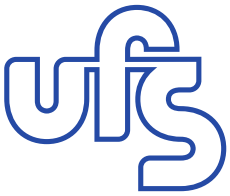


FEDERAL UNIVERSITY OF SERGIPE

CENTER FOR EXACT SCIENCES AND TECHNOLOGY

GRADUATE PROGRAM IN COMPUTER SCIENCE

A Comparative Survey of ARM and RISC-V: Architectures, Applications and Challenges

Master's Thesis

Andre Reges Souza Meira

São Cristóvão – Sergipe

2026

FEDERAL UNIVERSITY OF SERGIPE

CENTER FOR EXACT SCIENCES AND TECHNOLOGY

GRADUATE PROGRAM IN COMPUTER SCIENCE

Andre Reges Souza Meira

A Comparative Survey of ARM and RISC-V: Architectures, Applications and Challenges

Master's Thesis submitted to the Graduate Program in Computer Science as a partial requirement for obtaining the degree of Master in Computer Science.

Advisor: Edward David Moreno

São Cristóvão – Sergipe

2026

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: ANDRÉ REGES SOUZA MEIRA

Aos 22 dias do mês de julho do ano de dois mil e vinte seis, com início às 10hs, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **ANDRÉ REGES SOUZA MEIRA** que desenvolveu o trabalho intitulado: **"*A Comparative Survey of ARM and RISC-V: Architectures, Applications and Challenges*",** sob a orientação do Prof. **Dr. Edward David Moreno Ordonez.** A Sessão foi presidida pelo Prof. **Dr. Edward David Moreno Ordonez** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, o **Dr.Cesar Giacomini Penteado (UNIMAR)** e, em seguida, **Dr. Calebe Micael de Oliveira Conceição (DCOMP/UFS) e Dr. Rafael Oliveira Vasconcelos (PROCC/UFS)**. Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando **APROVADO**. Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 22 de julho de 2026.

gov.br Documento assinado digitalmente
EDWARD DAVID MORENO ORDONEZ
Data: 28/07/2026 09:45:52-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Edward David Moreno Ordonez
(PROCC/UFS)
Presidente

gov.br Documento assinado digitalmente
RAFAEL OLIVEIRA VASCONCELOS
Data: 28/07/2026 09:54:30-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rafael Oliveira Vasconcelos
(PROCC/UFS)
Examinador Interno

gov.br Documento assinado digitalmente
CALEBE MICAEL DE OLIVEIRA CONCEICAO
Data: 28/07/2026 11:34:21-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Calebe Micael de Oliveira Conceição
(UFS)
Examinador Externo ao programa

gov.br Documento assinado digitalmente
CESAR GIACOMINI PENTEADO
Data: 28/07/2026 14:43:04-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Cesar Giacomini Penteado
(UNIMARE)
Examinador Externo à Instituição

ANDRÉ REGES SOUZA MEIRA
Candidato

gov.br Documento assinado digitalmente
ANDRE REGES SOUZA MEIRA
Data: 28/07/2026 15:15:56-0300
Verifique em https://validar.iti.gov.br

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

M514c

Meira, Andre Reges Souza

A comparative survey of ARM and RISC-V: architectures, applications and challenges / Andre Reges Souza Meira ; orientador Edward David Moreno. – São Cristóvão, 2026.

124 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2026.

1. Sistemas embarcados (Computadores). 2. Robótica. 3. Inteligência artificial. 4. Computadores. I. Moreno, Edward David orient. II. Título.

CDU 004.2

Acknowledgements

(This section is written in Portuguese.)

Agradeço à minha família, e em especial à minha mãe, Divanizia, pelo amor incondicional, pelo apoio em cada decisão e por sempre acreditar em mim, mesmo nos momentos mais difíceis.

Ao meu orientador, Prof. Dr. Edward David Moreno, pela orientação atenta, pela confiança depositada em mim e por todo o conhecimento compartilhado, fundamentais para a realização deste trabalho. Também, ao meu coorientador, Prof. Dr. Calebe Conceição, pelas valiosas contribuições e pelo acompanhamento ao longo da pesquisa.

Aos professores do Programa de Pós-Graduação em Ciência da Computação (PROCC), pelos ensinamentos e pela dedicação que enriqueceram a minha formação acadêmica.

À Universidade Federal de Sergipe (UFS), pela infraestrutura e pelas oportunidades que tornaram possível a realização deste mestrado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Abstract

The selection of a processor instruction set architecture (ISA) has become a foundational engineering decision that shapes the performance, power efficiency, security, and supply-chain independence of modern computing systems. Three converging trends make a renewed comparison of ARM and RISC-V especially timely: the expansion of computing into safety- and security-critical domains governed by standards such as ISO 26262 and IEC 61508; the rise of RISC-V as the first credible open alternative to ARM, with over ten billion cores shipped by 2024; and the growing demand for hardware auditability following microarchitectural attacks such as Spectre and Meltdown and supply-chain compromises such as the XZ Utils backdoor.

In this master's thesis, we provide a comprehensive, parallel-structured comparative survey of the ARM and RISC-V architectures, spanning ISA design and evolution, microarchitecture, vector processing, and virtualization, together with the application domains of robotics, artificial intelligence, and hardware security — dimensions that existing ISA-level, benchmarking, and industry analyses address only in isolation.

The work is an architectural survey rather than an experimental benchmarking study, grounded in ISA specifications, peer-reviewed literature from IEEE, ACM, USENIX, and Springer venues, vendor datasheets and safety documentation, and international functional-safety standards, and organised as a chapter-by-chapter parallel evaluation of the two architectures.

The analysis finds that the gap between ARM and RISC-V is increasingly one of ecosystem maturity rather than architectural capability: ARM offers a vertically integrated, broadly deployed ecosystem with hardware-enforced security defaults (PAC, MTE, BTI, CCA) and certified safety cores, whereas RISC-V offers openness, amenability to formal verification, and license-free customisation, with ISO 26262 ASIL D cores now emerging. For artificial-intelligence workloads, RISC-V demonstrates up to fourfold superior energy efficiency through custom extensions, while ARM delivers up to fifteenfold faster inference on complex networks through a mature toolchain; these figures are best-case, implementation-dependent trends rather than fixed architectural verdicts. Across robotics and security, architecture selection proves application- and certification-dependent rather than absolute, and the frontier is shifting from a binary ARM-versus-RISC-V choice toward heterogeneous designs that compose both architectures with domain-specific accelerators on a single platform.

Keywords: ARM. RISC-V. Instruction Set Architecture. Embedded Systems. Robotics. Artificial Intelligence. Functional Safety. Hardware Security.

Resumo

A escolha da arquitetura do conjunto de instruções (ISA) de um processador tornou-se uma decisão de engenharia fundamental, que determina o desempenho, a eficiência energética, a segurança e a independência da cadeia de suprimentos dos sistemas computacionais modernos. Três tendências convergentes tornam especialmente oportuna uma nova comparação entre ARM e RISC-V: a expansão da computação para domínios críticos quanto à segurança funcional (*safety*) e à segurança computacional (*security*), regidos por normas como a ISO 26262 e a IEC 61508; a ascensão do RISC-V como a primeira alternativa aberta verdadeiramente viável ao ARM, com mais de dez bilhões de núcleos produzidos até 2024; e a crescente demanda por auditabilidade do *hardware* após ataques microarquiteturais como Spectre e Meltdown e comprometimentos da cadeia de suprimentos como o *backdoor* no XZ Utils.

Nesta dissertação, apresentamos um levantamento comparativo abrangente e de estrutura paralela das arquiteturas ARM e RISC-V, abordando o projeto e a evolução da ISA, a microarquitetura, o processamento vetorial e a virtualização, juntamente com os domínios de aplicação de robótica, inteligência artificial e segurança de *hardware* — dimensões que as análises existentes em nível de ISA, de *benchmarking* e da indústria tratam apenas de forma isolada.

O trabalho constitui um levantamento arquitetural, e não um estudo experimental de *benchmarking*, fundamentado em especificações de ISA; na literatura revisada por pares, de veículos como IEEE, ACM, USENIX e Springer; em manuais e documentos de segurança de fabricantes; e em normas internacionais de segurança funcional; e organizado como uma avaliação paralela das duas arquiteturas, em cada capítulo desta dissertação.

A análise constata que a diferença entre ARM e RISC-V é cada vez mais uma questão de maturidade de ecossistema do que de capacidade arquitetural: o ARM oferece um ecossistema verticalmente integrado e amplamente difundido, com mecanismos de segurança impostos por *hardware* e ativados por padrão (PAC, MTE, BTI, CCA) e núcleos com certificação de segurança funcional, ao passo que o RISC-V oferece abertura, viabilidade de verificação formal e personalização livre de licenciamento, com núcleos certificados em ISO 26262 ASIL D já surgindo. Para cargas de trabalho de inteligência artificial, o RISC-V demonstra eficiência energética até quatro vezes superior por meio de extensões personalizadas, enquanto o ARM entrega inferência até quinze vezes mais rápida em redes complexas por meio de uma cadeia de ferramentas madura; tais valores constituem tendências de melhor caso, dependentes da implementação, e não veredictos arquiteturais fixos. Em robótica e segurança, a escolha da arquitetura mostra-se dependente da aplicação e da certificação, e não absoluta, e a fronteira está se deslocando de uma escolha binária entre ARM e RISC-V para projetos heterogêneos

que combinam ambas as arquiteturas com aceleradores específicos de domínio em uma única plataforma.

Palavras-chave: ARM. RISC-V. Arquitetura de Conjunto de Instruções. Sistemas Embarcados. Robótica. Inteligência Artificial. Segurança Funcional. Segurança de *Hardware*.

List of Figures

List of Tables

List of abbreviations and acronyms

AI	Artificial Intelligence
ALU	Arithmetic Logic Unit
ARM	Advanced RISC Machines
ASIL	Automotive Safety Integrity Level
BTI	Branch Target Identification
CCA	Confidential Compute Architecture
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSR	Control and Status Register
DCLS	Dual-Core Lockstep
DMA	Direct Memory Access
DNN	Deep Neural Network
DSP	Digital Signal Processor
FPGA	Field-Programmable Gate Array
FPU	Floating-Point Unit
GIC	Generic Interrupt Controller
GPR	General-Purpose Register
GPU	Graphics Processing Unit
HPC	High-Performance Computing
IMU	Inertial Measurement Unit
IOMMU	Input/Output Memory Management Unit
IP	Intellectual Property
IRQ	Interrupt Request
ISA	Instruction Set Architecture

KVM	Kernel-based Virtual Machine
ML	Machine Learning
MMU	Memory Management Unit
MPU	Memory Protection Unit
MTE	Memory Tagging Extension
NPU	Neural Processing Unit
OoO	Out-of-Order
PAC	Pointer Authentication Code
PID	Proportional–Integral–Derivative
PMP	Physical Memory Protection
PTE	Page-Table Entry
RISC	Reduced Instruction Set Computer
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
RTL	Register-Transfer Level
RTOS	Real-Time Operating System
RVV	RISC-V Vector extension
SIMD	Single Instruction, Multiple Data
SIMT	Single Instruction, Multiple Threads
SLAM	Simultaneous Localization and Mapping
SME	Scalable Matrix Extension
SoC	System-on-Chip
SRAM	Static Random-Access Memory
SVE	Scalable Vector Extension
TCB	Trusted Computing Base
TCM	Tightly Coupled Memory

TEE	Trusted Execution Environment
TLB	Translation Lookaside Buffer
TOPS	Tera Operations Per Second
UAV	Unmanned Aerial Vehicle
WCET	Worst-Case Execution Time
YOLO	You Only Look Once

Contents

1 Introduction

The computational demands of modern computing systems have undergone a fundamental transformation. From data centers running AI workloads at hyperscale to autonomous robots operating in unstructured environments alongside humans, the choice of processor architecture has become a foundational decision affecting performance, power efficiency, security, and, increasingly, supply-chain independence. For decades, two architectural paradigms have dominated this landscape: x86, which controls the desktop and server markets, and ARM (Advanced RISC Machines), which dominates mobile and embedded systems and, increasingly, data centers. The recent emergence of RISC-V as a credible alternative to ARM in the embedded and application processor space has introduced a third option that fundamentally changes the architectural decision-making process for system designers.

This master's thesis presents a comparative survey of the ARM and RISC-V architectures, examining their design philosophies, technical capabilities, application domains, and the challenges each faces in the current computing landscape. The comparison is particularly relevant now because of the convergence of three major trends in the industry.

The expansion of computing into safety-critical and security-critical domains. Autonomous systems, medical devices, automotive electronics, and industrial control systems are increasingly entering applications where software and hardware failures have physical consequences. These applications require functional safety certification under standards such as ISO 26262 (International Organization for Standardization, 2018) (automotive), IEC 61508 (International Electrotechnical Commission, 2010) (industrial), and DO-178C (aerospace), which impose rigorous requirements on hardware isolation, fault detection, and deterministic timing that must be satisfied at the processor level. The processor architecture is no longer merely a performance parameter; it is the foundation upon which the entire safety and security case is built.

The emergence of RISC-V as a credible alternative to ARM. ARM has been the

dominant architecture for embedded and mobile computing for over a decade. Virtually every commercially available smartphone, the majority of microcontrollers, and most commercial robotic platforms run on ARM-based Systems-on-Chip (SoCs). This dominance reflects a mature ecosystem, excellent power efficiency, and a comprehensive profile system (Cortex-A/R/M) targeting different computational roles (BURNS; DAVIS, 2017). However, RISC-V has emerged as the first credible alternative, with over 10 billion cores shipped by 2024 (RISC-V International, 2024e), commercial out-of-order implementations approaching ARM's single-threaded performance, and ISO 26262-certified safety cores now available from multiple vendors (SiFive, Inc., 2024).

The growing importance of hardware auditability and supply-chain trust. The discovery of Spectre (KOCHER et al., 2019) and Meltdown (LIPP et al., 2018) in 2018 demonstrated that security depends on microarchitectural properties invisible in the ISA specification. Simultaneously, supply-chain attacks such as the XZ Utils backdoor (CVE-2024-3094) (FREUND, 2024) revealed that trusting closed-source components creates systemic risk. For systems requiring high-assurance certification, the ability to formally verify hardware isolation mechanisms, possible with open-source RISC-V register-transfer-level (RTL) code but not with closed ARM intellectual property (IP), is shifting from a theoretical advantage to a practical certification enabler. European sovereignty initiatives such as the European Processor Initiative (European Processor Initiative Consortium, 2018), India's SHAKTI project (MADHUSUDAN et al., 2016), and substantial RISC-V investment across Asia reflect this shift at the geopolitical level.

1.1 Problem Statement

Despite the growing relevance of both ARM and RISC-V across embedded, mobile, automotive, and emerging application domains, to the best of our knowledge no comprehensive comparative analysis exists that evaluates these architectures across the full stack—from ISA design through pipeline implementation, vector processing, virtualization, application domains (robotics, artificial intelligence), and security—in a unified, structured framework.

Existing literature tends to fall into one of three categories: ISA-level analyses that compare instruction encoding and register architecture without addressing system-level concerns; performance benchmarking studies that measure throughput on specific workloads without evaluating the architectural features relevant to safety, security, or specific application domains; and industry whitepapers that advocate for one architecture without rigorous comparative analysis. None provides the structured, parallel evaluation that a system designer needs to make an informed architecture choice for a modern computing platform.

This master's thesis addresses this gap by providing a comprehensive, parallel-structure survey of both architectures, covering their internal design and evolution, their applications in robotics and artificial intelligence, and their respective approaches to the security challenges of

contemporary computing systems.

1.2 Scope and Methodology

This work is a comparative architectural analysis, a survey, rather than an experimental benchmarking study. The analysis is grounded in ISA specifications, published microarchitectural documentation, academic literature from IEEE, ACM, USENIX, and Springer venues, vendor datasheets and safety documentation, and international standards (ISO 26262, IEC 61508, CAST-32A).

The methodological approach is a structured parallel comparison: the ARM and RISC-V surveys (Chapters 2 and 3) follow an identical section organization (historical origins, ISA design, register architecture, privilege model, pipeline evolution, vector/SIMD processing, virtualization, and safety-critical systems), enabling direct feature-by-feature cross-architectural evaluation. The application chapters (robotics, artificial intelligence) examine how each architecture is deployed in commercial and research systems, drawing on benchmarking literature and case studies. The security chapter synthesizes findings from across the thesis into a unified six-layer threat model, comparing each architecture's mechanisms for hardware root of trust, privilege isolation, trusted execution environments, memory safety, side-channel resistance, and supply-chain trust.

The scope is intentionally broad rather than deep on any single dimension: the goal is to provide a system architect with a comprehensive map of both architectures' capabilities and limitations across multiple application domains, rather than an exhaustive treatment of any individual subsystem.

1.3 Contributions

The choice of processor architecture for modern computing systems is not merely a technical decision about performance and power efficiency, but a strategic decision affecting supply-chain independence, certification transparency, and long-term architectural control. This is reflected in the dual nature of the ARM–RISC-V comparison: while ARM offers a mature, vertically integrated ecosystem with the trade-off of vendor dependence, RISC-V offers an open, auditable architecture with the trade-off of ecosystem fragmentation and less mature tooling. In this master's thesis, we examine this comparison in depth and make the following contributions:

1. A comprehensive survey of the ARM architecture from its origins through ARMv9, structured for application across embedded, mobile, and safety-critical domains. The survey covers the A/R/M profile taxonomy, pipeline evolution from Cortex-A53 through Cortex-X4, the SIMD/vector processing lineage from NEON through SVE2 and SME, hardware virtualization via EL2 and the GICv3/v4, TrustZone and CCA security architectures, and the safety-critical ecosystem.

2. A parallel survey of the RISC-V architecture, structured to enable direct cross-architectural comparison with the ARM survey. The survey covers the modular extension system, privilege modes and PMP, the RVV 1.0 vector extension, H-extension virtualization, pipeline implementations from academic (Rocket, BOOM) through commercial (SiFive P870, Ventana Veyron, Tenstorrent Ascalon), and the emerging safety certification ecosystem.

3. A study of the application of ARM and RISC-V architectures in robotic systems, examining the dominance of ARM in commercial robotics platforms (NVIDIA Jetson, Qualcomm Robotics, NXP i.MX), the emerging RISC-V research platforms (PULP, GAP-8, SiFive-based prototypes), and the architectural features relevant to robotic workloads such as SLAM, sensor fusion, motion planning, and real-time control.

4. A study of the application of ARM and RISC-V architectures in artificial intelligence workloads, covering edge AI inference, training acceleration, the integration of NPUs and GPUs with each architecture, and the role of vector extensions (NEON, SVE, SVE2, SME for ARM; RVV for RISC-V) in machine learning performance.

5. A six-layer security architecture comparison evaluating ARM and RISC-V for modern computing platforms, covering hardware root of trust, privilege isolation, trusted execution environments, memory safety mechanisms, side-channel and speculative execution defenses, and supply-chain trust and auditability.

1.4 Thesis Organization

The remainder of this master's thesis is organized as follows:

Chapter 2 presents a comprehensive survey of the ARM architecture, covering its evolution from the original Acorn RISC Machine through the current ARMv9 generation. Chapter 3 provides a parallel survey of the RISC-V architecture, following the same organizational structure as Chapter 2 to enable direct cross-architectural comparison. Chapter 4 examines the application of both architectures in robotic systems, surveying commercial and research deployments and analyzing the architectural features relevant to robotic workloads. Chapter 5 examines the application of both architectures in artificial intelligence workloads, covering edge inference, training acceleration, and the role of vector and matrix extensions in machine learning performance. Chapter 6 presents a comparative security analysis structured as six layers of the security stack, evaluating how ARM and RISC-V each address the threat model of modern computing platforms. Chapter 7 concludes the master's thesis with a synthesis of findings across all chapters, a summary of the key architectural trade-offs, and directions for future work.

2

The ARM Architecture

This chapter provides a comprehensive technical overview of the ARM architecture's trajectory, detailing its transition from a modest 1985 design into a dominant force in global computing. The analysis focuses on the differentiation between the A (Application), R (Real-Time), and M (Microcontroller) profiles, explaining how each meets specific needs, ranging from smartphones and servers to safety-critical embedded systems. It explores the fundamental evolutions of the ARMv7, ARMv8, and ARMv9 versions, highlighting advancements in 64-bit processing, artificial intelligence, and confidential computing. We also emphasize the importance of innovations such as the Cortex-R82 and the use of the Rust programming language to ensure determinism and protection in industrial and automotive environments. Finally, the references describe how the licensing model and energy efficiency allowed ARM technology to become the foundation of nearly all modern digital infrastructure.

2.1 Introduction

The ARM instruction set architecture represents one of the most consequential developments in computing history. Originally designed by Sophie Wilson and Steve Furber at Acorn Computers in Cambridge, UK, the architecture has grown from a 25,000-transistor processor for the BBC Micro ecosystem to the foundation of an industry shipping over 280 billion chips cumulatively as of 2024 (FURBER, 2000). ARM-based processors now power approximately 95% of the world's smartphones, the majority of embedded systems, a rapidly growing share of datacenter infrastructure through platforms such as AWS Graviton, Google Axion, and Microsoft Azure Cobalt, and the world's first ARM-based exascale-class supercomputer (Fugaku, based on the Fujitsu A64FX) (KODAMA et al., 2020).

This extraordinary breadth of deployment is enabled by ARM's architectural philosophy: a modular, profile-based ISA that can be implemented across a range of microarchitectures

spanning sub-milliwatt microcontrollers (Cortex-M0+) to multi-gigahertz out-of-order server cores (Neoverse V2), with a consistent programming model at each tier. The architecture has undergone three major evolutionary phases that define the modern ARM landscape.

ARMv7 (2004–2013) formalized the three architecture profiles—Application (A), Real-Time (R), and Microcontroller (M)—and introduced the features that won the mobile market: Thumb-2 instruction encoding, NEON SIMD processing, hardware virtualization extensions, and the big.LITTLE heterogeneous computing model (Arm Ltd., 2018).

ARMv8 (2011–2020) introduced AArch64, a clean-sheet 64-bit ISA that broke from the accumulated complexity of 32-bit ARM. Through eight sub-versions (v8.0–v8.7), ARMv8 added Scalable Vector Extension (SVE) (STEPHENS et al., 2017a), Large System Extensions (LSE) atomics, Pointer Authentication Codes (PAC) (LILJESTRAND et al., 2019), Memory Tagging Extension (MTE) (SEREBRYANY et al., 2018), Virtualization Host Extensions (VHE) (DALL; NIEH, 2014), and BFloat16/matrix multiply instructions for AI inference (Arm Ltd., 2024b).

ARMv9 (2021–present) mandates SVE2 as the baseline vector ISA, introduces the Scalable Matrix Extension (SME/SME2) for matrix-oriented AI workloads (Arm Ltd., 2021b), and establishes the Confidential Compute Architecture (CCA) with hardware-enforced Realm isolation for cloud computing (LI et al., 2022).

This survey provides a comprehensive technical analysis of the ARM architecture across all three phases and all three profiles, with particular attention to the intersection of architecture design and safety-critical systems, a domain of growing importance as ARM processors penetrate automotive, aerospace, robotics, and industrial control applications. The remainder of this chapter is organized as follows: Section 2.2 traces ARM's historical origins; Sections 2.3 and 2.4 detail the architecture profiles and register model; Sections 2.5–2.7 provide in-depth analysis of each major architecture version; Sections 2.8–2.10 cover pipeline evolution, SIMD/vector processing, and virtualization; Sections 2.11–2.12 address safety-critical applications and the Cortex-R82; and Section 2.13 concludes with future directions.

2.2 Historical Origins

The ARM architecture originated as a direct consequence of the limitations Acorn Computers encountered while developing successors to the BBC Micro. In 1981, Acorn had won the contract to build the BBC Microcomputer for the BBC's Computer Literacy Project, a machine built around the 8-bit MOS 6502 processor. By 1983, Acorn engineers required a 32-bit processor for their next-generation platform but found no commercially available option satisfactory: the Motorola 68000 was considered too power-hungry and complex, the Intel 80286 was expensive and heavily CISC-oriented, and the National Semiconductor 32016 suffered from critical silicon defects (FURBER, 2000).

Wilson and Furber drew direct inspiration from the Berkeley RISC-I and Stanford MIPS academic projects (PATTERSON; DITZEL, 1980; PATTERSON, 1985a), which demonstrated that a simple processor with many registers and a load/store architecture could outperform complex CISC designs while using dramatically fewer transistors. The first ARM1 processor taped out in April 1985 at VLSI Technology, Inc. Implemented in 3 μm CMOS, ARM1 contained approximately 25,000 transistors—compared to the Motorola 68000's 68,000—and functioned correctly on first silicon, a remarkable achievement for the era. The chip consumed less than 0.1 watts, a characteristic that would prove prescient as mobile computing emerged a decade later.

The **ARM2** (1986) was deployed in the Acorn Archimedes personal computer and represented the first commercially significant ARM product. It featured a 3-stage pipeline (fetch, decode, execute), a 26-bit address space, 16 32-bit registers, and the barrel shifter, the ability to shift one operand as part of any data-processing instruction at no additional cycle cost, that would remain an ARM hallmark through all subsequent architecture versions.

In 1990, Acorn spun off its processor division as **ARM Ltd.** (Advanced RISC Machines), a joint venture with Apple Computer (seeking a low-power processor for the Newton PDA) and VLSI Technology. This decision established the *licensing business model* that would define ARM's trajectory: rather than manufacturing chips, ARM licensed its processor designs and ISA to semiconductor companies who implemented them in their own silicon. This capital-light model enabled ARM to proliferate across vendors and markets without the cost of fabrication facilities.

The **ARM7TDMI** (1993) was the breakthrough core that achieved mass deployment, with the designation encoding its key features: Thumb (16-bit compressed instruction set), Debug support, enhanced Multiplier, and ICE (In-Circuit Emulator) interface. The ARM7TDMI shipped in billions of units across applications including the Nintendo Game Boy Advance, early iPods, and the first generation of ARM-powered Nokia mobile phones. Its success established ARM as the dominant architecture for battery-powered devices.

Subsequent cores—ARM9TDMI (5-stage pipeline enabling higher clock frequencies), ARM11 (first with SIMD extensions, VFP floating-point, and the initial TrustZone security architecture)—progressively added features. However, the architecture's evolution remained somewhat ad hoc until ARMv7 formalized the three-profile taxonomy that structures the modern ARM ecosystem.

2.3 Architecture Profiles: A, R, and M

ARMv7 introduced the formal division of the ARM architecture into three distinct profiles, each targeting fundamentally different computing domains with different hardware guarantees to software. The profiles differ most visibly in their memory model: a full Memory Management Unit (MMU) with virtual-to-physical address translation in the A-profile, versus

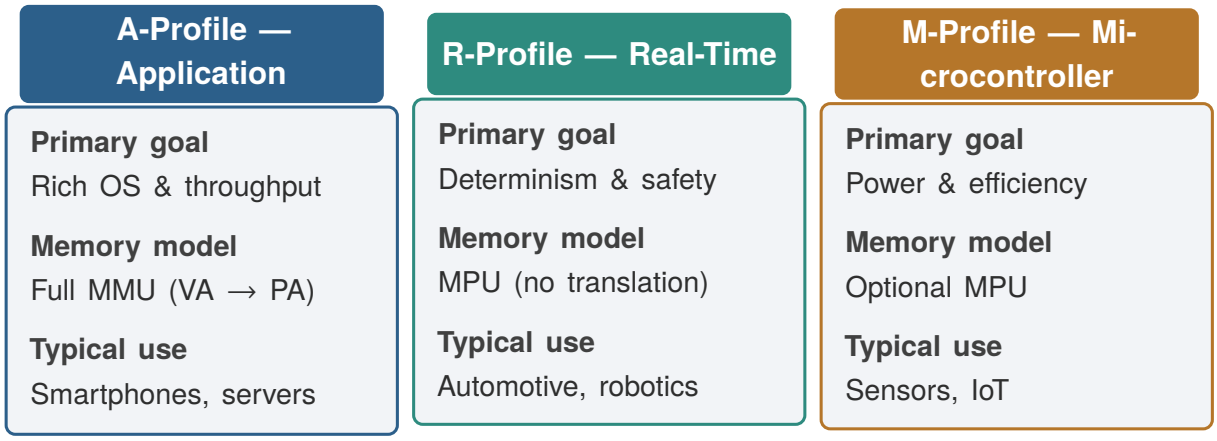


Figure 1 – The three ARM architecture profiles and their defining characteristics.

Source: prepared by the author, based on ARM Ltd. architecture documentation (Arm Ltd., 2018; Arm Ltd., 2023a).

a simpler Memory Protection Unit (MPU) without translation in the R- and M-profiles. This taxonomy persists through ARMv8 and ARMv9. Figure 1 summarises the three profiles at a glance; each is detailed in the following subsections.

2.3.1 A-Profile: Application Processors

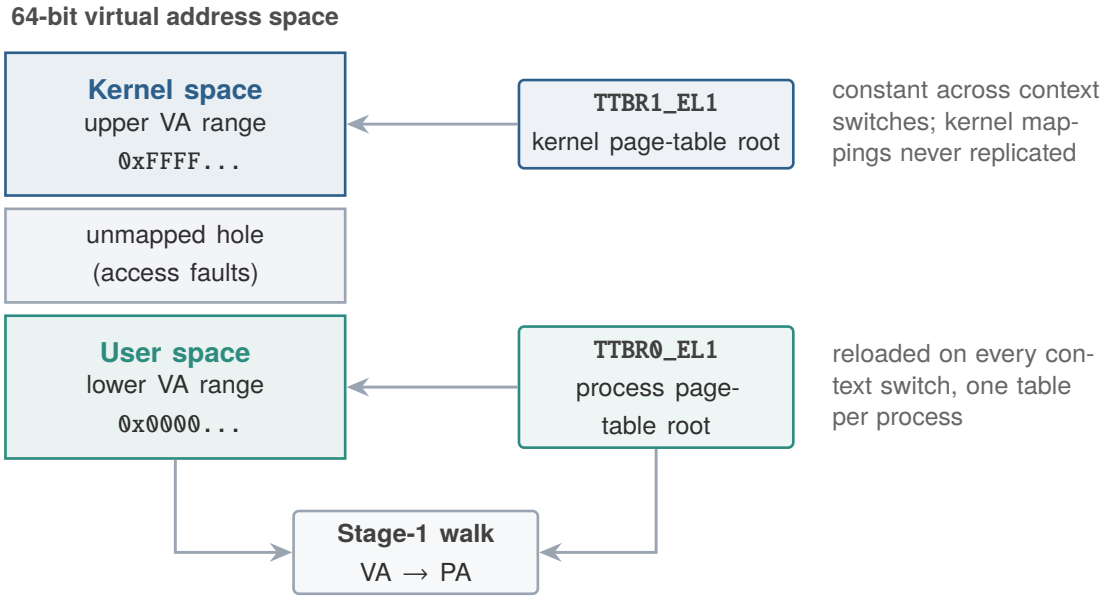
The A-profile is designed to run rich operating systems—Linux, Android, Windows, macOS—where the software stack assumes virtual memory, user/kernel privilege separation, and transparent context switching between many concurrent processes.

The defining feature is a full **MMU** with hardware page table walking. The MMU translates virtual addresses to physical addresses through a multi-level page table structure, supporting 4 KB, 16 KB, and 64 KB page granules with up to 4-level walks. The translation tables carry permission bits (read, write, execute), privilege-level restrictions (EL0/EL1), cacheability and shareability attributes, and Execute-Never (XN, PXN) bits for security (Arm Ltd., 2024b).

The A-profile employs a **two-TTBR design**: `TTBR0_EL1` maps the lower virtual address range (user space) and changes on every context switch, while `TTBR1_EL1` maps the upper range (kernel space) and remains constant. This guarantees that kernel mappings are always accessible without being replicated in each process's page tables; Figure 2 illustrates the scheme.

The A-profile uses a **weakly-ordered memory model** (PULTE et al., 2018), a fundamental difference from x86's Total Store Order (TSO). The hardware may reorder loads and stores relative to each other unless explicit barrier instructions are used: `DMB` (Data Memory Barrier) for ordering, `DSB` (Data Synchronization Barrier) for completion, and `ISB` (Instruction Synchronization Barrier) for pipeline flushing. Acquire/release semantics on load/store operations (`LDAR`/`STLR`) provide a lighter-weight synchronization primitive. This weak ordering gives the microarchitecture greater

Figure 2 – The AArch64 two-TTBR translation scheme: `TTBR0_EL1` covers the lower (user) virtual-address range and is reloaded on every context switch, while `TTBR1_EL1` covers the upper (kernel) range and remains constant.



Source: prepared by the author, based on the ARM Architecture Reference Manual (Arm Ltd., 2023a).

freedom to overlap and reorder memory operations, which is critical for out-of-order cores with deep load/store queues.

A-profile core families include: **Cortex-A** (general-purpose, from the energy-efficient A510/A520 to the high-performance A720/A725), **Cortex-X** (performance-unconstrained cores: X4, X925, with 8-wide decode and large ROBs), and **Neoverse** (server/infrastructure: N-series for cloud density, V-series for HPC, E-series for networking). The **big.LITTLE** and **DynamIQ** heterogeneous computing models pair big and little cores on a cache-coherent interconnect (JEFF, 2013; GREENHALGH, 2011).

2.3.2 R-Profile: Real-Time Processors

The R-profile exists because the A-profile's virtual memory introduces *unpredictable latency*. A translation lookaside buffer (TLB) miss can take 20–100 cycles; page faults can stall for milliseconds. For anti-lock brakes, jet engine controllers, or hard-disk heads, such variability is unacceptable.

The R-profile replaces the MMU with an **MPU** that partitions the address space into a fixed number of regions (typically 16–24), each with configurable permissions, but *without address translation*. Virtual addresses are physical addresses. MPU configuration is stored in system registers, not in memory, eliminating page table walk latency (Arm Ltd., 2018).

Tightly-Coupled Memory (TCM), small SRAMs (32–256 KB) wired directly to the

core, provides guaranteed single-cycle access latency. Unlike cache, TCM content is never evicted, ensuring that interrupt handlers and safety-critical code paths execute with fully deterministic timing (Arm Ltd., 2020c).

Several R-profile cores (Cortex-R5, R52, R52+) support **lockstep execution**, where two identical cores execute the same instruction stream and a hardware comparator checks outputs every cycle. Divergence indicates a hardware fault and raises an immediate error. This provides the diagnostic coverage required by ISO 26262 ASIL D and DO-254 DAL-A (ITURBE et al., 2016; International Organization for Standardization, 2018).

R-profile cores are typically **in-order**, as deep speculation creates variable-latency states difficult to bound for worst-case execution time (WCET) analysis (WILHELM et al., 2008). Key implementations include Cortex-R4, R5, R8, R52, R52+, and the landmark Cortex-R82 (Section 2.12).

2.3.3 M-Profile: Microcontroller Processors

The M-profile is architecturally distinct. It runs **Thumb/Thumb-2 only**—no classic 32-bit ARM instructions. The instruction stream mixes 16-bit and 32-bit encodings detected by the hardware (YIU, 2013; Arm Ltd., 2023b).

The programming model has two execution modes: **Thread mode** (normal code) and **Handler mode** (exceptions), with two privilege levels: privileged and unprivileged. This simplicity allows interrupt handlers to be plain C functions.

The **Nested Vectored Interrupt Controller (NVIC)** is architecturally integrated, managing up to 240 interrupts with configurable priorities and hardware-managed preemption. On exception entry, the processor automatically pushes eight registers (`R0`–`R3`, `R12`, `LR`, `PC`, `xPSR`) in hardware, in parallel with the vector fetch, enabling 12-cycle interrupt entry on Cortex-M3/M4 (YIU, 2013).

Two key optimizations enhance throughput: **tail-chaining** (back-to-back ISRs skip unstack/restack, saving ∼12 cycles each) and **late arrival** (higher-priority interrupts arriving during stacking redirect without wasting the stacking work).

The family spans Cortex-M0+ (∼12K gates, 2-stage pipeline) through M4 (DSP + FPU), M7 (6-stage superscalar, 600+ MHz), M33 (TrustZone-M), M55 (Helium MVE), to M85 (PACBTI + Helium + TrustZone) (Arm Ltd., 2019).

TrustZone for Armv8-M uses hardware SAU/IDAU partitioning rather than firmware-mediated world switching. Secure functions are invoked via the `SG` (Secure Gateway) instruction, with hardware automatic register clearing and return state saving in ∼4 cycles (PINTO; SANTOS, 2019).

2.4 Register Architecture and Instruction Encoding

AArch64's register architecture comprises a general-purpose integer file, a floating-point and vector file shared with NEON and SVE, and a set of banked system registers, all accessed through a uniform fixed-width instruction encoding.

The general-purpose file provides 31 64-bit registers, `X0`–`X30`, each of which can also be accessed as a 32-bit `W` register; writing to a `W` register zeros the upper 32 bits, eliminating partial-register dependencies. The encoding for "register 31" is context-dependent, resolving at decode time to either `XZR`, a hardwired zero, or `SP`, the stack pointer (Arm Ltd., 2024b). By the AAPCS64 calling convention, `X0`–`X7` carry arguments and return values, `X19`–`X28` are callee-saved, `X29` serves as the frame pointer, and `X30` is the link register set by `BL`.

Floating-point, NEON, and SVE share a single bank of 32 128-bit registers (`V0`–`V31`), which can be viewed as `Bn` (8-bit), `Hn` (16-bit), `Sn` (32-bit), `Dn` (64-bit), or `Qn` (128-bit) depending on the operation; SVE extends these to the scalable `Z0`–`Z31` registers (128–2048 bits) together with 16 predicate registers `P0`–`P15` (STEPHENS et al., 2017a). Each exception level additionally has its own banked system registers—`SP_ELn`, the exception-return address `ELR_ELn`, and the saved processor state `SPSR_ELn`—so that on exception entry the hardware automatically saves PSTATE into the target level's SPSR and the return address into ELR. The individual `PSTATE` fields, including the NZCV (Negative, Zero, Carry, oVerflow) condition flags, the current exception level, stack-pointer selection, and the DAIF interrupt masks, remain individually accessible.

Instruction Encoding

AArch64 uses uniform 32-bit encoding. Key classes include:

Data processing: Arithmetic, logical, shift, and bitfield. The barrel shifter integrates into the ALU path—e.g., `ADD X0, X1, X2, LSL #3` computes $X1 + (X2 \ll 3)$ in one operation.

Load/store: Base+offset, base+register (with shift), pre-index, post-index. `LDP`/`STP` (load/store pair) transfer two 64-bit registers, critical for prologues/epilogues.

Branch: `B`, `B.cond`, `BL` (call), `BR`/`BLR`/`RET` (register-indirect), `CBZ`/`CBNZ` (compare-and-branch).

System: `MSR`/`MRS`, `SVC`/`HVC`/`SMC`, barriers (`DMB`, `DSB`, `ISB`), cache maintenance.

Comparison: x86-64 provides 16 general-purpose registers (GPRs) (32 with APX), variable-length encoding (1–15 bytes), monolithic RFLAGS. RISC-V provides 31 GPRs, hardwired zero (`x0`), no condition flags. AArch64 balances register richness with encoding regularity (BLEM et al., 2015; WEAVER et al., 2023).

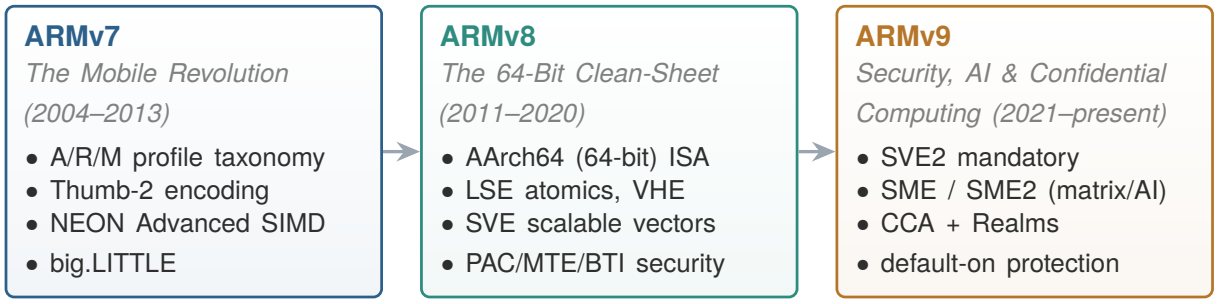


Figure 3 – Evolution of the ARM architecture across the ARMv7, ARMv8, and ARMv9 generations.

Source: prepared by the author, based on the ARM version history surveyed in this chapter (Arm Ltd., 2018; Arm Ltd., 2023a).

2.5 ARMv7: The Mobile Revolution (2004–2013)

Figure 3 traces the ARM architecture across the three generations examined in this and the following two sections, from the profile-based ARMv7 to the security- and AI-focused ARMv9.

The Cortex-A8 (2005) was the first ARMv7-A implementation—a dual-issue, in-order core with a 13-stage pipeline running at 1 GHz on a 65 nm process—and it powered a generation of early mobile devices, from TI OMAP3 (early Android, BeagleBoard) and Samsung Exynos to the Apple A4 in the iPad 1 and iPhone 4 (Arm Ltd., 2018).

At the instruction-set level, ARMv7 introduced two innovations that defined the mobile era. Thumb-2 unified the 16-bit Thumb encoding (present since ARMv4T) with new 32-bit instructions, creating a variable-length ISA that combined Thumb-class code density with ARM-state performance; because the compiler could mix the two encodings instruction by instruction, and because denser code used instruction caches more effectively, the improved density often *improved* performance rather than merely saving space. NEON Advanced SIMD, in turn, added 128-bit registers (`Q0`–`Q15`) operating on vectors ranging from 16×8-bit to 2×64-bit elements, though it remained *optional* in ARMv7-A and therefore required runtime feature detection, a persistent portability concern.

On the system side, the Cortex-A15 added hardware virtualization through HYP mode (PL2) with two-stage address translation, enabling KVM/ARM (DALL; NIEH, 2014), while the big.LITTLE model (JEFF, 2013; GREENHALGH, 2011) paired a wide out-of-order A15 with a small in-order A7 on a cache-coherent interconnect for workload-adaptive power management.

By 2012, however, the architecture's limits had become apparent: a 32-bit virtual-address ceiling of roughly 3 GB per process, only 13 usable general-purpose registers, a 4-bit condition field carried on every ARM-state instruction yet used in fewer than 5% of Thumb-2 code, and the lingering complexity of ARM/Thumb interworking. Together these motivated the clean-sheet move to AArch64 (BLEM; MENON; SANKARALINGAM, 2013).

2.6 ARMv8: The 64-Bit Clean-Sheet Redesign (2011–2020)

ARMv8-A, announced October 2011, was the most significant change in ARM's history. AArch64 is not an extension of ARMv7; it is a clean-sheet design.

2.6.1 AArch64 Design Decisions

AArch64 was a clean-sheet design that systematically resolved the weaknesses accumulated in 32-bit ARM. The key decisions, each addressing a specific limitation discussed above, were the following:

31 GPRs with hardwired zero (`XZR`). The larger file reduces stack spill/fill, improving cache behavior and power.

No implicit condition flag updates. NZCV flags are set only by the dedicated comparison instructions `CMP`/`TST` or by S-variants (`ADDS`), eliminating false dependencies in out-of-order (OoO) pipelines.

No per-instruction predication. Replaced by `CSEL`, `CCMP`, conditional increment/negate, a compact set covering useful cases without the 4-bit encoding cost.

Fixed 32-bit encoding with no ARM/Thumb mode. Decoder hardware is dramatically simpler.

PC removed from GPR file. Readable via `ADR`/`ADRP` only; modifiable only by branches. Eliminates pipeline-depth-dependent behavior and PC-relative attacks.

2.6.2 Sub-Version Feature Cascade (v8.0–v8.7)

Rather than arriving as a single revision, the capabilities that distinguish ARMv8 accumulated across roughly a decade of point releases. Each designation v8.*x* below denotes an incremental revision of the *architecture specification*—a set of optional features that silicon vendors may implement—and not a new hardware block or processor generation. The cascade below is best read as a timeline of that accretion rather than as a list to be memorized: tracing it from v8.0 to v8.7, two trajectories stand out—a steady hardening of security (Pointer Authentication in v8.3, Branch Target Identification and Memory Tagging in v8.5, faulting PAC in v8.7) and a growing orientation toward machine learning (FP16 in v8.2, BFloat16 and integer matrix multiply in v8.6). It is precisely these two threads that ARMv9 would later consolidate into mandatory, default-on features.

v8.0 (2011): Base AArch64, EL0–EL3 exception model, mandatory NEON/FP, optional crypto. First cores: Cortex-A53 (in-order, most-shipped 64-bit ARM) and A57 (OoO). Apple A7 (September 2013) was first shipping silicon.

v8.1 (2014): *Virtualization Host Extensions (VHE)*—host kernel at EL2 directly, eliminat-

ing EL1↔EL2 transitions (DALL et al., 2016). *LSE atomics*—`CAS`, `SWP`, `LDADD/LDCLR/LDSET`, replacing retry-based `LDXR/STXR` loops. Essential for databases and lock-free structures on many-core servers. *PAN*—prevents kernel from accidentally accessing user memory.

v8.2 (2016): *SVE* (STEPHENS et al., 2017a)—scalable 128–2048 bit vectors, predicate registers, VLA programming. *FP16* processing. *RAS* (Reliability, Availability, Serviceability) error reporting. *SPE* (Statistical Profiling). *LVA/LPA*—52-bit virtual and physical addresses.

v8.3 (2016): *PAC* (Pointer Authentication) (LILJESTRAND et al., 2019)—cryptographic pointer signing via `PACIA/AUTIA`. Forged return addresses cause faults. Hardware defense against ROP/JOP. Deployed in Apple A12+. *Nested Virtualization*. *FJCVTZS*—JavaScript FP-to-int conversion.

v8.4 (2017): *S-EL2*—secure world virtualization for Secure Partition Managers. *DIT* (Data Independent Timing)—constant-time execution for crypto.

v8.5 (2018): *BTI* (Branch Target Identification)—`BTI` instructions mark valid indirect branch targets; non-BTI landings fault. JOP defense. *MTE* (Memory Tagging) (SEREBRYANY et al., 2018)—4-bit tags on pointers and 16-byte granules; mismatches signal faults. Hardware use-after-free/overflow detection. Deployed on Pixel devices. *RNG*—`RNDR/RNDRRS` hardware entropy.

v8.6 (2019): *BFloat16* and *Int8 matrix multiply* (`BFMMLA`, `SMMLA`, `UMMLA`)—targeting neural network inference. *ECV* (Enhanced Counter Virtualization). *FGT* (Fine-Grained Traps) for hypervisor efficiency.

v8.7 (2020): *WFET* (Wait-For-Event with Timeout). *FPAC* (Faulting PAC)—immediate trap on authentication failure. PMU snapshot. Features lead into ARMv9.

2.6.3 Key Implementations and Datacenter Expansion

The **Neoverse N1** (2018) was ARM's first purpose-built server core, deployed in AWS Graviton2 and Ampere Altra. It demonstrated competitiveness with Intel Xeon on cloud workloads, not only on power efficiency but on absolute performance for web servers, databases, and in-memory caches. LSE atomics were critical: without single-instruction atomic operations, lock-free data structures suffered severe contention penalties at high core counts (DALL et al., 2016).

Apple's **M1** (2020, based on ARMv8.5) demonstrated that ARM could match or exceed x86 desktop/laptop performance while consuming dramatically less power, disrupting the assumption that ARM was only suitable for mobile and embedded applications.

2.7 ARMv9: Security, AI, and Confidential Computing (2021–Present)

ARMv9 forked from ARMv8 rather than replacing it: each v9.x release shares a common base with v8.(x+5)—v9.0 corresponds to v8.5 plus SVE2 and RME, and v9.2 to v8.7 plus SME—while AArch32 becomes optional, allowing implementations to be AArch64-only at every exception level (Apple having already dropped AArch32, a choice ARMv9 formalizes as an architectural option). The architecture is organized around three pillars: default-on security, vector and matrix compute for AI, and confidential computing.

The security pillar elevates protection from optional extensions to baseline: PAC, BTI, and MTE become mandatory or strongly encouraged, and the Guarded Control Stack (GCS, v9.4) adds a hardware shadow call stack in which every `BL` pushes the return address to both the regular stack and a protected shadow stack, faulting on any mismatch at `RET`. GCS is stronger than PAC precisely because it requires no cryptographic secrets and therefore exposes no key-leakage attack vector (KOCHER et al., 2019), and architectural Spectre/Meltdown mitigations are built in.

The AI and compute pillar makes SVE2 mandatory, guaranteeing that every v9 processor supports vector-length-agnostic code; as a full NEON superset adding saturating arithmetic, bitwise permutation, polynomial multiplication, and complex multiply-accumulate (STEPHENS et al., 2017a; RICO; RAMIREZ; VALERO, 2017), it effectively deprecates NEON. SME (v9.2) (Arm Ltd., 2021b) then introduces a fundamentally new computational model built on a streaming SVE mode and dedicated ZA tile storage (SVL×SVL bits): its key operation is the outer product—a column vector times a row vector, accumulated into ZA—which is the fundamental GEMM primitive, with ZA holding 32 KB of matrix data at 512-bit SVL. SME2 (v9.4) extends this with multi-vector operations over groups of two or four ZA rows and range-indexed lookup tables for activation functions such as GELU and SiLU, targeting transformer inference, and FP8 arrives in v9.5.

The confidential-computing pillar introduces the Confidential Compute Architecture (CCA) (LI et al., 2022), which adds a fourth security state, the Realm, alongside Secure, Non-Secure, and Root to address the cloud trust problem in which hypervisors otherwise hold full access to guest memory. A Granule Protection Table (GPT) tags every physical page with its world assignment, and on every memory access the hardware checks the GPT and blocks cross-world accesses, with only Root (EL3) firmware able to modify it. The hypervisor still manages a realm's lifecycle—creation, scheduling, and destruction—but *cannot read or write realm memory*; instead, the Realm Management Monitor (RMM) at R-EL2 handles encryption keys, attestation, and realm stage-2 translation, and its small codebase enables formal verification. The result provides isolation comparable to Intel TDX and AMD SEV-SNP, but with per-page GPT enforcement and a distinct trust boundary (LI et al., 2022; LI et al., 2021).

2.8 Pipeline Evolution and Microarchitectural Trends

ARM defines an ISA, not a pipeline. Different microarchitectures implement the same ISA with radically different designs, and the progression reflects ARM's expanding ambitions.

The **ARM7TDMI** (1993): 3-stage pipeline, single-issue, in-order, no branch prediction. The **Cortex-A8** (2005): 13-stage dual-issue in-order, 1 GHz. The **Cortex-A9** (2008): first ARM OoO core, dual-issue with register renaming and speculative execution (BLEM; MENON; SANKARALINGAM, 2013).

Modern big cores represent a qualitative leap. The **Cortex-X4** (2023) and Apple Firestorm/Avalanche feature 6–8-wide decode, reorder buffers exceeding 300 entries, 10+ execution ports, and TAGE-based branch predictors. Register renaming maps 31 architectural GPRs onto 200+ physical registers. Retirement occurs in program order (BLEM et al., 2015).

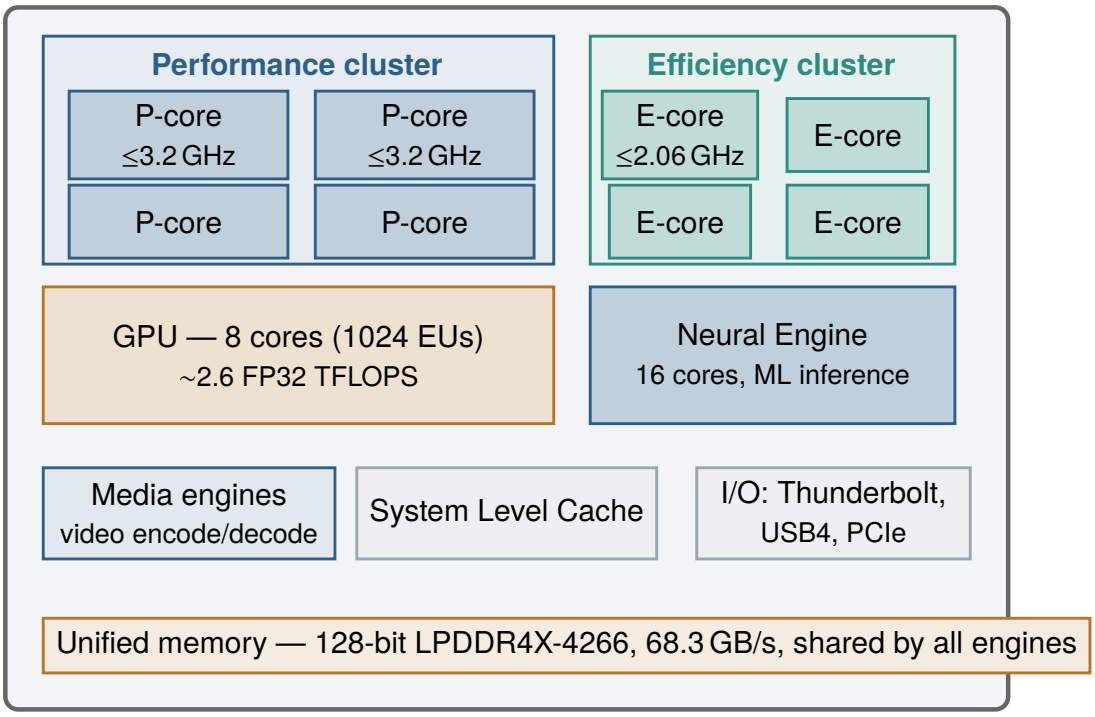
Little cores (Cortex-A510, A520) are in-order or narrow OoO for efficiency. The **DynamIQ** cluster model allows asymmetric configurations (e.g., 1×X4 + 3×A720 + 4×A520) with hardware-assisted migration and shared L3. The most prominent commercial expression of this heterogeneous philosophy is Apple's M1 (2020), the first ARM-based SoC to displace x86 across an entire personal-computer product line (FRUMUSANU, 2020). As Figure 4 illustrates, the M1 integrates two asymmetric CPU clusters (four performance and four efficiency cores), an eight-core GPU, a sixteen-core Neural Engine for on-die ML inference, and dedicated media engines around a unified 128-bit LPDDR4X memory system shared by all engines. The design shows where the A-profile ecosystem concentrates its advantage: not in the ISA alone, but in dense heterogeneous integration — the same pattern examined for RISC-V in Figure 5, at a far higher integration level than any RISC-V part shipping today.

Server cores (**Neoverse V2**) push throughput: moderate clocks, very wide issue, large caches, high memory bandwidth (KODAMA et al., 2020).

2.9 SIMD and Vector Processing Evolution

ARM's vector processing has evolved through four generations on the application-class profile, complemented by a dedicated extension for microcontrollers. The first generation is NEON (Advanced SIMD), which uses a fixed set of 128-bit registers with lane-based processing; software must handle loop tails explicitly, and the extension is optional in ARMv7 but mandatory in ARMv8. The second generation, SVE (Scalable Vectors) (STEPHENS et al., 2017a), decoupled software from hardware width: its `Z0`–`Z31` registers range from 128 to 2048 bits, and the Vector-Length-Agnostic (VLA) model lets the same code run on any width without recompilation, while predicate registers `P0`–`P15` provide per-lane masking and the `WHILELT` instruction naturally handles partial vectors; Fujitsu's A64FX, for instance, implements 512-bit SVE for the Fugaku supercomputer (KODAMA et al., 2020; SATO et al., 2020). The third generation, SVE2,

Figure 4 – Block-level organization of the Apple M1 (2020), the reference example of heterogeneous integration in a consumer ARM SoC.



Source: prepared by the author, based on die annotations by Locuza (Locuza, 2021) and AnandTech's microarchitectural analysis (FRUMUSANU, 2020).

mandatory in ARMv9 as a NEON superset, adds saturating arithmetic, bitwise permutation, polynomial multiplication over GF(2), complex multiply-accumulate, and narrowing/widening operations (RICO; RAMIREZ; VALERO, 2017), relegating NEON to a legacy compatibility layer. The fourth generation brings matrix processing: SME contributes ZA tile storage and outer-product accumulation, while SME2 adds multi-vector operations and activation lookup tables (Arm Ltd., 2021b), together providing a dedicated GEMM path for neural-network inference, with FP8 support arriving in v9.5.

Distinct from this application-class lineage, Helium, the M-Profile Vector Extension (MVE) (Arm Ltd., 2019), targets the tighter constraints of microcontrollers. Rather than adding dedicated vector registers, it reuses the floating-point register file (eight 128-bit `Q` registers) and employs beat-based execution, in which each 128-bit operation decomposes into four 32-bit "beats" spread over four cycles that interleave with scalar instructions; this adds minimal hardware while still delivering vectorized throughput for ML inference (int8/int16 dot products), DSP workloads (FIR/IIR filtering, FFT), and image processing.

Table 1 situates this SIMD lineage alongside competing vector architectures.

Table 1 – SIMD/Vector Architecture Comparison

Extension	Width	VLA	Notes
NEON	128b	No	Optional in v7
SVE/SVE2	128–2048b	Yes	Mandatory v9
SME/SME2	Tile	Yes	Outer products
Helium	128b	No	Beat-based, M
x86 SSE	128b	No	16 XMM regs
x86 AVX-512	512b	No	32 ZMM, masks
RISC-V RVV	128–65Kb	Yes	LMUL grouping

2.10 Virtualization Architecture

ARM's virtualization uses two-stage address translation. Guest VMs at EL1 translate VA to *intermediate physical addresses (IPA)*. EL2 installs stage-2 tables (via `VTTBR_EL2`) translating IPA to true PA. This double translation is hardware-performed with no per-access software overhead (DALL; NIEH, 2014; DALL et al., 2016).

`HCR_EL2` provides fine-grained trap bits: WFI, SMC, cache maintenance, TLB invalidation, debug access. Fewer traps yield better performance; KVM/ARM minimizes trapping (DALL; NIEH, 2014).

VHE (v8.1) optimizes Type-2 hypervisors by running the host kernel at EL2, with hardware remapping EL1 register accesses to EL2 equivalents. This eliminates EL1↔EL2 transitions for host operations (DALL et al., 2016).

GICv3/v4 (Arm Ltd., 2021a) supports direct virtual interrupt injection to guest VMs without hypervisor trapping. GICv4.1 adds virtual SGI direct injection.

The **SMMU** (System MMU) (Arm Ltd., 2020a) provides address translation for I/O devices (ARM's IOMMU), enabling near-native direct memory access (DMA) performance for device passthrough.

S-EL2 (v8.4) brings virtualization to the Secure world, enabling Secure Partition Managers for multiple isolated trusted applications.

2.11 Safety-Critical Systems: PREEMPT_RT, ELISA, and WCET

ARM's expansion into automotive, aerospace, robotics, and industrial control demands safety certification under ISO 26262 (International Organization for Standardization, 2018), IEC 61508 (International Electrotechnical Commission, 2010), and DO-178C.

On the software side, PREEMPT_RT (the Linux kernel's real-time preemption patch set) transforms Linux into a real-time system through fully preemptible kernel code, sleeping spinlocks (rt_mutex), threaded IRQ handlers, and priority inheritance (REGHENZANI; MASSARI; FORNACIARI, 2019a; OLIVEIRA et al., 2020). A-profile hardware, however, introduces irreducible latency—TLB-miss jitter (20–100 cycles), out-of-order retirement variability, cache-eviction spikes from co-running tasks, and GIC routing overhead—so that worst-case IRQ latency reaches roughly 50–150 μs on Cortex-A, against only 1–5 μs on an R-profile core with TCM.

This hardware dependency is most visible in WCET analysis, which provides the static upper bounds required for IEC 61508 SIL-3+ and ISO 26262 ASIL C/D (WILHELM et al., 2008). On A-profile out-of-order cores with caches and speculation, WCET bounds are 10–50× pessimistic, rendering them useless for system design (WILHELM et al., 2008; DAVIS; BURNS, 2011); on an R-profile core in MPU mode with TCM—an in-order pipeline with fixed stages, single-cycle TCM access, no page walks, and bounded misprediction penalty, the bounds are tight and credible, and therefore suitable as certification evidence under IEC 61508 Annex D (WILHELM et al., 2008; GRACIOLI et al., 2015).

Certification, rather than timing, is the concern of ELISA (BRASH; SHERWOOD, 2020), which builds safety-case arguments identifying which subset of Linux participates in the safety function—full MC/DC coverage of 30M+ lines of code being infeasible. Three hardware challenges constrain ELISA on A-profile: temporal isolation, since shared caches create inter-core interference that violates the freedom-from-interference requirements of ISO 26262 Part 6 Clause 7.4.12 (International Organization for Standardization, 2018; GRACIOLI et al., 2015); fault detection, since the absence of lockstep leaves only expensive and incomplete software techniques (ITURBE et al., 2016); and WCET evidence, whose pessimistic bounds lack credibility.

Beyond a single criticality level, mixed-criticality systems (BURNS; DAVIS, 2017; ZÜPKE; KAISER, 2023) require hardware partitioning for independent fault containment, while memory-safe languages, particularly Rust (JUNG et al., 2021a), strengthen safety cases by eliminating buffer overflows, use-after-free, and data races at compile time. Under IEC 61508 Table A.3, a "language subset" that prevents dangerous constructs is recommended for SIL-2+ (International Electrotechnical Commission, 2010), a role Rust's ownership model and borrow checker fill as a mechanically verified subset; the Tock OS (LEVY et al., 2017) demonstrates the approach on ARM Cortex-M for safety-critical embedded systems.

2.12 The Cortex-R82: Bridging Real-Time and Application Domains

The Cortex-R82 (Arm Ltd., 2020c) is architecturally unique: the first R-profile core with AArch64 and optional MMU. It resolves the fundamental tension between A-profile richness and R-profile determinism.

Architecturally, the R82 shares the A-profile ISA, the AAPCS64 calling convention, and the ELF format, so that code compiled for a Cortex-A53 runs unmodified, eliminating the "two-toolchain" problem of traditional heterogeneous SoCs. Its MMU is configurable: enabled for Linux (full stage-1 translation with hardware page walks) or disabled for bare-metal real-time use (MPU-only, flat physical addressing, TCM), selectable at boot or at runtime. Multi-core R82 systems can therefore be partitioned, running MMU-on cores under Linux alongside MPU-only cores under an RTOS or bare metal, communicating through shared memory under the same ARM memory model on both sides without protocol translation or message-passing serialization.

This configurability enables ASIL decomposition. A safety partition (R82 in MPU mode, lockstep, TCM) runs at ASIL D with small, WCET-bounded, fault-detected code whose in-order pipeline, deterministic TCM, and absence of page walks yield tight WCET bounds (WILHELM et al., 2008), while a non-safety partition (R82 with the MMU enabled, running Linux) runs at ASIL B or QM under the ELISA methodology of targeted testing and architectural argumentation, with Rust components strengthening the case (JUNG et al., 2021a; BRASH; SHERWOOD, 2020). Crucially, because TCM is physically separate from cached memory, no eviction, TLB interference, or bandwidth contention can affect the real-time path, a freedom from interference that is trivially provable and satisfies ISO 26262's most demanding requirement (International Organization for Standardization, 2018).

These properties suit several domains: in automotive, safety-critical actuator control runs on lockstep cores while infotainment and ADAS run under Linux on MMU cores; in 5G baseband, hard-real-time signal processing runs on MPU cores with the protocol stack on Linux cores; and in robotics, 1 kHz+ servo loops (proportional–integral–derivative (PID) control, motor commutation) execute from TCM on a bare-metal R82 while trajectory planning, ROS2, and ML inference run on a Linux R82, the two communicating through shared memory with minimal IPC (PALOSSI et al., 2019a; CAMPOS et al., 2021).

2.13 Final Remarks and Future Trends

Table 2 summarizes key architectural differences between profiles.

Modern SoCs combine profiles: NXP S32G (A53 + M7 + lockstep R52), TI TDA4VM (A72 + R5F), Renesas R-Car. The R82's dual-mode capability represents the latest convergence step.

Table 2 – ARM Architecture Profile Comparison

Feature	**A-Profile**	**R-Profile**	**M-Profile**
Memory management	Full MMU, HW page walks	MPU (no translation)	Optional MPU
Address translation	VA → IPA → PA	Flat physical	Flat physical
Privilege model	EL0–EL3	PL0–PL1 / EL0–EL2 (R82)	Thread + Handler
Interrupt controller	GIC (external)	GIC (external)	NVIC (integrated)
Exception entry	Software context save	Software context save	HW auto-stack (12 cy)
Pipeline	Deep OoO (6–8 wide)	In-order / narrow OoO	2–6 stage in-order
Deterministic timing	No	Yes (TCM, lockstep)	Yes (NVIC, fixed)
Lockstep support	No	Yes (R5, R52, R52+)	No
ISA state	AArch64 (+AArch32 opt.)	AArch32 or AArch64 (R82)	Thumb/Thumb-2 only
TrustZone	EL3 monitor firmware	Optional	SAU/IDAU hardware
Virtualization	Stage-2 MMU (EL2)	Optional MMU (R82)	No
Vector/SIMD	NEON, SVE, SVE2, SME	Optional NEON/FP (R82)	Helium (MVE)
Typical gate count	Millions	Hundreds of thousands	12K–300K

The ARM architecture has undergone a remarkable evolution from the 25,000-transistor ARM1 of 1985 to the ARMv9 ecosystem of 2025. Three generations define the modern landscape:

ARMv7 established the profile-based taxonomy and won mobile through Thumb-2 density, NEON acceleration, and big.LITTLE. Its 32-bit limitations motivated the most significant ISA redesign in ARM's history.

ARMv8 introduced AArch64 and through incremental sub-versions added datacenter viability (LSE atomics, VHE, SVE, RAS), security (PAC, MTE, BTI), and AI inference (BFloat16, matrix multiply). By the end of the v8 era, ARM was in every smartphone, most tablets, a growing share of laptops (Apple Silicon), and a significant fraction of cloud servers.

ARMv9 mandates security defaults, provides scalable AI compute (SVE2, SME/SME2), and establishes CCA for confidential cloud computing. Each generation roughly doubled architectural ambition: v7 targeted mobile, v8 added servers, v9 addresses the entire computing stack.

For safety-critical applications, the Cortex-R82 enables unified real-time and application computing on a single AArch64 ISA with hardware-enforced mixed-criticality partitioning. Combined with ELISA methodology and memory-safe languages like Rust, the R82 provides a credible path to certifiable Linux-based safety systems.

Future directions include: continued SME evolution for larger matrix tiles and higher-

precision AI; post-quantum cryptographic extensions; further A/R profile convergence; deeper hardware support for formal verification and runtime monitoring; and integration of domain-specific accelerators within the ARM coherent memory fabric.

3

The RISC-V Architecture

3.1 Introduction

The proliferation of intelligent robotic systems, from autonomous mobile robots and collaborative manipulators to unmanned aerial vehicles, has created demanding requirements for processor architectures that must simultaneously support real-time control loops, AI-driven perception pipelines, and general-purpose middleware execution. These heterogeneous workloads have traditionally been served by ARM-based systems-on-chip (SoCs), which dominate the embedded Linux ecosystem with over 95% smartphone market share and an extensive real-time operating system (RTOS) ecosystem through the Cortex-R profile (Arm Ltd., 2023a).

RISC-V, an open-standard instruction set architecture (ISA) originating from the University of California, Berkeley (WATERMAN et al., 2014), has emerged as a compelling alternative. Unlike ARM's proprietary licensing model—where architecture licenses can exceed $10M with additional per-chip royalties (TS2 Tech, 2025)—RISC-V is freely available under permissive open-source licenses. This openness has attracted over 100 member organizations to RISC-V International and enabled an estimated 10+ billion RISC-V cores shipped by 2024 (SiFive, 2023).

This survey provides a comprehensive analysis of the RISC-V ISA with specific focus on aspects relevant to real-time Linux and robotics applications. We examine the architecture from foundational design principles through commercial silicon implementations, drawing explicit parallels to ARM (ARMv8-A/ARMv9-A) and x86_64 throughout. This chapter serves as a companion to the ARM architecture survey in Chapter 2 and is structured to enable direct cross-architectural comparison.

The remainder of this chapter is organized as follows. Section 3.2 covers the historical origins and design philosophy. Section 3.3 describes the modular extension system. Section 3.4 analyzes the register architecture. Section 3.5 details the privilege mode hierarchy. Section 3.6 examines Linux requirements. Section 3.7 surveys pipeline implementations. Section 3.8 provides

a comprehensive vector/SIMD comparison including an analysis of why CPU vector extensions fundamentally differ from GPU matrix throughput. Section 3.9 covers the H-extension for virtualization. Section 3.10 addresses safety-critical systems. Section 3.11 discusses future directions, and Section 3.12 concludes.

3.2 Origins and Design Philosophy

The RISC-V ISA was conceived in 2010 at the University of California, Berkeley, by David Patterson, Krste Asanović, and graduate students Yunsup Lee and Andrew Waterman (WATERMAN et al., 2014; PATTERSON; HENNESSY, 2017). Patterson, co-author (with John L. Hennessy) of the foundational textbook *Computer Organization and Design* (PATTERSON; HENNESSY, 2017) and a pioneer of the original Berkeley RISC project in the early 1980s (PATTERSON, 1985b), was motivated by the absence of a clean, modern, freely available ISA for research and education. The name "RISC-V" denotes the fifth generation of Berkeley RISC designs, following RISC-I, RISC-II, SOAR, and SPUR, with the Roman numeral V also evoking "variation" and "vector" (WATERMAN et al., 2014). The initial design, first documented in a technical report (WATERMAN et al., 2011), was subsequently refined into the RISC-V ISA specification, now maintained by RISC-V International, a non-profit foundation with over 4,000 member organizations across more than 70 countries (RISC-V International, 2026a).

Three foundational principles distinguish RISC-V from ARM and x86. The first is openness as a fundamental property: the ISA specification is freely available and reference implementations such as the Rocket core (ASANOVIĆ et al., 2016) are released under permissive BSD licenses, so any entity may implement a RISC-V processor without licensing fees or royalties—in contrast to ARM's tiered licensing model and the effective Intel–AMD duopoly around x86 (TS2 Tech, 2025). The second is stability over backward compatibility: once ratified, base ISA instructions are frozen and never modified, with new functionality added exclusively through extensions, so that code compiled for the base ISA remains valid indefinitely—whereas AArch64 retains an AArch32 compatibility mode and x86_64 carries decades of backward-compatible encodings (PATTERSON; HENNESSY, 2017). The third is modularity over monolithism: rather than ARM's curated profiles (Cortex-A, -R, and -M with prescribed feature sets), RISC-V provides a minimal base ISA and lets implementers select precisely the extensions they need, scaling from a tiny 12,000-gate embedded core to a datacenter-class server processor (WATERMAN; ASANOVIĆ, 2019).

Governance of this process rests with RISC-V International, which oversees specification development through technical committees and task groups; ratification involves public review, prototype implementations, and software-ecosystem validation before an extension is frozen (RISC-V International, 2026a). Key ecosystem milestones include Linux kernel support upstream since v4.15 (2018), GCC and LLVM compiler backends reaching production quality,

and the October 2024 ratification of the RVA23 profile, which defines the baseline for server-class implementations (RISC-V International, 2024g; SiFive, 2025a).

3.3 Modular Extension System

3.3.1 Base Integer ISAs

RISC-V defines four base integer ISAs that establish the fundamental instruction set:

- **RV32I**: 32-bit base with 32 general-purpose registers, 47 instructions
- **RV64I**: 64-bit extension of RV32I with additional word-width operations
- **RV32E**: Embedded variant of RV32I with only 16 registers (x0–x15)
- **RV128I**: 128-bit variant (largely theoretical, not yet ratified)

The base ISAs are deliberately minimal, providing only integer arithmetic, logical operations, loads/stores, branches, and system calls. All additional functionality—including multiplication, floating point, atomics, and vector processing—is added through extensions (WATERMAN; ASANOVIĆ, 2019).

3.3.2 Standard Extensions

Table 3 summarizes the principal standard extensions. The naming convention concatenates extension letters: "RV64GC" denotes RV64I + M + A + F + D + C, where "G" (General) is an abbreviation for the IMAFD bundle. This is the baseline configuration for Linux-capable processors (WATERMAN; ASANOVIĆ, 2019).

3.3.3 Comparison to ARM Profiles

ARM's three profiles—Cortex-A (applications), Cortex-R (real-time), Cortex-M (microcontroller)—each define a curated set of features, privilege modes, and optional extensions. A Cortex-M core necessarily implements the Thumb-2 instruction set, a simplified exception model, and an optional MPU; a Cortex-A core necessarily implements AArch64, an MMU, and NEON SIMD (Arm Ltd., 2023a).

RISC-V's approach is orthogonal: there is no "real-time profile" or "application profile" imposed by the ISA. Instead, the RVA23 profile specification (RISC-V International, 2024g) defines mandatory and optional extensions for application-class processors (analogous to Cortex-A requirements), while embedded implementations freely select a minimal subset such as RV32IMC (analogous to Cortex-M class). The reserved custom opcode space (four major opcodes reserved for non-standard extensions (WATERMAN; ASANOVIĆ, 2019)) enables vendors to add proprietary accelerator instructions without conflicting with standard extensions, a capability ARM does not offer in comparable form.

Table 3 – Principal RISC-V Standard Extensions

Ext.	Name	Description
M	Multiply/Divide	Integer mul, div, rem
A	Atomics	LR/SC, AMO instructions for SMP
F	Single-precision FP	32-bit float, adds f0–f31 regs
D	Double-precision FP	64-bit float, widens FP regs
C	Compressed	16-bit encodings for common ops
V	Vector	RVV 1.0, scalable vector processing
H	Hypervisor	Hardware virtualization support
B	Bit manipulation	Count, rotate, extract, deposit
Zicsr	CSR access	Control/Status Register instructions
Zifencei	Instruction fence	I-cache/D-cache coherence fence
Zk*	Cryptography	AES, SHA, SM3/SM4 acceleration
Zba/Zbb/Zbs	Bitmanip subsets	Address generation, basic, single-bit

3.3.4 Profile Evolution: RVA20, RVA22, and RVA23

While RISC-V's modularity enables maximum flexibility, it also creates a fragmentation risk, a chip implementing RV64GC with unusual cache behavior or missing misaligned access support may technically boot Linux but fail to run portable userspace software reliably. To address this, RISC-V International introduced *profiles*: curated bundles of ratified extensions and behavioral guarantees that define what a given class of processor must implement for ecosystem compatibility (RISC-V International, 2023c; FPROX, 2023).

Three generations of application processor profiles have been ratified, each building on the previous:

RVA20 (ratified 2023, targeting ~2020-era designs) established the initial baseline: RV64I + M + A + F + D + C + Zicsr + Zifencei—essentially the "G" bundle plus CSR and fence instructions. Almost everything else, including hardware performance counters and bit manipulation, was optional (RISC-V International, 2023c).

RVA22 (ratified 2023, targeting ~2022-era designs) significantly expanded the mandatory set. Beyond the RVA20 extensions, RVA22 mandates bit-manipulation instructions (Zba, Zbb, Zbs—collectively the B extension), hardware performance counters (Zihpm), cache-block management operations (Zicbom, Zicboz, Zicbop), the pause hint (Zihintpause), and critically, a set of memory system behavioral guarantees: 64-byte cache blocks (Zic64b), misaligned load/store support (Zicclsm), atomic instruction guarantees on cacheable memory (Ziccamoa), and LR/SC forward progress (Ziccrse) (RISC-V International, 2023c; RISC-V International, 2024b). The vector extension (V) and hypervisor extension (H) remain optional. RVA22 is

the current compliance target for the RISC-V OS-A Platform Specification, which defines requirements for Linux server platforms (RISC-V International, 2023b).

RVA23 (ratified October 2024) represents the current design target for competitive application processors. The most significant change is that the **vector extension (V) becomes mandatory**—it was optional in RVA22 (RISC-V International, 2024g; RISC-V International, 2024f). Additionally, RVA23 mandates integer conditional operations (Zicond), additional compressed instructions (Zcb), additional floating-point instructions (Zfa), vector basic bit-manipulation (Zvbb), vector data-independent execution latency (Zvkt), and pointer masking (Supm). Scalar cryptography extensions (Zkn, Zks) that were optional in RVA22 are dropped in favor of mandatory vector crypto, since vectors are now mandatory and vector crypto is substantially faster (RISC-V International, 2024f).

Table 4 summarizes the profile progression and its ARM equivalence.

Table 4 – RISC-V Application Profile Evolution

Level	Role	ARM Equivalent
RV64GC	Bare minimum for kernel boot	ARMv8.0-A base
RVA20	Initial ecosystem baseline	ARMv8.0-A mandated
RVA22	Portable Linux platform	~ARMv8.2-A baseline
RVA23	Modern competitive platform	~ARMv9-A with SVE2

An important architectural distinction: profiles do not define new instructions or version the ISA. The RISC-V base ISA is frozen and never changes. Profiles are curated selections from *already-ratified* extensions—they move extensions from "optional" to "mandatory" for a given processor class. For example, the RVV 1.0 vector extension was ratified in 2021, but only became a profile-mandatory extension in RVA23 (2024), giving chip designers three years to implement and validate vector support before it became an ecosystem requirement (RISC-V International, 2023c; FPROX, 2023). This contrasts with ARM's approach, where a new architecture version (e.g., ARMv8.4) simultaneously introduces and mandates new instructions such as dot-product operations (Arm Ltd., 2023a).

The profile system reflects RISC-V's pragmatic acknowledgment that pure modularity creates fragmentation. Ubuntu announced that starting from October 2025, RISC-V images would require RVA23-compliant hardware (RISC-V International, 2024g), making the profile transition a concrete software ecosystem milestone that will drive hardware convergence.

3.4 Register Architecture

3.4.1 Integer Register File

RISC-V provides 32 general-purpose integer registers (x0–x31), each XLEN bits wide (32 or 64 bits for RV32/RV64). Register x0 is architecturally hardwired to zero; any write to x0 is discarded, and any read returns zero. This design enables elegant pseudo-instruction mappings: `nop` is `addi x0, x0, 0`, `mv rd, rs` is `addi rd, rs, 0`, and `not rd, rs` is `xori rd, rs, -1` (WATERMAN; ASANOVIĆ, 2019).

AArch64 provides 31 general-purpose registers (X0–X30) plus the zero register (XZR/WZR) and the stack pointer (SP). The stack pointer and zero register share the same encoding, with context determining interpretation, a source of decode complexity absent in RISC-V, where x2 (sp) is a conventional designation enforced by the ABI rather than hardware (Arm Ltd., 2023a; WATERMAN; ASANOVIĆ, 2019).

3.4.2 Absence of Condition Flags

A defining characteristic of RISC-V is the *absence of a condition flags register*. ARM's PSTATE contains NZCV (Negative, Zero, Carry, oVerflow) flags set by comparison and arithmetic instructions, consumed by subsequent conditional branches or selects. x86's FLAGS/EFLAGS register serves an analogous role (PATTERSON; HENNESSY, 2017).

RISC-V conditional branches directly compare two registers: `beq x5, x6, label` (branch if x5 equals x6). This eliminates the implicit data dependency between a compare instruction and subsequent branches through the flags register. For superscalar and out-of-order implementations, this simplifies dependency tracking: the branch depends only on its explicit source registers, not on an implicit flags register that may be written by unrelated instructions (WATERMAN; ASANOVIĆ, 2019; WATERMAN et al., 2014).

The tradeoff is reduced code density for certain patterns: ARM can execute a single comparison followed by multiple conditional operations consuming the same flags, whereas RISC-V requires separate comparisons. However, as we discuss in Section 3.7, high-performance RISC-V implementations recover this through instruction fusion.

3.4.3 Floating-Point Register File

When the F (single-precision) and D (double-precision) extensions are implemented, RISC-V provides 32 *separate* floating-point registers (f0–f31), completely independent from the integer file. ARM's AArch64 shares the V0–V31 register file between NEON/SIMD and scalar floating-point operations, with D-registers and S-registers aliasing the lower portions of V-registers (Arm Ltd., 2023a).

RISC-V's separation means integer and floating-point pipelines can operate without register file port contention, at the cost of additional silicon for the separate file. This architectural choice is carried through to the vector extension (Section 3.8), where the SiFive P870 implements separate FP and vector register files with independent rename resources (Chips and Cheese, 2023).

3.4.4 Control and Status Registers

RISC-V defines a 12-bit CSR address space supporting up to 4,096 control and status registers, accessed via dedicated `csrrw/csrrs/csrrc` instructions (Zicsr extension). CSRs are organized by privilege level (M-mode, S-mode, U-mode) and include machine status (`mstatus`), trap vectors (`mtvec/stvec`), exception cause (`mcause/scause`), performance counters, and timer registers (WATERMAN; ASANOVIĆ, 2021). This compares to ARM's extensive system register space accessed via MSR/MRS instructions, organized by exception level (Arm Ltd., 2023a).

3.5 Privilege Modes and System Architecture

RISC-V defines three base privilege modes, augmented by an optional fourth mode introduced by the H-extension (Section 3.9). Machine mode (M) is the most privileged and always present, executing firmware and bootloader code; it is analogous to ARM's EL3 secure monitor in privilege but is mandatory rather than optional. Supervisor mode (S) runs the OS kernel and is optional, needed only for virtual memory and OS-level separation, corresponding to ARM EL1. User mode (U) runs unprivileged applications, corresponding to ARM EL0. A minimal embedded system may implement only M-mode, running bare-metal code with full hardware access, simpler than even the Cortex-M Handler/Thread distinction with its optional MPU (WATERMAN; ASANOVIĆ, 2021; Arm Ltd., 2023a).

Within this hierarchy, M-mode can selectively delegate specific exceptions and interrupts to S-mode through the `medeleg` and `mideleg` CSRs, where each bit corresponds to a particular cause; when set, that exception is handled directly by S-mode without first trapping to M-mode, much as ARM routes exceptions through SCR_EL3 and HCR_EL2 (WATERMAN; ASANOVIĆ, 2021). Memory isolation is provided by Physical Memory Protection (PMP), which lets M-mode restrict S- and U-mode access to physical address regions through up to 16 implementation-defined entries carrying read/write/execute permissions; PMP is the RISC-V counterpart to the Cortex-R/M MPU and the primary isolation mechanism for bare-metal and RTOS systems without virtual memory (WATERMAN; ASANOVIĆ, 2021). Notably, RISC-V's base specification has no equivalent of ARM TrustZone's hardware-enforced secure/non-secure world split; that gap is filled instead by approaches such as SiFive's WorldGuard multi-domain partitioning (SiFive, 2021b), the open-source Keystone trusted execution environment (TEE) framework (LEE et al., 2020a), and the emerging CoVE confidential-VM extension (RISC-V International, 2024a).

System services are mediated by the Supervisor Binary Interface (SBI), a standardized set of firmware calls between the S-mode kernel and M-mode platform firmware, invoked via the `ecall` instruction and covering console I/O, timer programming, inter-processor interrupts, remote fences (TLB shootdowns), system reset, and performance monitoring (RISC-V International, 2022). Its reference implementation, OpenSBI, remains resident after boot to service ongoing calls (RISC-V International, 2026b), much as ARM Trusted Firmware stays at EL3 to handle PSCI requests, though the SBI interface is more formally specified and carries higher call frequency, since every timer reprogramming and IPI delivery traverses it. The Sstc extension mitigates this overhead by exposing a direct supervisor-mode timer-compare register (`stimecmp`), reducing the frequency of M-mode traps (RISC-V International, 2021c).

3.6 Linux Requirements on RISC-V

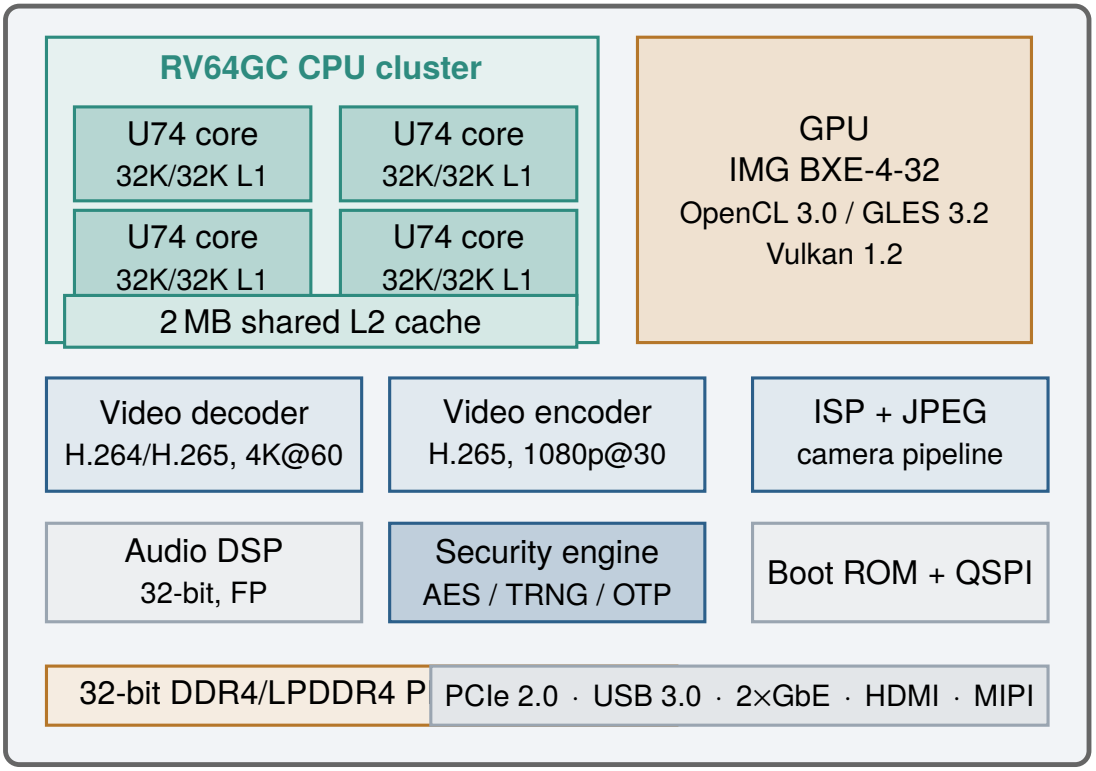
Running Linux on RISC-V requires the RV64GC configuration, in which each extension underpins specific kernel or userspace functionality: RV64I supplies the 64-bit addressing needed for virtual memory; M provides hardware multiply and divide for scheduler, timekeeping, and allocator arithmetic; A supplies the atomic instructions (LR/SC) behind spinlocks, RCU, and the lock-free structures essential to SMP operation; F and D provide hardware floating-point for userspace math libraries (the kernel itself avoids floating-point); and C contributes the compressed instructions that all mainline toolchains emit by default for code density. The Zicsr and Zifencei extensions are additionally required, for CSR access and instruction-cache coherence respectively (WATERMAN; ASANOVIĆ, 2019; StarFive, 2024b).

Linux also requires an MMU implementing one of the RISC-V page-table formats. Sv39 is the most common, providing 39-bit virtual addresses through a three-level walk with 4 KiB pages, where each level uses 9 bits of the virtual address to index a 512-entry table; the `satp` CSR holds the root page-table physical page number, an ASID for TLB tagging, and a mode field selecting the scheme (WATERMAN; ASANOVIĆ, 2021). Sv48 extends this to four-level, 48-bit translation (256 TiB), matching ARM's standard 48-bit configuration, and Sv57 to five-level, 57-bit translation for very large address spaces (WATERMAN; ASANOVIĆ, 2021). The page-table entry (PTE) format is unified across levels—leaf entries (with R, W, or X set) carry physical page numbers while non-leaf entries point to the next level—and superpages (2 MiB at level 1, 1 GiB at level 2) are formed by setting permission bits at intermediate levels (WATERMAN; ASANOVIĆ, 2021).

The kernel further needs a timer source and an interrupt controller. The CLINT or ACLINT supplies per-hart `mtime` and `mtimecmp` registers, analogous to ARM's Generic Timer (WATERMAN; ASANOVIĆ, 2021; Arm Ltd., 2023a), while external interrupts are routed by the PLIC with priority and claim/complete semantics; the newer Advanced Interrupt Architecture adds the APLIC and IMSIC for the MSI-based delivery that PCIe and high-performance I/O

require, paralleling ARM's GICv3/GICv4 with ITS (RISC-V International, 2023a). Bringing these together, the boot flow proceeds from the zero-stage boot loader in mask ROM through an FSBL/U-Boot SPL stage that loads OpenSBI, which then stays resident in M-mode, to U-Boot or GRUB in S-mode loading the kernel and DeviceTree blob, and finally to the kernel's `start_kernel`; as on ARM, hardware is described by a DeviceTree whose pointer is passed in register `a1` at kernel entry (RISC-V International, 2026b; StarFive, 2024b). Production silicon already exercises this path: Linux-capable RISC-V SoCs such as the StarFive JH7110 (SiFive U74 cores, VisionFive 2 board), the Sophgo SG2042 (64 T-Head C920 cores), and the SpacemiT K1 (X60 cores) all implement RV64GC with Sv39 and boot mainline Linux via OpenSBI (BROWN et al., 2024; SpacemiT, 2024). Figure 5 illustrates the block-level organization of one such SoC: alongside the U74 CPU cluster, the JH7110 integrates a GPU, an image-signal processor, video codecs, and a hardware security engine on a single die—the same heterogeneous-integration pattern that characterizes commercial ARM SoCs, here in an entry-level, fully Linux-capable RISC-V part.

Figure 5 – Block-level organization of the StarFive JH7110, a Linux-capable RISC-V SoC (quad-core SiFive U74, RV64GC, 28 nm).



Source: prepared by the author, based on the vendor's datasheet block diagram (StarFive Technology, 2023).

3.7 Pipeline Implementations

The RISC-V implementation ecosystem rests on two open-source cores developed at UC Berkeley. Rocket (ASANOVIĆ et al., 2016) is a single-issue, in-order, five-stage pipeline written in Chisel (a Scala-based hardware description language); implementing RV64GC, it has been fabricated in silicon many times, delivers Cortex-A5/A7-class performance at 1.0–1.5 GHz on modern nodes, and, through the Rocket Chip SoC generator, can be instantiated with custom accelerators via the RoCC interface. BOOM, the Berkeley Out-of-Order Machine (CELIO; PATTERSON; ASANOVIĆ, 2015), is a superscalar, out-of-order core in the same language, configurable from two- to four-wide decode with 10–12 stages; it shows that RISC-V's design choices (no condition flags, simple addressing modes) compose effectively with aggressive speculation, and Tenstorrent's open-source Ocelot extends it with RVV 1.0 vector support (Tenstorrent, 2023).

Commercial silicon has progressed through three broad generations. The first generation comprises in-order workhorses: the SiFive U74, a dual-issue, eight-stage core delivering Cortex-A55-class performance at roughly 1.5 GHz on 28 nm (SiFive, 2021a); the single-issue, five-stage T-Head C906 used in the Allwinner D1 (T-Head Semiconductor, 2020); and the SpacemiT X60, which reaches about 1.3× the performance of a Cortex-A55 at 0.8× the power with RVV 1.0 support (VLEN=256) (SpacemiT, 2024).

The second generation introduced narrow out-of-order execution. Alibaba DAMO Academy's T-Head C910/C920, first described at ISCA 2020 (CHEN et al., 2020b), is a three-wide, twelve-stage core implementing RV64GCV with dual-issue load/store units and a 64 KB instruction cache that stores predecode bits to resolve the C-extension's variable instruction boundaries (Chips and Cheese, 2025); the Sophgo SG2042 packs 64 C920 cores for server workloads at roughly 20–35% of AMD Zen 2 single-core performance (BROWN et al., 2024). SiFive's first out-of-order core, the three-wide, thirteen-stage P550 (2022), and Ventana's four-wide Veyron V1, which shipped without vector execution, round out this generation (Ventana Micro Systems, 2022).

The third generation brought wide out-of-order designs. The SiFive P870, presented at Hot Chips 2023 (SiFive, 2023), marks a major leap: it pairs six-wide decode (36 bytes/cycle, nine RISC-V instructions) with an aggressive fusion engine, an eight-table, 16K-entry TAGE branch predictor rivalling Zen 3's BTB capacity, and a notably short ~8-cycle misprediction penalty, well under Cortex-X3 (~11), Zen 4 (~13), or Golden Cove (~15+) (Chips and Cheese, 2023); its ~1120-entry pre-fusion reorder buffer (~280 effective fused macro-ops) and dual 128-bit RVV 1.0 pipes place it in Cortex-X2/Zen 4c territory at 17 SPECint2006/GHz, and the P870-A automotive variant adds lockstep, ECC caches, and RAS for ASIL B safety (ServeTheHome, 2023; SiFive, 2024b). The T-Head C930 (March 2025) is a six-wide, sixteen-stage RVA23-compatible core targeting servers, PCs, and autonomous vehicles (The Register, 2025), while Tenstorrent's Ascalon-X—an eight-wide core from Jim Keller's team, with dual 256-bit RVV 1.0

units—reaches roughly 21–22 SPECint2006/GHz (WARD-FOXTON, 2025; XPU.pub, 2025) and scales across the Ascalon family from two- to eight-wide decode for heterogeneous SoCs (Tom's Hardware, 2023).

Table 5 summarizes these pipeline parameters against contemporary ARM and x86 cores.

Table 5 – Cross-Architecture Pipeline Comparison (High-Performance Cores, 2023–2025)

Property	**SiFive P870**	**Ascalon-X**	**Cortex-X4**	**AMD Zen 5**	**Intel Golden Cove**
Decode width	6	8	10 (fused)	6 (op cache)	6 (op cache)
Fetch bandwidth	36 B/cyc	~32+ B/cyc	32 B/cyc	32 B / 12 μops	32 B / 6 μops
Instruction format	Native RISC-V	Native RISC-V	Native AArch64	x86 → μops	x86 → μops
μop cache	No	No	MOP cache	Yes (6.75K)	Yes (4K)
Mispredict penalty	~8 cyc	~10–12 cyc	~11 cyc	~13 cyc	~15–17 cyc
ROB depth	~280 (fused)	Large (undiscl.)	384	448	512
Integer ALUs	6	6 + 2 branch	6	6	5
Vector width	2×128b RVV	2×256b RVV	4×128b SVE2	2×512b AVX-512	2×256b AVX
SPECint2006/GHz	~17	~21–22	~20–22	~22–24	~19–20

RISC-V's ISA simplicity necessitates a distinctive approach to performance recovery. Where ARM encodes complex operations in single instructions—conditional selects, fused compare-and-branch, indexed addressing—RISC-V delegates that complexity to the microarchitecture through instruction fusion: the P870's decoder recognizes common multi-instruction patterns and welds them into single internal operations, recovering the code-density loss at the hardware level without burdening the ISA with complex encodings (Chips and Cheese, 2023). The implication is significant, since different implementations can fuse different patterns—simple in-order cores fusing nothing, high-end cores dozens—so the ISA stays stable while hardware capability evolves across generations, a fundamentally different upgrade path from ARM's practice of adding new ISA-level instructions such as ARMv8.4 dot products or ARMv9 SVE2 (WATERMAN; ASANOVIĆ, 2019).

Pipeline depth carries directly into real-time suitability, since it governs the complexity of worst-case execution-time (WCET) analysis. RISC-V's simpler decode lets equivalent performance be reached with shorter pipelines—the P870's roughly 10–12 stages (implied by its 8-cycle misprediction penalty) against Cortex-X3's ~14 and Golden Cove's ~20+—which yields lower misprediction penalties, more predictable timing, and simpler pipeline models for static timing-analysis tools (Chips and Cheese, 2023; REGHENZANI; MASSARI; FORNACIARI, 2019b).

3.8 Vector and SIMD Extensions

Data-parallel execution on CPUs has evolved through three paradigms: (1) fixed-width packed SIMD (ARM NEON at 128 bits, x86 SSE/AVX/AVX-512 at 128/256/512 bits), where software is compiled for a specific register width; (2) vector-length-agnostic (VLA) processing (ARM SVE/SVE2, RISC-V RVV 1.0), where the same binary runs across implementations with different physical vector widths; and (3) matrix/tensor extensions (ARM SME/SME2, Intel

AMX), providing direct 2D matrix operation support (RISC-V International, 2021a; Arm Ltd., 2023c; STEPHENS et al., 2017b).

3.8.1 RVV 1.0 Architecture

The RISC-V Vector Extension (RVV), ratified in September 2021 (RISC-V International, 2021a), introduces 32 vector registers (v0–v31) with implementation-defined width VLEN (minimum 128 bits, implementations range from 128 to 4096+ bits in research cores (CAVALCANTE et al., 2022)). The extension adds approximately 302 instructions plus configurable load/store operations (MUŁA, 2024).

Key architectural parameters include:

SEW (Selected Element Width): Dynamically configurable element size (8, 16, 32, or 64 bits), set via the `vsetvl` instruction. A VLEN=256 register holds 8×32-bit floats or 32×8-bit integers, enabling mixed-precision processing within the same register file (RISC-V International, 2021a).

LMUL (Length Multiplier): A mechanism unique to RVV allowing grouping of 2, 4, or 8 consecutive vector registers into a single logical vector. With LMUL=4 on VLEN=256, the effective vector length becomes 1024 bits from four grouped registers, trading register count for vector length. Fractional LMUL (1/2, 1/4, 1/8) conversely provides more logical registers with shorter vectors. Neither ARM SVE nor x86 AVX offers an equivalent mechanism (RISC-V International, 2021a; MUŁA, 2024).

vsetvl loop pattern: The canonical VLA programming pattern uses `vsetvli t0, a0, e32, m1` to query the hardware for the maximum number of 32-bit elements processable in one iteration. The hardware returns this count in `t0`, and the loop adjusts accordingly; the same binary automatically adapts to any VLEN (RISC-V International, 2021a; Samsung Research, 2024).

3.8.2 Memory Access Modes

RVV provides three memory access patterns: unit-stride (`vle32.v`) for contiguous access, strided (`vlse32.v`) for regular-offset patterns (e.g., matrix column access from row-major storage), and indexed/gather-scatter (`vluxei32.v`/`vsuxei32.v`) for irregular access patterns (RISC-V International, 2021a).

RVV additionally supports segment loads (`vlseg2e32.v` through `vlseg8e32.v`), which load interleaved data and de-interleave into separate registers, ideal for RGB pixel processing, multi-axis sensor data, and struct-of-arrays transformations. ARM NEON provides `VLD2`/`VLD3`/`VLD4` for similar patterns; SVE's equivalent is more limited; x86 AVX has no native segment load (RISC-V International, 2021a; MUŁA, 2024).

3.8.3 Masking and Control Flow

RVV uses vector register v0 as the sole masking operand for most instructions, interpreting the VL lowest bits as a mask. This differs from AVX-512's dedicated mask register file (k0–k7) and SVE's separate predicate registers (p0–p15). RVV's approach simplifies hardware (no additional register file) but creates register pressure and scheduling constraints (RISC-V International, 2021a; MUŁA, 2024; CDR, 2024).

Both RVV and SVE support first-fault loads (`vle32ff.v` / `LDFF1`), enabling safe speculative vectorization of loops where the trip count may cross page boundaries, a capability absent from NEON and all x86 extensions (RISC-V International, 2021a; STEPHENS et al., 2017b).

RVV defines both ordered (`vfredosum`) and unordered (`vfredusum`) floating-point reductions. Ordered reductions guarantee bit-identical results regardless of VLEN, critical for numerical reproducibility in safety-certified systems. SVE's `FADDA` provides equivalent ordered semantics; x86 AVX lacks any ordered reduction instruction (RISC-V International, 2021a).

3.8.4 Comprehensive Cross-Architecture Comparison

Table 6 presents a detailed feature comparison across the five principal vector/SIMD extensions.

3.8.5 ARM SME/SME2 and the RISC-V Matrix Gap

ARM's Scalable Matrix Extension (SME), ratified in ARMv9-A, introduces a 2D tile register (ZA array) with dedicated outer-product instructions (`FMOPA`, `BFMOPA`) that directly accelerate the GEMM (General Matrix Multiply) inner loop central to neural network inference (Arm Ltd., 2024d). SME2 adds multi-vector instructions with on-the-fly dequantization for 2-bit and 4-bit quantized weights, achieving up to 6× faster LLM chat responses on SME2-enabled hardware (InfoQ, 2025). Even ARM's newest efficiency core (C1-Nano, successor to Cortex-A520) includes shared SME2 support (Cyber Raiden, 2026).

RISC-V's Integrated Matrix Extension Task Group (IME-TG) is developing a matrix instruction set that reuses vector registers for matrix operations, targeting HPC workloads (RISC-V International, 2025). A separate effort proposes standalone matrix registers analogous to Intel AMX or ARM SME's ZA. Neither has been ratified as of 2025, leaving RISC-V reliant on custom vendor extensions for matrix acceleration, a significant gap for AI inference workloads (RISC-V International, 2025).

Table 6 – Cross-Architecture Vector/SIMD Extension Comparison

Feature	**NEON**	**SVE/SVE2**	**RVV 1.0**	**AVX2**	**AVX-512**
Register Architecture					
Vector registers	32×128b	32×VL bits	32×VLEN bits	16×256b	32×512b
Width model	Fixed 128b	VLA: 128–2048b	VLA: impl-defined	Fixed 256b	Fixed 512b
Mask registers	None	16 predicates	v0 only	None	8 (k0–k7)
Register grouping	None	None	LMUL (1/8 to 8)	None	None
Memory Access					
Strided load	No	Yes	Yes	No	No
Gather/scatter	No	Yes	Yes	Gather only	Yes
Segment load	VLD2/3/4	LD2/3/4 (limited)	vlseg2–vlseg8	No	No
First-fault load	No	Yes (LDFF1)	Yes (vle32ff)	No	No
Control Flow					
Active-length ctrl.	No (fixed)	Predicates	vsetvl	No	Mask-based
Tail handling	Manual	Predicate zeroes	Undisturbed/agnostic	Manual	Zero/merge
Ordered reduction	No	Yes (FADDA)	Yes (vfredosum)	No	No
Ecosystem (2025–2026)					
Compiler auto-vec	Mature	Mature	Good (GCC 13+)	Mature	Mature
Matrix extension	None	SME/SME2	IME (in progress)	None	AMX

3.8.6 CPU Vector Extensions vs. GPU Matrix Throughput

The performance gap between CPU vector/matrix extensions and GPU tensor cores is structural rather than evolutionary, arising from five fundamental architectural differences:

Parallelism granularity: An NVIDIA H100 GPU contains 16,896 FP32 ALUs and 528 Tensor Cores across 132 streaming multiprocessors (SMs), delivering ~990 TFLOPS of FP16 matrix throughput. A CPU with ARM SME2 achieves ~1.7 TFLOPS across 8 performance cores—a 500–1000× throughput gap arising from unit replication count, not instruction quality (NVIDIA, 2022).

SIMT (Single-Instruction, Multiple-Thread) execution model: GPUs hide memory latency through massive thread-level parallelism (thousands of concurrent warps), using zero-cost context switching between warps rather than the speculation and prediction hardware that consumes the majority of CPU die area (NVIDIA, 2024).

Memory bandwidth: H100 provides 3.35 TB/s HBM3 bandwidth versus ~460 GB/s for high-end server DDR5—a 7× gap that compounds with the compute gap (NVIDIA, 2022).

Systolic dataflow: GPU Tensor Cores use fixed-function systolic arrays where data flows through hardwired multiply-accumulate cells without register file arbitration, achieving ~20× the area efficiency of equivalent general-purpose FMA units (MARKIDIS et al., 2018).

Arithmetic intensity: Matrix multiplication has arithmetic intensity proportional to matrix dimension (~2n/3 ops per element for $n \times n$ matrices), making it overwhelmingly compute-bound for large matrices—perfectly matched to GPU architecture but underutilizing CPU caches designed for low-latency random access (WILLIAMS; WATERMAN; PATTERSON, 2009).

For robotics, this gap motivates heterogeneous SoC designs where RISC-V CPU cores handle control loops, middleware, and sensor fusion while dedicated accelerators (GPU cores, custom matrix engines via RoCC) handle perception inference (SiFive, 2024a).

3.9 H-Extension Virtualization

The RISC-V Hypervisor extension (H-extension), ratified as part of privileged specification v1.12 (WATERMAN; ASANOVIĆ, 2021), virtualizes the supervisor-level architecture by transforming S-mode into a hypervisor-extended supervisor mode (HS-mode) rather than adding a dedicated hypervisor privilege level. A single-bit virtualization indicator, V, distinguishes host context (V=0) from guest context (V=1) (XCP-ng, 2024): when V=1, the hart runs in virtual supervisor mode (VS-mode) or virtual user mode (VU-mode), and the supervisor CSRs (`sstatus`, `satp`, `stvec`, and others) are transparently redirected to virtual copies (`vsstatus`, `vsatp`, `vstvec`) so that unmodified guest operating systems execute without awareness of virtualization. This contrasts with ARM's dedicated EL2 hypervisor exception level, which occupies a distinct privilege ring between EL1 and EL3; where ARM later added Virtualization Host Extensions (VHE, ARMv8.1) to let the host kernel run at EL2 and reduce transition overhead, RISC-V's HS-mode provides the equivalent by design (Arm Ltd., 2023a).

Both architectures implement two-stage memory translation for guest isolation: stage 1, guest-controlled, translates guest-virtual to guest-physical addresses, while stage 2, hypervisor-controlled, translates guest-physical to real supervisor-physical addresses. In RISC-V, stage 1 uses `vsatp` and stage 2 uses `hgatp` with identical PTE formats across both stages, whereas ARM uses `TTBR0_EL1`/`TTBR1_EL1` for stage 1 and `VTTBR_EL2` for stage 2 (WATERMAN; ASANOVIĆ, 2021; Arm Ltd., 2023a). VMID (Virtual Machine Identifier) tagging in `hgatp` lets TLB entries from multiple guests coexist without flushing, with implementations supporting up to 14-bit VMIDs—16,384 concurrent VM identifiers.

Interrupt and I/O virtualization complete the model. The Advanced Interrupt Architecture (AIA) provides the IMSIC (Incoming Message-Signaled Interrupt Controller) with per-hart, per-guest interrupt files that deliver interrupts directly to guests without hypervisor intervention, the equivalent of ARM GICv4's virtual LPI injection, while the RISC-V IOMMU specification (ratified 2024) supplies DMA address translation for device passthrough, analogous to ARM's SMMUv3 and Intel VT-d (RISC-V International, 2023a; RISC-V International, 2024d).

A growing hypervisor ecosystem targets these mechanisms: Linux KVM (upstream since roughly v5.16, using the H-extension for type-2 virtualization) (CORBET, 2021), Xvisor (a

type-1 monolithic hypervisor for embedded real-time) (PATEL; DARBARI, 2020), Bao (type-1 static partitioning for mixed-criticality embedded systems, with under 1% overhead) (MARTINS et al., 2021), and Xen (porting in progress for server and cloud deployment) (XCP-ng, 2024). In robotics, this lets a safety-certified RTOS handling motor control and safety monitoring run alongside general-purpose Linux handling ROS2, perception, and networking on the same SoC, with hardware-enforced isolation through stage-2 page tables, IOPMP, and IMSIC-based direct interrupt routing; static-partitioning hypervisors such as Bao configure that isolation at boot and impose near-zero runtime overhead, keeping the arrangement compatible with safety-certification requirements.

Underlying all of this is temporal rather than spatial multiplexing: only one VM runs per hart at any instant, with the V-bit swap providing fast context switching through CSR duplication. This differs fundamentally from simultaneous multithreading (SMT), which multiplexes spatially by having two threads share pipeline resources every cycle. The distinction is decisive for safety certification, because temporal multiplexing gives clean per-VM isolation without the per-cycle interference that renders SMT incompatible with most safety-critical guidelines such as CAST-32A and ISO 26262 multicore guidance (REGHENZANI; MASSARI; FORNACIARI, 2019b; EASA, 2016).

3.10 Safety-Critical Systems

Functional safety certification (ISO 26262 for automotive, IEC 61508 for industrial, DO-178C for aerospace) rests on three pillars: fault detection, fault containment, and deterministic timing (International Organization for Standardization, 2018; International Electrotechnical Commission, 2010). RISC-V addresses each pillar through a combination of architectural features and vendor-implemented safety mechanisms.

3.10.1 Fault Detection: Lockstep and Error Protection

Dual-Core Lockstep (DCLS) is the primary mechanism for achieving ASIL D (automotive) or SIL 3/4 (industrial) diagnostic coverage. Two identical cores execute the same instruction stream, with a cycle-by-cycle comparator detecting divergence from transient or permanent faults (International Organization for Standardization, 2018).

Multiple RISC-V vendors now offer ISO 26262-certified lockstep cores:

- **SiFive E6-AB/AD/AS**: 32-bit embedded cores certified to ASIL B (E6-AB) through ASIL D in lockstep mode (E6-AS), with split-lock capability for dual independent operation at ASIL B (SiFive, 2025b)
- **SiFive S7-AD**: 64-bit core optimized for ASIL D zonal controllers (SiFive, 2025b)
- **Andes D23-SE/D45-SE**: DCLS with split-lock supporting both ASIL B (independent) and ASIL D (lockstep) operation (Andes Technology, 2025)

- **SiFive P870-A**: High-performance out-of-order core with ASIL B lockstep, ECC caches, and advanced RAS, combining performance-class computing with functional safety (SiFive, 2024b)

SiFive additionally achieved ISO/SAE 21434:2021 product certification for automotive cybersecurity (SiFive, 2025b).

3.10.2 Fault Containment: Memory Protection and Isolation

RISC-V provides a layered isolation stack:

PMP (Physical Memory Protection): M-mode-controlled physical address region permissions, the RISC-V equivalent of ARM Cortex-R/M's MPU (WATERMAN; ASANOVIĆ, 2021).

WorldGuard: SiFive's multi-domain security partitioning with more than two isolation worlds, exceeding ARM TrustZone's binary secure/non-secure split (SiFive, 2021b).

H-extension with stage-2 page tables: VM-level spatial isolation through hypervisor-controlled physical memory mapping (Section 3.9) (WATERMAN; ASANOVIĆ, 2021).

IOPMP (I/O Physical Memory Protection): DMA isolation preventing compromised peripherals from accessing unauthorized memory regions, complementing the RISC-V IOMMU for virtualized environments (RISC-V International, 2024d).

Cache coloring: Implemented by hypervisors like Bao to eliminate shared-cache interference between partitions, addressing the multicore temporal isolation challenge central to CAST-32A and ISO 26262 multicore guidance (MARTINS et al., 2021; EASA, 2016).

3.10.3 Deterministic Timing and WCET Analysis

WCET analysis complexity is directly influenced by pipeline design. RISC-V's ISA characteristics provide several advantages for timing analysis:

- **No condition flags**: Eliminates implicit dependencies through a shared flags register, simplifying data-flow analysis
- **Simple addressing modes**: Base-register-plus-immediate only for load/store, removing variable-latency addressing calculations
- **Regular instruction encoding**: Fixed register field positions enable straightforward instruction decode modeling
- **Shorter pipelines**: The P870's ~10–12 stages (vs. Cortex-X3's ~14, Golden Cove's ~20+) reduce the pipeline state space for timing analysis

For in-order safety cores (SiFive E6, Andes D23-SE), pipeline timing is near-deterministic: each instruction has a known cycle count, cache hit/miss latencies are bounded, and there is no speculative execution variability. The availability of open RTL enables WCET tool vendors to

build exact pipeline models derived directly from the hardware description, rather than relying on vendor documentation that may be incomplete (THOMAS, 2025; REGHENZANI; MASSARI; FORNACIARI, 2019b).

ISO 26262 requires precise documentation of hardware-software interfaces (HSI). RISC-V's modular design supports this by clearly separating base ISA functionality from extensions, enabling per-extension HSI documentation (THOMAS, 2025; International Organization for Standardization, 2018).

3.10.4 PREEMPT_RT on RISC-V

On September 20, 2024, PREEMPT_RT was fully merged into mainline Linux (kernel v6.12) with RISC-V as one of four first-class supported architectures alongside x86, x86_64, and ARM64 (CORBET, 2024). This milestone makes RISC-V a native real-time Linux platform without requiring out-of-tree patches.

Research on the VisionFive 2 (SiFive JH7110, U74 cores) demonstrates maximum scheduling latencies consistently under 150 μs with PREEMPT_RT enabled, with latency distributions largely unaffected by varying system loads (JÄMBÄCK, 2022; SCHAFFNER; ALTENBERG; WALLENTOWITZ, 2025). These results are comparable to measurements on ARM and x86 platforms, confirming RISC-V's suitability for soft real-time applications (JÄMBÄCK, 2022).

Direct robotics validation has been demonstrated through implementation of motion control for a six-axis collaborative robot running on RISC-V PREEMPT_RT Linux with EtherCAT fieldbus communication (YOO; CHOI, 2024).

The architecture-specific PREEMPT_RT work for RISC-V was minimal, as the majority of real-time functionality is implemented in architecture-independent kernel code (StarFive, 2024a; SCHAFFNER; ALTENBERG; WALLENTOWITZ, 2025).

The ELISA (Enabling Linux in Safety Applications) project, a Linux Foundation initiative, develops processes and tools for certifying Linux-based safety-critical systems (The Linux Foundation, 2026). ELISA's methodology—including STPA (System-Theoretic Process Analysis), kernel tracing infrastructure, and documentation frameworks—is architecture-independent and applies equally to RISC-V and ARM64 (The Linux Foundation, 2026).

3.10.5 The Open-RTL Advantage

RISC-V's most distinctive safety advantage is *full-stack auditability*. The entire trust chain—from open-source RTL cores (Rocket, CVA6, or commercially licensed RTL with source access), through OpenSBI firmware, Bao hypervisor, and the Linux kernel—can be inspected, formally verified, and independently assessed. ARM's Cortex-R cores are delivered as encrypted netlists or hard macros; the licensee cannot independently verify lockstep comparator coverage or safety mechanism correctness (THOMAS, 2025).

For ISO 26262 Part 11 (semiconductor-level safety analysis), demonstrating safety mechanism effectiveness through formal proofs on inspectable RTL provides qualitatively richer certification evidence than relying on vendor documentation alone. RISC-V's open architecture also enables dissimilar redundancy by employing different vendor implementations of the same ISA, valuable for DO-178C DAL-A common-mode failure protection (THOMAS, 2025).

3.10.6 Toolchain and Ecosystem

The safety toolchain ecosystem for RISC-V includes IAR Embedded Workbench (ISO 26262 certified for RISC-V, with TÜV SÜD safety certificate) (IAR Systems, 2023), Green Hills Software (functional safety compiler) (Green Hills Software, 2026), LDRA (WCET analysis, code coverage, requirements traceability) (THOMAS, 2025), and Imperas (virtual platform models for Ascalon and other RISC-V cores) (Imperas Software, 2023).

Gaps remain in safety RTOS availability (QNX and INTEGRITY lack RISC-V ports), AUTOSAR Classic stack maturity, and deployment track record, areas where ARM retains a 15+ year advantage (Eureka PatSnap, 2024; International Organization for Standardization, 2018).

3.11 Discussion and Future Directions

RISC-V has reached architectural feature parity with ARM for safety-critical embedded systems: lockstep execution (ASIL D), memory protection through PMP, hypervisor-based isolation via the H-extension, first-class real-time Linux support (PREEMPT_RT in v6.12+), and pipeline performance approaching the Cortex-X and Neoverse V classes. The primary remaining gap is not architectural but ecosystem-level: deployment track record, the availability of safety-certified RTOSes, AUTOSAR support, and assessor familiarity.

The architecture's modular ISA and open extension mechanism point toward heterogeneous SoCs that combine in-order lockstep safety cores (ASIL D, for motor control and safety monitoring), out-of-order performance cores (ASIL B, for Linux and ROS2 workloads), and custom accelerators for AI inference, all sharing a single coherent address space with hardware-enforced isolation through the H-extension and IOPMP (SiFive, 2024b; SiFive, 2025b). This directly answers the mixed-criticality demands of modern robotic systems, where one SoC must simultaneously sustain deterministic 10 kHz control loops, soft-real-time perception pipelines, and best-effort middleware.

Several developments are worth monitoring: the RISC-V Integrated Matrix Extension (IME) for standardized matrix acceleration (RISC-V International, 2025); Tenstorrent's Callandor, the next-generation core after Ascalon, targeting roughly twice the IPC by 2027 (XPU.pub, 2025); the XiangShan open-source high-performance core from the Chinese Academy of Sciences; and growing adoption by automotive tier-1 suppliers including Bosch and Infineon (WANG, 2024).

Finally, the contrast with ARM's Cortex-R82 is instructive. The R82 bridges the real-time and application domains within a single 64-bit, Linux-capable core that offers optional lockstep and deterministic pipeline behavior; RISC-V achieves an equivalent bridge instead through heterogeneous SoC composition, pairing E6-class safety cores with P870-class performance cores under a Bao hypervisor, which offers potentially greater flexibility at the cost of more system-level integration effort (SiFive, 2024b; MARTINS et al., 2021).

3.12 Final Remarks

This survey has presented a comprehensive analysis of the RISC-V instruction set architecture across all dimensions relevant to real-time Linux and robotics applications. RISC-V's modular extension system, clean register architecture without condition flags, well-defined privilege hierarchy with the SBI firmware abstraction, and VLA vector processing through RVV 1.0 provide a sound architectural foundation. Pipeline implementations have progressed through three generations, from academic in-order cores to commercial 8-wide out-of-order designs approaching ARM Neoverse V3 and AMD Zen 5 class performance. The H-extension provides elegant hardware-assisted virtualization for mixed-criticality systems, and multiple vendors have achieved ISO 26262 certification through ASIL D with lockstep RISC-V cores.

The gap between RISC-V and ARM is one of maturity, not capability. ARM's 15+ year deployment history in safety-critical automotive and industrial systems, mature RTOS ecosystem, and established assessor familiarity provide significant advantages for near-term production designs. However, for projects with 3–5 year development horizons, and especially those where full-stack transparency provides certification value, RISC-V represents a viable and increasingly compelling alternative.

The unique combination of an open ISA amenable to formal verification, the ability to compose heterogeneous safety and performance cores on a single SoC, and the freedom to add domain-specific accelerators through custom extensions positions RISC-V as a strong candidate for next-generation robotic platforms that demand both ASIL D safety guarantees for actuator control and high-throughput AI-driven perception—all on a fully auditable, royalty-free foundation.

4

Robotic Applications of ARM and RISC-V

4.1 Introduction

Embedded processors form the computational backbone of modern robotic systems, executing everything from low-level motor control loops at microsecond timescales to high-level AI inference consuming billions of operations per second. The choice of processor architecture fundamentally shapes a robot's capabilities, power budget, cost structure, and time-to-market. Two instruction set architectures (ISAs) now define this landscape: ARM, the entrenched incumbent commanding over 50% of the embedded processor market, and RISC-V, the open-source challenger growing at 10.73% CAGR with an estimated 60 billion cores manufactured to date (CSIS, 2024).

The motivation for comparing these architectures in robotics stems from a confluence of trends. Modern robots increasingly demand heterogeneous computing—simultaneous execution of real-time control, sensor fusion, computer vision, SLAM, and neural network inference—creating complex architectural requirements that stress any single processor family (TU et al., 2025). As detailed in Chapter 2, ARM's decades of ecosystem investment have produced a rich stack from safety-certified Cortex-R cores to the 275-TOPS Jetson AGX Orin, making it the default choice for commercial robotics (NVIDIA, 2026a). RISC-V's open ISA, however, enables custom instruction extensions purpose-built for robotic workloads, zero-royalty scaling for cost-sensitive applications, and formal verification advantages critical for safety-critical systems (KALAPOTHAS et al., 2023).

This chapter systematically examines both architectures across the full spectrum of robotic applications. Section 4.2 covers ARM's role in robotics. Section 4.3 analyzes RISC-V's emerging presence. Section 4.4 surveys AI workloads on embedded robotic platforms. Section 4.5 provides comparative analysis. Sections 4.6 and 4.7 address market dynamics and future directions. The analysis draws on over 60 recent papers from IEEE Xplore, ACM Digital Library, arXiv, and

related venues, with emphasis on work published between 2020 and 2025.

4.2 ARM Architecture in Robotics

ARM's presence in robotics spans the entire computational hierarchy, from microcontrollers executing PID loops to application processors running deep neural networks. This vertical integration across the Cortex-M, Cortex-R, and Cortex-A families gives ARM a unique position: a single vendor's ISA can address every processing tier within a robotic system.

4.2.1 Cortex-M Series: The Real-Time Motor Control Workhorse

The ARM Cortex-M family, particularly through STMicroelectronics' STM32 line, dominates low-level robotic control. The STM32F103 (Cortex-M3, 72 MHz) appears in the majority of published robotic motor control designs, handling PID regulation, PWM generation, encoder reading, and CAN bus communication. Li e Wang (2023) demonstrated voice-controlled dancing robot motion using STM32F103C8T6 with integrated inertial measurement unit (IMU) sensing and DC motor control. Jiang et al. (2022) deployed the same processor as the bottom-layer controller in a greenhouse mobile robot, managing PID-based motor control with encoder feedback while an upper layer based on the Robot Operating System (ROS) handled 3D LiDAR SLAM navigation. More ambitiously, researchers implemented real-time path planning based on Rapidly-exploring Random Trees (RRT) for unmanned aerial vehicles (UAVs) entirely on an STM32F103C8T6, demonstrating that collision-free trajectory tracking is achievable even under severe resource constraints of 20 KB of static random-access memory (SRAM) at 72 MHz (TSAI; YANG; SUN, 2025).

The Cortex-M4's floating-point unit enables more computationally demanding robotic applications. A 4-DoF delta parallel robot was prototyped on STM32F4 using Matlab/Simulink with the Waijung Blockset for rapid control prototyping, solving inverse kinematics in real-time (LASKI; SMYKOWSKI, 2021). The hierarchical dual-processor architecture, Cortex-M for low-level control paired with a Cortex-A device for high-level processing, has become a standard design pattern in service robotics (ZHAI; DONG, 2025).

Newer Cortex-M variants push deeper into AI territory. The Cortex-M55 pairs with ARM's Ethos-U55 micro-NPU for on-device TinyML inference, while ARM's CMSIS-NN library provides optimized neural network kernels for Cortex-M processors, enabling object classification and anomaly detection directly on microcontrollers (ABADADE et al., 2023).

4.2.2 Cortex-A and GPU-Equipped SoCs: The AI Inference Powerhouse

The NVIDIA Jetson platform has become the de facto standard for embedded AI in robotics. The current flagship, Jetson AGX Orin 64GB, integrates 12 ARM Cortex-A78AE cores

at 2.2 GHz with a 2048-core Ampere GPU and 64 Tensor Cores, delivering 275 TOPS of INT8 inference at 15–60 W (NVIDIA, 2026a). Kumar, Park e Behera (2023) developed Jetson-SLAM, a GPU-accelerated stereo visual SLAM system achieving over 90 FPS on Jetson NX at 320×240 resolution through innovations including Bounded Rectification and Pyramidal Culling that eliminate costly CPU–GPU memory transfers. Benchmarking of YOLOv8 variants on Jetson Orin NX demonstrated that TensorRT-optimized INT8 models achieve 15.16 ms per inference iteration, with the Orin Nano reaching 23.16 ms—both sufficient for real-time robotic perception at 30+ FPS (ALJAMI et al., 2026).

A survey by Galagain et al. (2025) examined whether Semantic SLAM is ready for embedded systems, benchmarking Geometric SLAM, NeRF-based, and Gaussian Splatting approaches on Jetson AGX Orin. The work revealed that while NeRF and 3DGS methods remain too resource-intensive for embedded deployment, semantic geometric SLAM offers the best balance of accuracy and computational feasibility. The authors noted a critical practical challenge: many libraries need to be recompiled, and algorithms optimized for x86+CUDA create compatibility barriers when ported to ARM.

Beyond Jetson, the Qualcomm Robotics RB5 platform (QRB5165, octa-core Kryo 585, 15 TOPS AI Engine) supports up to 7 concurrent cameras for SLAM and autonomous navigation with optional 5G connectivity (QUALCOMM, 2026). NXP's i.MX 8M Plus (Cortex-A53 + 2.3 TOPS NPU) targets service robots and AMRs with native ROS 2 support, while the newer i.MX 95 (6× Cortex-A55 + Cortex-M7 + 2 TOPS Neutron NPU) adds ASIL B/SIL-2 certified safety features for industrial applications (ADVANTECH, 2024).

4.2.3 Cortex-R Series: Safety-Critical Robotic Systems

ARM Cortex-R processors serve the most demanding safety requirements in robotics. The Cortex-R52, purpose-built for functional safety, supports ISO 26262 ASIL D and IEC 61508 SIL-3 certification with hardware-enforced software separation and hypervisor-enabled virtualization (ARM, 2026). TI's Sitara AM2x MCUs use Cortex-R5F cores delivering up to 6400 DMIPS for industrial servo drives and factory robots with EtherCAT support (Texas Instruments, 2026). The newer Cortex-R82AE extends to 64-bit ARMv8-R architecture supporting up to 1 TB address space for high-performance safety-related autonomous driving subsystems.

4.2.4 ARM's Commercial Robotics Footprint

The breadth of ARM's deployment in production robots is unmatched. Boston Dynamics Spot uses Jetson Xavier NX in its Spot CORE I/O payload for on-robot AI (Boston Dynamics, 2026). Zipline autonomous delivery drones operate with dual Jetson Orin NX modules per aircraft. Universal Robots integrates NVIDIA Isaac ROS through its AI Accelerator toolkit (NVIDIA, 2024). NVIDIA's ecosystem encompasses 850,000+ Jetson developers and 6,000+ companies

building commercial products (NVIDIA, 2026a). ARM announced at CES 2026 that partners including Boston Dynamics, Caterpillar, LG Electronics, and NEURA Robotics are showcasing ARM-powered robotic systems (ARM Newsroom, 2026).

4.3 RISC-V Architecture in Robotics

While RISC-V's commercial footprint in robotics remains nascent compared to ARM, its open ISA has catalyzed a vibrant research ecosystem producing novel architectures, custom accelerators, and energy-efficient platforms that push the boundaries of what is possible in resource-constrained robotic systems.

4.3.1 The PULP/GAP Ecosystem: Dominant Platform for Edge-AI Robotics

The Parallel Ultra-Low-Power Processing Platform (PULP), a joint project of ETH Zurich and the University of Bologna, has produced over 40 ASIC tape-outs and represents the most mature RISC-V ecosystem for robotic applications (ETH Zurich, 2026). The commercially available GAP8 processor—8+1 RISC-V cores (RI5CY, RV32IMC + XpulpV2 extensions) with a hardware CNN accelerator—has become the standard computing platform for autonomous nano-drone research (FLAMAND et al., 2018).

With PULP-DroNet, an open-source autonomous-navigation system for nano-drones, Palossi et al. (2019b) deployed the DroNet CNN on GAP8 aboard a 27-gram Crazyflie nano-drone, achieving autonomous visual navigation at 6–18 fps within just 64–284 mW. The subsequent PULP-Frontnet (PALOSSI et al., 2021) extended this work to human-drone pose estimation, reaching 135 fps at under 87 mW with 8-bit quantized models on the same platform. These results demonstrate RISC-V's exceptional energy efficiency for targeted AI workloads in extremely constrained robotic platforms.

The GAP9 processor advances the platform significantly with a 9-core RISC-V cluster, the NE16 neural engine (32.2 GMAC/s on INT8), and transprecision floating-point support (FP32/FP16/BF16). The GAP9Shield module (MÜLLER; KARTSCH; BENINI, 2024), described in a 2024 preprint, runs YOLO object detection, SLAM, and localization under 100 mW with 18–250 ms latency on nano-drones—a 20% weight reduction over prior designs. NanoSLAM, deployed on a 44-gram drone with GAP9, achieves real-world mapping at 87.9 mW, demonstrating that complete SLAM pipelines can execute on milliwatt-class RISC-V hardware.

4.3.2 RISC-V ISA Extensions for Robotic Acceleration

RISC-V's defining advantage, the ability to add custom instruction set extensions, has produced several innovations relevant to robotics.

The V (Vector) extension, ratified as v1.0, enables SIMD-style parallelism critical for CNN inference and signal processing. The Arrow accelerator (ASSIR et al., 2021) demonstrated 2–78× speedup over scalar RISC-V cores for ML inference on Xilinx FPGA. Kirschner, Dudzik e Becker (2024) presented a 16-core RISC-V vector processor targeting ResNet50 and YOLOv5-small with timing-predictable execution for automated driving.

XpulpV2 custom extensions (used in GAP8/GAP9) add hardware loops, post-increment load/store, SIMD vectorial arithmetic, and 8/16-bit dot product instructions specifically optimized for quantized neural network inference. Garofalo et al. (2020) published XpulpNN at DATE 2020, demonstrating ISA extensions that accelerate quantized neural networks on PULP processors—a key enabler for AI inference on ultra-low-power robotic platforms. Research published at ACM HEART 2024 (M et al., 2024) showed that custom RISC-V instructions for memory operations achieve approximately 50% clock cycle reduction and 30% power reduction for deep neural network (DNN) workloads.

Wang et al. (2024) proposed RV-SCNN, a RISC-V processor with SIMD custom instruction extensions accelerating both spiking and convolutional neural networks, sharing core operators between SNN and CNN modes—a design particularly relevant for neuromorphic robotic perception.

4.3.3 ROS Compatibility on RISC-V: A Maturing Ecosystem

ROS/ROS 2 support on RISC-V has progressed substantially. Lee et al. (2020b) demonstrated the first RISC-V FPGA platform running ROS at IEEE FPL 2020, using a BOOM core on Xilinx Zynq-7000 to control an iRobot vacuum cleaner. Dehnavi et al. (2021) presented CompROS at IROS 2021—a composable architecture with a hard real-time RISC-V subsystem in FPGA programmable logic communicating with an ARM subsystem via bare-metal XRCE-DDS, achieving ROS 2 middleware in under 200 KB.

In 2024, Acceleration Robotics and Microchip achieved a milestone: the first ROS 2 Humble release on a RISC-V SoC FPGA (Microchip PolarFire), with ROS 2-API compatible IP cores for perception and coordinate transformations (Microchip Technology, 2024). Siemens' isar-riscv project provides additional ROS 2 support through Yocto layers targeting RISC-V hardware (SIEMENS, 2026).

4.3.4 Industrial and Real-Time RISC-V for Robotic Control

Yoo e Choi (2024) benchmarked RISC-V real-time performance using FreeRTOS on SiFive HiFive1 Rev B and Preempt-RT Linux on StarFive VisionFive 2, implementing an EtherCAT master to control a six-axis collaborative robot and comparing response times against ARM and x86-64 architectures. The MEbots framework (POLLO et al., 2025) integrates Webots robotics simulation with RISC-V virtual platforms for energy-aware robotic system co-design.

Valente et al. (2024) presented a heterogeneous RISC-V based SoC for secure nano-UAV navigation, combining a cluster of RISC-V cores with cryptographic accelerators.

4.4 AI Workloads on Embedded Robotic Processors

Modern robots must execute a diverse portfolio of AI workloads simultaneously, each with distinct computational profiles and latency requirements. The choice of processor architecture interacts strongly with workload characteristics, creating a complex optimization landscape.

4.4.1 Computer Vision and Object Detection

YOLO-family models dominate real-time object detection in robotics. Benchmarking studies show YOLOv8-nano achieving 30+ FPS on Jetson Orin Nano with TensorRT INT8 optimization, which yields up to 4× speedup over unoptimized PyTorch inference (ALJAMI et al., 2026). A comparative study by Vaghela et al. (2026) evaluated YOLOv5 through YOLOv10 for real-time object detection on a semi-autonomous robot using a custom campus-navigation dataset, with YOLOv9c achieving the highest accuracy at 82.2% mAP50 and the YOLOv8 variants close behind at roughly 81%. For humanoid robots, YOLOv9 integration into the ROBOTIS-OP3 platform demonstrated viability under hardware constraints using ROS and PyTorch (POTTIER; LAU, 2025).

On RISC-V platforms, the Tiny-PULP-Dronets (LAMBERTI et al., 2022) work combined obstacle avoidance and object detection on GAP8 at 134 mW total power. Cereda et al. (2023) presented neural architecture search for visual pose estimation deployed on RISC-V nano-UAV processors at ICRA 2023, targeting sub-10 cm accuracy aboard 27-gram drones. Conti et al. (2024) demonstrated on-device self-supervised learning on GAP9 requiring only 1 MB memory and 19 mW, reducing positioning error by up to 26%.

4.4.2 SLAM on Embedded Platforms

SLAM implementations have matured significantly on ARM-based embedded GPUs. Peng et al. (2020) evaluated the visual-SLAM software libraries ORB-SLAM2 and OpenVSLAM on Jetson Nano, TX2, and Xavier, finding the TX2 offers the best power-performance tradeoff. Data-flow modifications to ORB-SLAM2 achieve approximately 30 FPS on Jetson TX2 through efficient CPU/GPU load distribution (ALDEGHERI et al., 2019). NVIDIA's cuVSLAM reaches 250 FPS localization on Jetson platforms with best-in-class KITTI benchmark results (NVIDIA, 2026b). For multi-robot scenarios, Xu et al. (2024) demonstrated D^2SLAM, a decentralized collaborative visual-inertial SLAM system for aerial swarms deployed on Jetson embedded platforms.

FPGA-based approaches offer lower power alternatives. eSoC-SLAM, an FPGA-accelerated SLAM design implemented on the Intel Cyclone V SoC (ARM Cortex-A9 + FPGA), achieved 6.5× speedup over CPU-only at approximately 32 FPS (GERLEIN et al., 2021). The Navion accelerator by Suleiman et al. (2019) represents the extreme of domain-specific design: a fully integrated visual-inertial odometry accelerator consuming just 2 mW for nano-drone navigation.

4.4.3 TinyML for Resource-Constrained Robots

TinyML, the umbrella term for running machine-learning inference on microcontroller-class hardware, extends AI capability to devices operating at milliwatt power levels. Abadade et al. (2023) published a comprehensive TinyML survey in IEEE Access (2023) covering hardware platforms, software frameworks, and robotics applications on both ARM and RISC-V microcontrollers. Beltrán-Escobar et al.'s 2024 review (BELTRÁN-ESCOBAR et al., 2024) specifically examined TinyML for embedded vision in robotics, covering implementations on Kendryte K210 (RISC-V), Sipeed MAIX-II, and Raspberry Pi.

Key frameworks include TensorFlow Lite Micro (supporting ARM Cortex-M and RISC-V), Edge Impulse (end-to-end TinyML development), STM32Cube.AI (neural network deployment on STM32), and GAPflow (GreenWaves' toolset for GAP8/GAP9) (SHUVO et al., 2023). A climbing inspection robot was demonstrated using FOMO (Faster Objects, More Objects), a compact object-detection model architecture, on Jetson Nano (under 300 ms), Arduino Portenta H7, and Arduino 33 BLE Sense, with INT8 quantization enabling real-time bolt-defect detection on the most constrained platforms (LIN; CHANG; PUTRANTO, 2024).

4.4.4 Foundation Models and LLMs at the Robotic Edge

The deployment of large language models and vision-language-action (VLA) models on robotic platforms represents the newest frontier. On Jetson AGX Orin (64 GB), small VLMs of 4–20 billion parameters run at approximately 40 tokens/second via vLLM, while the upcoming Jetson AGX Thor (128 GB) will support models up to 120 billion parameters (NVIDIA, 2025b). NVIDIA's TensorRT Edge-LLM framework provides C++ inference optimized for embedded automotive and robotics platforms with EAGLE-3 speculative decoding and NVFP4 quantization (NVIDIA, 2025a).

Firoozi et al.'s comprehensive survey (FIROOZI et al., 2023) identifies key challenges for edge deployment of foundation models in robotics: data scarcity, safety guarantees, uncertainty quantification, and the fundamental gap between model computational requirements and embedded hardware capabilities. NVIDIA's Isaac GR00T N1 represents a paradigm shift: a VLA foundation model that ingests camera streams and language commands to directly output joint positions and motor velocities, replacing traditional programmatic robot control with end-to-end learned behavior.

4.4.5 Reinforcement Learning Deployment

RL policies for robotic control follow a consistent deployment pattern: training in simulation (Isaac Gym, MuJoCo) produces small MLPs (2–3 layers, 64–256 units) that run at 200–500 Hz on embedded CPUs. Badalian et al.'s LExCI framework (BADALIAN et al., 2024) bridges RL libraries with embedded hardware, enabling training and deployment with real-time constraints. A critical finding from the Air Learning work (KRISHNAN et al., 2021) is that differences between training hardware (GPU) and deployment hardware (embedded CPU) can cause fundamentally different agent behavior, highlighting architecture-awareness as essential for robotic RL.

Stewart et al. at the Naval Research Laboratory demonstrated converting RL policies for the Astrobee robot into spiking neural networks for Intel's Loihi 2 neuromorphic processor, achieving low-latency, energy-efficient control (CHOWDHURY et al., 2025)—pointing toward neuromorphic computing as an alternative acceleration paradigm for robotic RL.

4.4.6 Sensor Fusion and Real-Time Processing

Sensor fusion represents a critical workload that spans the full computational stack. Extended Kalman Filters (EKF) and complementary filters for IMU/GPS fusion typically execute on Cortex-M microcontrollers at 200–1000 Hz (ABADADE et al., 2023). Multi-modal fusion combining LiDAR, camera, and radar data requires Cortex-A class processors or GPUs. The RoSÉ infrastructure (KIM et al., 2023) enables pre-silicon evaluation of full-stack robotics SoCs, allowing architects to characterize sensor fusion bottlenecks before tape-out. Dong et al. (2025) characterized model-predictive control on embedded SoCs, identifying memory bandwidth as the primary bottleneck rather than compute throughput.

4.5 Comparative Analysis

4.5.1 Performance Benchmarks

An empirical comparison at ACM SC'23 (WEAVER; MCINTOSH-SMITH, 2023) analyzing HPC-style workloads found RISC-V and AArch64 instruction path lengths within 10% of each other, with RISC-V averaging only 2.3% longer paths. The authors concluded that the RISC-V ISA is not disadvantaged compared to AArch64 in terms of potential performance. However, this ISA-level parity does not translate directly to implementation-level parity. ARM Cortex-M cores achieve 3.4–5.0 CoreMark/MHz versus 2.5–3.5 for comparable RISC-V implementations (SIMSON; TAHIRAGA; ECKER, 2024), reflecting ARM's decades of microarchitectural optimization.

A study by Suárez, Almeida e Blanco (2024) in the *Journal of Supercomputing* evaluated energy efficiency across ARM and RISC-V SoCs using NAS Benchmarks and TFLite inference

(MobileNet v1–v3). The critical finding: RISC-V implementations exhibit lower average power consumption, but the performance-per-watt ratio does not necessarily favor RISC-V—a nuance often overlooked in comparisons.

For ML inference specifically, ARM-based platforms dominate current benchmarks. The Jetson AGX Orin delivers 275 TOPS at 60 W (~4.6 TOPS/W), while RISC-V's best commercial offering, GAP9 at 150 GOPS, targets a fundamentally different power envelope (sub-watt). Specialized accelerators like the Hailo-8 achieve class-leading ~10 TOPS/W at 26 TOPS, illustrating that the accelerator architecture matters more than the host ISA for inference workloads (CORDOVA-CARDENAS; AMOR; GUTIÉRREZ, 2025).

4.5.2 Ecosystem Maturity

ARM's ecosystem advantage extends far beyond compiler support (both architectures have mature GCC and LLVM toolchains). ROS 2 provides Tier 1 ARM64 support with pre-built binary packages; RISC-V has no official tier and requires source compilation (ROS, 2026). ARM's library ecosystem—CMSIS-NN for neural networks, CMSIS-DSP for signal processing, Kleidi libraries for ML/CV optimization—has no RISC-V equivalent. The debugging and profiling infrastructure for ARM (Keil, IAR, Arm Development Studio) vastly exceeds what is available for RISC-V, where OpenOCD and GDB remain the primary tools (RunTime Recruitment, 2024).

Safety certification represents perhaps the starkest gap. ARM Cortex-R processors have achieved ISO 26262 ASIL D and IEC 61508 SIL-3 certification across multiple implementations (ARM, 2026). RISC-V safety certification is emerging through the EMSA5-FS core and the Quintauris RT-Europa consortium (Infineon, Bosch, NXP, Qualcomm, Nordic) (WESSMAN et al., 2022), which published the first standardized RISC-V profile for safety-critical automotive in 2025—but commercial deployment of safety-certified RISC-V in robotics remains years away.

4.5.3 RISC-V's Open ISA Advantage

RISC-V's compelling advantage lies in ISA customizability. The XpulpV2 extensions in GAP8/GAP9 add specialized dot product, SIMD, and hardware loop instructions that achieve performance impossible on standard ARM cores at equivalent power (GAROFALO et al., 2020). Research at DATE 2020 and ACM HEART 2024 (M et al., 2024) demonstrated that custom RISC-V instructions can reduce clock cycles by 50% and power by 30% for DNN workloads compared to standard implementations.

The economic argument strengthens at scale: ARM licensing costs $1–10 million upfront plus 1–2% per-unit royalties, while RISC-V's open specification carries zero licensing fees (CSIS, 2024). For high-volume robotic products (consumer robots, agricultural drones, warehouse AMRs), these savings compound significantly.

Table 7 summarizes the key comparison dimensions.

Table 7 – Comparative Summary: ARM vs. RISC-V for Robotics

Dimension	**ARM**	**RISC-V**
Market share (2025)	~50–60% embedded	Growing (10.7% CAGR)
CoreMark/MHz	3.4–5.0	2.5–3.5
Best robotics AI	275 TOPS (Jetson Orin)	150 GOPS (GAP9)
ROS 2 support	Tier 1 (binaries)	Source-only
Safety certif.	Mature (ASIL D)	Emerging
Licensing cost	$1M–$10M + royalties	Zero (open ISA)
Custom ISA ext.	Not permitted	Unlimited
Robotics vendors	6,000+ companies	~10–20 groups

4.6 Market Analysis and Industry Landscape

4.6.1 ARM's Commanding Position

ARM's dominance in commercial robotics is effectively unchallenged in the current market. Over 55% of embedded processor units shipped in 2024 use ARM architectures, and this share increases further when considering robotics-specific platforms (INTELLIGENCE, 2024). Every major commercial robot manufacturer relies on ARM at some level: NVIDIA Jetson (Cortex-A78AE) for high-performance AI, STM32 (Cortex-M) for motor control, and Cortex-R for safety-critical subsystems. The NVIDIA Jetson ecosystem alone encompasses 850,000+ developers and 6,000+ companies (NVIDIA, 2026a).

The global edge AI hardware market, of which robotics is a key vertical, was valued at $27.01 billion in 2024 and is projected to reach $269.82 billion by 2032 (33.3% CAGR) (MARKETSANDMARKETS, 2032). Devices consuming 1–3 W account for 80.5% of edge AI hardware by volume, a segment where both ARM and RISC-V compete.

4.6.2 RISC-V's Growing but Strategic Presence

RISC-V's 60 billion manufactured cores are overwhelmingly in non-robotics applications—NVIDIA alone shipped over 1 billion RISC-V cores in 2024 as embedded controllers within GPUs (CSIS, 2024). In robotics specifically, RISC-V's footprint concentrates in three areas: academic nano-drone research (GAP8/GAP9 platform), FPGA-based hardware acceleration (PolarFire SoC, Zynq), and emerging industrial controllers (Renesas, Trinamic). The EU DARE project is developing "Titania," a RISC-V-based AI unit for autonomous European drones and industrial robots, reflecting strategic investment in RISC-V for technological sovereignty (CHRONIS et al., 2025).

RISC-V International has grown to over 4,500 members across 70 countries, and the RISE Project (2023) unites Google, Intel, NVIDIA, Qualcomm, Samsung, and MediaTek around RISC-V software ecosystem development (SIFIVE, 2024). However, CSIS's 2025 assessment concludes that RISC-V will not be competitive with ARM in mainstream consumer robotics for approximately 10 years (CSIS, 2025).

4.7 Challenges and Future Directions

4.7.1 Challenges Facing RISC-V Adoption

The most significant barrier to RISC-V in robotics is ecosystem fragmentation. RISC-V's configurable nature, its greatest technical strength, creates compatibility challenges when different vendors implement different extension combinations (RunTime Recruitment, 2024). The RVA23 Profile, ratified in late 2024, aims to standardize application-class processors, but proliferation of vendor-specific extensions (XpulpV2, Andes matrix extensions, Semidynamics tensor instructions) complicates software portability. No equivalent to ARM's CMSIS-DSP library exists for RISC-V, impacting DSP and ML workloads. ROS 2 lacks pre-built RISC-V packages, requiring hours of source compilation (Microchip Technology, 2024).

4.7.2 ARM's Continued Evolution with ARMv9

ARMv9 introduces capabilities directly relevant to next-generation robotics. Scalable Vector Extension 2 (SVE2) provides variable-length vectors (128–2048 bits) for scalable ML inference on CPUs. The Scalable Matrix Extension (SME/SME2) adds dedicated matrix multiplication instructions, potentially reducing dependence on separate NPUs for smaller models (ARM Newsroom, 2024b). The Cortex-A320, the first ultra-efficiency ARMv9 core for edge AI, delivers 10× ML performance over Cortex-A35, while pairing with the Ethos-U85 NPU enables AI models exceeding 1 billion parameters on-device (ARM Newsroom, 2024a).

4.7.3 Heterogeneous Computing as the Unifying Trend

The most important architectural direction transcends ISA choice entirely. Modern robotic computing increasingly requires heterogeneous SoCs combining CPU, GPU, NPU, FPGA, and potentially neuromorphic elements (TU et al., 2025). Both ARM and RISC-V cores appear in heterogeneous designs: AMD Versal AI Edge combines ARM Cortex-A72 with FPGA and AI engines; Tenstorrent pairs RISC-V Ascalon cores with Tensix AI accelerators.

Kim et al.'s RoSÉ (KIM et al., 2023) infrastructure enables pre-silicon evaluation of complete robotic SoC architectures, while Lin et al. (2023) demonstrated a 28 nm motion-control SoC combining ARM cores with custom accelerators for trajectory planning and obstacle avoidance.

Chiplet architectures represent the next evolution. ARM's Chiplet System Architecture (CSA) and the UCIe standard enable modular, composable SoCs mixing different compute elements. For robotics, this means domain-specific accelerators for SLAM, sensor fusion, and motor control can be combined in custom packages optimized for specific robotic workloads without full custom chip design—a paradigm that benefits both ARM and RISC-V equally.

4.7.4 Neuromorphic Computing for Robotics

Chowdhury et al.'s 2025 survey in Nature Communications Engineering (CHOWDHURY et al., 2025) advocates integrating event-based cameras, spiking neural networks, and neuromorphic hardware (Intel Loihi 2, IBM TrueNorth) for robotic vision. These architectures consume orders of magnitude less power than von Neumann systems for temporal processing tasks, though large-scale integration and standardization remain open challenges.

4.7.5 The Open-Source Hardware Movement

The convergence of open-source hardware (RISC-V), open-source software (ROS 2), and open FPGA toolchains is creating a fully open robotic computing stack for the first time (POLLO et al., 2025). Acceleration Robotics' ROBOTCORE® framework maps ROS 2 computational graphs to RISC-V + FPGA hardware, with open-source IP cores for perception and coordinate transformations (Microchip Technology, 2024).

4.8 Final Remarks

This chapter reveals a robotic processor landscape characterized by ARM's deep commercial entrenchment and RISC-V's accelerating academic innovation. Three key insights emerge from the analysis.

First, the ISA matters less than the SoC architecture for robotic performance. With path lengths within 10% between RISC-V and AArch64 (WEAVER; MCINTOSH-SMITH, 2023), and with AI inference dominated by GPU/NPU/FPGA accelerators rather than CPU cores, the critical design decisions involve heterogeneous integration, memory hierarchy, and accelerator architecture rather than instruction set choice.

Second, ARM and RISC-V occupy complementary niches that are unlikely to fully converge within this decade. ARM's mature ecosystem, safety certifications, and commercial platform availability (Jetson's 275 TOPS) make it the only viable choice for production robotic AI today. RISC-V's custom instruction extensions and zero-royalty model make it optimal for ultra-low-power edge robotics (GAP8/GAP9 at milliwatt levels), high-volume cost-sensitive products, and FPGA-based hardware acceleration research.

Third, the frontier is shifting from architecture selection to architecture composition. Chiplet-based designs, heterogeneous SoCs, and the emerging VLA model paradigm suggest that future robotic systems will combine ARM application cores, RISC-V real-time controllers, specialized NPUs, and FPGA accelerators within unified platforms—making the ARM-versus-RISC-V framing increasingly less relevant than the question of how to optimally orchestrate diverse compute resources for robotic intelligence. The next five years will determine whether RISC-V can close the ecosystem gap sufficiently to challenge ARM's commercial dominance, or whether it will find its lasting contribution as a specialized, customizable component within fundamentally heterogeneous robotic computing architectures.

5

AI Applications for Edge, Fog, and IoT

5.1 Introduction

The convergence of artificial intelligence and distributed computing paradigms has created unprecedented demand for efficient neural network inference at the network edge. IoT Analytics estimated approximately 18.8 billion connected IoT devices in 2024, with projections reaching 21.1 billion by the end of 2025 and 39 billion by 2030 (SINHA, 2025), necessitating intelligent processing capabilities that minimize latency, preserve privacy, and reduce bandwidth requirements. This transformation has positioned processor architecture selection as a critical design decision for edge AI systems.

Two instruction set architectures (ISAs) dominate the embedded AI landscape: ARM, with its established ecosystem spanning microcontrollers to server-class processors, and RISC-V, the open-source ISA gaining momentum through customization and elimination of licensing fees. The ARM architecture powers an estimated 70% of microcontrollers worldwide, with more than 310 billion Arm-based chips shipped cumulatively to date (Arm Ltd., 2025a). Meanwhile, RISC-V has emerged as a compelling alternative, particularly for custom AI acceleration, with commercial implementations from GreenWaves Technologies, SiFive, and Alibaba T-Head demonstrating competitive performance.

This chapter provides a comprehensive analysis of both architectures across three distributed computing paradigms: edge computing, which brings AI inference to endpoint devices; fog computing, which distributes processing across intermediate network nodes; and IoT systems, which operate under extreme resource constraints. The main contributions are 1) a systematic review of RISC-V and ARM architectural features that enable AI acceleration, including vector extensions, custom instructions, and neural processing units, alongside 2) a comparative analysis of performance benchmarks, energy efficiency metrics, and ecosystem maturity across both architectures. The study also 3) examines key software frameworks such as

TensorFlow Lite Micro, CMSIS-NN, and PULP-NN and their respective optimization strategies for each architecture, while 4) identifying emerging trends such as heterogeneous computing, in-memory processing, and neuromorphic approaches.

The remainder of this chapter is organized as follows: Section 5.2 provides architectural background on RISC-V and ARM AI capabilities. Sections 5.3, 5.4, and 5.5 examine applications in edge computing, fog computing, and IoT, respectively. Section 5.6 presents a comparative analysis with quantitative benchmarks. Section 5.7 discusses future directions, and Section 5.8 concludes with key findings.

5.2 ARM and RISC-V

Both ARM and RISC-V architectures have evolved to address the growing demand for efficient artificial intelligence acceleration across edge, embedded, and server-class systems. At a fundamental level, both ecosystems leverage a combination of SIMD/vector extensions, mixed-precision arithmetic, and tightly coupled accelerators to improve the performance and energy efficiency of neural network inference. However, while ARM emphasizes a vertically integrated ecosystem with standardized extensions, RISC-V relies on its architectural openness and extensibility to enable domain-specific instruction set extensions and custom accelerators tailored to specific AI workloads. This distinction shapes trade-offs in programmability, ecosystem maturity, and long-term flexibility for edge AI system designers.

5.2.1 RISC-V Architecture for AI/ML

RISC-V is an open-standard ISA based on reduced instruction set computer (RISC) principles, originally developed at UC Berkeley in 2010. Its modular design philosophy enables custom extensions without licensing constraints, making it particularly attractive for domain-specific AI acceleration (GAROFALO et al., 2020).

The RISC-V Vector Extension (RVV), ratified as version 1.0 in 2021, provides scalable vector processing support for neural network workloads. Unlike fixed-width SIMD architectures, RVV adopts a vector-length agnostic (VLA) programming model, enabling implementations with vector lengths ranging from 32 to 2048 bits while preserving software portability (CAVALCANTE et al., 2020). RVV supports mixed-precision operations (INT8, INT16, INT32, FP16, FP32, and FP64), Length Multiplier (LMUL) for register grouping and increased effective vector width, and predicated execution for efficient per-element masking in irregular computations. The Ara vector processor, implemented in GlobalFoundries 22FDX FD-SOI, achieves greater than 98.5% FPU utilization on matrix operations with energy efficiency exceeding 35 DP-GFLOPS/W (PEROTTI et al., 2022).

Despite the availability of standardized extensions, the open nature of RISC-V further enables the design of deeper, domain-specific customizations. A representative example is the

XpulpNN extension developed at ETH Zurich and University of Bologna which introduces sub-byte SIMD operations for quantized neural networks (QNNs). XpulpNN supports packed 8-bit, 4-bit, and 2-bit data types with load-fused dot-product instructions, achieving a 1.64× improvement in peak MACs per cycle over baseline implementations (GAROFALO et al., 2020). Building on these extensions, the PULP-NN library attains 15.5 MACs/cycle for INT8 inference on an 8-core cluster, corresponding to a 30× speedup compared to ARM CMSIS-NN on comparable STM32 microcontrollers (GAROFALO et al., 2019).

There are several commercial RISC-V processors targeting AI workloads. The GAP9 processor from GreenWaves Technologies combines a 9-core RISC-V cluster with the NE16 neural engine supporting INT2 to INT16 precision. It achieves 0.62ms image classification latency on MLPerf Tiny benchmarks with power consumption of approximately 50mW (FLAMAND et al., 2018). The C908 processor, from Xuantie Series of Alibaba T-Head implements a 9-stage dual-issue in-order pipeline with RVV 1.0, achieving 3.5× faster MLPerf Tiny inference compared to its C906 predecessor. The server-grade C930 targets AI-HPC workloads with greater than 15 SPECint2006/GHz (CHEN et al., 2020a). It is also worth mentioning Gemmini, from UC Berkeley, an open-source systolic array generator supporting weight-stationary and output-stationary dataflows through the RoCC (Rocket Custom Coprocessor) interface, fabricated in TSMC 16nm (GENC et al., 2021).

5.2.2 ARM Architecture for AI/ML

ARM's AI ecosystem spans a wide range of platforms, from ultra-low-power microcontrollers to cloud inference servers, with architectures optimized for each performance tier. These solutions can be broadly grouped into three main categories: the Cortex-M series for TinyML workloads, Ethos neural processing units (NPUs), and SIMD/vector acceleration through NEON and the Scalable Vector Extension (SVE).

NEON is the classic ARM's extension that provides 128-bit SIMD operations across all Cortex-A and Cortex-R processors, supporting vector multiply-accumulate, dot products, and complex arithmetic. The Scalable Vector Extension (SVE/SVE2) in Neoverse processors enables vector lengths from 128 to 2048 bits, with BFloat16 and Int8 matrix multiplication support in AWS Graviton3/4 deployments (CASTELLÓ et al., 2022).

The Cortex-M family forms the foundation for microcontroller-based AI inference. Cortex-M4 and Cortex-M7 processors incorporate DSP extensions, including single-cycle multiply-accumulate (MAC) instructions, enabling efficient fixed-point neural network execution. When combined with CMSIS-NN optimizations, the Cortex-M7 achieves up to 249 MOps/s on convolutional neural network (CNN) inference (LAI; SUDA; CHANDRA, 2018). More recent designs further enhance ML performance: the Cortex-M55 introduces Helium (M-Profile Vector Extension), providing up to a 15× improvement over earlier generations (Arm Ltd., 2019) through 128-bit vector operations supporting 8-, 16-, and 32-bit data types, while the Cortex-M85

integrates Helium with improved branch prediction to deliver up to a 4× AI/ML performance increase compared to the Cortex-M7.

Ethos Neural Processing Units are ARM's dedicated NPU family, which provides orders-of-magnitude performance improvements for neural network workloads. Some details are presented in Table 1. The Ethos-U55 microNPU occupies approximately 0.1mm^2 while providing a 480× ML performance boost when paired with Cortex-M55. The main feature of U55 is 480× ML boost. Similarly, the main feature of U65 is Higher throughput, and the U85 is Transformer support.

Table 8 – ARM Ethos NPU Family Specifications

NPU	**Target**	**Performance**
Ethos U55	MCUs	32-128 MACs
Ethos U65	Edge	2× U55
Ethos U85	IoT/Edge	4 TOPS @1GHz

5.3 Edge Computing Applications

These presented architectural advances enable the offloading of AI inference from centralized servers to resource-constrained edge devices, bringing AI inference capabilities to devices at the network periphery, minimizing latency and bandwidth requirements while preserving data privacy. This section examines architectural optimizations and benchmark results for edge AI deployment.

5.3.1 ML Inference Optimization Techniques

In order to achieve an efficient edge inference, besides co-optimization of model architecture, solutions can employ adjusts in quantization strategy and hardware-specific kernels. Quantization reduces precision from FP32 to INT8 or lower, enabling 2-4× memory reduction and leveraging integer SIMD/MAC units. Post-training quantization (PTQ) provides straightforward deployment, while quantization-aware training (QAT) maintains accuracy within 1% of FP32 baselines (BANBURY et al., 2021). Sub-byte quantization (INT4, INT2) enables further compression with accuracy trade-offs. The main characteristics of quantization are shown in Table 2. In this Table, the last column is accuracy.

Modern edge SoCs integrate heterogeneous compute elements combining CPU cores with dedicated accelerators. One example is Google Coral Edge TPU, which achieves 4 TOPS at approximately 2W for TensorFlow Lite quantized models, with inference latency of approximately 3ms on MobileNetV2 (BALLER et al., 2021). The NVIDIA Jetson Orin Nano Provides 275 TOPS through GPU acceleration at 10-15W, excelling on complex models requiring parallel

Table 9 – Quantization Impact on Model Efficiency

Precision	**Memory**	**Speedup**	**Accura**
FP32 (baseline)	1×	1×	100%
INT8 PTQ	4×	2-4×	99%
INT4	8×	4-8×	95-98%
Binary/Ternary	16-32×	10-58×	90-95%

computation (ALQAHTANI; CHEEMA; TOOSI, 2025). Another example is Hailo-8, which delivers 26 TOPS at 2.5W (10 TOPS/W) through dataflow architecture optimized for CNNs (MERENDA; PORCARO; IERO, 2020).

5.3.2 Comparative Edge AI Benchmarks

Christofas et al. (2023) present a comprehensive comparative evaluation between RISC-V and ARM-based edge AI inference machines equipped with dedicated neural network accelerators. The study examines the Sipeed Maixduino, featuring the Kendryte K210 dual-core RISC-V 64-bit processor with an integrated KPU (Kendryte Processing Unit) capable of 1 TOPS at sub-watt power consumption, against the Raspberry Pi 4B combined with Google's Coral USB Accelerator, which utilizes the Edge TPU achieving 4 TOPS at approximately 2W. The benchmark employed MobileNet v1 architectures with varying depth multipliers (1.0, 0.75, 0.50, 0.25) on the CIFAR-100 dataset for image classification tasks representative of edge computing workloads.

In another study, DeepEdgeBench (BALLER et al., 2021) evaluated multiple edge platforms including Raspberry Pi 4, Google Coral Dev Board, and NVIDIA Jetson Nano across TensorFlow and TensorFlow Lite frameworks. Key findings indicate that Google Coral provides optimal performance for quantized models, while Jetson Nano excels when inference computation exceeds 29.3% of total execution time.

5.3.3 Real-Time AI Processing

Real-time edge AI applications demand deterministic latency guarantees. Research on RISC-V multicore vector processors with Vicuna coprocessors demonstrates compiler-based worst-case execution time (WCET) analysis for INT8 quantized neural networks, enabling formal verification of timing constraints (KIRSCHNER; DUDZIKK; BECKER, 2024).

On the other hand, object detection benchmarks on ARM-based Raspberry Pi platforms (ALQAHTANI; CHEEMA; TOOSI, 2025) reveal a balanced accuracy-latency trade-off with YOLOv8 Nano, the highest mAP but 3-5× higher energy consumption with YOLOv8 Medium, and SSD MobileNet V1 as the most energy-efficient, fastest inference.

5.4 Fog Computing Applications

Fog computing extends cloud capabilities to network edge through distributed processing across intermediate nodes, enabling hierarchical AI workload distribution with optimized latency-energy trade-offs.

Regarding distributed AI processing architectures, the edge-fog-cloud continuum enables flexible AI task allocation based on computational requirements and latency constraints (FIROUZI; FARAHANI; MARINŠEK, 2022). Typical architectures employ three or four tiers: the IoT/Edge Layer comprises Sensor data acquisition, lightweight preprocessing, the Fog Layer contains intermediate inference, model aggregation, latency-critical processing, while the model training, complex inference, long-term storage relies in the cloud layer.

A more in depth discussion on how to distribute responsibilities on AI processing chain can be found in (IFTIKHAR et al., 2023), which provides a comprehensive taxonomy of AI/ML-based resource management for fog/edge computing, analyzing machine learning, deep learning, and reinforcement learning techniques for dynamic task scheduling and resource allocation.

Fog computing significantly reduces end-to-end latency compared to cloud-only processing. Key optimization approaches include task offloading, federated learning in fog, and architecture-specific fog optimizations.

Deep reinforcement learning (DRL) enables adaptive offloading decisions balancing computation, transmission, and queuing delays. The Optimized Latency Fog Computing (OLFC) framework combines fuzzy inference with neural network evolution strategies, achieving 52% latency reduction compared to baseline fog models in healthcare IoT applications (SHUKLA et al., 2019).

For Federated Learning in Fog, the framework FedHealthFog (TRIPATHY et al., 2024) elevates fog nodes as local aggregators in federated learning, demonstrating 87.01% reduction in communication latency, 57.98% reduction in energy consumption, and privacy preservation through decentralized model training. Alternatively, the FLight framework (ZHU; GOUDARZI; BUYYA, 2024) extends FogBus2 for asynchronous federated learning with worker selection policies optimizing training efficiency across heterogeneous fog nodes.

Processor architecture selection impacts fog node efficiency. ARM-based fog nodes benefit from mature container orchestration (Docker/Kubernetes) and established ML frameworks. Cortex-A series processors with NEON optimization provide predictable performance for CNN inference. RISC-V-based fog nodes, in turn, offer customization potential for specific workloads. The ability to add domain-specific extensions enables optimized matrix operations and reduced-precision inference at fog layer without general-purpose overhead.

Dynamic task allocation using fuzzy logic (JIN; REZAEIPANAH, 2025) provides

real-time scheduling for fog-cloud systems, adapting to changing network conditions through membership functions for latency, resource utilization, and deadline constraints.

5.5 IoT and TinyML Applications

TinyML enables machine learning inference on devices operating under 1mW with memory constraints of 32KB-1MB SRAM, targeting the billions of deployed microcontrollers worldwide (DAVID et al., 2021).

5.5.1 Ultra-Low Power AI Inference frameworks

TensorFlow Lite Micro (TFLM) is an interpreter-based inference framework designed for ultra-low-power and resource-constrained devices. It avoids dynamic memory allocation and relies on static memory planning, enabling deterministic execution across heterogeneous embedded platforms (DAVID et al., 2021). When integrated with CMSIS-NN, TFLM achieves up to a 4.6× reduction in runtime and a 4.9× improvement in energy efficiency on ARM Cortex-M processors. Experimental results on a 96 MHz Cortex-M4 report 4,857.7 k cycles for the optimized Visual Wake Words workload and 36.4 k cycles for optimized Keyword Spotting inference, while interpreter overhead remains below 0.1% for long-running models.

MCUNet adopts a joint optimization approach that co-designs the neural network architecture and the inference engine through TinyNAS and TinyEngine, respectively. This methodology enabled the first implementation to surpass 70% Top-1 accuracy on ImageNet using a commercial microcontroller, while requiring 3.5× less SRAM and 5.7× less Flash memory compared to quantized MobileNetV2. These gains are achieved through a memory scheduling strategy that exploits network topology awareness rather than conventional layer-wise allocation, significantly improving memory efficiency for ultra-low-power inference.

The PULP-NN library (GAROFALO et al., 2019) provides highly optimized neural network kernels for RISC-V PULP clusters, achieving up to 15.5 MACs/cycle for INT8 inference on an 8-core configuration and a 63× performance improvement over a sequential RV32IMC baseline. It supports aggressive quantization down to INT4, INT2, and INT1, enabling efficient execution under strict memory and energy constraints, and demonstrates energy efficiency of up to 24 GMAC/s/W on the GAP8 platform, making it well suited for ultra-low-power edge AI workloads.

Table 10 presents a comparative overview of representative embedded machine learning frameworks, highlighting their primary targets and supported architectures. Each framework emphasizes a distinct design objective: TensorFlow Lite Micro prioritizes portability across heterogeneous embedded platforms; CMSIS-NN focuses on architecture-specific SIMD optimization for ARM Cortex-M processors; PULP-NN exploits sub-byte quantization and parallel execution on RISC-V PULP clusters; TinyEngine emphasizes code generation tailored to resource-

constrained microcontrollers; and STM32Cube.AI provides automated model conversion and deployment for STM32 devices. Together, these frameworks illustrate the diverse optimization strategies adopted to enable efficient AI inference in embedded and edge environments.

Table 10 – Embedded ML Framework Comparison

Framework	**Target**	**Architecture**
TFLite Micro	Universal	ARM, RISC-V
CMSIS-NN	Cortex-M	ARM only
PULP-NN	PULP cluster	RISC-V
TinyEngine	MCUs	ARM
STM32Cube.AI	STM32	ARM

5.5.2 TinyML Application Case Studies

Several application case studies illustrate the practical deployment of ultra-low-power AI inference on embedded and edge platforms. Always-on keyword spotting exemplifies workloads operating under strict energy constraints: on a Cortex-M4, wake-word detection achieves 91% accuracy with 9 ms latency and an approximate 240 KB Flash footprint, enabling continuous operation in the sub-milliwatt range (BANBURY et al., 2021). In industrial Internet of Things (IIoT) scenarios, anomaly detection for predictive maintenance relies on lightweight models such as autoencoders and isolation forests; sensor fusion across vibration, temperature, and audio modalities yields F1-scores between 0.92 and 0.94 using quantized TinyML models (CHEVTCHENKO et al., 2023). Visual wake words further demonstrate the feasibility of always-on vision at the edge by triggering higher-power processing only when relevant events are detected, with the MCUNetV2 model family achieving over 90% accuracy within a 500 KB SRAM budget, thereby validating practical always-on visual intelligence.

5.5.3 Hardware Platform Comparison

Table 11 provides a comparative overview of representative TinyML hardware platforms, highlighting key architectural and computational characteristics relevant to ultra-low-power AI inference. The selected platforms span both ARM- and RISC-V–based designs and illustrate the trade-offs between on-chip memory capacity and available compute throughput. In addition to the parameters listed, these platforms operate under distinct power envelopes that further influence deployment choices: typical power consumption is approximately 10 mW for STM32L4, 100 mW for STM32H7, and 50 mW for GAP8, while the RA8P1 supports a wider and application-dependent power range due to its integration of a Cortex-M85 processor with an Ethos-U55 NPU. Together, these characteristics contextualize the performance and efficiency results discussed in the subsequent benchmark analysis.

Table 11 – TinyML Hardware Platform Specifications

Platform	**Arch.**	**SRAM**	**MACs/cyc**
STM32L4	Cortex-M4	128KB	~1
STM32H7	Cortex-M7	1MB	~2
GAP8	RISC-V	512KB	15.5 (8-core)
RA8P1	M85+U55	2MB	256 GOPS

5.6 Comparative Analysis

MLPerf Inference (REDDI et al., 2020) provides standardized benchmarking across four orders of magnitude in power and five orders in performance. MLPerf Tiny specifically targets microcontroller-class devices with four benchmarks: keyword spotting, visual wake words, image classification, and anomaly detection.

The AI-RISC processor (VERMA, 2022) demonstrates 2.45× average outperformance versus ARM Cortex-A72 on complete MLPerf Tiny inference benchmarks through tightly-coupled AI accelerator integration, with 17.63× speedup on vector-matrix multiply kernels.

An optimization analysis of TensorFlow Lite (LOUIS et al., 2019) highlights architecture-dependent performance advantages across different model types. For convolution-dominated workloads, RISC-V implementations with vector optimizations achieve up to an 8× reduction in instruction count compared to baseline executions. In contrast, LSTM-dominated models benefit more from ARM-specific optimizations, particularly for block-based vector–matrix multiplication. Furthermore, RISC-V designs employing wider 256-bit vector registers demonstrate an additional 2× reduction in instruction count, underscoring the impact of vector width and ISA-level optimizations on inference efficiency.

Table 12 shows a comparison of energy efficiency on different platforms. In this case, the Hailo-8 platform has a TOPS = 26, and Axelera Metis has a TOPS of 214. Remind that TOPS means Tera Operations Per Second, and the GAP8 platform is PULP-NN.

Table 12 – Energy Efficiency Comparison (TOPS/W)

Platform	**Archi**	**TOPS/W**
Hailo-8	ASIC	~10
Axelera Metis	In-memory	~15
Open-RIMC	RISC-V	5.6-14.0
GAP8	RISC-V	24 GMAC/s/W
Ara (RVV 1.0)	RISC-V	35 GFLOPS/W

Comprehensive analysis by (SUÁREZ; ALMEIDA; BLANCO, 2024) confirms that RISC-V demonstrates lower average power consumption than ARM, though this does not automatically translate to superior performance-per-watt across all workloads.

The maturity of the ecosystem and toolchain differs between ARM and RISC-V, reflecting their distinct design philosophies and commercialization strategies. ARM benefits from a highly mature software stack, including CMSIS-NN with Helium and NEON optimizations, well-established integration with Arm NN and TensorFlow Lite, extensive commercial support and documentation, and production-proven deployment of Ethos NPUs. In contrast, RISC-V emphasizes openness and flexibility, offering license-free deployment, the ability to introduce custom ISA extensions without intellectual property constraints, and rapidly expanding support across major machine learning frameworks such as TensorFlow Lite, PyTorch, and ONNX Runtime, complemented by commercial-grade development tools including Synopsys ASIP Designer and Codasip Studio.

5.6.1 Summary Comparison

Table 13 shows the main aspects of RISC-V and ARM architectures. Here, we can see aspects such as (1) power consumption, (2) inference speed, (3) custom acceleration, (4) toolchain maturity, (5) licensing costs, and (6) vector processing. The values it reports are indicative, best-case ("up to") trends synthesised from the studies surveyed above rather than fixed constants; the precise margins depend on the specific core generation and on any custom extensions or accelerators integrated into a given design.

Table 13 – RISC-V vs ARM Architecture Comparison

Aspect	**RISC-V**	**ARM**
(1)	4× lower	Baseline
(2)	Baseline	Up to 15× faster
(3)	High flexibility	Limited
(4)	Developing	Mature
(5)	None	Established fees
(6)	Scalable (RVV)	Fixed (NEON 128-bit)

5.7 Emerging Trends and Future Directions

The optimal architecture for edge AI increasingly integrates general-purpose CPU cores with domain-specific accelerators to balance flexibility and performance. In the RISC-V ecosystem, the RoCC interface enables tight coupling of custom coprocessors, while ARM platforms leverage Ethos NPUs as turnkey acceleration solutions. Representative implementations include the Mobileye EyeQ Ultra, which combines twelve RISC-V cores with dedicated neural accelerators, the Renesas RA8P1, integrating a Cortex-M85 core with an Ethos-U55 NPU delivering up to 256 GOPS, and a heterogeneous system-on-chip developed at UC Berkeley that combines BOOM, Hwacha, and Gemmini accelerators, achieving an energy efficiency of 106.1 GOPS/W (GONZALEZ et al., 2021).

In-memory computing (IMC) eliminates the von Neumann bottleneck by performing matrix operations within memory arrays. IBM's 64-core phase-change memory (PCM) chip achieves 63.1 TOPS peak with 9.76 TOPS/W for 8-bit matrix-vector multiplication (THOMPSON, 2023). RRAM-based implementations demonstrate 10-100 TOPS/W for reduced-precision operations, suggesting transformative efficiency gains for edge AI inference.

Event-driven neuromorphic processors offer 10- 1000× energy efficiency improvements for sparse, temporal workloads. Intel Loihi 2 and BrainChip Akida demonstrate practical implementations, with software frameworks (PyNN, Lava) enabling accessible development (MUIR; SHEIK, 2025). Integration with RISC-V or ARM control processors provides hybrid architectures combining conventional and spiking neural network capabilities.

5.7.1 RISC-V AI Extension Roadmap

The RISC-V community is actively developing and standardizing instruction set extensions to support AI and signal-processing workloads. The Accelerated Matrix Extension (AME), currently under development, targets efficient execution of dense matrix operations, while the ratified Packed-SIMD extension enables sub-word parallelism for low-precision data types. In addition, the ratified P extension provides DSP- and SIMD-oriented operations optimized for signal processing, collectively strengthening RISC-V's capabilities for efficient AI inference on embedded and edge platforms.

Commercial implementations including SiFive Intelligence X390 (1024-bit VLEN), Alibaba C930, and GreenWaves GAP10 demonstrate accelerating adoption trajectory.

5.7.2 ARM AI Roadmap

ARM continues to expand its AI capabilities across the entire computing continuum, from ultra-low-power embedded systems to cloud-scale platforms. The Ethos-U85 introduces native support for transformer-based networks, while the Cortex-X series targets high-performance edge inference workloads. At the software level, KleidiAI enhances integration with PyTorch and ExecuTorch for optimized deployment on Neoverse-based cloud systems. In parallel, ARM's Total Compute solutions provide tightly integrated CPU, GPU, and NPU platforms, enabling end-to-end optimization of AI workloads across heterogeneous architectures.

5.8 Final Remarks

This comprehensive survey examined RISC-V and ARM architecture applications for artificial intelligence across fog computing, edge computing, and IoT domains. Our analysis of 35 academic sources from 2020-2025 reveals several key findings:

Architecture selection is fundamentally driven by application requirements. RISC-V demonstrates up to four times superior power efficiency and enables custom AI acceleration without licensing constraints, making it well-suited for power-constrained deployments and domain-specific optimization. In contrast, ARM delivers up to fifteen times faster inference for complex networks through a mature and well-integrated toolchain, favoring performance-critical applications that demand rapid time-to-market. These magnitudes are indicative best-case figures rather than definitive architectural verdicts, and they depend heavily on the specific core generation, degree of customisation, and workload involved. The fifteenfold figure, in particular, reflects the generational uplift of ARM's Helium vector extension (Arm Ltd., 2019) over earlier M-profile cores rather than a direct ARM-versus-RISC-V measurement, while the fourfold energy-efficiency advantage is realised primarily by purpose-built RISC-V accelerators and custom extensions rather than by general-purpose cores. This distinction matters: in a like-for-like comparison of general-purpose SoCs, (SUÁREZ; ALMEIDA; BLANCO, 2024) found that the RISC-V platform drew lower average power than its ARM counterparts yet did not achieve a superior performance-per-watt ratio, and—crucially—attributed the observed differences to device-specific characteristics and the comparatively nascent state of RISC-V implementations rather than to the underlying ISA, anticipating that these gaps will narrow as the ecosystem matures. The central disparity between the two architectures is therefore better understood as one of ecosystem and implementation maturity than of inherent architectural capability; as both architectures and their toolchains evolve, these margins are expected to shift.

The maturity of the software ecosystem remains a key differentiating factor between ARM and RISC-V platforms. ARM benefits from production-ready deployment paths through CMSIS-NN, tight TensorFlow Lite integration, and established Ethos NPU support. Although the RISC-V ecosystem is evolving rapidly with initiatives such as PULP-NN, TensorFlow Lite ports, and emerging commercial tools, limitations persist in terms of optimization depth and comprehensive documentation.

Quantization techniques play a central role in enabling efficient edge AI deployment. INT8 quantization combined with quantization-aware training achieves accuracy within one percent of FP32 models while enabling two- to fourfold reductions in memory footprint and substantial inference speedups on both architectures. Sub-byte quantization formats such as INT4 and INT2 further improve compression, offering practical benefits for TinyML applications with acceptable accuracy trade-offs.

Fog computing architectures significantly reduce end-to-end inference latency in distributed AI systems. Federated learning and distributed inference across fog nodes achieve latency reductions ranging from 50 to 87 percent when compared to cloud-only processing, while deep reinforcement learning techniques enable adaptive and efficient task offloading decisions.

Heterogeneous architectures represent the most effective path forward for scalable and efficient edge AI. The combination of RISC-V or ARM CPU cores with domain-specific

accelerators, including NPUs, systolic arrays, and in-memory computing units, provides a balanced solution that integrates flexibility, energy efficiency, and high performance across diverse deployment scenarios.

Future research directions include standardized RISC-V AI extensions, neuromorphic-conventional hybrid architectures, and continued advancement of in-memory computing for transformative efficiency improvements. As edge AI applications proliferate across healthcare, industrial IoT, autonomous systems, and smart infrastructure, both RISC-V and ARM architectures will play essential roles in enabling intelligent processing at the network edge.

6

Security Mechanisms for ARM and RISC-V

6.1 Introduction

Security in processor architectures for safety-critical robotics systems encompasses a multi-layered set of challenges that extends well beyond traditional access control. A modern autonomous robot—whether a warehouse logistics platform, a humanoid manipulator, or an industrial collaborative arm—executes heterogeneous workloads with different trust levels, real-time requirements, and failure consequences on a single System-on-Chip (SoC). Hard real-time motor controllers must be isolated from best-effort perception pipelines. Cryptographic keys for fleet authentication must be protected even from a compromised operating system. Firmware updates must be verified before execution. The hardware must enforce these boundaries with mechanisms that are certifiable under ISO 26262 (International Organization for Standardization, 2018) and IEC 61508 (International Electrotechnical Commission, 2010).

ARM and RISC-V approach these challenges with fundamentally different philosophies. As detailed in Chapter 2, ARM provides a vertically integrated security ecosystem—TrustZone, CCA, PAC, MTE—with mature tooling and extensive deployment history, but closed hardware implementations that require trust in the IP vendor. RISC-V provides transparent, auditable building blocks—PMP, M-mode isolation, open-source TEE frameworks—with the advantage of independent verification but the challenge of ecosystem fragmentation and less mature hardware mechanisms.

This chapter provides a structured comparison of the two architectures in security capabilities across six layers: hardware root of trust (Section 6.2), privilege isolation and memory protection (Section 6.3), trusted execution environments (Section 6.4), memory safety mechanisms (Section 6.5), side-channel and speculative execution defenses (Section 6.6), and supply-chain trust and auditability (Section 6.7). The analysis focuses specifically on the security implications for mixed-criticality robotic platforms.

6.2 Hardware Root of Trust

Every secure system requires a trust anchor—a component whose integrity is assumed and upon which all other security properties depend. The root of trust answers the question: when the processor powers on, how does the system know the first instruction it executes is legitimate?

6.2.1 ARM: TrustZone and Secure Boot

ARM's root of trust is built around TrustZone (ALVES; FELTON, 2004; PINTO; SANTOS, 2019) and the EL3 Secure Monitor. The processor exits reset in Secure EL3, executing immutable on-chip ROM that initiates a chain of trust: BL1 (ROM) verifies BL2, which verifies BL31 (the EL3 runtime firmware, typically ARM Trusted Firmware / TF-A), BL31 verifies BL32 (the secure-world OS, such as OP-TEE) and BL33 (the normal-world bootloader), and each subsequent stage authenticates the next before handing off control (Arm Ltd., 2009).

ARM formalized this architecture through the Platform Security Architecture (PSA) (PSA Joint Stakeholders, 2021), which defines three certification levels: Level 1 (questionnaire-based), Level 2 (lab evaluation of secure boot and key storage), and Level 3 (full penetration testing). PSA provides OEMs with a standardized evaluation framework for ARM-based SoC security posture.

For Cortex-M microcontrollers, Trusted Firmware-M (TF-M) (Trusted Firmware Project (Linaro), 2023) provides a reference Secure Processing Environment (SPE) that boots first, validates the non-secure firmware, then hands off to the application. The Secure Attribution Unit (SAU) and Implementation-Defined Attribution Unit (IDAU) enforce the secure/non-secure memory map from the first cycle.

6.2.2 RISC-V: M-mode and Open-Source Roots of Trust

RISC-V's root of trust is architecturally simpler but less standardized. The processor exits reset in M-mode (Machine mode)—the highest privilege level. The M-mode firmware (typically OpenSBI (RISC-V International, 2019) or a vendor-specific boot ROM) serves as the trust anchor. Unlike ARM, RISC-V has no orthogonal security dimension (Secure/Non-Secure worlds) in the base specification; M-mode is both the highest privilege level and the trust anchor (LU, 2021).

Several open-source and commercial projects address this gap:

OpenTitan (MEZA et al., 2023; CIANI et al., 2024) is a standalone open-source silicon root of trust built around the Ibex RISC-V core (RV32IMC, from lowRISC/ETH Zurich). It integrates AES, HMAC-SHA256, CSRNG, RSA/ECC accelerators, OTP fuse controller, and lifecycle management. OpenTitan boots before the main application processor and provides measured boot and attestation services across a physically separate trust boundary. Its entire RTL is publicly auditable under the Apache 2.0 license.

Table 14 – Hardware Root of Trust (RoT) Ecosystems

Feature	**ARM Architecture**	**RISC-V Architecture**
Primary trust anchor	TrustZone and EL3	M-mode firmware (OpenSBI)
Standardization	PSA (Platform Security Architecture) Certified	Fragmented / independent; emerging standards like CoVE
Key projects / IP	TF-A (Trusted Firmware) and TF-M	OpenTitan, Caliptra, Keystone
Auditability	Closed-source hardware; requires trust in IP vendor	Open-source RTL; independently verifiable (Ibex core)
Certification path	Structured levels (PSA L1–L3) based on lab evaluations	Path through formal verification of open RTL (SAIL ISA spec)

Caliptra (Open Compute Project (AMD, Google, Microsoft, NVIDIA), 2022), a joint project from Google, Microsoft, AMD, and NVIDIA under the Open Compute Project, provides a RISC-V-based root-of-trust IP block designed for datacenter SoC integration, offering measured boot, firmware integrity verification, and hardware-backed device identity.

Keystone (LEE et al., 2020a) from MIT and UC Berkeley uses RISC-V's PMP (Physical Memory Protection) to create isolated enclaves at the M-mode level without requiring custom hardware extensions. The Security Monitor (SM), approximately 10K lines of C, manages enclave lifecycle, PMP configuration, and attestation.

SiFive WorldGuard (SiFive, 2021b) provides multi-domain security partitioning by tagging bus transactions with a World ID, supporting more than two isolation domains—exceeding ARM TrustZone's binary Secure/Non-Secure split.

6.2.3 Comparative Assessment

ARM's root-of-trust ecosystem is mature and standardized (PSA Certified), with vertical integration from boot ROM through secure OS. However, the hardware implementation is closed, requiring trust in ARM and the silicon vendor. RISC-V offers genuine supply-chain diversity—system designers can choose OpenTitan (open-source, Google/lowRISC-governed), Caliptra (multi-vendor consortium), Keystone (academic, MIT/Berkeley), SiFive Shield (commercial), or build their own root of trust from open-source cores. This diversity provides supply-chain independence but creates a fragmented ecosystem without ARM's unified certification framework.

Table 14 summarizes the main aspects of hardware root of trust (RoT) ecosystems.

6.3 Privilege Isolation and Memory Protection

Privilege isolation answers the question central to mixed-criticality robotics: how can a hard real-time motor controller and a Linux-based perception pipeline coexist on the same SoC without mutual interference?

6.3.1 ARM Exception Levels and Memory Protection

ARM AArch64 defines four exception levels (EL0–EL3), each entered only through controlled entry points (SVC, HVC, SMC instructions) (Arm Ltd., 2024a). The MMU provides page-granularity (4KB minimum) virtual-to-physical translation with per-page permission bits. Stage-2 translation, controlled by EL2, adds hypervisor-enforced isolation: the hypervisor controls which physical memory each VM can access via Independent Physical Address (IPA) mapping.

ARM's MPU (Memory Protection Unit), used in Cortex-R/M profiles, provides region-based access control without address translation. The MPU offers constant-time, deterministic access checks—critical for WCET-analyzable real-time systems. The Cortex-R82 uniquely supports both MMU and MPU, enabling Linux (via MMU) and bare-metal RTOS (via MPU) on the same core.

The System Memory Management Unit (SMMUv3) (Arm Ltd., 2020a) extends isolation to DMA-capable devices. Each device, identified by a Stream ID, has its own page tables controlled by the hypervisor, enabling safe device passthrough to VMs.

6.3.2 RISC-V Privilege Modes and PMP

RISC-V defines three privilege modes: M-mode (Machine), S-mode (Supervisor), and U-mode (User) (WATERMAN; ASANOVIĆ; RISC-V Foundation, 2021). The H-extension adds VS-mode and VU-mode for virtualization, with two-stage address translation analogous to ARM's EL1/EL2 model (SÁ; MARTINS; PINTO, 2022).

PMP provides region-based physical address access control configured by M-mode (WATERMAN; ASANOVIĆ; RISC-V Foundation, 2021; CHEANG et al., 2022). With typically 16 entries (the specification supports up to 64), PMP defines address ranges with R/W/X permissions using TOR (Top of Range) or NAPOT (Naturally Aligned Power-of-Two) matching modes. PMP applies to S-mode and U-mode; M-mode is unrestricted unless the Lock bit is set.

The enhanced PMP (ePMP) extension (RISC-V International, 2021b) introduces the Machine Mode Lockdown (MML) bit in the `mseccfg` CSR, subjecting M-mode itself to PMP restrictions. This provides defense-in-depth by enforcing least-privilege even at the highest privilege level—a property ARM achieves through software discipline in TF-A rather than hardware enforcement.

For application-class processors, RISC-V supports Sv39 (39-bit VA, 3-level page tables), Sv48 (48-bit VA, 4-level), and Sv57 (57-bit VA, 5-level) virtual memory modes. Physical Memory Attributes (PMA) are configured independently of page tables, typically in M-mode—unlike ARM, where memory type attributes are encoded in the PTE itself. RISC-V's approach prevents a compromised OS kernel from misconfiguring memory attributes for device registers.

6.3.3 DMA Isolation: IOPMP and IOMMU

DMA-capable peripherals can bypass CPU-level memory protection. ARM addresses this with the SMMUv3, which provides page-table-based translation and access control for device transactions with two-stage support for virtualization (Arm Ltd., 2020a).

RISC-V provides two complementary mechanisms: IOPMP (RISC-V IOPMP Task Group, 2024) for region-based DMA access control (analogous to PMP for devices), and the RISC-V IOMMU (JEZNACH; PATEL et al., 2024) for full page-table-based device translation with two-stage support. IOPMP targets embedded systems with known, fixed peripheral sets; the IOMMU targets application-class systems requiring per-page device isolation and VM passthrough.

6.3.4 Cache Partitioning

Shared caches create both performance interference and security side-channel risks in multi-domain systems. ARM addresses this with MPAM (Memory Partitioning and Monitoring) (Arm Ltd., 2024c; ZINI; CASINI; BIONDI, 2023), which assigns cache partitions and memory bandwidth allocations per VM or priority level.

RISC-V lacks a standardized cache partitioning mechanism. The current approach is software-managed cache coloring (MARTINS et al., 2020), implemented by hypervisors such as Bao (MARTINS et al., 2020), which assigns physical page ranges to VMs based on the cache indexing function. Some vendors (SiFive, T-Head) provide proprietary cache partition registers, but no standard exists.

6.3.5 Heterogeneous SoC Memory Architecture

A robotics SoC typically combines small RV32 real-time cores (M-mode only, with Tightly Coupled Memory for deterministic access) and larger RV64 application cores (with full MMU, H-extension, and DRAM access). Both core types can access shared DRAM and on-chip SRAM, but the 32-bit core is limited to the 4 GB physical address space. Inter-core communication uses shared memory regions configured as uncacheable via PMA, with signaling via inter-processor interrupts (IPIs) through the CLINT/ACLINT. PMP on the RV32 core, stage-2 tables on the RV64 core, and IOPMP at the bus level provide layered defense-in-depth isolation.

Table 15 focuses on how the architectures isolate different workloads, such as real-time controllers and general-purpose pipelines.

Table 15 – Privilege Isolation and Memory Protection

Mechanism	**ARM Solution**	**RISC-V Solution**
Privilege levels	Four exception levels (EL0–EL3)	Three modes (U-mode, S-mode, M-mode)
Isolation granularity	MMU (page-based) and MPU (region-based)	PMP (Physical Memory Protection)
Enhanced isolation	Stage-2 translation (EL2 / hypervisor)	H-extension (VS/VU modes)
DMA protection	SMMUv3 (Stream-ID-based page tables)	IOPMP (region-based) and RISC-V IOMMU
Cache management	MPAM (Partitioning and Monitoring)	Software-managed cache coloring (Bao hypervisor)

6.4 Trusted Execution Environments

TEEs create environments where sensitive code and data are protected even from a compromised operating system. The core threat model assumes the OS kernel is hostile—a realistic assumption given the Linux kernel's millions of lines of code and exposure to network-delivered exploits.

6.4.1 ARM TrustZone

TrustZone splits the system into Secure and Non-Secure worlds, with every bus transaction carrying a hardware-driven NS (Non-Secure) bit on the AMBA bus fabric (ALVES; FELTON, 2004; PINTO; SANTOS, 2019). World transitions occur exclusively through the Secure Monitor at EL3 via the SMC instruction, ensuring controlled entry points.

The Secure world typically runs OP-TEE, providing a trusted kernel at S-EL1 and isolated Trusted Applications (TAs) at S-EL0 via a GlobalPlatform-compliant TEE Internal API. TrustZone's limitations include the binary two-world model (addressed by S-EL2 Secure Partitions via FF-A (Arm Ltd., 2025b)), world-switch overhead (2–5 µs hardware, 20–100 µs full software stack), and TCB size (OP-TEE: ~200K+ lines of C). Known vulnerabilities in TrustZone-assisted TEE systems are systematically catalogued by Cerdeira *et al.* (CERDEIRA et al., 2020).

6.4.2 ARM CCA (Confidential Computing Architecture)

CCA (LI et al., 2022) introduces a fourth security state, the Realm, alongside Secure, Non-Secure, and Root. The Granule Protection Table (GPT) tags every physical page with its world assignment; hardware blocks cross-world accesses. Realm memory is encrypted in DRAM, protecting against physical attacks. The Realm Management Monitor (RMM) at R-EL2 manages Realm lifecycle and attestation. CCA enables multi-stakeholder robotic platforms where an OEM's motor control algorithm, a third-party perception neural network, and a fleet operator's logistics software each run in isolated Realms that even the hypervisor cannot read.

CCA entry/exit costs 10–20 µs, roughly 3–5× more expensive than a TrustZone world switch, due to GPT checks, memory encryption engine engagement, and RMM involvement.

6.4.3 RISC-V Keystone

Keystone (LEE et al., 2020a) creates ephemeral PMP-based enclaves managed by a Security Monitor (SM) in M-mode (~10K lines of C). The SM dynamically reconfigures PMP entries on every context switch between the untrusted OS and an enclave, granting enclave memory access and revoking OS access. Keystone's TCB is dramatically smaller than TrustZone's, making formal verification and safety certification more tractable. The limitation is PMP entry count: with 16 entries, 3–4 simultaneous enclaves are practical. The H-extension-based evolution (using stage-2 tables instead of PMP for enclave isolation) eliminates this constraint (KÜHNE et al., 2025).

6.4.4 MultiZone

MultiZone (Hex Five Security) (Hex Five Security, Inc., 2020) partitions the system into isolated zones at boot time using PMP, with a nanokernel in M-mode (<6KB) mediating all inter-zone communication via message passing. Its minimal TCB makes it the most certifiable TEE in this comparison. MultiZone's fixed-partition model maps directly to multi-domain robotic architectures: one zone for safety supervision, one for motor control, one for perception.

6.4.5 RISC-V CoVE

CoVE (SAHITA et al., 2023) is RISC-V's specification for confidential computing, providing CCA/TDX/SEV-SNP-equivalent functionality. The TEE Security Manager (TSM) in M-mode manages confidential VMs (TVMs) using PMP and H-extension stage-2 tables. Unlike CCA, CoVE does not introduce new hardware security states; it builds on existing privilege architecture. Memory encryption is supported but not mandated—the most significant gap relative to CCA, which requires it.

6.4.6 TEE Comparison

Table 16 summarizes the key TEE characteristics relevant to safety-critical robotics. For safety certification, TCB size directly impacts certification feasibility: certifying 6KB of MultiZone to ASIL D is tractable; certifying 200K+ lines of OP-TEE is impractical without massive investment. None of these TEEs are suitable for per-sample data-path security at kHz rates; hardware crypto accelerators (ARM CryptoCell, RISC-V Zk* extensions (MARSHALL et al., 2022)) provide inline cryptographic operations without TEE transition overhead.

Table 16 – TEE Architecture Comparison for Robotics

Property	**TZ**	**CCA**	**KS**	**MZ**
TCB size (lines)	~200K	>200K	~10K	<6K
Max domains	2[a]	Many	3–4[b]	4–8[b]
Switch cost (µs)	2–5	10–20	5–15	5–15
Memory encryption	No[c]	Yes	No	No
HW requirements	TZ	v9+RME	PMP	PMP
Open-source	Partial	No	Yes	SDK

TZ=TrustZone, CCA=Arm CCA, KS=Keystone, MZ=MultiZone
[a]More with S-EL2. [b]PMP-limited. [c]Vendor-specific options exist.

6.5 Memory Safety Mechanisms

Memory safety vulnerabilities—buffer overflows, use-after-free, type confusion—account for approximately 60–70% of exploitable vulnerabilities in large C/C++ codebases, a statistic consistently confirmed by Microsoft (MILLER, 2019) and Google (Vander Stoep, 2022).

6.5.1 ARM: Hardware Enforcement

ARM has invested heavily in hardware mechanisms that catch memory safety violations at runtime with minimal performance overhead:

PAC (Pointer Authentication Codes, ARMv8.3) (LILJESTRAND et al., 2019) computes a truncated cryptographic MAC in the unused upper bits of 64-bit pointers using a hardware key. On authentication failure, the pointer is corrupted, causing a fault on dereference. PAC adds approximately 1–2% overhead and defends against ROP/JOP attacks. The MAC width (7–16 bits) allows probabilistic brute-force forgery; FPAC (ARMv8.6) mitigates this by faulting immediately on authentication failure.

MTE (Memory Tagging Extension, ARMv8.5) (SEREBRYANY et al., 2018; SEREBRYANY, 2019) assigns 4-bit tags to every 16-byte memory granule and to pointers. On every memory access, the hardware compares pointer and memory tags; mismatches indicate spatial (buffer overflow) or temporal (use-after-free) violations. MTE operates in synchronous mode (3–5% overhead, precise faults) or asynchronous mode (1–2% overhead, imprecise). Google has deployed MTE on Android (Pixel devices) for heap hardening.

BTI (Branch Target Identification, ARMv8.5) marks valid indirect branch targets with `BTI` instructions; branches landing on non-BTI instructions fault. Provides coarse-grained forward-edge CFI at essentially zero performance cost.

GCS (Guarded Control Stack, ARMv9.4) provides a hardware-protected shadow call stack. Every `BL` pushes the return address to both the regular and guarded stacks; `RET` compares them. GCS pages cannot be written by normal store instructions. GCS provides deterministic

return address protection without PAC's probabilistic forgery risk.

These mechanisms compose as defense-in-depth: PAC protects all pointer types, BTI restricts branch targets, MTE catches spatial/temporal errors, and GCS provides deterministic return address integrity.

6.5.2 RISC-V: Software-Oriented with Emerging Hardware

The RISC-V base ISA contains no memory safety mechanisms comparable to PAC, MTE, BTI, or GCS. The ratified extensions **Zicfilp** (Landing Pad) and **Zicfiss** (Shadow Stack) (RISC-V International, 2024c) provide equivalents to BTI and GCS respectively—coarse-grained forward-edge CFI and deterministic return address protection. These give RISC-V parity with ARM on control flow integrity but leave the data protection gap (PAC and MTE equivalents) unaddressed.

Vendor-specific solutions include Andes StackSafe (hardware stack bounds checking) and Dover Microsystems CoreGuard (DHAWAN et al., 2015) (a general-purpose micropolicy enforcement engine based on the PUMP architecture).

6.5.3 CHERI: Capability Hardware

CHERI (Capability Hardware Enhanced RISC Instructions) (WOODRUFF et al., 2014; WATSON et al., 2015), from the University of Cambridge and SRI International, replaces raw pointers with hardware-enforced capabilities carrying address, bounds, permissions, and a validity tag. Capabilities cannot be forged—they can only be derived from existing valid capabilities with equal or narrower authority. CHERI prevents buffer overflows (hardware bounds checking), use-after-free (capability revocation), and pointer forgery (validity tag) with deterministic, not probabilistic, enforcement.

CHERI has been prototyped on RISC-V (CHERI-RISC-V (WATSON et al., 2023b)) and ARM (the Morello evaluation platform (WATSON et al., 2023a)). CheriBSD (DAVIS et al., 2019) demonstrates a full POSIX-compliant OS with CHERI enforcement. Performance overhead is 5–15% from widened 128-bit capabilities. ARM has not committed to commercial CHERI adoption, pursuing PAC/MTE/BTI/GCS instead; RISC-V's open extension model means CHERI could become a ratified extension, potentially providing the strongest memory safety of any production ISA.

6.5.4 The Rust Dimension

Both architectures benefit from memory-safe languages. Rust provides compile-time memory safety without hardware support, but `unsafe` blocks (unavoidable in embedded firmware) and FFI boundaries to C libraries remain unprotected (JUNG et al., 2021b; JUNG et al., 2018; SHARMA et al., 2024). ARM's hardware mechanisms catch bugs in these unprotected regions;

Table 17 – Memory Safety and Control Flow Integrity

Mechanism Type	**ARM Hardware Enforcement**	**RISC-V Mechanism**
Pointer integrity	PAC (Pointer Authentication Codes)	No base-ISA equivalent; vendor-specific only
Memory tagging	MTE (Memory Tagging Extension)	No base-ISA equivalent; relies on Rust / software
Branch target ID	BTI (coarse-grained forward-edge CFI)	Zicfilp (Landing Pad)
Shadow stack	GCS (Guarded Control Stack)	Zicfiss (Shadow Stack)
Capability model	Morello (CHERI prototype)	CHERI-RISC-V (research / prototype)

RISC-V without PAC/MTE equivalents does not. Google's Android data shows a ~1000× reduction in vulnerability density for Rust versus C/C++ components (Vander Stoep, 2025).

For the paper's context: ARM provides pragmatic hardware hardening for existing C/C++ codebases; RISC-V relies more heavily on language-level safety (Rust) and has a potential future advantage if CHERI is adopted; the ideal is both (Rust + hardware enforcement).

Table 17 highlights hardware-assisted defenses against spatial and temporal errors.

6.6 Side-Channel and Speculative Execution Defenses

Side-channel attacks exploit implementation artifacts—cache timing, speculative execution state, power consumption—that are invisible in the ISA specification but present in physical hardware.

6.6.1 Spectre and Meltdown

Meltdown (LIPP et al., 2018) exploits transient execution of permission-violating loads that affect cache state before faults are raised. ARM's exposure was limited (primarily Cortex-A75); most ARM cores used in robotics (Cortex-M, Cortex-R, Cortex-A53) were immune. RISC-V in-order cores (Ibex, SiFive E-series, Rocket) are structurally immune; out-of-order implementations (BOOM, SiFive P-series) require explicit verification.

Spectre (KOCHER et al., 2019) exploits speculative execution steered by attacker-manipulated prediction mechanisms. Both ARM and RISC-V out-of-order cores are susceptible. ARM mitigations include SSBS (preventing speculative store bypass), CSV2/CSV3 (partitioning branch predictions by security context), DIT (data-independent timing for constant-time execution), and FEAT_SPECRES (prediction invalidation on context switches) (Arm Ltd., 2020b).

RISC-V's Spectre response varies by implementation tier: in-order embedded cores (the majority in safety-critical robotics) are inherently immune; mid-range out-of-order cores have a smaller attack surface (shorter speculation windows, simpler predictors); high-performance cores (P870, Veyron) require ARM-comparable mitigations. For open-source cores (BOOM),

the attack surface is fully characterizable from RTL (GONZALEZ et al., 2019). The **Zkt** extension (MARSHALL et al., 2022) specifies data-independent timing for cryptographic instructions, analogous to ARM's DIT.

6.6.2 Cache Timing Attacks

Cache timing attacks on cryptographic implementations (BERNSTEIN, 2005; OSVIK; SHAMIR; TROMER, 2006) exploit data-dependent cache access patterns to recover secret keys. These affect any processor with a cache, regardless of ISA. Mitigations include constant-time implementations (bitsliced AES), ARM's DIT/FEAT_DIT, and RISC-V's Zkt. Cache partitioning (ARM MPAM; RISC-V cache coloring via Sanctum (COSTAN; LEBEDEV; DEVADAS, 2016) or Bao (MARTINS et al., 2020)) prevents cross-domain cache interference.

6.6.3 Physical Side Channels

Differential Power Analysis (DPA) (KOCHER; JAFFE; JUN, 1999) and electromagnetic emanation attacks exploit the physics of CMOS circuits and are ISA-independent. Hardware countermeasures (random instruction insertion, dual-rail logic, power trace randomization) are implemented at the physical design level. RISC-V's open RTL enables custom side-channel hardening—OpenTitan's Ibex core includes register file scrambling and random pipeline stall insertion (MITELOUDI et al., 2024). ARM provides hardened core variants (SecurCore SC000/SC300) for smart card applications but does not expose RTL for custom modifications.

6.6.4 Robotics Threat Model

A standalone robot communicating with known servers via authenticated channels has a smaller microarchitectural attack surface than cloud infrastructure. The primary Spectre-class threat is a compromised ROS 2 node (exploited via network vulnerability) using cache timing to extract secrets from a co-resident TEE or safety partition. Cache partitioning (MPAM or cache coloring) and domain isolation mitigate this risk. Supply-chain attacks, such as the XZ Utils backdoor (FREUND, 2024), that achieve code execution on the robot represent the more likely attack vector, connecting directly to the supply-chain trust analysis in Section 6.7.

Table 18 shows that both architectures address microarchitectural threats that exploit hardware implementation artifacts.

6.7 Supply-Chain Trust and Auditability

The ARM-vs-RISC-V comparison has a dimension absent from Intel-vs-AMD analyses: hardware auditability. This layer ties together every previous security mechanism by asking whether their implementations can be independently verified.

Table 18 – Side-Channel and Speculative Execution Defenses

Threat Category	**ARM Mitigations**	**RISC-V Mitigations**
Spectre / Meltdown	SSBS, CSV2, CSV3, and FEAT_SPECRES	Structural immunity for in-order cores; explicit verification for BOOM
Timing attacks	DIT (Data-Independent Timing)	Zkt extension for constant-time crypto
Power analysis	Hardened cores (SecurCore)	RTL-level hardening (OpenTitan pipeline stalls)
Cache isolation	MPAM for cache partitioning	Sanctum or Bao for cache coloring

6.7.1 The Trust Stack

Using a closed-source processor core requires trusting multiple entities: the IP vendor's RTL correctly implements the specification, the synthesis tools do not introduce errors, the foundry does not insert hardware trojans (BHUNIA; TEHRANIPOOR, 2018; XUE et al., 2020), and counterfeit components are excluded from the supply chain. ARM provides safety manuals and FMEDA (Failure Modes, Effects, and Diagnostic Analysis) data for certified cores, but independent reproduction of these results requires access to confidential RTL.

6.7.2 RISC-V's Transparency Advantage

Open-source RISC-V cores (Rocket, BOOM, Ibex, CVA6) enable independent verification at the RTL level. Formal verification can prove security-critical properties, "PMP always blocks S-mode access to M-mode-protected regions", with mathematical certainty, as demonstrated by Cheang *et al.* (CHEANG et al., 2022) for the Rocket Chip's PMP implementation using the UCLID5 model checker. The SAIL formal ISA specification (ARMSTRONG et al., 2019), adopted by RISC-V International, provides a machine-readable reference for verifying that implementations conform to the ISA.

OpenTitan (MEZA et al., 2023) exemplifies this advantage: formal verification of security properties (PMP enforcement, lifecycle state transitions, crypto module correctness) is part of the standard design flow. ARM's CCA has been formally verified (LI et al., 2022), but only by ARM and their collaborators using the confidential RTL.

6.7.3 Certification Implications

For ISO 26262 ASIL D certification, which recommends formal verification for demonstrating freedom from systematic faults, open RTL provides a clear compliance path: the certification body can independently reproduce fault injection campaigns and formal proofs. With closed ARM RTL, the body must accept the vendor's FMEDA at face value or commission ARM to provide additional evidence—an arrangement some European certification bodies are increasingly uncomfortable with.

This connects to the broader European sovereignty agenda: the European Processor

Initiative (EPI) (European Processor Initiative Consortium, 2018) and the RISC-V Commons project explicitly invest in RISC-V for auditable processors for European critical infrastructure. India's SHAKTI project (MADHUSUDAN et al., 2016) (IIT Madras) pursues similar sovereignty objectives for defense applications.

The honest comparison is: ARM provides *vendor-asserted* security properties (backed by internal verification and reputation); open-source RISC-V provides *independently-verifiable* security properties (backed by the auditor's own analysis). For regulated industries, independently-verifiable properties carry greater evidentiary weight in certification proceedings.

6.8 Discussion: Multi-Domain Robotic Security Architecture

A modern autonomous robot requires at least four security/safety domains: a safety supervisor (ASIL D, lockstep cores, bare-metal M-mode, PMP-locked), motor control (ASIL B, RTOS, per-joint-group isolation), perception and AI inference (QM, Linux/ROS 2, GPU/NPU via IOMMU), and planning/communication (QM, Linux, network interfaces). The security architecture must enforce isolation between domains, authenticate inter-domain communication, protect cryptographic keys, and verify firmware updates—while meeting hard real-time deadlines on the motor control path.

ARM's approach deploys heterogeneous SoCs (e.g., NXP S32G: Cortex-A53 + Cortex-R52 + Cortex-M7) with profile-specific security: TrustZone for the security boundary, stage-2 tables for VM isolation, SMMU for DMA isolation, MPAM for cache partitioning, and GIC for interrupt routing. The advantage is maturity—this architecture has years of production deployment in automotive (NVIDIA Orin/Thor). The disadvantage is complexity: three ISA variants, three toolchains, and closed hardware requiring vendor trust.

RISC-V's approach uses ISA-uniform heterogeneous SoCs (RV32IMC safety cores alongside RV64GCV application cores) with composable security: PMP for M-mode partitioning, H-extension stage-2 for VM isolation, IOPMP/IOMMU for DMA protection, Keystone/MultiZone for TEE, and WorldGuard for bus-level domain separation. The advantage is transparency (one ISA, auditable RTL, certifiable TCB) and customizability (vendor-specific security extensions in reserved opcode space). The disadvantage is maturity: the full RISC-V security stack is still being validated in production, while ARM's ecosystem has extensive deployment history.

The architectural tension is fundamental: ARM provides purpose-built mechanisms for each security layer (TrustZone, CCA, PAC, MTE, MPAM) within a closed ecosystem; RISC-V provides fewer, simpler mechanisms (PMP, M-mode, H-extension) composable into equivalent architectures within an open ecosystem. Neither is inherently more secure—the question is whether the system's certification requirements demand *verifiable* security (RISC-V advantage) or *battle-tested* security (ARM advantage).

6.9 Final Remarks

This chapter has provided a structured six-layer comparison of ARM and RISC-V security architectures for safety-critical robotics. The key findings are:

ARM leads in deployed hardware security mechanisms (PAC, MTE, BTI, GCS for memory safety; TrustZone and CCA for trusted execution; MPAM for cache partitioning; SMMUv3 for DMA isolation) with over two decades of TrustZone deployment and an emerging CCA ecosystem. The closed hardware model requires trust in ARM but delivers a mature, vertically integrated security stack.

RISC-V leads in transparency and certifiability (open RTL enabling formal verification, smaller TCB in Keystone/MultiZone, SAIL formal specification). Its security mechanisms are architecturally simpler (PMP, M-mode isolation) but compose flexibly to address the same threat model. The open extension model positions CHERI as a potential future advantage for deterministic memory safety. Current limitations include the absence of PAC/MTE equivalents, no standardized cache partitioning, and ecosystem fragmentation.

For mixed-criticality robotic platforms, both architectures can provide adequate security if correctly designed. The choice depends on the specific certification pathway (RISC-V's auditability advantage for high-assurance certification), the ecosystem maturity requirements (ARM's advantage for near-term deployment), and the supply-chain trust model (RISC-V's advantage for vendor independence).

Future work should examine the practical certification cost differential between open and closed hardware isolation mechanisms, benchmark TEE switching overhead in realistic robotic workloads, and evaluate CHERI-RISC-V's capability model for mixed-criticality domain isolation.

7 Conclusions

In this master's thesis, we set out to address a gap identified in Chapter 1: the absence of a unified, parallel-structured comparison of the ARM and RISC-V architectures spanning instruction set design, microarchitecture, and the application domains of robotics, artificial intelligence, and security. Through five surveys conducted under a common analytical framework (Chapters 2 through 6), a consistent picture has emerged: the contemporary choice between ARM and RISC-V is no longer primarily a question of raw architectural capability, but of ecosystem maturity, certification pathway, and the strategic value of openness.

7.1 Principal Findings

Across the dimensions examined, four cross-cutting findings recur.

First, **the gap between ARM and RISC-V is one of maturity rather than capability.** RISC-V's modular extension system, clean register model, and ratified vector and virtualization extensions provide an architectural foundation comparable to AArch64, and commercial out-of-order implementations now approach ARM Neoverse-class performance. What ARM retains is the accumulated advantage of more than fifteen years of deployment: a mature toolchain, established assessor familiarity, and a broad catalogue of certified safety cores.

Second, **the instruction set architecture matters less than the surrounding system architecture.** In the robotics analysis (Chapter 4), measured instruction-path lengths between RISC-V and AArch64 fell within roughly ten percent, and AI inference performance was dominated by GPU, NPU, and FPGA accelerators rather than the scalar CPU core. The decisive engineering decisions therefore lie in heterogeneous integration, memory hierarchy, and accelerator design rather than in the choice of ISA itself.

Third, **architecture selection is application- and certification-dependent rather than absolute.** The artificial-intelligence analysis (Chapter 5) found RISC-V delivering up to fourfold

superior energy efficiency through custom extensions, while ARM delivered up to fifteenfold faster inference on complex networks through its mature toolchain — indicative best-case magnitudes rather than fixed architectural verdicts, and a trade-off that resolves differently for an ultra-low-power sensor node than for a latency-critical perception pipeline. The six-layer security comparison (Chapter 6) reached an analogous conclusion: ARM leads in deployed hardware mechanisms (PAC, MTE, BTI, CCA, TrustZone), while RISC-V leads in transparency and certifiability through open RTL and formal specification, with both able to satisfy the same threat model when correctly designed.

Fourth, **openness and auditability are shifting from theoretical to practical advantages.** As certification regimes increasingly demand evidence of hardware isolation, and as supply-chain trust becomes a first-order concern, RISC-V's open RTL — amenable to formal verification and independent audit — is becoming a concrete certification enabler rather than an abstract philosophical preference.

The parallel structure of this survey allows the comparison to be stated compactly. Table 19 consolidates the cross-architecture findings dimension by dimension, following the order of the parallel sections of Chapters 2 and 3 and the application analyses of Chapters 4 through 6; each row summarizes the corresponding section-level comparison developed in the body of the thesis.

7.2 Implications for System Designers

These findings carry direct implications for practice. For near-term production systems, particularly in automotive and industrial domains with established certification requirements, ARM remains the lower-risk choice by virtue of ecosystem maturity. For systems with longer development horizons, cost-sensitivity at volume, ultra-low-power constraints, or a need for full-stack auditability, RISC-V is increasingly compelling. Most significantly, the analysis suggests that the framing of "ARM versus RISC-V" is itself becoming less relevant: the trajectory of the field points toward heterogeneous platforms that compose ARM application cores, RISC-V real-time and accelerator cores, and domain-specific accelerators within a single coherent system. The relevant design question is shifting from architecture selection to architecture composition.

7.3 Limitations

Several limitations qualify these conclusions. This work is an architectural survey rather than an experimental study; its comparisons rest on published specifications, peer-reviewed literature, and vendor documentation rather than on original benchmarking under controlled conditions, and quantitative figures drawn from heterogeneous sources should be read as indicative rather than definitive. The field is also evolving rapidly: RISC-V ratifies new extensions and

Table 19 – Cross-architecture synthesis: ARM and RISC-V along the principal dimensions of this survey

Dimension	**ARM**	**RISC-V**
ISA model & governance	Proprietary, licensed (1985; ARM Ltd.)	Open, royalty-free (2010; RISC-V International)
Design approach	Rich ISA; new instructions added with each version	Minimal frozen base plus modular extensions; complexity delegated to the microarchitecture (instruction fusion)
Vector processing	NEON (fixed 128-bit) evolving to scalable SVE/SVE2/SME	RVV 1.0, vector-length agnostic from its first ratified version
Virtualization	EL2 with a mature ecosystem (KVM, Xen)	H-extension ratified; ecosystem maturing (KVM, Xvisor, Bao)
Pipeline evolution	Four decades of refinement; Cortex-X and Neoverse flagship out-of-order cores	Three generations in a single decade; wide out-of-order designs approaching parity
Ecosystem & tooling	Mature; more than fifteen years of deployment lead	Growing rapidly; fragmentation addressed by profiles (RVA23)
Safety-critical support	Cortex-R lineage; broad catalogue of ASIL D-certified cores	Certified lockstep cores emerging (SiFive, Andes); architectural feature parity reached
Security mechanisms	Deployed hardware defenses: TrustZone, PAC, MTE, BTI, CCA	Transparency-oriented: PMP, open TEEs (Keystone, WorldGuard, CoVE)
Auditability & supply chain	Closed IP; trust vested in the vendor	Open RTL; independent audit and formal verification possible
Deployment today (robotics, AI)	Dominant in shipping products (Jetson, i.MX, RB5)	Strong in research and the milliwatt class (PULP, GAP9)

profiles on a continuing basis, and any snapshot of relative maturity is necessarily bound to its moment. Finally, the analysis concentrates on the ARM and RISC-V architectures themselves and does not treat competing or complementary options such as x86 or specialised AI-accelerator ISAs in comparable depth.

7.4 Future Works

Several directions follow naturally from this work. **First,** the most direct is experimental validation: controlled benchmarking of representative ARM and RISC-V platforms on robotics and edge-AI workloads would convert the indicative figures reported here into directly comparable measurements. **A second direction** is the deeper study of memory-safety mechanisms, in particular the CHERI capability model, whose composition with RISC-V may offer deterministic memory safety beyond the probabilistic guarantees of ARM's MTE. **A third** is the maturation of RISC-V profiles and safety certification, where tracking the emergence of ASIL D-certified cores and standardised cache-partitioning mechanisms would refine the maturity assessment offered here. **A fourth direction** applies these architectures to brain–computer interface (BCI)

devices, an emerging safety-critical biomedical domain in which implantable and wearable neural interfaces must pair hard real-time signal processing with strict safety and reliability guarantees. This connects directly to the real-time Linux (PREEMPT_RT) and ELISA efforts examined in this thesis: the deterministic, certifiable execution they target is precisely what such neural-interface systems demand, and the low-power, auditable ARM and RISC-V platforms studied here offer complementary foundations for them. **Finally,** the heterogeneous-composition paradigm identified above warrants dedicated study: methodologies for partitioning mixed-criticality robotic workloads across ARM, RISC-V, and accelerator cores on a single SoC remain an open and increasingly important research problem.

7.5 Closing Remarks

The emergence of RISC-V as a credible peer to ARM marks a structural shift in the processor landscape — the first in decades in which the dominant embedded and application architecture faces an open, royalty-free alternative of comparable technical merit. Whether RISC-V ultimately challenges ARM's commercial dominance or settles into a complementary role as a customisable component within fundamentally heterogeneous systems, the architectural decision it forces designers to confront is no longer a narrow question of performance, but a broad one encompassing security, certifiability, cost, and long-term technological sovereignty. In this master's thesis, we have sought to map that decision space; the choices within it will increasingly be made not between ARM and RISC-V, but about how best to combine them. Finally, it is transparently declared that Generative Artificial Intelligence (AI) tools were used as technical assistants throughout the development of this work. The use of these tools was concentrated on assisting with linguistic revision, orthographic refinement, and the logical structuring of the text across all chapters of this academic document, without prejudice to the author's technical and authorial responsibility for the content presented.

Bibliography

ABADADE, Y. et al. *A Comprehensive Survey on TinyML*. 2023. IEEE Access. Y. Abadade *et al.*, "A Comprehensive Survey on TinyML," *IEEE Access*, vol. 11, pp. 96892–96922, 2023, doi: 10.1109/ACCESS.2023.3292575. Cited 3 times on pages 61, 66, and 67.

ADVANTECH. *High-Efficiency Edge AI Solution with NXP i.MX 95*. 2024. Advantech, "High-Efficiency Edge AI Solution with NXP i.MX 95," 2024. Cited on page 62.

ALDEGHERI, S. et al. *Data Flow ORB-SLAM for Real-Time Performance on Embedded GPU Boards*. 2019. Proc. IEEE/RSJ IROS. "Data Flow ORB-SLAM for Real-Time Performance on Embedded GPU Boards," in *Proc. IEEE/RSJ IROS*, 2020, doi: 10.1109/IROS45743.2020.8967814. Cited on page 65.

ALJAMI, H. M. et al. *Benchmarking YOLOv8 Variants for Object Detection on Jetson Orin NX for Edge Computing*. 2026. MDPI Computers. "Benchmarking YOLOv8 Variants for Object Detection on Jetson Orin NX for Edge Computing," *MDPI Computers*, vol. 15, no. 2, p. 74, 2025. Cited 2 times on pages 62 and 65.

ALQAHTANI, D. K.; CHEEMA, M. A.; TOOSI, A. N. Benchmarking deep learning models for object detection on edge computing devices. In: GAALOUL, W. et al. (Ed.). *Service-Oriented Computing*. Singapore: Springer Nature Singapore, 2025. p. 142–150. ISBN 978-981-96-0805-8. Cited on page 77.

ALVES, T.; FELTON, D. *TrustZone: Integrated Hardware and Software Security*. [S.l.], 2004. Cited 2 times on pages 87 and 91.

Andes Technology. *Andes Technology Announces D23-SE: A Functional Safety RISC-V Core with DCLS and Split-Lock for ASIL-B/D Automotive Applications*. 2025. Press Release. Cited on page 55.

ARM. *Cortex-R52: Real-Time Safety for Robotics & Healthcare*. 2026. ARM, "Cortex-R52: Real-Time Safety for Robotics & Healthcare," [Online]. Available: <https://www.arm.com/products/silicon-ip-cpu/cortex-r/cortex-r52>. Cited 2 times on pages 62 and 68.

Arm Ltd. *ARM Security Technology—Building a Secure System using TrustZone Technology*. [S.l.], 2009. Document PRD29-GENC-009492C. Cited on page 87.

Arm Ltd. *ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition*. [S.l.], 2018. Document DDI 0406C.d. Cited 4 times on pages 23, 25, 26, and 29.

Arm Ltd. Arm Helium technology: M-Profile vector extension (MVE) for Arm Cortex-M processors. *Arm White Paper*, 2019. Available at developer.arm.com. Cited 4 times on pages 27, 34, 75, and 84.

Arm Ltd. *ARM System Memory Management Unit Architecture Specification SMMU architecture version 3*. [S.l.], 2020. Document IHI 0070. Cited 3 times on pages 35, 89, and 90.

Arm Ltd. *Cache Speculation Side-channels*. [S.l.], 2020. Version 2.5. Cited on page 95.

Arm Ltd. *Cortex-R82 Processor Technical Reference Manual*. [S.l.], 2020. Document DDI 0586, revision r0p0. Cited 2 times on pages 27 and 37.

Arm Ltd. *ARM Generic Interrupt Controller Architecture Specification GIC architecture version 3 and version 4*. [S.l.], 2021. Document IHI 0069. Cited on page 35.

Arm Ltd. *Scalable Matrix Extension for the Armv9-A Architecture*. 2021. Arm Architecture White Paper. Available at community.arm.com. Cited 3 times on pages 23, 32, and 34.

Arm Ltd. *Arm Architecture Reference Manual for A-Profile Architecture*. [S.l.], 2023. Cited 10 times on pages 25, 26, 29, 40, 42, 44, 45, 46, 47, and 54.

Arm Ltd. *Armv8-M Architecture Reference Manual*. [S.l.], 2023. Document DDI 0553B. Cited on page 27.

Arm Ltd. *Scalable Vector Extensions (SVE) for the Armv8-A Architecture*. 2023. Arm Developer, 2023. Cited on page 51.

Arm Ltd. *Arm Architecture Reference Manual for A-profile Architecture*. [S.l.], 2024. Document DDI 0487. Cited on page 89.

Arm Ltd. *Arm Architecture Reference Manual for A-profile architecture (Armv8-A/Armv9-A)*. [S.l.], 2024. Document DDI 0487, available at developer.arm.com. Cited 3 times on pages 23, 25, and 28.

Arm Ltd. *Arm Architecture Reference Manual Supplement—Memory System Resource Partitioning and Monitoring (MPAM)*. [S.l.], 2024. Cited on page 90.

Arm Ltd. *Scalable Matrix Extension (SME) for Armv9-A*. 2024. Arm Community Blog, 2024. Cited on page 52.

Arm Ltd. *The Arm Ecosystem: Powering AI Everywhere – From Cloud to Edge*. 2025. Arm Newsroom. Accessed in 2025. Disponível em: <https://newsroom.arm.com/blog/arm-computex-2025>. Cited on page 73.

Arm Ltd. *Arm Firmware Framework for Arm A-profile*. [S.l.], 2025. Document DEN0077, Version 1.2. Cited on page 91.

ARM Newsroom. *Arm Accelerates Edge AI with Latest Generation Ethos-U NPU and New IoT Reference Design Platform*. 2024. ARM Newsroom, "Arm Accelerates Edge AI with Latest Generation Ethos-U NPU and New IoT Reference Design Platform," 2024. Cited on page 70.

ARM Newsroom. *Embracing the Future with Cortex-A320: A Deep Dive into the General Armv9 Architecture Adoption*. 2024. ARM Newsroom, "Embracing the Future with Cortex-A320: A Deep Dive into the General Armv9 Architecture Adoption," 2024. Cited on page 70.

ARM Newsroom. *The Next Platform Shift: Physical and Edge AI, Powered by Arm*. 2026. ARM Newsroom, "The Next Platform Shift: Physical and Edge AI, Powered by Arm," 2026. Cited on page 63.

ARMSTRONG, A. et al. ISA semantics for ARMv8-A, RISC-V, and CHERI-MIPS. *Proc. ACM Program. Lang.*, v. 3, n. POPL, p. 71, 2019. Cited on page 97.

ASANOVIĆ, K. et al. *The Rocket Chip Generator*. [S.l.], 2016. Cited 2 times on pages 41 and 49.

ASSIR, I. A. et al. *Arrow: A RISC-V Vector Accelerator for Machine Learning Inference*. 2021. Proc. CARRV Workshop. I. Al Assir *et al.*, "Arrow: A RISC-V Vector Accelerator for Machine Learning Inference," in *Proc. CARRV Workshop*, 2021, arXiv:2107.07169. Cited on page 64.

BADALIAN, K. et al. *LExCI: A Framework for Reinforcement Learning with Embedded Systems*. 2024. Applied Intelligence. K. Badalian *et al.*, "LExCI: A Framework for Reinforcement Learning with Embedded Systems," *Applied Intelligence*, Springer, 2024, arXiv:2312.02739. Cited on page 67.

BALLER, S. P. et al. Deepedgebench: Benchmarking deep neural networks on edge devices. In: *2021 IEEE International Conference on Cloud Engineering (IC2E)*. [S.l.: s.n.], 2021. p. 20–30. Cited 2 times on pages 76 and 77.

BANBURY, C. et al. *MLPerf Tiny Benchmark*. 2021. Disponível em: <https://arxiv.org/abs/2106.07597>. Cited 2 times on pages 76 and 80.

BELTRÁN-ESCOBAR, D. et al. *A Review on Resource-Constrained Embedded Vision Systems-Based Tiny Machine Learning for Robotic Applications*. 2024. Algorithms. D. Beltrán-Escobar *et al.*, "A Review on Resource-Constrained Embedded Vision Systems-Based Tiny Machine Learning for Robotic Applications," *Algorithms*, vol. 17, no. 11, p. 476, 2024. Cited on page 66.

BERNSTEIN, D. J. *Cache-timing attacks on AES*. [S.l.], 2005. Cited on page 96.

BHUNIA, S.; TEHRANIPOOR, M. M. *The Hardware Trojan War: Attacks, Myths, and Defenses*. [S.l.]: Springer, 2018. Cited on page 97.

BLEM, E.; MENON, J.; SANKARALINGAM, K. Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures. In: *Proc. 19th IEEE Int. Symp. on High Performance Computer Architecture (HPCA)*. [S.l.: s.n.], 2013. p. 1–12. Cited 2 times on pages 29 and 33.

BLEM, E. et al. ISA wars: Understanding the relevance of ISA being RISC or CISC to performance, power, and energy on modern architectures. In: . [S.l.: s.n.], 2015. v. 33, n. 1, p. 1–34. Cited 2 times on pages 28 and 33.

Boston Dynamics. *Doing More with Spot*. 2026. Boston Dynamics, "Doing More with Spot," [Online]. Available: <https://bostondynamics.com/blog/doing-more-with-spot/>. Cited on page 62.

BRASH, D.; SHERWOOD, T. Enabling Linux in safety applications. *Linux Foundation Technical Report*, 2020. Available at elisa.tech. Cited 2 times on pages 36 and 37.

BROWN, N. et al. Performance characterisation of the 64-core SG2042 RISC-V CPU for HPC. *arXiv preprint arXiv:2406.12394*, jun. 2024. Cited 2 times on pages 48 and 49.

BURNS, A.; DAVIS, R. I. A survey of research into mixed criticality systems. *ACM Computing Surveys*, v. 50, n. 6, p. 1–37, 2017. Cited 2 times on pages 19 and 36.

CAMPOS, C. et al. ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, v. 37, n. 6, p. 1874–1890, 2021. Cited on page 37.

CASTELLó, A. et al. High performance and energy efficient inference for deep learning on multicore arm processors using general optimization techniques and blis. *Journal of Systems Architecture*, v. 125, p. 102459, 2022. ISSN 1383-7621. Cited on page 75.

CAVALCANTE, M. et al. 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*, v. 28, n. 2, p. 530–543, 2020. Cited on page 74.

CAVALCANTE, M. et al. An open source highly efficient RISC-V V 1.0 vector unit design. In: *Proc. IEEE*. [S.l.: s.n.], 2022. Cited on page 51.

CDR camel. *RISC-V Vector Extension for Integer Workloads: An Informal Gap Analysis*. 2024. GitHub Gist. Cited on page 52.

CELIO, C.; PATTERSON, D. A.; ASANOVIĆ, K. *The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor*. [S.l.], 2015. Cited on page 49.

CERDEIRA, D. et al. SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems. In: *Proc. 2020 IEEE Symp. Security and Privacy (S&P)*. [S.l.: s.n.], 2020. p. 1416–1432. Cited on page 91.

CEREDA, E. et al. *Deep Neural Network Architecture Search for Accurate Visual Pose Estimation Aboard Nano-UAVs*. 2023. Proc. IEEE ICRA. E. Cereda *et al.*, "Deep Neural Network Architecture Search for Accurate Visual Pose Estimation Aboard Nano-UAVs," in *Proc. IEEE ICRA*, 2023, doi: 10.1109/ICRA48891.2023.10160730. Cited on page 65.

CHEANG, K. et al. *Verifying RISC-V Physical Memory Protection*. 2022. ArXiv:2211.02179. Cited 2 times on pages 89 and 97.

CHEN, C. et al. Xuantie-910: A commercial multi-core 12-stage pipeline out-of-order 64-bit high performance risc-v processor with vector extension : Industrial product. In: *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. [S.l.: s.n.], 2020. p. 52–64. Cited on page 75.

CHEN, Y. et al. Xuantie-910: A commercial multi-core 12-stage pipeline out-of-order 64-bit high performance RISC-V processor with vector extension. In: *Proc. ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. [S.l.: s.n.], 2020. p. 752–764. Cited on page 49.

CHEVTCHENKO, S. F. et al. Predictive maintenance model based on anomaly detection in induction motors: A machine learning approach using real-time iot data. In: *Proceeding of the 33rd European Safety and Reliability Conference*. Research Publishing Services, 2023. (ESREL), p. 3173–3180. Disponível em: <http://dx.doi.org/10.3850/978-981-18-8071-1_P578-cd>. Cited on page 80.

Chips and Cheese. *Hot Chips 2023: SiFive's P870 Takes RISC-V Further*. 2023. <https://chipsandcheese.com/p/hot-chips-2023-sifives-p870-takes-risc-v-further>. Cited 3 times on pages 46, 49, and 50.

Chips and Cheese. *Alibaba/T-HEAD's Xuantie C910*. 2025. <https://chipsandcheese.com/p/alibabat-heads-xuantie-c910>. Cited on page 49.

CHOWDHURY, S. S. et al. *Neuromorphic Computing for Robotic Vision: Algorithms to Hardware Advances*. 2025. Nature Commun. Eng. S. S. Chowdhury *et al.*, "Neuromorphic Computing for Robotic Vision: Algorithms to Hardware Advances," *Nature Commun. Eng.*, 2025, doi: 10.1038/s44172-025-00492-5. Cited 2 times on pages 67 and 71.

CHRISTOFAS, V. et al. Comparative evaluation between accelerated risc- v and arm ai inference machines. In: *2023 6th World Symposium on Communication Engineering (WSCE)*. [S.l.: s.n.], 2023. p. 108–113. Cited on page 77.

CHRONIS et al. *Deploying Large AI Models on Micro-Electronics with RISC-V: Federated Learning for Energy Monitoring and Robotics*. 2025. Proc. ICICT 2025. Chronis *et al.*, "Deploying Large AI Models on Micro-Electronics with RISC-V: Federated Learning for Energy Monitoring and Robotics," in *Proc. ICICT 2025*, Springer LNNS, vol. 1444, 2025. Cited on page 69.

CIANI, M. et al. Unleashing OpenTitan's potential: a silicon-ready embedded secure element for root of trust and cryptographic offloading. *ACM Trans. Embedded Computing Systems*, 2024. Cited on page 87.

CONTI, F. et al. *Training on the Fly: On-Device Self-Supervised Learning Aboard Nano-Drones Within 20 mW*. 2024. IEEE Trans. Comput.-Aided Design. F. Conti *et al.*, "Training on the Fly: On-Device Self-Supervised Learning Aboard Nano-Drones Within 20 mW," *IEEE Trans. Comput.-Aided Design*, vol. 43, pp. 3685, 2024. Cited on page 65.

CORBET, J. *Why RISC-V Doesn't (Yet) Support KVM*. 2021. LWN.net. Cited on page 54.

CORBET, J. *The realtime preemption end game — for real this time*. 2024. LWN.net, <https://lwn.net/Articles/989212/>. PREEMPT_RT enabled in mainline Linux v6.12 for x86, x86_64, ARM64, and RISC-V; merged Sep. 20, 2024. Cited on page 57.

CORDOVA-CARDENAS, R.; AMOR, D.; GUTIÉRREZ, Á. *Edge AI in Practice: A Survey and Deployment Framework for Neural Networks on Embedded Systems*. 2025. MDPI Electronics. "Edge AI in Practice: A Survey and Deployment Framework for Neural Networks on Embedded Systems," *MDPI Electronics*, vol. 14, no. 24, p. 4877, 2025. Cited on page 68.

COSTAN, V.; LEBEDEV, I.; DEVADAS, S. Sanctum: Minimal hardware extensions for strong software isolation. In: *Proc. 25th USENIX Security Symp.* [S.l.: s.n.], 2016. p. 857–874. Cited on page 96.

CSIS. *What RISC-V Means for the Future of Chip Development*. 2024. CSIS, "What RISC-V Means for the Future of Chip Development," 2024. Cited 3 times on pages 60, 68, and 69.

CSIS. *Sustaining Standards Leadership: The United States Cannot Disengage from RISC-V*. 2025. CSIS, "Sustaining Standards Leadership: The United States Cannot Disengage from RISC-V," 2025. Cited on page 70.

Cyber Raiden. *Arm C1-Nano Technical Overview: Power Efficiency, SME2 Matrix Acceleration, and Security in the Lumex Era*. 2026. Cited on page 52.

DALL, C. et al. ARM virtualization: Performance and architectural implications. In: *Proc. 43rd IEEE/ACM International Symposium on Computer Architecture (ISCA)*. [S.l.: s.n.], 2016. p. 304–316. Cited 2 times on pages 31 and 35.

DALL, C.; NIEH, J. KVM/ARM: The design and implementation of the Linux ARM hypervisor. In: *Proc. 19th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. [S.l.: s.n.], 2014. p. 333–348. Cited 3 times on pages 23, 29, and 35.

DAVID, R. et al. *TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems*. 2021. Disponível em: <https://arxiv.org/abs/2010.08678>. Cited on page 79.

DAVIS, B. et al. CheriABI: Enforcing valid pointer provenance and minimizing pointer privilege in the POSIX C run-time environment. In: *Proc. 24th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. [S.l.: s.n.], 2019. Cited on page 94.

DAVIS, R. I.; BURNS, A. A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys*, v. 43, n. 4, p. 1–44, 2011. Cited on page 36.

DEHNAVI, S. et al. *CompROS: A Composable ROS2 Based Architecture for Real-Time Embedded Robotic Development*. 2021. Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS). S. Dehnavi *et al.*, "CompROS: A Composable ROS2 Based Architecture for Real-Time Embedded Robotic Development," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021, doi: 10.1109/IROS51168.2021.9636590. Cited on page 64.

DHAWAN, U. et al. Architectural support for software-defined metadata processing. In: *Proc. 20th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '15)*. [S.l.: s.n.], 2015. Cited on page 94.

DONG, S. et al. *Characterizing and Optimizing Real-Time Optimal Control for Embedded SoCs*. 2025. Proc. IEEE IISWC. S. Dong *et al.*, "Characterizing and Optimizing Real-Time Optimal Control for Embedded SoCs," in *Proc. IEEE IISWC*, 2025, doi: 10.1109/IISWC66894.2025.00047. Cited on page 67.

EASA. *Certification Authorities Software Team Position Paper CAST-32A: Multi-Core Processors*. 2016. Cited 2 times on pages 55 and 56.

ETH Zurich. *PULP Platform*. 2026. ETH Zurich, "PULP Platform," [Online]. Available: <https://iis-projects.ee.ethz.ch/index.php/PULP>. Cited on page 63.

Eureka PatSnap. *RISC-V in AUTOSAR and ISO 26262 Contexts: Safety Pathways*. 2024. Cited on page 58.

European Processor Initiative Consortium. *EPI — European Processor Accelerators Based on RISC-V*. 2018. EU Horizon 2020 Project. Cited 2 times on pages 19 and 98.

FIROOZI, R. et al. *Foundation Models in Robotics: Applications, Challenges, and the Future*. 2023. R. Firoozi *et al.*, "Foundation Models in Robotics: Applications, Challenges, and the Future," arXiv:2312.07843, 2023. Cited on page 66.

FIROUZI, F.; FARAHANI, B.; MARINŠEK, A. The convergence and interplay of edge, fog, and cloud in the ai-driven internet of things (iot). *Inf. Syst.*, Elsevier Science Ltd., GBR, v. 107, n. C, jul. 2022. ISSN 0306-4379. Disponível em: <https://doi.org/10.1016/j.is.2021.101840>. Cited on page 78.

FLAMAND, E. et al. *GAP-8: A RISC-V SoC for AI at the Edge of the IoT*. 2018. Proc. IEEE 29th Int. Conf. ASAP. E. Flamand *et al.*, "GAP-8: A RISC-V SoC for AI at the Edge of the IoT," in *Proc. IEEE 29th Int. Conf. ASAP*, 2018, doi: 10.1109/ASAP.2018.8445101. Cited 2 times on pages 63 and 75.

FPROX. *RISC-V Profiles: Defining Sets of Extensions for Coherent Ecosystems*. 2023. Substack. <https://fprox.substack.com/p/risc-v-profiles>. Cited 2 times on pages 43 and 44.

FREUND, A. *Backdoor in Upstream xz/liblzma Leading to SSH Server Compromise*. 2024. Oss-security mailing list. CVE-2024-3094. Cited 2 times on pages 19 and 96.

FRUMUSANU, A. *The 2020 Mac Mini Unleashed: Putting Apple Silicon M1 to the Test*. 2020. AnandTech, <https://www.anandtech.com/show/16252/>. Microarchitectural analysis of the Apple M1 SoC. Cited 2 times on pages 33 and 34.

FURBER, S. *ARM System-on-Chip Architecture*. 2nd. ed. [S.l.]: Pearson Education, 2000. ISBN 0201675196. Cited 2 times on pages 22 and 23.

GALAGAIN, C. et al. *Is Semantic SLAM Ready for Embedded Systems? A Comparative Survey*. 2025. C. Galagain *et al.*, "Is Semantic SLAM Ready for Embedded Systems? A Comparative Survey," arXiv:2505.12384, 2025. Cited on page 62.

GAROFALO, A. et al. *XpulpNN: Accelerating Quantized Neural Networks on RISC-V Processors Through ISA Extensions*. 2020. Proc. Design, Autom. & Test in Europe (DATE). A. Garofalo *et al.*, "XpulpNN: Accelerating Quantized Neural Networks on RISC-V Processors Through ISA Extensions," in *Proc. Design, Autom. & Test in Europe (DATE)*, 2020, doi: 10.23919/DATE48585.2020.9116588. Cited 4 times on pages 64, 68, 74, and 75.

GAROFALO, A. et al. Pulp-nn: accelerating quantized neural networks on parallel ultra-low-power risc-v processors. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v. 378, n. 2164, p. 20190155, 12 2019. ISSN 1364-503X. Cited 2 times on pages 75 and 79.

GENC, H. et al. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*. [S.l.: s.n.], 2021. p. 769–774. Cited on page 75.

GERLEIN, E. A. et al. *Embedded System-on-Chip 3D Localization and Mapping—eSoC-SLAM*. 2021. MDPI Electronics. "Embedded System-on-Chip 3D Localization and Mapping—eSoC-SLAM," *MDPI Electronics*, vol. 10, no. 12, p. 1378, 2021. Cited on page 66.

GONZALEZ, A. et al. Replicating and mitigating Spectre attacks on an open source RISC-V microarchitecture. In: *Proc. Third Workshop Computer Architecture Research with RISC-V (CARRV 2019)*. [S.l.: s.n.], 2019. Cited on page 96.

GONZALEZ, A. et al. A 16mm2 106.1 gops/w heterogeneous risc-v multi-core multi-accelerator soc in low-power 22nm finfet. In: *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*. [S.l.: s.n.], 2021. p. 259–262. Cited on page 82.

GRACIOLI, G. et al. A survey on cache management mechanisms for real-time embedded systems. *ACM Computing Surveys*, v. 48, n. 2, p. 1–36, 2015. Cited on page 36.

Green Hills Software. *MIPS and Green Hills Software Accelerate Safety Certified Product Development for MIPS RISC-V Microcontrollers*. 2026. Cited on page 58.

GREENHALGH, P. big.LITTLE processing with ARM Cortex-A15 & Cortex-A7. *ARM White Paper*, 2011. Available at arm.com. Cited 2 times on pages 26 and 29.

Hex Five Security, Inc. *MultiZone Security TEE for RISC-V*. 2020. <https://hex-five.com/>. Cited on page 92.

IAR Systems. *IAR Systems Brings Functional Safety Tools to RISC-V with Certification for IEC 61508 and ISO 26262*. 2023. Press Release. Cited on page 58.

IFTIKHAR, S. et al. Ai-based fog and edge computing: A systematic review, taxonomy and future directions. *Internet of Things*, v. 21, p. 100674, 2023. ISSN 2542-6605. Disponível em: <https://www.sciencedirect.com/science/article/pii/S254266052200155X>. Cited on page 78.

Imperas Software. *Imperas to Provide Model of Tenstorrent Ascalon RISC-V Core*. 2023. Cited on page 58.

InfoQ. *Arm Scalable Matrix Extension 2 Coming to Android to Accelerate On-Device AI*. 2025. Cited on page 52.

INTELLIGENCE, M. *Embedded Computing Market—Size, Share & Growth*. 2024. Mordor Intelligence, "Embedded Computing Market—Size, Share & Growth," 2024. Cited on page 69.

International Electrotechnical Commission. *IEC 61508:2010 — Functional safety of electrical/electronic/programmable electronic safety-related systems*. 2010. Parts 1–7. Cited 4 times on pages 18, 36, 55, and 86.

International Organization for Standardization. *ISO 26262:2018 — Road vehicles — Functional safety*. 2018. Parts 1–12. Cited 8 times on pages 18, 27, 36, 37, 55, 57, 58, and 86.

ITURBE, X. et al. ARM cortex-R5 based fault-tolerant processor for safety-critical applications. In: *Proc. IEEE Int. Conf. on Dependable Systems and Networks Workshops (DSN-W)*. [S.l.: s.n.], 2016. Cited 2 times on pages 27 and 36.

JÄMBÄCK, M. *Evaluation of Real-Time Linux on RISC-V Processor Architecture*. Dissertação (Mestrado) — Tampere University, Finland, mar. 2022. Cited on page 57.

JEFF, B. big.LITTLE system architecture from ARM: saving power through heterogeneous multiprocessing and task context migration. In: *Proc. Design Automation Conference (DAC)*. [S.l.: s.n.], 2013. Cited 2 times on pages 26 and 29.

JEZNACH, T.; PATEL, A. et al. *RISC-V IOMMU Architecture Specification, Version 1.0.1*. [S.l.], 2024. Cited on page 90.

JIANG, S. et al. *Autonomous Navigation System of Greenhouse Mobile Robot Based on 3D LiDAR and 2D LiDAR SLAM*. 2022. Frontiers Plant Sci. S. Jiang *et al.*, "Autonomous Navigation System of Greenhouse Mobile Robot Based on 3D LiDAR and 2D LiDAR SLAM," *Frontiers Plant Sci.*, 2022, doi: 10.3389/fpls.2022.815218. Cited on page 61.

JIN, W.; REZAEIPANAH, A. Dynamic task allocation in fog computing using enhanced fuzzy logic approaches. *Scientific Reports*, v. 15, n. 1, p. 18513, 2025. ISSN 2045-2322. Disponível em: <https://doi.org/10.1038/s41598-025-03621-4>. Cited on page 78.

JUNG, R. et al. RustBelt: Securing the foundations of the Rust programming language. *Proc. ACM Program. Lang.*, v. 2, n. POPL, 2018. Cited on page 94.

JUNG, R. et al. Safe systems programming in Rust. In: . [S.l.: s.n.], 2021. v. 64, n. 4, p. 144–152. Cited 2 times on pages 36 and 37.

JUNG, R. et al. Safe systems programming in Rust. *Communications of the ACM*, v. 64, n. 4, p. 144–152, 2021. Cited on page 94.

KALAPOTHAS, S. et al. *A Survey on RISC-V-Based Machine Learning Ecosystem*. 2023. Information. S. Kalapothas *et al.*, "A Survey on RISC-V-Based Machine Learning Ecosystem," *Information*, vol. 14, no. 2, p. 64, 2023, doi: 10.3390/info14020064. Cited on page 60.

KIM, S. et al. *RoSÉ: A Hardware-Software Co-Simulation Infrastructure Enabling Pre-Silicon Full-Stack Robotics SoC Evaluation*. 2023. Proc. 50th ISCA. S. Kim *et al.*, "RoSÉ: A Hardware-Software Co-Simulation Infrastructure Enabling Pre-Silicon Full-Stack Robotics SoC Evaluation," in *Proc. 50th ISCA*, ACM, 2023, doi: 10.1145/3579371.3589099. Cited 2 times on pages 67 and 70.

KIRSCHNER, M.; DUDZIK, K.; BECKER, J. *Work-in-Progress: Real-Time Neural Network Inference on a Custom RISC-V Multicore Vector Processor*. 2024. M. Kirschner, K. Dudzik, and J. Becker, "Work-in-Progress: Real-Time Neural Network Inference on a Custom RISC-V Multicore Vector Processor," arXiv:2410.10340, 2024. Cited on page 64.

KIRSCHNER, M.; DUDZIKK, K.; BECKER, J. Work-in-progress: Real-time neural network inference on a custom risc-v multicore vector processor. In: *2024 IEEE Real-Time Systems Symposium (RTSS)*. [S.l.: s.n.], 2024. p. 427–430. Cited on page 77.

KOCHER, P. et al. Spectre attacks: Exploiting speculative execution. In: *Proc. IEEE Symposium on Security and Privacy (S&P)*. [S.l.: s.n.], 2019. p. 1–19. Cited 3 times on pages 19, 32, and 95.

KOCHER, P.; JAFFE, J.; JUN, B. Differential power analysis. In: *Advances in Cryptology—CRYPTO '99*. [S.l.]: Springer, 1999. (LNCS, v. 1666), p. 388–397. Cited on page 96.

KODAMA, Y. et al. Preliminary performance evaluation of application kernels using ARM SVE with multiple vector lengths. 2020. Cited 2 times on pages 22 and 33.

KRISHNAN, S. et al. *Air Learning: A Deep Reinforcement Learning Gym for Autonomous Aerial Robot Visual Navigation*. 2021. Machine Learning. "Air Learning: A Deep Reinforcement Learning Gym for Autonomous Aerial Robot Visual Navigation," *Machine Learning*, Springer, 2021, doi: 10.1007/s10994-021-06006-6. Cited on page 67.

KÜHNE, M. et al. Dorami: Privilege separating security monitor on RISC-V TEEs. In: *Proc. USENIX Security 2025*. [S.l.: s.n.], 2025. ArXiv:2410.03653. Cited on page 92.

KUMAR, A.; PARK, J.; BEHERA, L. *Jetson-SLAM: High-Speed Stereo Visual SLAM for Low-Powered Computing Devices*. 2023. IEEE Robot. Autom. Lett. A. Kumar, J. Park, and L. Behera, "Jetson-SLAM: High-Speed Stereo Visual SLAM for Low-Powered Computing Devices," *IEEE Robot. Autom. Lett.*, 2023, presented at ICRA 2024. Cited on page 62.

LAI, L.; SUDA, N.; CHANDRA, V. *CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs*. 2018. Disponível em: <https://arxiv.org/abs/1801.06601>. Cited on page 75.

LAMBERTI, L. et al. *Tiny-PULP-Dronets: Squeezing Neural Networks for Faster and Lighter Inference on Multi-Tasking Autonomous Nano-Drones*. 2022. Proc. IEEE AICAS. L. Lamberti *et al.*, "Tiny-PULP-Dronets: Squeezing Neural Networks for Faster and Lighter Inference on Multi-Tasking Autonomous Nano-Drones," in *Proc. IEEE AICAS*, 2022, doi: 10.1109/AICAS54282.2022.9869995. Cited on page 65.

LASKI, P. A.; SMYKOWSKI, M. *Using a Development Platform with an STM32 Processor to Prototype an Inexpensive 4-DoF Delta Parallel Robot*. 2021. PMC. "Using a Development Platform with an STM32 Processor to Prototype an Inexpensive 4-DoF Delta Parallel Robot," *PMC*, 2021. Cited on page 61.

LEE, D. et al. Keystone: An open framework for architecting trusted execution environments. In: *Proc. EuroSys*. [S.l.: s.n.], 2020. Cited 3 times on pages 46, 88, and 92.

LEE, J. et al. *RISC-V FPGA Platform Toward ROS-Based Robotics Application*. 2020. Proc. 30th Int. Conf. Field-Programmable Logic and Applications (FPL). J. Lee, H. Chen, J. Young, and H. Kim, "RISC-V FPGA Platform Toward ROS-Based Robotics Application," in *Proc. 30th Int. Conf. Field-Programmable Logic and Applications (FPL)*, IEEE, 2020. Cited on page 64.

LEVY, A. et al. Multiprogramming a 64kb computer safely and efficiently. In: *Proc. 26th ACM Symposium on Operating Systems Principles (SOSP)*. [S.l.: s.n.], 2017. p. 234–251. Tock OS — Rust-based embedded RTOS. Cited on page 36.

LI, X. et al. Design and verification of the Arm confidential compute architecture. In: *Proc. 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. [S.l.: s.n.], 2022. p. 465–484. Cited 4 times on pages 23, 32, 91, and 97.

LI, X. et al. A secure and formally verified Linux KVM hypervisor. In: *Proc. IEEE Symposium on Security and Privacy (S&P)*. [S.l.: s.n.], 2021. p. 1782–1799. Cited on page 32.

LI, Y.-J.; WANG, Y.-J. *Control and Design of Dancing Robot Based on STM32*. 2023. J. Electrical Eng. Autom. Y.-J. Li and Y.-J. Wang, "Control and Design of Dancing Robot Based on STM32," *J. Electrical Eng. Autom.*, vol. 5, no. 1, pp. 29–45, 2023. Cited on page 61.

LILJESTRAND, H. et al. PAC it up: Towards pointer integrity using ARM pointer authentication. In: *Proc. 28th USENIX Security Symposium*. [S.l.: s.n.], 2019. p. 177–194. Cited 3 times on pages 23, 31, and 93.

LIN, I.-T. et al. *A 28nm 142mW Motion-Control SoC for Autonomous Mobile Robots*. 2023. Proc. IEEE ISSCC. I.-T. Lin *et al.*, "A 28nm 142mW Motion-Control SoC for Autonomous Mobile Robots," in *Proc. IEEE ISSCC*, 2023. Cited on page 70.

LIN, T.-H.; CHANG, C.-T.; PUTRANTO, A. *Tiny Machine Learning Empowers Climbing Inspection Robots for Real-Time Multiobject Bolt-Defect Detection*. 2024. Eng. Appl. Artif. Intell. "Tiny Machine Learning Empowers Climbing Inspection Robots for Real-Time Multiobject Bolt-Defect Detection," *Eng. Appl. Artif. Intell.*, Elsevier, 2024. Cited on page 66.

LIPP, M. et al. Meltdown: Reading kernel memory from user space. In: *Proc. 27th USENIX Security Symposium*. [S.l.: s.n.], 2018. p. 973–990. Cited 2 times on pages 19 and 95.

Locuza. *Die Analysis of the Apple M1 Family*. 2021. Locuza's Semiconductor Newsletter, <https://locuza.substack.com/>. Annotated die-shot analysis of the M1, M1 Pro, and M1 Max. Cited on page 34.

LOUIS, M. S. et al. Towards deep learning using tensorflow lite on risc-v. In: ACM. *Third Workshop on Computer Architecture Research with RISC-V (CARRV)*. [S.l.], 2019. v. 1. Cited on page 81.

LU, T. A survey on RISC-V security: Hardware and architecture. *arXiv preprint*, 2021. ArXiv:2107.04175. Cited on page 87.

M, A. K. et al. *Implementation and Analysis of Custom Instructions on RISC-V for Edge-AI Applications*. 2024. Proc. HEART'24. "Implementation and Analysis of Custom Instructions on RISC-V for Edge-AI Applications," in *Proc. HEART'24*, ACM, 2024, doi: 10.1145/3665283.3665342. Cited 2 times on pages 64 and 68.

MADHUSUDAN, G. S. et al. *SHAKTI: An Open-Source Processor Ecosystem*. [S.l.], 2016. Cited 2 times on pages 19 and 98.

MARKETSANDMARKETS. *Edge AI Hardware Market Size, Share, Trends and Industry Analysis 2032*. 2032. MarketsandMarkets, "Edge AI Hardware Market Size, Share, Trends and Industry Analysis 2032," 2024. Cited on page 69.

MARKIDIS, S. et al. NVIDIA tensor core programmability, performance & precision. In: *Proc. IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. [S.l.: s.n.], 2018. Cited on page 53.

MARSHALL, B. et al. *RISC-V Cryptography Extensions Volume I: Scalar & Entropy Source Instructions, Version 1.0.1*. [S.l.], 2022. Cited 2 times on pages 92 and 96.

MARTINS, J. et al. A first look at RISC-V virtualization from an embedded systems perspective. *arXiv preprint arXiv:2103.14951*, 2021. Cited 3 times on pages 55, 56, and 59.

MARTINS, J. et al. Bao: A lightweight static partitioning hypervisor for modern multi-core embedded systems. In: *Workshop on Next Generation Real-Time Embedded Systems (NG-RES 2020)*. [S.l.: s.n.], 2020. (OASIcs, v. 77), p. 3:1–3:14. Cited 2 times on pages 90 and 96.

MERENDA, M.; PORCARO, C.; IERO, D. Edge machine learning for ai-enabled iot devices: A review. *Sensors*, v. 20, n. 9, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/9/2533>. Cited on page 77.

MEZA, A. et al. Security verification of the OpenTitan hardware root of trust. *IEEE Security & Privacy*, v. 21, n. 3, p. 27–36, 2023. Cited 2 times on pages 87 and 97.

Microchip Technology. *Integrating ROS 2 With Microchip's PolarFire SoC FPGA*. 2024. Microchip Technology, "Integrating ROS 2 With Microchip's PolarFire SoC FPGA," 2024. Cited 3 times on pages 64, 70, and 71.

MILLER, M. *Trends, Challenges, and Strategic Shifts in the Software Vulnerability Mitigation Landscape*. 2019. BlueHat IL 2019, Microsoft Security Response Center. Cited on page 93.

MITELOUDI, K. et al. Plan your defense: A comparative analysis of leakage detection methods on RISC-V cores. In: *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS 2024)*. [S.l.]: Springer, 2024. (LNCS, v. 15380), p. 139–151. Cited on page 96.

MUIR, D. R.; SHEIK, S. The road to commercial success for neuromorphic technologies. *Nature Communications*, v. 16, n. 1, p. 3586, 2025. ISSN 2041-1723. Disponível em: <https://doi.org/10.1038/s41467-025-57352-1>. Cited on page 83.

MUŁA, W. *RISC-V Vector Extension Overview*. 2024. <http://0x80.pl/notesen/2024-11-09-riscv-vector-extension.html>. Cited 2 times on pages 51 and 52.

MÜLLER, H.; KARTSCH, V.; BENINI, L. *GAP9Shield: A 150GOPS AI-Capable Ultra-Low Power Module for Vision and Ranging Applications on Nano-Drones*. 2024. "GAP9Shield: A 150GOPS AI-Capable Ultra-Low Power Module for Vision and Ranging Applications on Nano-Drones," preprint, 2024. Cited on page 63.

NVIDIA. *NVIDIA H100 Tensor Core GPU Architecture*. 2022. NVIDIA Whitepaper. Cited on page 53.

NVIDIA. *CUDA C++ Programming Guide*. 2024. V12.3. Cited on page 53.

NVIDIA. *NVIDIA Brings Generative AI Tools, Simulation and Perception Workflows to ROS Developer Ecosystem*. 2024. NVIDIA, "NVIDIA Brings Generative AI Tools, Simulation and Perception Workflows to ROS Developer Ecosystem," 2024. Cited on page 62.

NVIDIA. *Accelerating LLM and VLM Inference for Automotive and Robotics with NVIDIA TensorRT Edge-LLM*. 2025. NVIDIA, "Accelerating LLM and VLM Inference for Automotive and Robotics with NVIDIA TensorRT Edge-LLM," NVIDIA Technical Blog, 2025. Cited on page 66.

NVIDIA. *Getting Started with Edge AI on NVIDIA Jetson: LLMs, VLMs, and Foundation Models for Robotics*. 2025. NVIDIA, "Getting Started with Edge AI on NVIDIA Jetson: LLMs, VLMs, and Foundation Models for Robotics," NVIDIA Technical Blog, 2025. Cited on page 66.

NVIDIA. *Embedded Systems Developer Kits & Modules from NVIDIA Jetson*. 2026. NVIDIA, "Embedded Systems Developer Kits & Modules from NVIDIA Jetson," [Online]. Available: <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/>. Cited 4 times on pages 60, 62, 63, and 69.

NVIDIA. *Isaac ROS Visual SLAM*. 2026. NVIDIA, "Isaac ROS Visual SLAM," [Online]. Available: <https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_visual_slam>. Cited on page 65.

OLIVEIRA, D. B. de et al. Demystifying the real-time Linux scheduling latency. In: *Proc. 32nd Euromicro Conference on Real-Time Systems (ECRTS)*. [S.l.: s.n.], 2020. Cited on page 36.

Open Compute Project (AMD, Google, Microsoft, NVIDIA). *Caliptra: A Datacenter System on a Chip (SoC) Root of Trust (RoT)*. [S.l.], 2022. Revision 2.0. Cited on page 88.

OSVIK, D. A.; SHAMIR, A.; TROMER, E. Cache attacks and countermeasures: The case of AES. In: *Topics in Cryptology—CT-RSA 2006*. [S.l.]: Springer, 2006. (LNCS, v. 3860), p. 1–20. Cited on page 96.

PALOSSI, D. et al. A 64-mw DNN-based visual navigation engine for autonomous nano-drones. In: . [S.l.: s.n.], 2019. v. 6, n. 5, p. 8357–8371. Cited on page 37.

PALOSSI, D. et al. *A 64mW DNN-Based Visual Navigation Engine for Autonomous Nano-Drones*. 2019. IEEE Internet Things J. D. Palossi *et al.*, "A 64mW DNN-Based Visual Navigation Engine for Autonomous Nano-Drones," *IEEE Internet Things J.*, 2019, doi: 10.1109/JIOT.2019.2917066. Cited on page 63.

PALOSSI, D. et al. *Fully Onboard AI-Powered Human-Drone Pose Estimation on Ultra-Low Power Autonomous Flying Nano-UAVs*. 2021. IEEE Internet Things J. D. Palossi *et al.*, "Fully Onboard AI-Powered Human-Drone Pose Estimation on Ultra-Low Power Autonomous Flying Nano-UAVs," *IEEE Internet Things J.*, 2021, doi: 10.1109/JIOT.2021.3091643. Cited on page 63.

PATEL, A.; DARBARI, M. *Xvisor: Open Source Type-1 Hypervisor for ARM and RISC-V*. 2020. Cited on page 55.

PATTERSON, D. A. Reduced instruction set computers. *Communications of the ACM*, v. 28, n. 1, p. 8–21, 1985. Cited on page 24.

PATTERSON, D. A. Reduced instruction set computers. *Communications of the ACM*, v. 28, n. 1, p. 8–21, jan. 1985. Cited on page 41.

PATTERSON, D. A.; DITZEL, D. R. The case for the reduced instruction set computer. In: *Proc. 8th Annual Symposium on Computer Architecture (ISCA)*. [S.l.]: ACM, 1980. p. 25–33. Cited on page 24.

PATTERSON, D. A.; HENNESSY, J. L. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. [S.l.]: Morgan Kaufmann, 2017. Cited 2 times on pages 41 and 45.

PENG, T. et al. *An Evaluation of Embedded GPU Systems for Visual SLAM Algorithms*. 2020. Proc. IS&T Int. Symp. Electronic Imaging. T. Peng *et al.*, "An Evaluation of Embedded GPU Systems for Visual SLAM Algorithms," in *Proc. IS&T Int. Symp. Electronic Imaging*, 2020. Cited on page 65.

PEROTTI, M. et al. A "new ara" for vector computing: An open source highly efficient risc-v v 1.0 vector processor design. In: *2022 IEEE 33rd Intl. Conference on Application-specific Systems, Architectures and Processors (ASAP)*. [S.l.: s.n.], 2022. p. 43–51. Cited on page 74.

PINTO, S.; SANTOS, N. Demystifying Arm TrustZone: A comprehensive survey. In: . [S.l.: s.n.], 2019. v. 51, n. 6, p. 1–36. Cited 3 times on pages 27, 87, and 91.

POLLO, G. et al. *MEbots: Integrating a RISC-V Virtual Platform with a Robotic Simulator for Energy-Aware Design*. 2025. G. Pollo *et al.*, "MEbots: Integrating a RISC-V Virtual Platform with a Robotic Simulator for Energy-Aware Design," arXiv:2505.16682, 2025. Cited 2 times on pages 64 and 71.

POTTIER, N.; LAU, M. C. *A Modular Object Detection System for Humanoid Robots Using YOLO*. 2025. "A Modular Object Detection System for Humanoid Robots Using YOLO," arXiv:2510.13625, 2024. Cited on page 65.

PSA Joint Stakeholders. *Platform Security Model 1.1*. [S.l.], 2021. PSA Certified Specification, Document JSADEN014. Cited on page 87.

PULTE, C. et al. Simplifying ARM concurrency: Multicopy-atomic axiomatic and operational models for ARMv8. *Proc. ACM on Programming Languages (POPL)*, v. 2, n. POPL, p. 1–29, 2018. Cited on page 25.

QUALCOMM. *Qualcomm Robotics RB5 Development Kit*. 2026. Qualcomm, "Qualcomm Robotics RB5 Development Kit," [Online]. Available: <https://www.thundercomm.com/product/qualcomm-robotics-rb5-development-kit/>. Cited on page 62.

REDDI, V. J. et al. Mlperf inference benchmark. In: *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. [S.l.: s.n.], 2020. p. 446–459. Cited on page 81.

REGHENZANI, F.; MASSARI, G.; FORNACIARI, W. The real-time Linux kernel: A survey on PREEMPT_RT. In: . [S.l.: s.n.], 2019. v. 52, n. 1, p. 1–36. Cited on page 36.

REGHENZANI, F.; MASSARI, G.; FORNACIARI, W. The real-time Linux kernel: A survey on PREEMPT_RT. *ACM Computing Surveys*, v. 52, n. 1, p. Art. 18, fev. 2019. Cited 3 times on pages 50, 55, and 57.

RICO, A.; RAMIREZ, A.; VALERO, M. Available parallelism with the ARM scalable vector extension. In: . [S.l.: s.n.], 2017. v. 14, n. 2. Cited 2 times on pages 32 and 34.

RISC-V International. *RISC-V Supervisor Binary Interface (SBI) Specification*. [S.l.], 2019. Cited on page 87.

RISC-V International. *RISC-V 'V' Vector Extension, Version 1.0*. [S.l.], 2021. Ratified Sep. 2021. Cited 2 times on pages 51 and 52.

RISC-V International. *Smepmp Extension for PMP Enhancements for Memory Access and Execution Prevention in Machine Mode, Version 1.0*. [S.l.], 2021. Cited on page 89.

RISC-V International. *Sstc: Supervisor-Mode Timer Compare Extension*. 2021. In RISC-V Privileged Specification. Cited on page 47.

RISC-V International. *RISC-V Supervisor Binary Interface Specification*. [S.l.], 2022. Cited on page 47.

RISC-V International. *RISC-V Advanced Interrupt Architecture, Version 1.0*. [S.l.], 2023. Ratified 2023. Cited 2 times on pages 48 and 54.

RISC-V International. *RISC-V Platform Specification*. 2023. <https://lpc.events/event/11/contributions/1101/attachments/803/1570/riscv-platform-spec.pdf>. OS-A platform requires RVA22 profile compliance. Cited on page 44.

RISC-V International. *RISC-V Profiles, Version 1.0*. 2023. <https://docs.riscv.org/reference/profiles/rva20-rvi20-rva22/_attachments/RISC-V_Profiles.pdf>. Ratified Mar. 2023. Defines RVI20, RVA20, and RVA22 profiles. Cited 2 times on pages 43 and 44.

RISC-V International. *CoVE: Confidential VM Extension*. 2024. Task Group, 2024. Cited on page 46.

RISC-V International. *RISC-V Architecture Profiles Repository*. 2024. <https://github.com/riscv/riscv-profiles/blob/main/src/profiles.adoc>. Cited on page 43.

RISC-V International. *RISC-V Control-Flow Integrity Extensions: Zicfilp and Zicfiss*. [S.l.], 2024. Cited on page 94.

RISC-V International. *RISC-V IOMMU Specification, Version 1.0*. 2024. <https://github.com/riscv-non-isa/riscv-iommu>. Ratified 2024. Cited 2 times on pages 54 and 56.

RISC-V International. *RISC-V: The Open Standard Instruction Set Architecture*. 2024. <https://riscv.org>. Cited on page 19.

RISC-V International. *RVA23 Profile Specification Source*. 2024. <https://github.com/riscv/riscv-profiles/blob/main/src/rva23-profile.adoc>. Cited on page 44.

RISC-V International. *RVA23 Profile Specification, Version 1.0*. 2024. <https://lf-riscv.atlassian.net/wiki/display/HOME/RISC-V+Technical+Specifications>. Cited 2 times on pages 42 and 44.

RISC-V International. *Towards an Integrated Matrix Extension: Workload Analysis of CNN Inference with QEMU TCG Plugins*. 2025. Cited 2 times on pages 52 and 58.

RISC-V International. *About RISC-V*. 2026. <https://riscv.org/about/>. Accessed: 2026-03-25. Cited on page 41.

RISC-V International. *OpenSBI: RISC-V Open Source Supervisor Binary Interface*. 2026. <https://github.com/riscv-software-src/opensbi>. Accessed: 2026-03-25. Cited 2 times on pages 47 and 48.

RISC-V IOPMP Task Group. *RISC-V IOPMP Architecture Specification*. [S.l.], 2024. Cited on page 90.

ROS. *ROS 2 on Raspberry Pi*. 2026. ROS, "ROS 2 on Raspberry Pi," [Online]. Available: <https://docs.ros.org>. Cited on page 68.

RunTime Recruitment. *The Future of RISC-V in Embedded Systems: Opportunities and Challenges*. 2024. RunTime Recruitment, "The Future of RISC-V in Embedded Systems: Opportunities and Challenges," 2024. Cited 2 times on pages 68 and 70.

SÁ, B.; MARTINS, J.; PINTO, S. A first look at RISC-V virtualization from an embedded systems perspective. *IEEE Trans. Computers*, v. 71, n. 9, p. 2177–2190, 2022. Cited on page 89.

SAHITA, R. et al. CoVE: Towards confidential computing on RISC-V platforms. In: *Proc. Workshop Hardware and Architectural Support for Security and Privacy (HASP '23)*. [S.l.: s.n.], 2023. Cited on page 92.

Samsung Research. *RISC-V and Vectorization*. 2024. <https://research.samsung.com/blog/RISC-V-and-Vectorization>. Cited on page 51.

SATO, M. et al. Co-Design for A64FX manycore processor and Tofu interconnect D for the supercomputer Fugaku. *Fujitsu Technical Review*, 2020. Available at fujitsu.com/global/documents/about/resources/publications/fstj/. Cited on page 33.

SCHAFFNER, T.; ALTENBERG, J.; WALLENTOWITZ, S. Real-time Linux on RISC-V: Long-term performance analysis of PREEMPT_RT patches. In: CARRO, L.; REGAZZONI, F.; PILATO, C. (Ed.). *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. [S.l.]: Springer, 2025. Cited on page 57.

SEREBRYANY, K. ARM memory tagging extension and how it improves C/C++ memory safety. *;login: The USENIX Magazine*, v. 44, n. 2, p. 12–16, 2019. Cited on page 93.

SEREBRYANY, K. et al. Memory tagging and how it improves C/C++ memory safety. In: . [S.l.: s.n.], 2018. Cited 3 times on pages 23, 31, and 93.

ServeTheHome. *SiFive P870 RISC-V Processor at Hot Chips 2023*. 2023. <https://www.servethehome.com/sifive-p870-risc-v-processor-at-hot-chips-2023/>. Cited on page 49.

SHARMA, A. et al. Rust for embedded systems: Current state, challenges and open problems. In: *Proc. ACM CCS 2024*. [S.l.: s.n.], 2024. Cited on page 94.

SHUKLA, S. et al. An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment. *PLOS ONE*, Public Library of Science, v. 14, n. 11, p. 1–31, 11 2019. Disponível em: <https://doi.org/10.1371/journal.pone.0224934>. Cited on page 78.

SHUVO, M. M. H. et al. *Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review*. 2023. Proc. IEEE. M. M. H. Shuvo *et al.*, "Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review," *Proc. IEEE*, vol. 111, no. 1, pp. 42–91, 2023, doi: 10.1109/JPROC.2022.3226481. Cited on page 66.

SIEMENS. *isar-riscv: ROS 2 Support for RISC-V*. 2026. Siemens, "isar-riscv: ROS 2 Support for RISC-V," [Online]. Available: <https://github.com/siemens/isar-riscv>. Cited on page 64.

SiFive. *SiFive U74 Core IP*. 2021. <https://www.sifive.com/cores/u74>. Cited on page 49.

SiFive. *WorldGuard Specification*. 2021. Cited 3 times on pages 46, 56, and 88.

SiFive. SiFive Performance P870 processor. In: *Hot Chips 2023*. Stanford, CA: [s.n.], 2023. Cited 2 times on pages 40 and 49.

SIFIVE. *RISC-V Software Updates: Great Progress in 2024 and Much More Ahead in 2025*. 2024. SiFive, "RISC-V Software Updates: Great Progress in 2024 and Much More Ahead in 2025," 2024. Cited on page 70.

SiFive. *SiFive Automotive Family*. 2024. <https://www.sifive.com/cores/automotive>. Cited on page 54.

SiFive. *SiFive Performance P800 Series – P870-D*. 2024. <https://www.sifive.com/cores/performance-p800>. Cited 4 times on pages 49, 56, 58, and 59.

SiFive. *RISC-V Software Updates: Great Progress in 2024 and Much More Ahead in 2025*. 2025. SiFive Blog. Cited on page 42.

SiFive. *SiFive Enhances RISC-V Automotive Safety Leadership with Critical Functional Safety Certification*. 2025. SiFive Blog. Cited 3 times on pages 55, 56, and 58.

SiFive, Inc. *SiFive Functional Safety Cores*. 2024. <https://www.sifive.com/cores>. Cited on page 19.

SIMSON, N.; TAHIRAGA, A.; ECKER, W. *A Comparative Analysis of ARM and RISC-V ISAs for Deeply Embedded Systems*. 2024. Proc. IEEE (VDE) Conf. "A Comparative Analysis of ARM and RISC-V ISAs for Deeply Embedded Systems," in *Proc. IEEE (VDE) Conf.*, 2024, IEEE Xplore: 10564506. Cited on page 67.

SINHA, S. *State of IoT 2025: Number of connected IoT devices growing 14% to 21.1 billion globally*. 2025. IoT Analytics. Published October 28, 2025. Accessed in 2025. Disponível em: <https://iot-analytics.com/number-connected-iot-devices/>. Cited on page 73.

SpacemiT. *Key Stone K1 Octa-Core RISC-V Processor*. 2024. Cited 2 times on pages 48 and 49.

StarFive. *Analysis of Running Real-Time Linux on VisionFive 2*. 2024. <https://doc-en.rvspace.org/VisionFive2/RTLinux/>. Cited on page 57.

StarFive. *VisionFive 2 Documentation: PREEMPT_RT for RISC-V*. 2024. <https://doc-en.rvspace.org/VisionFive2/RTLinux/>. Cited 2 times on pages 47 and 48.

StarFive Technology. *JH7110 SoC Platform: Product Brief and Datasheet*. 2023. StarFive Technology Co., Ltd., <https://www.starfivetech.com/>. Quad-core SiFive U74 (RV64GC), 28 nm, Linux-capable multimedia SoC. Cited on page 48.

STEPHENS, N. et al. The ARM scalable vector extension. In: . [S.l.: s.n.], 2017. v. 37, n. 2, p. 26–39. Cited 5 times on pages 23, 28, 31, 32, and 33.

STEPHENS, N. et al. The ARM scalable vector extension. *IEEE Micro*, v. 37, n. 2, p. 26–39, mar. 2017. Cited 2 times on pages 51 and 52.

SUÁREZ, D.; ALMEIDA, F.; BLANCO, V. Comprehensive analysis of energy efficiency and performance of arm and risc-v socs. *J. Supercomput.*, Kluwer Academic Publishers, USA, v. 80, n. 9, p. 12771–12789, fev. 2024. ISSN 0920-8542. Disponível em: <https://doi.org/10.1007/s11227-024-05946-9>. Cited 3 times on pages 67, 81, and 84.

SULEIMAN, A. et al. *Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones*. 2019. IEEE J. Solid-State Circuits. A. Suleiman *et al.*, "Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones," *IEEE J. Solid-State Circuits*, vol. 54, no. 4, pp. 1106–1119, 2019, doi: 10.1109/JSSC.2018.2886342. Cited on page 66.

T-Head Semiconductor. *XuanTie C906 User Manual*. 2020. Cited on page 49.

Tenstorrent. *Ocelot: Open-Source Out-of-Order RISC-V Processor with Vector Extension*. 2023. <https://tenstorrent.com/en/ip/risc-v-cpu>. Cited on page 49.

Texas Instruments. *Arm Cortex-R MCUs*. 2026. Texas Instruments, "Arm Cortex-R MCUs," [Online]. Available: <https://www.ti.com>. Cited on page 62.

The Linux Foundation. *ELISA: Enabling Linux in Safety Applications*. 2026. <https://elisa.tech/>. Accessed: 2026-03-25. Cited on page 57.

The Register. *Alibaba Launches Server-Grade RISC-V CPU Design*. 2025. <https://www.theregister.com/2025/03/05/china_alibaba_risc_v_c930/>. Cited on page 49.

THOMAS, J. RISC-V: Achieving functional safety and security. *Microcontroller Tips*, mar. 2025. Cited 2 times on pages 57 and 58.

THOMPSON, A. Addressing the memory bottleneck with in-memory computing. *Scilight*, v. 2023, n. 7, p. 071106, 02 2023. ISSN 2572-7907. Disponível em: <https://doi.org/10.1063/10.0017431>. Cited on page 83.

Tom's Hardware. *Tenstorrent Shares Roadmap of Ultra-High-Performance RISC-V CPUs and AI Accelerators*. 2023. Cited on page 50.

TRIPATHY, S. S. et al. Fedhealthfog: A federated learning-enabled approach towards healthcare analytics over fog computing platform. *Heliyon*, v. 10, n. 5, p. e26416, 2024. ISSN 2405-8440. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405844024024472>. Cited on page 78.

Trusted Firmware Project (Linaro). *Trusted Firmware-M Technical Overview*. [S.l.], 2023. Cited on page 87.

TS2 Tech. *RISC-V vs ARM vs x86: The 2025 Silicon Architecture Showdown*. 2025. <https://ts2.tech/en/risc-v-vs-arm-vs-x86-the-2025-silicon-architecture-showdown/>. Cited 2 times on pages 40 and 41.

TSAI, S.-E.; YANG, S.-M.; SUN, W.-C. *The Study on Real-Time RRT-Based Path Planning for UAVs Using a STM32 Microcontroller*. 2025. "The Study on Real-Time RRT-Based Path Planning for UAVs Using a STM32 Microcontroller," Preprints.org, 2025. Cited on page 61.

TU, F. et al. *Robotic Computing System and Embodied AI Evolution: An Algorithm-Hardware Co-Design Perspective*. 2025. J. Semiconductors. F. Tu *et al.*, "Robotic Computing System and Embodied AI Evolution: An Algorithm-Hardware Co-Design Perspective," *J. Semiconductors*, 2025. Cited 2 times on pages 60 and 70.

VAGHELA, R. et al. Optimizing object detection for autonomous robots: a comparative analysis of yolo models. *Measurement*, v. 257, p. 118676, 2026. ISSN 0263-2241. Cited on page 65.

VALENTE, L. et al. *A Heterogeneous RISC-V Based SoC for Secure Nano-UAV Navigation*. 2024. IEEE Trans. Circuits Syst. I. L. Valente *et al.*, "A Heterogeneous RISC-V Based SoC for Secure Nano-UAV Navigation," *IEEE Trans. Circuits Syst. I*, 2024, arXiv:2401.03531. Cited on page 65.

Vander Stoep, J. *Memory Safe Languages in Android 13*. 2022. Google Security Blog. <https://security.googleblog.com/2022/12/memory-safe-languages-in-android-13.html>. Cited on page 93.

Vander Stoep, J. *Rust in Android: move fast and fix things*. 2025. Google Security Blog. <https://security.googleblog.com/2025/11/rust-in-android-move-fast-fix-things.html>. Cited on page 95.

Ventana Micro Systems. *Veyron V1 RISC-V Processor*. 2022. Cited on page 49.

VERMA, V. Ai-risc: Scalable risc-v processor for iot edge ai applications. *University of Virginia, Electrical Engineering-School of Engineering and Applied Science, PHD (Doctor of Philosophy)*, 2022. Cited on page 81.

WANG, S. The rise of RISC-V and ISO 26262 compliance. *Electronic Design*, maio 2024. Cited on page 58.

WANG, S. et al. *RV-SCNN: A RISC-V Processor With Customized Instruction Set for SNN and CNN Inference Acceleration on Edge Platforms*. 2024. IEEE Trans. Comput.-Aided Design. S. Wang *et al.*, "RV-SCNN: A RISC-V Processor With Customized Instruction Set for SNN and CNN Inference Acceleration on Edge Platforms," *IEEE Trans. Comput.-Aided Design*, 2024, doi: 10.1109/TCAD.2024.3472293. Cited on page 64.

WARD-FOXTON, S. Tenstorrent productizes RISC-V CPU and AI IP. *EE Times*, set. 2025. Cited on page 50.

WATERMAN, A.; ASANOVIĆ, K. *The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA, Document Version 20191213*. [S.l.], 2019. Cited 5 times on pages 41, 42, 45, 47, and 50.

WATERMAN, A.; ASANOVIĆ, K. *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 20211203*. [S.l.], 2021. Cited 4 times on pages 46, 47, 54, and 56.

WATERMAN, A.; ASANOVIĆ, K.; RISC-V Foundation. *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture*. [S.l.], 2021. Version 20211203. Cited on page 89.

WATERMAN, A. et al. *The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA*. [S.l.], 2011. Cited on page 41.

WATERMAN, A. et al. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0*. [S.l.], 2014. Cited 3 times on pages 40, 41, and 45.

WATSON, R. N. M. et al. The Arm Morello evaluation platform—validating CHERI-based security in a high-performance system. *IEEE Micro*, v. 43, n. 3, p. 50–57, 2023. Cited on page 94.

WATSON, R. N. M. et al. *Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 9)*. [S.l.], 2023. Cited on page 94.

WATSON, R. N. M. et al. CHERI: A hybrid capability-system architecture for scalable software compartmentalization. In: *Proc. 36th IEEE Symp. Security and Privacy (Oakland)*. [S.l.: s.n.], 2015. p. 20–37. Cited on page 94.

WEAVER, D.; MCINTOSH-SMITH, S. *An Empirical Comparison of the RISC-V and AArch64 Instruction Sets*. 2023. Proc. SC'23: Int. Conf. HPC. "An Empirical Comparison of the RISC-V and AArch64 Instruction Sets," in *Proc. SC'23: Int. Conf. HPC*, ACM, 2023, doi: 10.1145/3624062.3624233. Cited 2 times on pages 67 and 71.

WEAVER, V. M. et al. ISA comparison on supercomputing workloads. In: *Proc. ACM/IEEE Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*. [S.l.: s.n.], 2023. Cited on page 28.

WESSMAN, N.-J. et al. *De-RISC: A Complete RISC-V Based Space-Grade Platform*. 2022. Proc. IEEE Conf. "De-RISC: The First RISC-V Space-Grade Platform for Safety-Critical Systems," in *Proc. IEEE Conf.*, 2021, IEEE Xplore: 9546286. Cited on page 68.

WILHELM, R. et al. The worst-case execution-time problem—overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems (TECS)*, v. 7, n. 3, p. 1–53, 2008. Cited 3 times on pages 27, 36, and 37.

WILLIAMS, S.; WATERMAN, A.; PATTERSON, D. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, v. 52, n. 4, p. 65–76, abr. 2009. Cited on page 54.

WOODRUFF, J. et al. The CHERI capability model: Revisiting RISC in an age of risk. In: *Proc. ACM/IEEE 41st Int. Symp. Computer Architecture (ISCA)*. [S.l.: s.n.], 2014. p. 457–468. Cited on page 94.

XCP-ng. *Xen and the RISC-V Hypervisor Extension*. 2024. <https://xcp-ng.org/blog/2021/03/12/xen-and-the-risc-v-hypervisor-extension/>. Cited 2 times on pages 54 and 55.

XPU.pub. *Tenstorrent Licenses RISC-V CPU and NPU IP*. 2025. <https://xpu.pub/2025/10/09/tenstorrent-ascalon/>. Cited 2 times on pages 50 and 58.

XU, H. et al. *D^2SLAM: Decentralized and Distributed Collaborative Visual-Inertial SLAM System for Aerial Swarm*. 2024. IEEE Trans. Robot. H. Xu *et al.*, "D^2SLAM: Decentralized and Distributed Collaborative Visual-Inertial SLAM System for Aerial Swarm," *IEEE Trans. Robot.*, vol. 40, pp. 3445–3464, 2024, doi: 10.1109/TRO.2024.3422003. Cited on page 65.

XUE, M. et al. Ten years of hardware trojans: A survey from the attacker's perspective. *IET Computers & Digital Techniques*, v. 14, n. 6, p. 231–246, 2020. Cited on page 97.

YIU, J. The definitive guide to ARM Cortex-M3 and Cortex-M4 processors. In: . [S.l.]: Newnes/Elsevier, 2013. ISBN 978-0124080829. Cited on page 27.

YOO, T.; CHOI, B. W. *Real-Time Performance Benchmarking of RISC-V Architecture: Implementation and Verification on an EtherCAT-Based Robotic Control System*. 2024. Electronics. T. Yoo and B. W. Choi, "Real-Time Performance Benchmarking of RISC-V Architecture: Implementation and Verification on an EtherCAT-Based Robotic Control System," *Electronics*, vol. 13, no. 4, p. 733, 2024, doi: 10.3390/electronics13040733. Cited 2 times on pages 57 and 64.

ZHAI, W.; DONG, L. *Two-Wheeled Self-Balancing Mobile Robot Control System Based on STM32-MAT and Android*. 2025. J. Measurements Eng. W. Zhai and L. Dong, "Two-Wheeled Self-Balancing Mobile Robot Control System Based on STM32-MAT and Android," *J. Measurements Eng.*, vol. 13, no. 3, 2025. Cited on page 61.

ZHU, W.; GOUDARZI, M.; BUYYA, R. Flight: A lightweight federated learning framework in edge and fog computing. *Software: Practice and Experience*, v. 54, n. 5, p. 813–841, 2024. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3300>. Cited on page 78.

ZINI, M.; CASINI, D.; BIONDI, A. Analyzing Arm's MPAM from the perspective of time predictability. *IEEE Trans. Computers*, v. 72, n. 1, p. 59–74, 2023. Cited on page 90.

ZÜPKE, A.; KAISER, R. Deterministic real-time guarantees in the presence of virtual memory on ARM multicore platforms. In: *Proc. IEEE Real-Time Systems Symposium (RTSS)*. [S.l.: s.n.], 2023. Cited on page 36.