to light; The multi-objective optimization problem faced by circuit architects does not always favour the most performant or the most energy-efficient novel technology. At the system level, the well-balanced performance and cost metrics of digital CMOS make it remain a very competitive solution for real-life near-sensor analytics compute fabrics.

Having explored the challenge of energy-proportional computing from various angles, we found that a multi-stage architecture template with increasing degrees of compute performance is very well suited for the task. With our contributions proposed throughout this thesis, we aspired to advance the ULP research field one step stone further on the very long path towards the natural limits of energy efficiency and performance density. To continue this journey, we can think of several possible future research directions:

Research Error-Resilient Control-Flow Architectures for HDC

The current design of Hypnos exploits HDC's data-path error-resiliency by operating on vectors with large degrees of implicit redundancy. However, in order to support even more aggressive voltage scaling, not only the data path needs to be resilient to device failures, but even more importantly, the control path, that is, the control logic that orchestrates the *order* of primitive HD-computing operations needs to be hardened. In the author's view, this problem can be approached from a number of different angles, of which classical redundant lockstep computing is just one possibility. Potential other directions are the use HDC based state machines or stochastic control flows, where the HDC algorithm, i.e. the order of operations is vector dimension specific and could thus share similar noise-resiliency properties as the data contained in the vectors itself.

HDC Algorithm Development and Deployment Framework

The stellar growth of DNN in recent years can partially be explained by the availability of the proper tools like *PyTorch*, *TensorFlow* and the like for rapid prototyping and easy deployment of new network architectures on complex accelerator hardware. The VSA research community would most definitely profit from developing a framework in that direction that would simplify the testing of novel HDC algorithm ideas and the eventual deployment on systems like Hypnos.

Expand the Idea of Cognitive Wake-Up Unit to ML-enhanced Voltage and Frequency Scaling

One possible way to push the energy-proportionality of heterogenous SoCs or SiPs to the next level could be to extend the idea of "smart-On/Off" control to smart workload deployment and performance control. An extended version of the CWU could not only select between discarding an event or offloading an event-of-interest to the next processing stage but could also decide on the most suitable HW unit for the offloading operation and the most appropriate operating corner to meet the latency constraints.

Investigate Ideas for Energy-Proportional Interconnect Fabric for the Memory Hierarchy

Finally, on the memory-hierarchy side, there might be an opportunity to depart from the static-at-design time latency of the interconnect fabric in SoCs. While pipeline stages are the go-to solution to improve throughput at the expense of access latency, they are also well-known to be detrimental to energy efficiency beyond a certain point, which depends on the particular design. Thus another idea to increase the energy-proportionality of the entire system would be to design an interconnect that can be dynamically, meaning at runtime, tuned to only use as much pipelining as necessary to meet the currently required bandwidth demands. Possible ways to achieve such behaviour could be NoC architectures with bypassable pipeline stages that tune the bypass control signal to the current corner, i.e. criticality of the network edge

at hand, the latency sensitivity of the deployed application or the workload-dependent Pareto frontier in energy efficiency versus the number of pipeline stages.

Appendix A

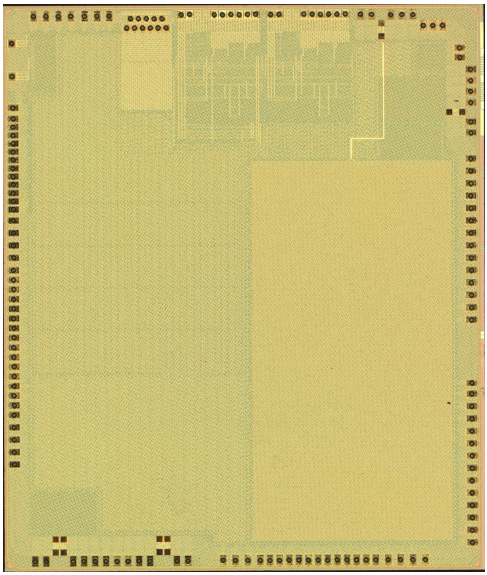
Chip Gallery

In this appendix we provide a summary of every fabricated IC with involvment of the author in the design process. The chips are grouped by relvance to the thesis topic. The complete up-to date list can be found online at `http://asic.ethz.ch/authors/Manuel_Eggimann.html`.

A.1 Treated in this Thesis

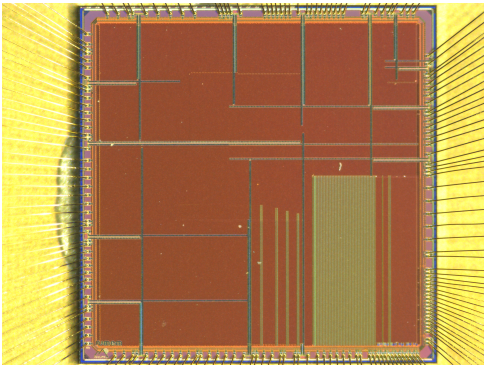
Name	Rosetta
Designers	M. Eggimann, A. Di Mauro, R. Gitermann, O. Harel, A. Burg, A. Levisse, S. William, M. Rios, B. Muheim
Application / Publications	PULP, HDC / Research Project [64]
Technology / Package	TSMC 65nm / QFN64
Dimensions	4100 µm×3000 µm
Gates	6 MGE
Voltage	1.2 V
Clock	190 MHz

Rosetta is a PULPissimo based architecture which uses the RI5CY core with the newest vector processing ISA extension and several accelerators for HDC and Spiking Neural Network (SNN). In addition to the Integrated Circuits Laboratory, the chip contains an a computational-SRAM contributed by the Embedded Systems Laboratory and an eDRAM developed by the Telecommunication Circuits Laboratory of EPFL our sister university from Lausanne.



Name	Vega
Designers	D. Rossi, F. Conti, M. Eggimann, S. Mach, A. Di Mauro, M. Guermandi, G. Tagliavini, A. Pullini, I. Loi, J. Chen, E. Flamand
Application / Publications	PULP, Always-on Sensing / Research Project [84], [122]
Technology / Package	GF 22nm / BGA169
Dimensions	4000 µm×3000 µm
Gates	100 MGE
Voltage	0.5 V to 0.8 V
Clock	32 kHz to 100 MHz

Vega is an always-on IoT end-node SoC capable of scaling from a 1.7 µW fully retentive cognitive sleep mode up to 32.2 GOPS (@49.4 mW) peak performance on NSAAs, including mobile DNN inference, exploiting 1.6 MiB of state- retentive SRAM, and 4 MiB of non-volatile MRAM. To meet the performance and flexibility requirements of NSAAs, the SoC features 10 RISC-V cores: one core for SoC and IO management and a 9-core cluster supporting multi-precision SIMD integer and floating- point computation. Two programmable machine-learning accelerators boost energy efficiency in sleep and active state, respectively.



Name	Siracusa
Designers	M. Eggimann, A. Di Mauro, A. Prasad, F. Conti, D. Rossi
Application / Publications	PULP, ML Smart Sensing / Research Project
Technology / Package	TSMC 16nm / BGA225
Dimensions	0 µm×0 µm
Gates	0 MGE
Voltage	0.55 V to 0.9 V
Clock	530 MHz

Siracusa is a multicore PULP based ULP heterogenous digital signal processing SoC for AR applications developed in collaboration with Meta Realitiy Labs. Besides its 8+1 RISC-V core architecture with custo NN SIMD ISA extensions, it features a high performance Neureka, a quantized neural network accelerator coupled through a custom high-bandwidth low-latency hybrid interconnect with 4 MiB of SRAM and 4 MiB. The ASIC cluster runs at up to 530 MHz consuming around 220 mW at its peak-performance operating mode.

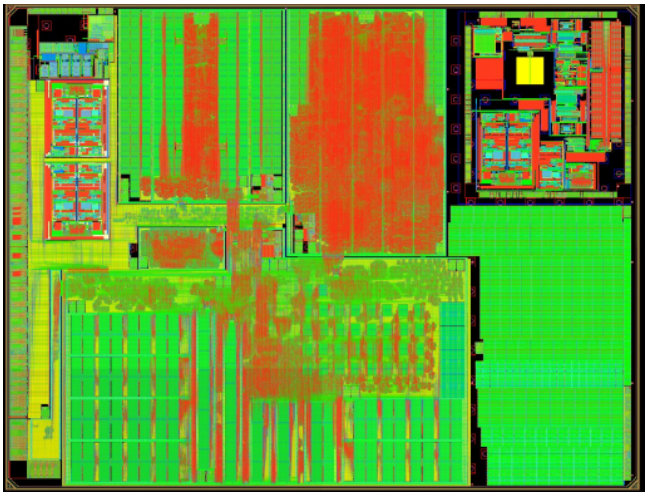
A.2 Other ASICs

Name	Hydra
Designers	M. Eggimann, C. Gloor, M. Schaffner, L. Cavigelli
Application / Publications	Video Processing / Semester Thesis [131]
Technology / Package	UMC 65nm / QFN64
Dimensions	1875 µm×1875 µm
Gates	1400 kGE
Voltage	1.2 V
Clock	333 MHz

Edge aware filters are an important precursor to various graphics processing algorithms. An algorithm which allows for an efficient hardware implementation of a Permeability Filter has been developed in collaboration with Disney Research Zurich recently. In this semester project we nished implementing one part of the algorithm as proposed in the previous paper and developed a hardware architecture which realizes it. The hardware was implemented on an ASIC prototype built in umcL65 technology. It filters 30 frames per second in HD-resolution (1280 ×720 pixels) for a single colour channel. It operates with a floating point precision of 24 bits. An efficient design has been found in order to handle the large amounts of data which need to be processed. The design is easily scalable for higher resolutions or multiple channels.

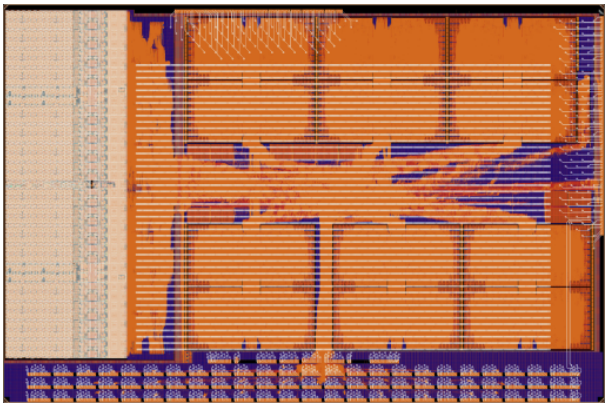
Name	Kraken
Designers	A. Di Mauro, M. Scherer, A. Prasad, T. Fischer, O. Castaneda, M. Eggimann, M. Spallanzani, G. Rutishauser
Application / Publications	PULP, autonomous UAVs / Research Project
Technology / Package	GF 22nm / QFN88
Dimensions	3000 µm×3000 µm
Gates	80 MGE
Voltage	0.8 V
Clock	400 MHz

Kraken is an IoT processor based around our OpenPULP architecture. Kraken features one fabric controller (a 32-bit RI5CY/CV32E40P core) and a computing cluster with eight further RISC-V cores accessing a 128 KiB scratchpad memory. The cluster also has 16 instances of PULPO, an accelerator to solve 1st order optimization problems. The system is further enhanced by an additional external hardware processing engine block that hosts a spiking neural engine (SNE) and a ternary inference engine (CUTIE) that focuses on minimizing non-computational energy and switching activity. The chip also includes a rich set of I/O peripherals to allow it to be connected to existing event driven camera systems



Name	Marsellus
Designers	D. Rossi, A. Di Mauro, F. Conti, G. Paulin, A. Garofalo, G. Ottavi, H. Okuhara, M. Eggimann
Application / Publications	PULP, always-on sensing / Research Project
Technology / Package	GF 22nm / BGA100
Dimensions	4950 µm×3780 µm
Gates	80 MGE
Voltage	0.8 V
Clock	420 MHz

Marsellus is an all-digital AI-IoT end-node heterogeneous SoC fabricated in GlobalFoundries 22nm FDX that combines three key contributions to enable aggressive scaling and fine tuning of performance and energy: a general-purpose cluster of 16 RISC-V DSP cores supporting instruction extensions for a highly diverse range of numerical precision requirements: from 2-bit and 4-bit integer operands (Xpulpnn), combined with expanding, fused MAC & LOAD (M&L), to 32-bit floating-point; a 2-8 bit Reconfigurable Binary Engine to accelerate 3x3 and 1x1 (pointwise) convolutions in DNNs; a set of On-Chip Monitoring (OCM) blocks connected to an Adaptive Body Bias (ABB) generator and a hardware control loop, enabling on-the-fly adaptation of transistor threshold voltages.



Name	Occamy
Designers	G. Paulin, F. Zaruba, S. Mach, M. Eggimann, M. Cavalcante, P. Scheffler, Y. Zhang, T. Fischer, N. Wistoff, L. Bertaccini, T. Benz, L. Colagrande, A. Di Mauro, A. Kurth, S. Riedel, N. Huetter, G. Ottavi, Z. Jiang, B. Muheim, F. Gurkaynak, D. Rossie, L. Benini
Application / Publications	Scale-out HPC / Research Project
Technology / Package	GF 12nm / Custom 3D IC
Dimensions	10 500 µm×6950 µm
Gates	300 MGE
Voltage	0.8 V
Clock	1 GHz

Occamy is a 2.5D integrated dual-chiplet system with 16GB private high-bandwidth memory (HBM) per chiplet. We have taped out the Occamy chiplet in 12nm FINFET technology, and the interposer Hedwig in 65nm purely passive silicon interposer technology. The entire project has been very generously sponsored by Globalfoundries and Rambus as well as support from Synopsys and Micron.

The scalable and ultra-efficient many-core Occamy chiplet architecture, designed to operate at 1GHz, is organized around Snitch, a lightweight, latency-tolerant RISC-V core. Snitch is coupled with an FPU featuring SIMD, Minifloat formats (8-bit, 16-bit) and fused expanding-sum-dot-product capabilities. Each chiplet contains more than 200 Snitch cores organized in groups of four compute clusters. Each cluster shares a tightly-coupled memory among eight compute cores and a high-bandwidth (512-bit) DMA engine orchestrating the data flow. A CVA6 Linux-capable RISC-V core manages all compute clusters and system peripherals. Each chiplet has a private 16GB high-bandwidth memory (HBM) and can communicate with each other over a 72GB/s source-synchronous DDR serial die-to-die link.

Appendix B

Notations and Acronyms

ADC . . . . . Analog-To-Digital Converter

AI . . . . . Artificial Inteligence

AM . . . . . Associative Memory

AMS . . . . . Analog Mixed Signal

AR . . . . . Augmented Reality

BEOL . . . . . Back-End-Of-Line

BER . . . . . Bit-Error Rate

BOM . . . . . Bill of Material

BSC . . . . . Binary Spatter Code

CF . . . . . Conductive Filament

CMOS . . . . . Complementary Metal-Oxide-Semiconductor

CNN . . . . . Convolutional Neural Network

CPU . . . . . Central Processing Unit

CWU . . . . . Cognitive Wakeup Unit

DNN . . . . . Deep Neural Network

DRAM . . . . . Dynamic Random Access Memory

DRC . . . . . Design Rule Check

DSP . . . . . Digital Signal Processing

DVCSL . . . . Differential Cascode Voltage Switch Logic

EDA . . . . . Electronic Design Automation

eDRAM . . . . Embedded Dynamic Random Access Memory

EMA . . . . . Exponential Moving Average

EMG . . . . . Electromyography

FC . . . . . Fabric Controller

FEOL . . . . . Front-End-Of-Line

FLL . . . . . Frequency-Locked Loop

FPU . . . . . Floating Point Unit

GALS . . . . . Globally-Asynchronous Locally-Synchronous

GAN . . . . . Generative Adversarial Network

GPU . . . . . Graphics Processing Unit

HD . . . . . Hyper-Dimensional

HDC . . . . . Hyper-Dimensional Computing

HDL . . . . . Hardware Description Language

HPC . . . . . High-Performance Computing

HRR . . . . . Holographic Reduced Representation

I2C . . . . . Inter-Integrated Circuit

IC . . . . . Integrated Circuit

IoT . . . . . Interent of Things

IRB . . . . . Inverted Residual Bottleneck

ISA . . . . . Instruction-Set Architecture

LSTM . . . . . Long Short-Term Memory

LVS . . . . . Layout-Versus-Schematic

MAP . . . . . Multiply-Add-Permute

MCU . . . . . Microcontroller Unit

MEC . . . . . Mobile Edge Computing

ML . . . . . Machine Learning

MLC . . . . . Multi-Level Cell

MOSFET . . . Metal-Oxide-Semiconductor Field-Effect Transistor

MRAM . . . . Magnetoresistive Random Access Memory

MTBF . . . . . Mean Time Between Failures

mtj . . . . . Magnetic Tunnel Junction

MTTF . . . . . Mean Time To Failure

NAS . . . . . Network Architecture Search

NN . . . . . Neural Network

NoC . . . . . Network-On-Chip

NRE . . . . . Non-Recurring Engineering Cost

NVM . . . . . Non-Volatile Memory

PCB . . . . . Printed Circuit Board

PCHB . . . . . Precharged Half-Buffer

PCM . . . . . Phase-Change Memory

PE . . . . . Processing Element

PPA . . . . . Power, Performance and Area

QDI . . . . . Quasi-Delay Insensitivity

QNN . . . . . Quantized Neural Network

ReLU . . . . . Rectified Linear Unit

ReRAM . . . . Resistive Random-Access Memory

ResNet . . . . . Residual Neural Network

RNN . . . . . Recurrent Neural Network

ROI . . . . . Region-Of-Interest

SAR . . . . . Successive-Approximation-Register

SBDR . . . . . Sparse Binary Distributed Represenation

SCM . . . . . Standard Cell Memory

SIMD . . . . . Single Instruction Multiple Data

SiP . . . . . System-In-Package

SLC . . . . . Single-Level Cell

SNN . . . . . Spiking Neural Network

SNR . . . . . Signal-To-Noise Ratio

SoA . . . . . State-Of-The-Art

SoC . . . . . System on Chip

SOT-MRAM . . Spin-Orbit-Torque Magnetoresistive RAM

SPI . . . . . Serial Peripheral Interface

SRAM . . . . . Static Random Access Memory

SSD . . . . . Solid-State Drive

STT-MRAM . . Spin-Transfer-Torque Magnetoresistive RAM

SVE . . . . . Scalable Vector Extension

TC-NVM . . . Tightly-Coupled Non-Volatile Memory

TMR . . . . . Tunnel Magnetoresistance

TPU . . . . . Tensor Processing Unit

UAV . . . . . Unmanned Aerial Vehicle

ULP . . . . . Ultra-Low Power

VLSI . . . . . Very Large-Scale Integration

VSA . . . . . Vector Symbolic Architecture

XR . . . . . Extended Reality

Bibliography

[1] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, Jul. 1961, ISSN: 0018-8646. DOI: 10.1147/rd.53.0183.

[2] H.-S. P. Wong, C.-S. Lee, J. Luo, and C.-H. Wang. "CMOS Technology Scaling Trend." (May 13, 2022), [Online]. Available: https://nano.stanford.edu/cmos-technology-scaling-trend (visited on 05/13/2022).

[3] R. Bogue, "Recent developments in MEMS sensors: A review of applications, markets and technologies," *Sensor Review*, vol. 33, Sep. 9, 2013. DOI: 10.1108/SR-05-2013-678.

[4] M. Alioto, Ed., *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems*. Springer International Publishing, 2017, ISBN: 978-3-319-51480-2.

[5] N. Dall'Ora, S. Centomo, and F. Fummi, "Industrial-IoT Data Analysis Exploiting Electronic Design Automation Techniques," in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*, Jun. 2019, pp. 103–109. DOI: 10.1109/IWASI.2019.8791344.

[6] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh, "Simulating the Power Consumption of Large-scale Sensor Network Applications," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, (Baltimore, MD, USA),

ser. SenSys '04, New York, NY, USA: ACM, Nov. 2004, pp. 188–200, ISBN: 978-1-58113-879-5. DOI: 10.1145/1031495.1031518.

[7] M. Abrash, "Creating the Future: Augmented Reality, the next Human-Machine Interface," in *2021 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2021, pp. 1.2.1–1.2.11. DOI: 10.1109/IEDM19574.2021.9720526.

[8] R. Azuma, "Tracking requirements for augmented reality," *Communications of the ACM*, vol. 36, no. 7, pp. 50–51, Jul. 1993, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/159544.159581.

[9] H. You, C. Wan, Y. Zhao, *et al.*, "EyeCoD: Eye Tracking System Acceleration via FlatCam-based Algorithm & Accelerator Co-Design," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, Jun. 18, 2022, pp. 610–622. DOI: 10.1145/3470496.3527443. arXiv: 2206.00877 [cs].

[10] J.-P. Stauffert, F. Niebling, and M. E. Latoschik, "Latency and Cybersickness: Impact, Causes, and Measures. A Review," *Frontiers in Virtual Reality*, vol. 1, 2020, ISSN: 2673-4192.

[11] B. Spencer, M. Ruiz Sandoval, and N. Kurata, "Smart Sensing Technology: Opportunities and Challenges," *Structural Control and Health Monitoring*, vol. 11, pp. 349–368, Oct. 1, 2004. DOI: 10.1002/stc.48.

[12] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Communications of the ACM*, vol. 62, no. 2, pp. 48–60, Jan. 28, 2019, ISSN: 0001-0782. DOI: 10.1145/3282307.

[13] Steve McCaskill. "Three quarters of smartphones will have AI chip by 2022," TechRadar. (Apr. 15, 2019), [Online]. Available: https://www.techradar.com/news/three-quarters-of-smartphones-will-have-ai-chip-by-2022 (visited on 03/15/2023).

[14] N. P. Jouppi, "A Domain-Specific TPU Supercomputer for Training Deep Neural Networks," Oct. 27, 2020.

[15] L. Su and S. Naffziger, "1.1 Innovation For the Next Decade of Compute Efficiency," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, Feb. 2023, pp. 8–12. DOI: 10.1109/ISSCC42615.2023.10067810.

[16] J. Postman and P. Chiang, "A Survey Addressing On-Chip Interconnect: Energy and Reliability Considerations," *International Scholarly Research Notices*, vol. 2012, e916259, Mar. 26, 2012. DOI: 10.5402/2012/916259.

[17] A. Di Mauro, M. Scherer, J. F. Mas, B. Bougenot, M. Magno, and L. Benini, "FlyDVS: An Event-Driven Wireless Ultra-Low Power Visual Sensor Node," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Feb. 2021, pp. 1851–1854. DOI: 10.23919/DATE51398.2021.9474260.

[18] G. Rovere, S. Fateh, and L. Benini, "A 2.2-µW Cognitive Always-On Wake-Up Circuit for Event-Driven Duty-Cycling of IoT Sensor Nodes," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 543–554, Sep. 2018. DOI: 10.1109/JETCAS.2018.2828505.

[19] X. Zhang and Y. Lian, "A 300-mV 220-nW Event-Driven ADC With Real-Time QRS Detection for Wearable ECG Sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 6, pp. 834–843, Dec. 2014, ISSN: 1940-9990. DOI: 10.1109/TBCAS.2013.2296942.

[20] B. Arnaldi, *Virtual Reality & Augmented Re: Myths and Realities*, 1st ed. Hoboken, NJ: John Wiley & Sons, May 8, 2018, 372 pp., ISBN: 978-1-78630-105-5.

[21] C. Hao, J. Dotzel, J. Xiong, L. Benini, Z. Zhang, and D. Chen, "Enabling Design Methodologies and Future Trends for Edge AI: Specialization and Codesign," *IEEE Design & Test*, vol. 38, no. 4, pp. 7–26, Aug. 2021, ISSN: 2168-2364. DOI: 10.1109/MDAT.2021.3069952.

[22] C.-J. Wu, D. Brooks, K. Chen, *et al.*, "Machine Learning at Facebook: Understanding Inference at the Edge," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2019, pp. 331–344. DOI: 10.1109/HPCA.2019.00048.

[23] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine Learning at the Network Edge: A Survey," *ACM Computing Surveys*, vol. 54, no. 8, 170:1–170:37, Oct. 4, 2021, ISSN: 0360-0300. DOI: 10.1145/3469029.

[24] A. Alhilal, T. Braud, and P. Hui. "Distributed Vehicular Computing at the Dawn of 5G: A Survey." arXiv: 2001.07077. (Jan. 20, 2020), [Online]. Available: http://arxiv.org/abs/2001.07077 (visited on 01/22/2020), preprint.

[25] R. P. Weicker, "Dhrystone: A synthetic systems programming benchmark," *Communications of the ACM*, vol. 27, no. 10, pp. 1013–1030, Oct. 1, 1984, ISSN: 0001-0782. DOI: 10.1145/358274.358283.

[26] E. M. B. Consortium *et al.*, *CoreMark: An EEMBC benchmark*, 2018.

[27] V. J. Reddi, C. Cheng, D. Kanter, *et al.*, "MLPerf Inference Benchmark," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, May 2020, pp. 446–459. DOI: 10.1109/ISCA45697.2020.00045.

[28] L. Carrington, M. Laurenzano, A. Snavely, R. L. Campbell, and L. P. Davis, "How well can simple metrics represent the performance of HPC applications?" In *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, Nov. 2005, pp. 48–48. DOI: 10.1109/SC.2005.33.

[29] I. D. Raji, E. M. Bender, A. Paullada, E. Denton, and A. Hanna. "AI and the Everything in the Whole Wide World Benchmark." arXiv: 2111.15366 [cs]. (Nov. 26, 2021), [Online]. Available: http://arxiv.org/abs/2111.15366 (visited on 04/12/2023), preprint.

[30] W. Kramer, "Top500 versus sustained performance - the top problems with the TOP500 list - and what to do about them," in *2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep. 2012, pp. 223–230.

[31] R. K. Mobley, *An Introduction to Predictive Maintenance, Second Edition*. Butterworth-Heinemann, Apr. 4, 2013, 458 pp., ISBN: 978-0-12-399637-4.

[32] T. R. Farrell and R. F. Weir, "The Optimal Controller Delay for Myoelectric Prostheses," *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 15, no. 1, pp. 111–118, Mar. 2007, ISSN: 1534-4320. DOI: 10.1109/TNSRE.2007.891391. pmid: 17436883.

[33] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, Oct. 1994, pp. 8–11. DOI: 10.1109/LPE.1994.573184.

[34] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "How to Evaluate Deep Neural Network Processors: TOPS/W (Alone) Considered Harmful," *IEEE Solid-State Circuits Magazine*, vol. 12, no. 3, pp. 28–41, Sum. 2020, ISSN: 1943-0582, 1943-0590. DOI: 10.1109/MSSC.2020.3002140.

[35] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299–316, Jun. 2000, ISSN: 1557-9999. DOI: 10.1109/92.845896.

[36] V. Iyer, H. Gaensbauer, T. L. Daniel, and S. Gollakota, "Wind dispersal of battery-free wireless devices," *Nature*, vol. 603, no. 7901, pp. 427–433, 7901 Mar. 2022, ISSN: 1476-4687. DOI: 10.1038/s41586-021-04363-9.

[37] M. Fourniol, V. Gies, V. Barchasz, E. Kussener, and H. Glotin, "Applications of an Ultra Low-Power Analog Wake-up Detector for Environmental IoT Networks and Military Smart Dust," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, Nov. 2018, pp. 16–22. DOI: 10.1109/IOTAIS.2018.8600893.

[38] H. Li, T. F. Wu, A. Rahimi, *et al.*, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2016, pp. 16.1.1–16.1.4. DOI: 10.1109/IEDM.2016.7838428.

[39] B. Reagen, U. Gupta, L. Pentecost, *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, Jun. 2018, pp. 1–6. DOI: 10.1109/DAC.2018.8465834.

[40] P. Clarke. "Intel goes foundry for 7nm due to yield issues," EENewsEurope. (Jul. 27, 2020), [Online]. Available: https://www.eenewsanalog.com/en/intel-goes-foundry-for-7nm-due-to-yield-issues/ (visited on 10/10/2022).

[41] D. Stow, I. Akgun, R. Barnes, P. Gu, and Y. Xie, "Cost analysis and cost-driven IP reuse methodology for SoC design based on 2.5D/3D integration," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2016, pp. 1–6. DOI: 10.1145/2966986.2980095.

[42] J. Hruska. "As Chip Design Costs Skyrocket, 3nm Process Node Is in Jeopardy." (Jul. 22, 2018), [Online]. Available: https://www.extremetech.com/computing/272096-3nm-process-node (visited on 10/11/2022).

[43] G. Yeric, "Moore's law at 50: Are we planning for retirement?" In *2015 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2015, pp. 1.1.1–1.1.8. DOI: 10.1109/IEDM.2015.7409607.

[44] IRDS, "International Roadmap for Devices and Systems 2022 Edition - Executive Summary," 2022.

[45] X. Yu, K. Ergun, L. Cherkasova, and T. Š. Rosing, "Optimizing Sensor Deployment and Maintenance Costs for Large-Scale Environmental Monitoring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3918–3930, Nov. 2020, ISSN: 1937-4151. DOI: 10.1109/TCAD.2020.3012232.

[46] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2745201.

[47] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2014, pp. 10–14. DOI: 10.1109/ISSCC.2014.6757323.

[48] Wm. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995, ISSN: 0163-5964. DOI: 10.1145/216585.216588.

[49] A. Shrestha, H. Fang, Z. Mei, D. P. Rider, Q. Wu, and Q. Qiu, "A Survey on Neuromorphic Computing: Models and Hardware," *IEEE Circuits and Systems Magazine*, vol. 22, no. 2, pp. 6–35, 2022, ISSN: 1558-0830. DOI: 10.1109/MCAS.2022.3166331.

[50] N. Verma, H. Jia, H. Valavi, *et al.*, "In-Memory Computing: Advances and Prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, Aug. 2019, ISSN: 1943-0590. DOI: 10.1109/MSSC.2019.2922889.

[51] M. Flynn, "Very high-speed computing systems," *Proceedings of the IEEE*, vol. 54, no. 12, pp. 1901–1909, Dec. 1966, ISSN: 1558-2256. DOI: 10.1109/PROC.1966.5273.

[52] A. Harrod. "Arm's solution to the future needs of AI, security and specialized computing is v9," Arm | The Architecture for the Digital World. (Mar. 30, 2021), [Online]. Available: https://www.arm.com/company/news/2021/03/arms-answer-to-the-future-of-ai-armv9-architecture (visited on 10/01/2022).

[53] T. Trader. "Arm Details Neoverse V1, N2 Platforms with New Mesh Interconnect, Advances Partner Ecosystem," HPCwire. (Apr. 27, 2021), [Online]. Available: https://www.hpcwire.com/2021/04/27/arm-launches-neoverse-platforms-new-interconnect-advances-partner-ecosystem/ (visited on 10/20/2022).

[54] top500.org. "Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D | TOP500." (), [Online]. Available: https://www.top500.org/system/179807/ (visited on 10/20/2022).

[55] Y. Kodama, T. Odajima, E. Arima, and M. Sato, "Evaluation of Power Management Control on the Supercomputer Fugaku," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2020, pp. 484–493. DOI: 10.1109/CLUSTER49012.2020.00069.

[56] RISC-V. Community News. "RISC-V Vector Processing is Taking Off | SiFive," RISC-V International. (Jun. 21, 2022), [Online]. Available: https://riscv.org/blog/2022/06/risc-v-vector-processing-is-taking-off-sifive/ (visited on 10/20/2022).

[57] N. Dahad. "EETimes - Domain Specific Accelerators Will Drive Vector Processing on RISC-V -," EETimes. (May 26, 2020), [Online]. Available: https://www.eetimes.com/domain-specific-accelerators-will-drive-vector-processing-on-risc-v/ (visited on 08/26/2020).

[58] M. Cavalcante, F. Schuiki, F. Zaruba, M. Schaffner, and L. Benini, "Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor With Multiprecision Floating-Point Support in 22-nm FD-SOI," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 530–543, Feb. 2020, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2019.2950087.

[59] M. Cavalcante, D. Wüthrich, M. Perotti, S. Riedel, and L. Benini. "Spatz: A Compact Vector Processing Unit for High-Performance and Energy-Efficient Shared-L1 Clusters." arXiv: 2207.07970 [cs]. (Jul. 16, 2022), [Online]. Available: http://arxiv.org/abs/2207.07970 (visited on 10/20/2022), preprint.

[60] K. Sato and C. Young. "An in-depth look at Google's first Tensor Processing Unit (TPU)," Google Cloud Blog. (May 12, 2017), [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu (visited on 10/22/2022).

[61] N. P. Jouppi, D. H. Yoon, G. Kurian, *et al.*, "A domain-specific supercomputer for training deep neural networks," *Communications of the ACM*, vol. 63, no. 7, pp. 67–78, Jun. 18, 2020, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3360307.

[62] Y. E. Wang, G.-Y. Wei, and D. Brooks. "Benchmarking TPU, GPU, and CPU Platforms for Deep Learning." (Jul. 2019), [Online]. Available: http://arxiv.org/abs/1907.10701, preprint.

[63] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, "Analog architectures for neural network acceleration based on non-volatile memory," *Applied Physics Reviews*, vol. 7, no. 3, p. 031 301, Jul. 9, 2020. DOI: 10.1063/1.5143815.

[64] O. Harel, E. N. Casarrubias, M. Eggimann, *et al.*, "64-kB 65-nm GC-eDRAM With Half-Select Support and Parallel Refresh Technique," *IEEE Solid-State Circuits Letters*, vol. 5, pp. 170–173, 2022, ISSN: 2573-9603. DOI: 10.1109/LSSC.2022.3182531.

[65] R. Giterman, A. Teman, and P. Meinerzhagen, "Hybrid GC-eDRAM/SRAM Bitcell for Robust Low-Power Operation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 12, pp. 1362–1366, Dec. 2017. DOI: 10.1109/TCSII.2017.2768102.

[66] P. Meinerzhagen, C. Roth, and A. Burg, "Towards generic low-power area-efficient standard cell based memory architectures," in *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, Aug. 2010, pp. 129–132. DOI: 10.1109/MWSCAS.2010.5548579.

[67] O. Andersson, B. Mohammadi, P. Meinerzhagen, A. Burg, and J. N. Rodrigues, "Ultra Low Voltage Synthesizable Memories: A Trade-Off Discussion in 65 nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 6, pp. 806–817, Jun. 2016, ISSN: 1558-0806. DOI: 10.1109/TCSI.2016.2537931.

[68] A. Teman, D. Rossi, P. Meinerzhagen, L. Benini, and A. Burg, "Power, Area, and Performance Optimization of Standard Cell Memory Arrays Through Controlled Placement," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 4, pp. 1–25, May 2016. DOI: 10.1145/2890498.

[69] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489–502, Apr. 2003, ISSN: 1558-2256. DOI: 10.1109/JPROC.2003.811702.

[70] H. Tanaka, M. Kido, K. Yahashi, *et al.*, "Bit Cost Scalable Technology with Punch and Plug Process for Ultra High Density Flash Memory," in *2007 IEEE Symposium on VLSI Technology*, Jun. 2007, pp. 14–15. DOI: 10.1109/VLSIT.2007.4339708.

[71] IRDS, "International Roadmap for Devices and Systems 2022 Edition - Mass Data Storage and Non-Volatile Memory Roadmap," IRDS, 2022.

[72] IRDS, "International Roadmap for Devices and Systems 2022 Edition - More than Moore White Paper," IRDS, 2022.

[73] J. Cooke. "Flash memory 101: An introduction to NAND flash," EDN. (Mar. 20, 2006), [Online]. Available: https://www.edn.com/flash-memory-101-an-introduction-to-nand-flash/ (visited on 10/28/2022).

[74] P. Wang, F. Xu, B. Wang, *et al.*, "Three-Dimensional nand Flash for Vector–Matrix Multiplication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 988–991, Apr. 2019, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2018.2882194.

[75] Y. J. Park, H. T. Kwon, B. Kim, *et al.*, "3-D Stacked Synapse Array Based on Charge-Trap Flash Memory for Implementation of Deep Neural Networks," *IEEE Transactions on Electron Devices*, vol. 66, no. 1, pp. 420–427, Jan. 2019, ISSN: 1557-9646. DOI: 10.1109/TED.2018.2881972.

[76] M. R. Mahmoodi and D. Strukov, "An Ultra-Low Energy Internally Analog, Externally Digital Vector-Matrix Multiplier Based on NOR Flash Memory Technology," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, Jun. 2018, pp. 1–6. DOI: 10.1109/DAC.2018.8465804.

[77] X. Guo, F. M. Bayat, M. Bavandpour, *et al.*, "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology," in *2017 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2017, pp. 6.5.1–6.5.4. DOI: 10.1109/IEDM.2017.8268341.

[78] M. Bavandpour, S. Sahay, M. R. Mahmoodi, and D. B. Strukov, "3D-aCortex: An ultra-compact energy-efficient neurocomputing platform based on commercial 3D-NAND flash memories," *Neuromorphic Computing and Engineering*, vol. 1, no. 1, p. 014 001, Jul. 2021, ISSN: 2634-4386. DOI: 10.1088/2634-4386/ac0775.

[79] M. LaPedus. "Embedded Flash Scaling Limits," Semiconductor Engineering. (Jul. 19, 2018), [Online]. Available: https://semiengineering.com/embedded-flash-scaling-limits/ (visited on 10/28/2022).

[80] N. Dahad. "The case for de-integrating embedded Flash," Embedded.com. (Aug. 19, 2022), [Online]. Available: https://www.embedded.com/the-case-for-de-integrating-embedded-flash/ (visited on 10/28/2022).

[81] H. Yoda, "MRAM Fundamentals and Devices," in *Handbook of Spintronics*, Y. Xu, D. D. Awschalom, and J. Nitta, Eds., Dordrecht: Springer Netherlands, 2016, pp. 1031–1064, ISBN: 978-94-007-6891-8. DOI: 10.1007/978-94-007-6892-5_39.

[82] Y. Chen, H. H. Li, I. Bayram, and E. Eken, "Recent Technology Advances of Emerging Memories," *IEEE Design & Test*, vol. 34, no. 3, pp. 8–22, Jun. 2017, ISSN: 2168-2364. DOI: 10.1109/MDAT.2017.2685381.

[83] S. Sakhare, M. Perumkunnil, T. H. Bao, *et al.*, "Enablement of STT-MRAM as last level cache for the high performance computing domain at the 5nm node," in *2018 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA: IEEE, Dec. 2018, pp. 18.3.1–18.3.4, ISBN: 978-1-72811-987-8. DOI: 10.1109/IEDM.2018.8614637.

[84] D. Rossi, F. Conti, M. Eggimann, *et al.*, "Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, Jan. 2022, ISSN: 1558-173X. DOI: 10.1109/JSSC.2021.3114881.

[85] M. Komalan, S. Sakhare, T. H. Bao, *et al.*, "Cross-layer design and analysis of a low power, high density STT-MRAM for embedded systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, USA: IEEE, May 2017, pp. 1–4, ISBN: 978-1-4673-6853-7. DOI: 10.1109/ISCAS.2017.8050923.

[86] B. Sun, D. Liu, L. Yu, *et al.*, "MRAM Co-designed Processing-in-Memory CNN Accelerator for Mobile and IoT Applications," Nov. 26, 2018. arXiv: 1811.12179 [eess].

[87] W. Gallagher, E. Chien, T.-W. Chiang, *et al.*, "22nm STT-MRAM for Reflow and Automotive Uses with High Yield, Reliability, and Magnetic Immunity and with Performance and Shielding Options," in *2019 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, pp. 2.7.1–2.7.4. DOI: 10.1109/IEDM19573.2019.8993469.

[88] IRDS, "International Roadmap for Devices and Systems 2022 Edition - More Moore," IRDS, 2022.

[89] D. Garbin, "A variability study of PCM and OxRAM technologies for use as synapses in neuromorphic systems," Université Grenoble Alpes, 2015.

[90] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, and E. Eleftheriou, "Projected phase-change memory devices," *Nature Communications*, vol. 6, no. 1, p. 8181, 1 Sep. 3, 2015, ISSN: 2041-1723. DOI: 10.1038/ncomms9181.

[91] S. Cosemans, B. Verhoef, J. Doevenspeck, *et al.*, "Towards 10000TOPS/W DNN Inference with Analog in-Memory Computing – A Circuit Blueprint, Device Options and Requirements," in *2019 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, pp. 22.2.1–22.2.4. DOI: 10.1109/IEDM19573.2019.8993599.

[92] H. Castro, "Accessing memory cells in parallel in a cross-point array," U.S. Patent 20150074326A1, Mar. 12, 2015.

[93] J. Emerson. "In Remembrance of Optane," Pure Storage Blog. (Aug. 1, 2022), [Online]. Available: https://blog.purestorage.com/perspectives/in-remembrance-of-optane/ (visited on 10/31/2022).

[94] Y. Zhang, G.-Q. Mao, X. Zhao, *et al.*, "Evolution of the conductive filament system in HfO2-based memristors observed by direct atomic-scale imaging," *Nature Communications*, vol. 12, no. 1, p. 7232, 1 Dec. 13, 2021, ISSN: 2041-1723. DOI: 10.1038/s41467-021-27575-z.

[95] J.-M. Hung, C.-X. Xue, H.-Y. Kao, *et al.*, "A four-megabit compute-in-memory macro with eight-bit precision based on CMOS and resistive random-access memory for AI edge devices," *Nature Electronics*, vol. 4, no. 12, pp. 921–930, 12 Dec. 2021, ISSN: 2520-1131. DOI: 10.1038/s41928-021-00676-9.

[96] J.-M. Portal, M. Bocquet, S. Onkaraiah, *et al.*, "Design and Simulation of a 128 kb Embedded Nonvolatile Memory Based on a Hybrid RRAM (HfO2)/28 nm FDSOI CMOS Technology," *IEEE Transactions on Nanotechnology*, vol. 16, no. 4, pp. 677–686, Jul. 2017. DOI: 10.1109/TNANO.2017.2703985.

[97] A. Shafiee, A. Nag, N. Muralimanohar, *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2016, pp. 14–26. DOI: 10.1109/ISCA.2016.12.

[98] X. Sun, S. Yin, X. Peng, R. Liu, J.-s. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2018, pp. 1423–1428. DOI: 10.23919/DATE.2018.8342235.

[99] S. Mittal, "A Survey of ReRAM-Based Architectures for Processing-In-Memory and Neural Networks," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 75–114, 1 Mar. 2019. DOI: 10.3390/make1010005.

[100] F. M. Puglisi, N. Zagni, L. Larcher, and P. Pavan, "Random Telegraph Noise in Resistive Random Access Memories: Compact Modeling and Advanced Circuit Design," *IEEE Transactions on Electron Devices*, vol. 65, no. 7, pp. 2964–2972, Jul. 2018. DOI: 10.1109/TED.2018.2833208.

[101] F. M. Puglisid, N. Zagni, L. Larcher, and P. Pavan, "A new verilog-A compact model of random telegraph noise in oxide-based RRAM for advanced circuit design," in *2017 47th European Solid-State Device Research Conference (ESSDERC)*, IEEE, Sep. 2017, pp. 204–207, ISBN: 978-1-5090-5978-2. DOI: 10.1109/ESSDERC.2017.8066627.

[102] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping Zhezhi," in *Proceedings of the 56th Annual Design Automation Conference 2019 on - DAC '19*, ACM Press, 2019, pp. 1–6, ISBN: 978-1-4503-6725-7. DOI: 10.1145/3316781.3317870.

[103] A. Lal Shimpi and G. Key. "The Dark Knight: Intel's Core i7." (Nov. 3, 2008), [Online]. Available: https://www.anandtech.com/show/2658 (visited on 11/01/2022).

[104] B. Ulmann, *Analog Computing*. Oldenbourg Wissenschaftsverlag, Jul. 22, 2013, ISBN: 978-3-486-75518-3. DOI: 10.1524/9783486755183.

[105] A. S. Rekhi, B. Zimmer, N. Nedovic, *et al.*, "Analog/Mixed-Signal Hardware Error Modeling for Deep Learning Inference," in *Proceedings of the 56th Annual Design Automation Conference 2019 on - DAC '19*, ACM Press, 2019, pp. 1–6, ISBN: 978-1-4503-6725-7. DOI: 10.1145/3316781.3317770.

[106] H. Kaeslin, *Top-Down Digital VLSI Design: From Architectures to Gate-Level Circuits and FPGAs*. Waltham, MA: Morgan Kaufmann, Dec. 8, 2014, 598 pp., ISBN: 978-0-12-800730-3.

[107] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *Proceedings of the 40th Annual Design Automation Conference*, ser. DAC '03, New York, NY, USA: Association for Computing Machinery, Jun. 2, 2003, pp. 622–627, ISBN: 978-1-58113-688-3. DOI: 10.1145/775832.775989.

[108] A. Kapoor, C. Groot, G. V. Piqué, *et al.*, "Digital Systems Power Management for High Performance Mixed Signal Platforms," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 961–975, Apr. 2014, ISSN: 1558-0806. DOI: 10.1109/TCSI.2014.2304662.

[109] M. Krstic, E. Grass, F. K. Gürkaynak, and P. Vivet, "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook," *IEEE Design & Test of Computers*, vol. 24, no. 5, pp. 430–441, Sep. 2007, ISSN: 1558-1918. DOI: 10.1109/MDT.2007.164.

[110] C. E. Cummings, "Clock Domain Crossing (CDC) Design & Verification Techniques Using SystemVerilog," presented at the Synopsys User Group 2008 (SNUG), 2008, p. 56.

[111] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," presented at the International Symposium on the Switching Theory, Harvard University, 1959.

[112] A. Martin and M. Nystrom, "Asynchronous Techniques for System-on-Chip Design," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1089–1120, Jun. 2006, ISSN: 1558-2256. DOI: 10.1109/JPROC.2006.875789.

[113] M. M. Nystrom, *Asynchronous Pulse Logic*, Softcover reprint of the original 1st ed. 2002 Edition. Springer, Oct. 4, 2013, 236 pp., ISBN: 978-1-4419-5284-4.

[114] A. J. Martin, A. Lines, R. Manohar, *et al.*, "The design of an asynchronous MIPS R3000 microprocessor," *Proceedings Seventeenth Conference on Advanced Research in VLSI*, Jan. 1, 1997.

[115] R. Siddagangappa and N. D. K, "Asynchronous NoC with Fault tolerant mechanism: A Comprehensive Review," in *2022 Trends in Electrical, Electronics, Computer Engineering Conference (TEECCON)*, May 2022, pp. 84–92. DOI: 10.1109/TEECCON54414.2022.9854837.

[116] M. Krstic, E. Grass, and X. Fan, "Asynchronous and GALS Design -Overview and Perspectives," in *2017 New Generation of CAS (NGCAS)*, Sep. 2017, pp. 85–88. DOI: 10.1109/NGCAS.2017.42.

[117] M. Davies, N. Srinivasa, T.-H. Lin, *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018, ISSN: 1937-4143. DOI: 10.1109/MM.2018.112130359.

[118] G. Orchard, E. P. Frady, D. B. D. Rubin, *et al.*, "Efficient Neuromorphic Signal Processing with Loihi 2," in *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2021, pp. 254–259. DOI: 10.1109/SiPS52927.2021.00053.

[119] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 8, 2014. DOI: 10.1126/science.1254642.

[120] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 7784 Nov. 2019, ISSN: 1476-4687. DOI: 10.1038/s41586-019-1677-2.

[121] M. Eggimann, A. Rahimi, and L. Benini, "A 5 µW Standard Cell Memory-Based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 10, pp. 4116–4128, Oct. 2021, ISSN: 1558-0806. DOI: 10.1109/TCSI.2021.3100266.

[122] D. Rossi, F. Conti, M. Eggimann, *et al.*, "4.4 A 1.3TOPS/W @ 32GOPS Fully Integrated 10-Core SoC for IoT End-Nodes with 1.7µW Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, Feb. 2021, pp. 60–62. DOI: 10.1109/ISSCC42613.2021.9365939.

[123] M. Eggimann, S. Mach, M. Magno, and L. Benini, "A RISC-V Based Open Hardware Platform for Always-On Wearable Smart Sensing," in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*, Jun. 2019, pp. 169–174. DOI: 10.1109/IWASI.2019.8791364.

[124] F. Conti, D. Rossi, G. Paulin, *et al.*, "22.1 A 12.4TOPS/W @ 136GOPS AI-IoT System-on-Chip with 16 RISC-V, 2-to-8b Precision-Scalable DNN Acceleration and 30%-Boost Adaptive Body Biasing," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, Feb. 2023, pp. 21–23. DOI: 10.1109/ISSCC42615.2023.10067643.

[125] M. Scherer, M. Magno, J. Erb, P. Mayer, M. Eggimann, and L. Benini, "TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition With Short Range Radars," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 336–10 346, Jul. 2021, ISSN: 2327-4662. DOI: 10.1109/JIOT.2021.3067382.

[126] M. Osta, A. Ibrahim, M. Magno, *et al.*, "An Energy Efficient System for Touch Modality Classification in Electronic Skin Applications," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–4. DOI: 10.1109/ISCAS.2019.8702113.

[127] X. Wang, L. Cavigelli, M. Eggimann, M. Magno, and L. Benini, "HR-SAR-Net: A Deep Neural Network for Urban Scene Segmentation from High-Resolution SAR Data," in *2020 IEEE Sensors Applications Symposium (SAS)*, Mar. 2020, pp. 1–6. DOI: 10.1109/SAS48726.2020.9220068.

[128] M. Magno, X. Wang, M. Eggimann, L. Cavigelli, and L. Benini, "InfiniWolf: Energy Efficient Smart Bracelet for Edge Computing with Dual Source Energy Harvesting," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2020, pp. 342–345. DOI: 10.23919/DATE48585.2020.9116218.

[129] M. Eggimann, J. Erb, P. Mayer, M. Magno, and L. Benini, "Low Power Embedded Gesture Recognition Using Novel Short-Range Radar Sensors," in *2019 IEEE SENSORS*, Oct. 2019, pp. 1–4. DOI: 10.1109/SENSORS43011.2019.8956617.

[130] K. Dheman, P. Mayer, M. Eggimann, S. Schuerle, and M. Magno, "ImpediSense:A long lasting wireless wearable bio-impedance sensor node," *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100 556, Jun. 1, 2021, ISSN: 2210-5379. DOI: 10.1016/j.suscom.2021.100556.

[131] M. Eggimann, C. Gloor, F. Scheidegger, *et al.*, "Hydra: An Accelerator for Real-Time Edge-Aware Permeability Filtering in 65nm CMOS," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, May 2018, pp. 1–5, ISBN: 978-1-5386-4881-0. DOI: 10.1109/ISCAS.2018.8351051.

[132] K. Dheman, P. Mayer, M. Eggimann, M. Magno, and S. Schuerle, "Towards Artefact-Free Bio-Impedance Measurements: Evaluation, Identification and Suppression of Artefacts at Multiple Frequencies," *IEEE Sensors Journal*, vol. 22, no. 1, pp. 589–600, Jan. 2022, ISSN: 1558-1748. DOI: 10.1109/JSEN.2021.3128555.

[133] R. David, J. Duke, A. Jain, *et al.*, "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 800–811, Mar. 15, 2021.

[134] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, ISSN: 1476-4687. DOI: 10.1038/nature14539.

[135] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, Massachusetts: The MIT Press, Nov. 18, 2016, 800 pp., ISBN: 978-0-262-03561-3.

[136] S. Raschka, *Python Machine Learning: Unlock Deeper Insights into Machine Learning with This Vital Guide to Cutting-Edge Predictive Analytics* (Community Experience Distilled). Birmingham Mumbai: Packt Publishing open source, 2016, 425 pp., ISBN: 978-1-78355-513-0.

[137] K. P. Murphy, *Probabilistic Machine Learning: An Introduction* (Adaptive Computation and Machine Learning Series). Cambridge, Massachusetts: The MIT Press, 2022, 826 pp., ISBN: 978-0-262-04682-4.

[138] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1, 1989, ISSN: 1435-568X. DOI: 10.1007/BF02551274.

[139] S. Hayou, A. Doucet, and J. Rousseau, "On the Impact of the Activation function on Deep Neural Networks Training," in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, May 24, 2019, pp. 2672–2680.

[140] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[141] Y. LeCun and I. Misra. "Self-supervised learning: The dark matter of intelligence." (Mar. 4, 2021), [Online]. Available: `https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/` (visited on 10/01/2022).

[142] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Jun. 1, 2015, pp. 448–456.

[143] F. V. Veen. "The Neural Network Zoo," The Asimov Institute. (Sep. 14, 2016), [Online]. Available: `https://www.asimovinstitute.org/neural-network-zoo/` (visited on 11/15/2022).

[144] S. Dave, R. Baghdadi, T. Nowatzki, S. Avancha, A. Shrivastava, and B. Li, "Hardware Acceleration of Sparse and Irregular Tensor Computations of ML Models: A Survey and Insights," *Proceedings of the IEEE*, vol. 109, no. 10, pp. 1706–1752, Oct. 2021, ISSN: 1558-2256. DOI: `10.1109/JPROC.2021.3098483`.

[145] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort. "A White Paper on Neural Network Quantization." arXiv: `2106.08295 [cs]`. (Jun. 15, 2021), [Online]. Available: `http://arxiv.org/abs/2106.08295` (visited on 10/02/2022), preprint.

[146] M. Nagel, M. van Baalen, T. Blankevoort, and M. Welling, "Data-Free Quantization Through Weight Equalization and Bias Correction," presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1325–1334.

[147] B. Jacob, S. Kligys, B. Chen, *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 2704–2713. DOI: `10.1109/CVPR.2018.00286`.

[148] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training Deep Neural Networks with 8-bit Floating Point Numbers," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018.

[149] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, Jun. 2009, ISSN: 1866-9956. DOI: 10.1007/s12559-009-9009-8.

[150] R. W. Gayler. "Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience." arXiv: cs/0412059. (Dec. 13, 2004), [Online]. Available: http://arxiv.org/abs/cs/0412059 (visited on 11/29/2022), preprint.

[151] D. Kleyko, M. Davies, E. P. Frady, *et al.*, "Vector Symbolic Architectures as a Computing Framework for Emerging Hardware," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1538–1571, Oct. 2022, ISSN: 1558-2256. DOI: 10.1109/JPROC.2022.3209104.

[152] G. E. Hinton, "Mapping Part-Whole Hierarchies into Connectionist Networks," *Artificial Intelligence*, no. 46, pp. 47–75, 1990, ISSN: 9780262256360.

[153] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory hyperdimensional computing," *Nature Electronics*, vol. 3, no. 6, pp. 327–337, 6 Jun. 2020, ISSN: 2520-1131. DOI: 10.1038/s41928-020-0410-3.

[154] IRDS, "International Roadmap for Devices and Systems 2022 Edition - Beyond CMOS," IRDS, 2022.

[155] IEEE Computer Society, *IEEE Standard for Floating-Point Arithmetic*, Jul. 2019.

[156] T. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, May 1995, ISSN: 1941-0093. DOI: 10.1109/72.377968.

[157] R. W. Gayler, *Multiplicative binding, representation operators & analogy (workshop poster)*, 1998.

[158] P. Kanerva, "Binary spatter-coding of ordered K-tuples," in *Artificial Neural Networks — ICANN 96*, C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 1996, pp. 869–873, ISBN: 978-3-540-68684-2. DOI: 10.1007/3-540-61510-5_146.

[159] D. A. Rachkovskij and S. V. Slipchenko, "Similarity-Based Retrieval with Structure-Sensitive Sparse Binary Distributed Representations," *Computational Intelligence*, vol. 28, no. 1, pp. 106–129, 2012, ISSN: 1467-8640. DOI: 10.1111/j.1467-8640.2011.00423.x.

[160] K. Schlegel, P. Neubert, and P. Protzel, "A comparison of vector symbolic architectures," *Artificial Intelligence Review*, vol. 55, no. 6, pp. 4523–4555, Aug. 1, 2022, ISSN: 1573-7462. DOI: 10.1007/s10462-021-10110-3.

[161] M. Schmuck, L. Benini, and A. Rahimi, "Hardware Optimizations of Dense Binary Hyperdimensional Computing: Rematerialization of Hypervectors, Binarized Bundling, and Combinational Associative Memory," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 4, 32:1–32:25, Oct. 10, 2019, ISSN: 1550-4832. DOI: 10.1145/3314326.

[162] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, IEEE, Oct. 2016, pp. 1–8, ISBN: 978-1-5090-1370-8. DOI: 10.1109/ICRC.2016.7738683.

[163] A. Moin, A. Zhou, A. Rahimi, *et al.*, "An EMG Gesture Recognition System with Flexible High-Density Sensors and Brain-Inspired High-Dimensional Classifier," *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, Feb. 2018, ISSN: 978-1-5386-4881-0. DOI: 10.1109/ISCAS.2018.8351613.

[164] A. Rahimi, A. Tchouprina, P. Kanerva, J. D. R. Millán, and J. M. Rabaey, "Hyperdimensional Computing for Blind and One-Shot Classification of EEG Error-Related Potentials," *Mobile Networks and Applications*, Oct. 2017. DOI: 10.1007/s11036-017-0942-6.

[165] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey, "Efficient Biosignal Processing Using Hyperdimensional Computing: Network Templates for Combined Learning and Classification of ExG Signals," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 123–143, Jan. 2019. DOI: 10.1109/JPROC.2018.2871163.

[166] A. Burrello, K. Schindler, L. Benini, and A. Rahimi, "One-shot Learning for iEEG Seizure Detection Using End-to-end Binary Operations: Local Binary Patterns with Hyperdimensional Computing," in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct. 2018, pp. 1–4. DOI: 10.1109/BIOCAS.2018.8584751.

[167] M. Imani, T. Nassar, A. Rahimi, and T. Rosing, "HDNA: Energy-efficient DNA sequencing using hyperdimensional computing," in *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, Mar. 2018, pp. 271–274. DOI: 10.1109/BHI.2018.8333421.

[168] T. A. Plate, "Estimating analogical similarity by dot-products of Holographic Reduced Representations," in *Advances in Neural Information Processing Systems*, vol. 6, Morgan-Kaufmann, 1993.

[169] B. Emruli, R. W. Gayler, and F. Sandin, "Analogical mapping and inference with binary spatter codes and sparse distributed memory," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013, pp. 1–8. DOI: 10.1109/IJCNN.2013.6706829.

[170] T. Yerxa, A. Anderson, and E. Weiss, "The hyperdimensional stack machine," *Cognitive Computing*, pp. 1–2, 2018.

[171] E. Osipov, D. Kleyko, and A. Legalov, "Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, Oct. 2017, pp. 3276–3281. DOI: 10.1109/IECON.2017.8216554.

[172] I. Nunes, M. Heddes, T. Givargis, and A. Nicolau. "An Extension to Basis-Hypervectors for Learning from Circular Data in Hyperdimensional Computing." arXiv: 2205.07920 [cs, math]. (May 16, 2022), [Online]. Available: http://arxiv.org/abs/2205.07920 (visited on 12/16/2022), preprint.

[173] A. Rahimi, S. Datta, D. Kleyko, *et al.*, "High-Dimensional Computing as a Nanoscalable Paradigm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2508–2521, Sep. 2017, ISSN: 1558-0806. DOI: 10.1109/TCSI.2017.2705051.

[174] P. Meinerzhagen, S. M. Y. Sherazi, A. Burg, and J. N. Rodrigues, "Benchmarking of Standard-Cell Based Memories in the Sub-V_T Domain in 65-nm CMOS Technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 173–182, Jun. 2011, ISSN: 2156-3365. DOI: 10.1109/JETCAS.2011.2162159.

[175] M. E. Sinangil, N. Verma, and A. P. Chandrakasan, "A reconfigurable 65nm SRAM achieving voltage scalability from 0.25–1.2V and performance scalability from 20kHz–200MHz," in *ESSCIRC 2008 - 34th European Solid-State Circuits Conference*, Sep. 2008, pp. 282–285. DOI: 10.1109/ESSCIRC.2008.4681847.

[176] B. Mohammadi, O. Andersson, J. Nguyen, L. Ciampolini, A. Cathelin, and J. N. Rodrigues, "A 128 kb 7T SRAM Using a Single-Cycle Boosting Mechanism in 28-nm FD–SOI," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1257–1268, Apr. 2018, ISSN: 1558-0806. DOI: 10.1109/TCSI.2017.2750762.

[177] J. D. Dixon, "The probability of generating the symmetric group," *Mathematische Zeitschrift*, vol. 110, no. 3, pp. 199–205, Jun. 1, 1969, ISSN: 1432-1823. DOI: 10.1007/BF01110210.

[178] L. Babai, "The probability of generating the symmetric group," *Journal of Combinatorial Theory, Series A*, vol. 52, no. 1, pp. 148–153, Sep. 1, 1989, ISSN: 0097-3165. DOI: 10.1016/0097-3165(89)90068-X.

[179] B. Chatterjee, N. Cao, A. Raychowdhury, and S. Sen, "Context-Aware Intelligence in Resource-Constrained IoT Nodes: Opportunities and Challenges," *IEEE Design & Test*, vol. 36, no. 2, pp. 7–40, Apr. 2019, ISSN: 2168-2364. DOI: 10.1109/MDAT.2019.2899334.

[180] S. Bagchi, T. F. Abdelzaher, R. Govindan, *et al.*, "New Frontiers in IoT: Networking, Systems, Reliability, and Security Challenges," *IEEE Internet of Things Journal*, pp. 1–1, Jul. 2020, ISSN: 2327-4662. DOI: 10.1109/JIOT.2020.3007690. arXiv: 2005.07338.

[181] D. Newell and M. Duffy, "Review of Power Conversion and Energy Management for Low-Power, Low-Voltage Energy Harvesting Powered Wireless Sensors," *IEEE Transactions on Power Electronics*, vol. 34, no. 10, pp. 9794–9805, Oct. 2019, ISSN: 1941-0107. DOI: 10.1109/TPEL.2019.2894465.

[182] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, 7 Jul. 2020, ISSN: 1748-3395. DOI: 10.1038/s41565-020-0655-z.

[183] I. Miro-Panades, B. Tain, J.-F. Christmann, *et al.*, "SamurAI: A 1.7MOPS-36GOPS Adaptive Versatile IoT Node with 15,000× Peak-to-Idle Power Reduction, 207ns Wake-Up Time and 1.3TOPS/W ML Efficiency," in *2020 IEEE Symposium on VLSI Circuits*, Jun. 2020, pp. 1–2. DOI: 10.1109/VLSICircuits18222.2020.9163000.

[184] D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, "Sensing, Computing, and Communications for Energy Harvesting IoTs: A Survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 1222–1250, Dec. 2020, ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2962526.

[185] J. S. P. Giraldo, S. Lauwereins, K. Badami, and M. Verhelst, "Vocell: A 65-nm Speech-Triggered Wake-Up SoC for 10- µW Keyword Spotting and Speaker Verification," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 868–878, Apr. 2020, ISSN: 1558-173X. DOI: 10.1109/JSSC.2020.2968800.

[186] W. Shan, M. Yang, J. Xu, *et al.*, "14.1 A 510nW 0.41V Low-Memory Low-Computation Keyword-Spotting Chip Using Serial FFT-Based MFCC and Binarized Depthwise Separable Convolutional Neural Network in 28nm CMOS," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb. 2020, pp. 230–232. DOI: 10.1109/ISSCC19947.2020.9063000.

[187] Y. Zhao, Z. Shang, and Y. Lian, "A 13.34 µW Event-Driven Patient-Specific ANN Cardiac Arrhythmia Classifier for Wearable ECG Sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 2, pp. 186–197, Apr. 2020, ISSN: 1940-9990. DOI: 10.1109/TBCAS.2019.2954479.

[188] Z. Wang, L. Ye, H. Zhang, *et al.*, "20.2 A 57nW Software-Defined Always-On Wake-Up Chip for IoT Devices with Asynchronous Pipelined Event-Driven Architecture and Time-Shielding Level-Crossing ADC," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb. 2020, pp. 314–316. DOI: 10.1109/ISSCC19947.2020.9062952.

[189] M. Cho, S. Oh, Z. Shi, *et al.*, "17.2 A 142nW Voice and Acoustic Activity Detection Chip for mm-Scale Sensor Nodes Using Time-Interleaved Mixer-Based Frequency Scanning," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb. 2019, pp. 278–280. DOI: 10.1109/ISSCC.2019.8662540.

[190] A. Joshi, J. T. Halseth, and P. Kanerva, "Language Geometry Using Random Indexing," in *Quantum Interaction*, J. A. de Barros, B. Coecke, and E. Pothos, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, Jan. 2017, pp. 265–274, ISBN: 978-3-319-52289-0. DOI: 10.1007/978-3-319-52289-0_21.

[191] D. Kleyko and E. Osipov, "Brain-like classifier of temporal patterns," in *2014 International Conference on Computer and Information Sciences (ICCOINS)*, Jun. 2014, pp. 1–6. DOI: 10.1109/ICCOINS.2014.6868349.

[192] T. F. Wu, H. Li, P.-C. Huang, *et al.*, "Hyperdimensional Computing Exploiting Carbon Nanotube FETs, Resistive RAM, and Their Monolithic 3D Integration," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3183–3196, Nov. 2018. DOI: 10.1109/JSSC.2018.2870560.

[193] S. Salamat, M. Imani, B. Khaleghi, and T. Rosing, "F5-HD: Fast Flexible FPGA-based Framework for Refreshing Hyperdimensional Computing," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19, New York, NY, USA: Association for Computing Machinery, Feb. 20, 2019, pp. 53–62, ISBN: 978-1-4503-6137-8. DOI: 10.1145/3289602.3293913.

[194] S. Salamat, M. Imani, and T. Rosing, "Accelerating Hyperdimensional Computing on FPGAs by Exploiting Computational Reuse," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1159–1171, Aug. 2020, ISSN: 1557-9956. DOI: 10.1109/TC.2020.2992662.

[195] S. Datta, R. A. G. Antonio, A. R. S. Ison, and J. M. Rabaey, "A Programmable Hyper-Dimensional Processor Architecture for Human-Centric IoT," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 3, pp. 439–452, Sep. 2019, ISSN: 2156-3365. DOI: 10.1109/JETCAS.2019.2935464.

[196] M. Gautschi, P. D. Schiavone, A. Traber, *et al.*, "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2017.2654506.

[197] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A Robust and Energy-Efficient Classifier Using Brain-Inspired Hyperdimensional Computing," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ser. ISLPED '16, San Francisco Airport, CA, USA: Association for Computing Machinery, Aug. 8, 2016, pp. 64–69, ISBN: 978-1-4503-4185-1. DOI: 10.1145/2934583.2934624.

[198] L. Ge and K. K. Parhi, "Classification using Hyperdimensional Computing: A Review," *IEEE Circuits and Systems Magazine*, vol. 20, no. 2, pp. 30–47, Apr. 2020, ISSN: 1531-636X, 1558-0830. DOI: 10.1109/MCAS.2020.2988388. arXiv: 2004.11204.

[199] S. Selcuk, "Predictive maintenance, its implementation and latest trends," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 231, no. 9, pp. 1670–1679, Jul. 1, 2017, ISSN: 0954-4054. DOI: 10.1177/0954405415601640.

[200] H. Qiu, J. Lee, J. Lin, and G. Yu, "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics," *Journal of Sound and Vibration*, vol. 289, no. 4,

pp. 1066–1090, Feb. 7, 2006, ISSN: 0022-460X. DOI: 10.1016/j.jsv.2005.03.007.

[201] J. Ben Ali, L. Saidi, A. Mouelhi, B. Chebel-Morello, and F. Fnaiech, "Linear feature selection and classification using PNN and SFAM neural networks for a nearly online diagnosis of bearing naturally progressing degradations," *Engineering Applications of Artificial Intelligence*, vol. 42, pp. 67–81, Jun. 1, 2015, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2015.03.013.

[202] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, "Quentin: An Ultra-Low-Power PULPissimo SoC in 22nm FDX," in *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, Oct. 2018, pp. 1–3. DOI: 10.1109/S3S.2018.8640145.

[203] A. Rahimi, I. Loi, M. R. Kakoee, and L. Benini, "A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters," in *2011 Design, Automation & Test in Europe*, IEEE, Mar. 2011, pp. 1–6, ISBN: 978-3-9810801-8-6. DOI: 10.1109/DATE.2011.5763085.

[204] A. Pullini, D. Rossi, G. Haugou, and L. Benini, "µDMA: An autonomous I/O subsystem for IoT end-nodes," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sep. 2017, pp. 1–8. DOI: 10.1109/PATMOS.2017.8106971.

[205] Y. Kaya, M. Uyar, R. Tekin, and S. Yıldırım, "1D-local binary pattern based feature extraction for classification of epileptic EEG signals," *Applied Mathematics and Computation*, vol. 243, pp. 209–219, Sep. 15, 2014, ISSN: 0096-3003. DOI: 10.1016/j.amc.2014.05.128.

[206] A. Burrello, L. Cavigelli, K. Schindler, L. Benini, and A. Rahimi, "Laelaps: An Energy-Efficient Seizure Detection Algorithm from Long-term Human iEEG Recordings without False Alarms," in *Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, Mar. 2019, pp. 752–757, ISBN: 978-3-9819263-2-3. DOI: 10.3929/ethz-b-000307983.

[207] T. Haine, Q.-K. Nguyen, F. Stas, L. Moreau, D. Flandre, and D. Bol, "An 80-MHz 0.4V ULV SRAM macro in 28nm FDSOI achieving 28-fJ/bit access energy with a ULP bitcell and on-chip adaptive back bias generation," in *ESSCIRC 2017 - 43rd IEEE European Solid State Circuits Conference*, 2017, pp. 312–315. DOI: 10.1109/ESSCIRC.2017.8094588.

[208] N. Reynders and W. Dehaene, "Variation-resilient building blocks for ultra-low-energy sub-threshold design," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 12, pp. 898–902, 2012. DOI: 10.1109/TCSII.2012.2231022.

[209] P. Prabhat, B. Labbe, G. Knight, *et al.*, "27.2 M0N0: A performance-regulated 0.8-to-38MHz DVFS ARM cortex-m33 SIMD MCU with 10nW sleep power," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 422–424. DOI: 10.1109/ISSCC19947.2020.9063136.

[210] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 246–247. DOI: 10.1109/ISSCC.2017.7870353.

[211] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 218–220. DOI: 10.1109/ISSCC.2018.8310262.

[212] S. Kang, D. Han, J. Lee, *et al.*, "GANPU: An energy-efficient multi-dnn training processor for GANs with speculative dual-sparsity exploitation," *IEEE Journal of Solid-State Circuits*, pp. 1–1, 2021. DOI: 10.1109/JSSC.2021.3066572.

[213] A. Agrawal, S. K. Lee, J. Silberman, *et al.*, "9.1 a 7nm 4-Core AI chip with 25.6TFLOPS hybrid FP8 training, 102.4TOPS INT4 inference and workload-aware throttling," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 144–146. DOI: 10.1109/ISSCC42613.2021.9365791.

[214] "HyperRAM & octal xSPI RAM memory." (), [Online]. Available: https://www.cypress.com/products/hyperram-octal-xspi-ram-memory (visited on 03/05/2021).

[215] "PSRAM & IoT RAM." (), [Online]. Available: https://www.apmemory.com/products/psram-iot-ram/ (visited on 03/05/2021).

[216] A. Pullini, D. Rossi, I. Loi, G. Tagliavini, and L. Benini, "Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 7, pp. 1970–1981, Jul. 2019, ISSN: 1558-173X. DOI: 10.1109/JSSC.2019.2912307.

[217] R. K. Y.B. and R. K. C.N., "Local binary pattern: An improved LBP to extract nonuniform LBP patterns with Gabor filter to increase the rate of face similarity," in *2016 Second International Conference on Cognitive Computing and Information Processing (CCIP)*, Aug. 2016, pp. 1–5. DOI: 10.1109/CCIP.2016.7802878.

[218] A. Moin, A. Zhou, A. Rahimi, *et al.*, "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition," *Nature Electronics*, vol. 4, no. 1, pp. 54–63, 1 Jan. 2021, ISSN: 2520-1131. DOI: 10.1038/s41928-020-00510-8.

[219] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey, "Exploring Hyperdimensional Associative Memory," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Austin, TX: IEEE, Feb. 2017, pp. 445–456, ISBN: 978-1-5090-4985-1. DOI: 10.1109/HPCA.2017.28.

[220] S. Mach, F. Schuiki, F. Zaruba, and L. Benini, "FPnew: An Open-Source Multiformat Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 774–787, Apr. 2021, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2020.3044752.

[221] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making VGG-style ConvNets Great Again," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 13 728–13 737. DOI: 10.1109/CVPR46437.2021.01352.

[222] A. Garofalo, M. Rusci, F. Conti, D. Rossi, and L. Benini, "PULP-NN: A Computing Library for Quantized Neural Network inference at the edge on RISC-V Based Parallel Ultra Low Power Clusters," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Nov. 2019, pp. 33–36. DOI: 10.1109/ICECS46596.2019.8965067.

[223] 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*, pp. 1–1, 2021. DOI: 10.1109/TC.2021.3066883.

[224] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware," *CoRR*, vol. abs/1812.00332, 2018. arXiv: 1812.00332.

[225] Á. Incze, H.-B. Jancsó, Z. Szilágyi, A. Farkas, and C. Sulyok, "Bird sound recognition using a convolutional neural network," in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, 2018, pp. 000 295–000 300. DOI: 10.1109/SISY.2018.8524677.

[226] B. Zhang, Y. Zhang, and S. Wang, "A Lightweight and Discriminative Model for Remote Sensing Scene Classification With Multidilation Pooling Module," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 8, pp. 2636–2653, 2019. DOI: 10.1109/JSTARS.2019.2919317.

[227] C. Schmidt, J. Wright, Z. Wang, *et al.*, "4.3 An Eight-Core 1.44GHz RISC-V Vector Machine in 16nm FinFET," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, Feb. 2021, pp. 58–60. DOI: 10.1109/ISSCC42613.2021.9365789.

[228] D. Bol, M. Schramme, L. Moreau, *et al.*, "SleepRunner: A 28-nm FDSOI ULP Cortex-M0 MCU With ULL SRAM and UFBR PVT Compensation for 2.6–3.6-µW/DMIPS 40–80-MHz Active Mode and 131-nW/kB Fully Retentive Deep-Sleep Mode," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2256–2269, Jul. 2021, ISSN: 1558-173X. DOI: 10.1109/JSSC.2021.3056219.

[229] E. Flamand, D. Rossi, F. Conti, *et al.*, "GAP-8: A RISC-V SoC for AI at the edge of the IoT," in *2018 IEEE 29th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, 2018, pp. 1–4. DOI: 10.1109/ASAP.2018.8445101.

[230] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. "A Survey of Quantization Methods for Efficient Neural Network Inference." arXiv: 2103.13630 [cs]. (Jun. 21, 2021), [Online]. Available: http://arxiv.org/abs/2103.13630 (visited on 04/10/2023), preprint.

[231] J. Gomez, S. Patel, S. S. Sarwar, *et al.* "Distributed On-Sensor Compute System for AR/VR Devices: A Semi-Analytical Simulation Framework for Power Estimation." arXiv: 2203.07474 [cs]. (Mar. 14, 2022), [Online]. Available: http://arxiv.org/abs/2203.07474 (visited on 01/09/2023), preprint.

[232] H. Murakami, E. Bohannon, J. Childs, *et al.*, "A 4.9Mpixel Programmable-Resolution Multi-Purpose CMOS Image Sensor for Computer Vision," in *2022 IEEE International Solid- State Circuits*

Conference (ISSCC), vol. 65, Feb. 2022, pp. 104–106. DOI: 10.1109/ISSCC42614.2022.9731607.

[233] S. Han, B. Liu, R. Cabezas, *et al.*, "MEgATrack: Monochrome egocentric articulated hand-tracking for virtual reality," *ACM Transactions on Graphics*, vol. 39, no. 4, 87:87:1–87:87:13, Aug. 12, 2020, ISSN: 0730-0301. DOI: 10.1145/3386569.3392452.

[234] R. Eki, S. Yamada, H. Ozawa, *et al.*, "9.6 A 1/2.3inch 12.3Mpixel with On-Chip 4.97TOPS/W CNN Processor Back-Illuminated Stacked CMOS Image Sensor," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, Feb. 2021, pp. 154–156. DOI: 10.1109/ISSCC42613.2021.9365965.

[235] 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)*, Aug. 2022, pp. 1–19. DOI: 10.1109/HCS55958.2022.9895621.

Curriculum Vitæ

Manuel Eggimann was born on February 1, 1993 and grew up in Zurich, Switzerland. He received his B.Sc and M.Sc. degree in "Electrical Engineering and Information Technology" at ETH Zurich in 2016 and 2018, respectively. Late 2018 he joined the Integrated Systems Laboratory under the supervision of Prof. Dr. Luca Benini to pursue a PhD degree. His research interest lies in energy-efficient integrated circuit design for embedded machine learning and system-level architectures for always-on near-sensor analytics.