Politecnico di Torino

Computer Engineering

A.y. 2023/2024

Graduation Session March 2024

Accelerating Real-Time Edge AI

Unraveling the potential of the VE2302 in the AMD landscape

Supervisors:

Bartolomeo Montrucchio

Candidate:

Lorenzo Radaele

Acknowledgements

I am profoundly grateful to all the teachers and individuals who played a significant role in my lifelong learning journey, shaping me into the person I am today; special thanks to Bartolomeo Montrucchio for accepting to be my supervisor. A heartfelt thanks to Avnet Embedded for providing me with the opportunity to work on this project for my thesis, and to all my colleagues for their unwavering support and collaboration. Special thanks to Mario Bergeron for helping me go through all the experiments and for answering all of my questions.

I extend my deepest gratitude to my friend Fabiana Racca for being always with me over these long years, I would still be in the third year without you. I also want to thank Gabriele Di Bartolomei, Davide, Xiao, Gabriele, Lorenzo Ferro, Alessandro, Fabio Trovero, and Fabio Sofer for their assistance in studying and tackling various projects together.

Special thanks to my roommates Guido Vitali and Lorenzo Castiglione, who shared this adventure with me in Turin. It was an incredible journey, and I had so much fun with both of you.

Heartfelt thanks to my parents, Carla and Piero, for allowing me to stay in Turin during my university years and for always believing in me. I appreciate my sister Simona for consistently being by my side through every experience we faced together. My gratitude extends to my grandfather for passing on his passion for engineering to me.

I also want to express my gratitude to all my friends and everyone who accompanied me throughout these years, supporting and believing in me. I am deeply thankful for the love I received during my course of study. I acknowledge that I wasn't always the easiest person to be around, and your constant support means the world to me. I wouldn't be here without you. It's been a roller coaster of emotions, and I have ridden every high and low with gratitude.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Context

In a world where Artificial Intelligence (AI) and machine learning are experiencing unprecedented growth, industries are in a race against time to deliver cutting-edge products. These products aim to excel in various aspects, whether it be speed, accuracy, or versatility, offering a multitude of possibilities for exploration and research. In the realm of real-time applications, AMD, bolstered by its acquisition of Xilinx, is endeavoring to introduce innovative products with minimal inference time. This strategic move opens doors to numerous opportunities.

The application of AI in real-time event processing allows businesses to discern patterns, identify trends, and respond swiftly to emerging threats and opportunities. Furthermore, the spotlight on self-driving applications in the automotive industry has intensified. Real-time data sourced from cameras, lidar, and radar plays a pivotal role in enabling AI-equipped self-driving cars to interpret their surroundings. Algorithms analyze this data in real-time, facilitating navigation, obstacle avoidance, and adherence to traffic rules. Projections indicate that the global automotive artificial intelligence market will reach a valuation of $74.5 billion by 2030 [1].

Xilinx has introduced a groundbreaking family of devices, the Versal Family, capable of achieving remarkable results. Versal adaptive System on Chips (SoCs) offer unparalleled value at both application and system levels, catering to cloud, network, and edge applications. These devices stand out for their dynamic customizability at both hardware and software levels, making them suitable for a wide spectrum of applications and workloads.

1.2 Research motivation

In the landscape of adaptive compute acceleration platforms, the Versal Series by Xilinx stands out as a versatile family designed to cater to specific application needs. Among these, the Versal AI Edge Series holds particular significance. Described on the Xilinx website as a series that "delivers high performance, low latency AI inference for intelligence in automated driving, predictive factory and healthcare systems, multi-mission payloads in aerospace and defense, and a breadth of other applications" [2], it has garnered attention for its capabilities in diverse real-world scenarios.

The VEK280, positioned as the most comprehensive board within the Versal AI Edge Series, has demonstrated remarkable performance. However, it faces challenges in reaching the market due to its high production cost, compounded by its original intent for evaluation purposes. Now under the ownership of AMD, Xilinx recognizes the need for a more accessible and cost-effective board that maintains high standards. The goal is to create a device that not only addresses affordability concerns but also delivers optimal performance, particularly in achieving the lowest possible inference time.

This strategic shift positions Xilinx, now part of AMD, competitively in the real-time application field. The goal is to offer a board that blends superior performance with an affordable price, unlocking possibilities across industries like automated driving, factory automation, healthcare, aerospace, and defense. The VE2302, explored in this thesis, aims to find the sweet spot between accessibility and high-performance in the realm of edge AI devices.

1.3 Contributions

My contributions are as follows:

- In **Chapter 3** I give a comprehensive exploration and design of the VE2302, navigating through the reference design made by the team I am working with, covering its core components, System-on-Module (SOM), and Carrier Board describing all the different interfaces that will be implemented.

- In **Chapter 4** I examine the Board Definition Files (BDF) creation process, and I write all the files needed for the development tools to be able to target the incoming board for the build. I group all the information from the various documents and guides to make a clearer explanation of how to implement these files.

- In **Chapter 5** I contributed extensively to the benchmarking phase, working collaboratively with Mario to assess the performance metrics of various non-Versal and Versal devices, including the VE2302. Employed meticulous testing procedures to measure throughput, latency, and power consumption, focusing on gathering valuable insights into the VE2302's performance characteristics. I engaged in the identification of optimal configurations and collaborated closely with Mario to provide a comparative analysis of different Versal boards and three other prominent boards. The data-driven conclusions drawn from this benchmarking effort offer valuable insights for decision-makers and researchers in the field of edge AI devices.

1.4 Pubblications

The material produced for this thesis in Chapter 4 has resulted in a peer-reviewed publication described below:

- Bryan Fletcher, Tom Curran, Daniel Rozwood, and Tnizan. "**Avnet Board Definition Files $\longrightarrow$ ve2302_iocc-dev**" `https://github.com/Avnet/bdf/tree/ve2302_iocc-dev`

1.5 Thesis structure

This thesis is organized into six chapters. In **Chapter 2**, I review all the material needed to understand the project, this includes an exploration of the Vitis AI library, an overview of the Versal Family devices, and a detailed examination of their AI Engine. Additionally, I introduce the leading board that will serve as a reference design for the new device.

Chapter 3 delves into the actual design of the new device, breaking it down into the main core, the System on Module (SOM), and the Carrier Board. These components collectively offer the various interfaces needed to fully exploit the functionalities of the DPU in the new VE2302.

In **Chapter 4**, I walk through the process of defining the Board Definition Files required to build and run applications on the upcoming board. This chapter provides a more detailed exploration of the various components and interfaces of the two devices considered as a whole. It also describes how the various XML files are written.

Chapter 5 involves a thorough evaluation where my colleague Mario and I compare different boards that we obtained. Additionally, we assess how the architecture of the new board is expected to perform compared to the best-available device from the same family.

Chapter 6 concludes the thesis with a discussion of the obtained results and offers perspectives on the future of the project and potential related works.

Chapter 2

Premises

2.1 Vitis AI

In the rapidly evolving landscape of artificial intelligence (AI) and machine learning (ML), optimizing the performance of AI workloads has become paramount. Vitis AI stands out as a robust and versatile development platform specifically tailored for accelerating AI applications on Xilinx hardware. Vitis AI harnesses the power of Field-Programmable Gate Arrays (FPGAs) and adaptable System on Chips (SoCs) to deliver efficient and high-performance AI inferencing capabilities [3].
The Vitis AI solution consists of three primary components:

- The Deep-Learning Processor unit (DPU), a hardware engine for optimized the inferencing of ML models
- Model development tools, to compile and optimize ML models for the DPU
- Model deployment libraries and APIs, to integrate and execute the ML models on the DPU engine from a SW application

The Vitis AI solution is packaged and delivered as follows:

- AMD open download: pre-built target images integrating the DPU
- Vitis AI docker containers: model development tools
- Vitis AI github repository: model deployment libraries, setup scripts, examples and reference design

2.1.1 Deep-learning Processor Unit

The Deep-learning Processor Unit (DPU) stands as a programmable engine meticulously optimized for the swift execution of deep neural networks. Featuring an efficient tensor-level instruction set, the DPU is engineered to support and accelerate a spectrum of widely adopted convolutional neural networks. Notable examples include VGG, ResNet, GoogLeNet, YOLO, SSD, and MobileNet, among others.

Notably versatile, the DPU extends its support to a range of platforms, encompassing AMD Zynq UltraScale+ MPSoCs, the Kria KV260, Versal, and Alveo cards. Its scalability caters to diverse application needs, including variations in throughput, latency, scalability, and power efficiency.

AMD further simplifies the deployment of the DPU by providing pre-built platforms tailored for both edge and data-center cards. These platforms empower data scientists to initiate the development and testing of models without necessitating hardware development expertise.

For embedded applications, the integration of the DPU into a custom platform is essential, aligning seamlessly with other programmable logic functions within the FPGA or adaptive SoC device. Hardware designers can accomplish this integration using either the Vitis flow or the Vivado Design Suite. This flexibility ensures that the DPU can be seamlessly incorporated into custom solutions, offering a tailored approach to meeting specific application requirements.

2.1.2 Vitis AI Library

The Vitis AI Library represents a comprehensive suite of high-level libraries and APIs meticulously crafted for efficient AI inference utilizing the Deep-Learning Processor Unit (DPU). Built upon the Vitis AI runtime and seamlessly supporting XRT 2023.1, this library streamlines the development process by providing unified APIs and encapsulating various efficient and high-quality neural networks.

Designed to facilitate AI application development, the Vitis AI Library offers an accessible and unified interface, making it user-friendly even for those without prior knowledge of deep learning or FPGA intricacies. By abstracting the complexities of underlying hardware, it empowers developers to focus on crafting applications rather than dealing with intricate hardware details.

The Vitis AI Library comprises four essential components, as illustrated in the block diagram:

1. **Base Libraries:**
   - These libraries furnish a fundamental programming interface with the DPU, encompassing available post-processing modules for each model.
   - Key components include:

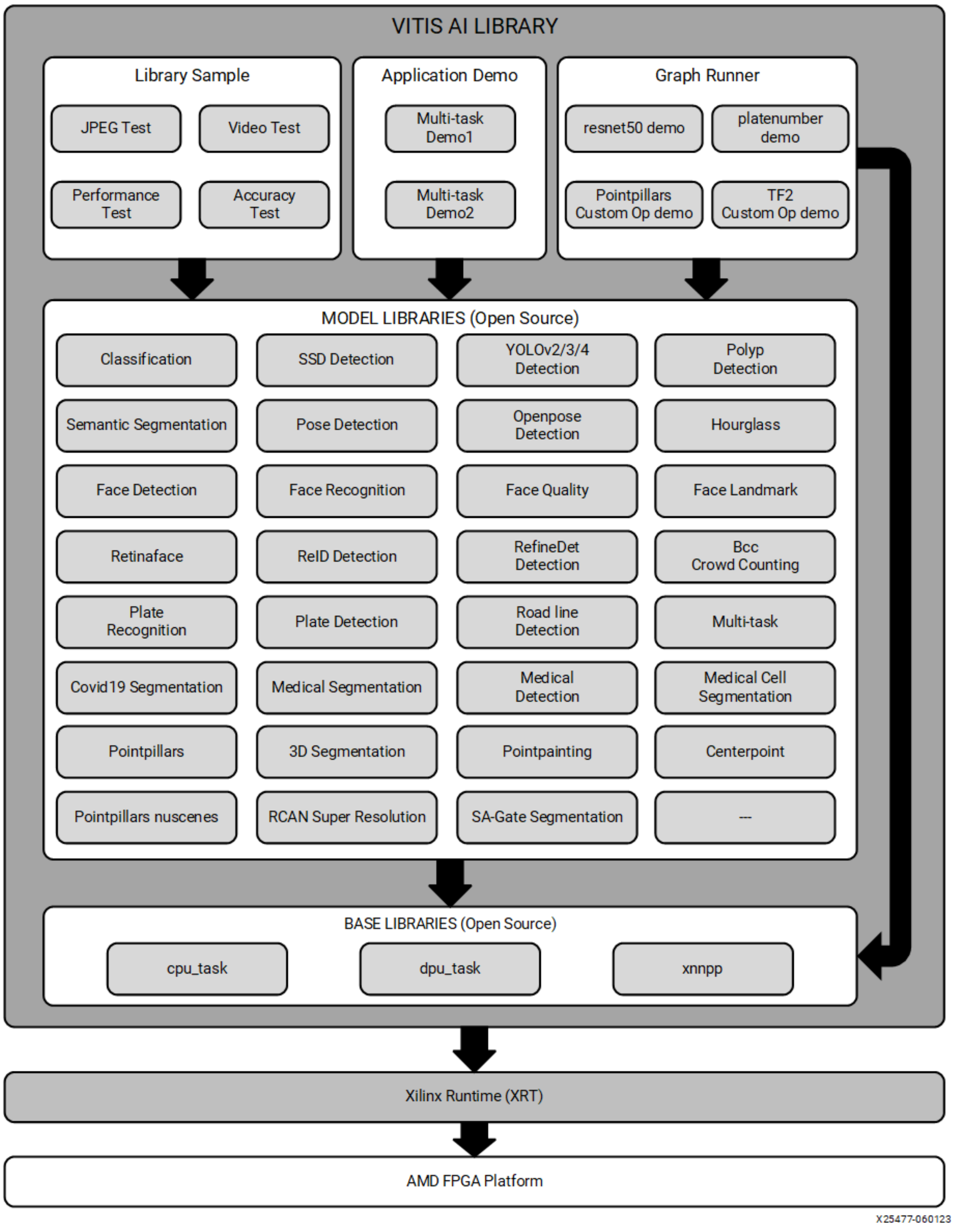


Figure 2.1: Vitis AI Library Block Diagram

    - `dpu_task`: The interface library for DPU operations.
    - `cpu_task`: The interface library for operations assigned to the CPU.
    - `xnnpp`: The post-processing library for each model, featuring built-in modules like optimization and acceleration.

2. **Model Libraries:**
    - These libraries embody a broad spectrum of open-source neural network deployments, covering common types like classification, detection, segmentation, and more.
    - They provide a unified and straightforward interface applicable to AMD models or custom models.

3. **Library Samples:**
    - Test samples within the library serve as quick tools for evaluating and testing model libraries.

4. **Application Demos:**
    - Application demos showcase how the Vitis AI Library can be effectively utilized to develop diverse applications.

Key Features:

- A comprehensive, end-to-end application solution.
- Optimized pre-processing and post-processing functions/libraries.
- Inclusion of open-source model libraries.
- Unified operation interface covering DPU, pre-processing, and post-processing.
- Practical, application-based model libraries, pre-processing and post-processing libraries, along with application examples.

In essence, the Vitis AI Library is a powerful toolset offering a simplified yet robust environment for AI inference development, fostering efficiency and ease of use throughout the application development lifecycle.

2.2 AI Engine

AMD Versal adaptive system-on-chips (SoCs) integrate Scalar Engines, Adaptable Engines, and Intelligent Engines, alongside cutting-edge memory and interfacing technologies. This combination delivers potent heterogeneous acceleration suitable for a diverse range of applications. Importantly, Versal adaptive SoCs are designed to be programmable and optimized by data scientists, software developers, and hardware developers alike. The hardware and software ecosystem is supported by a comprehensive set of tools, software, libraries, IP, middleware, and frameworks, facilitating integration into industry-standard design flows.

Built on TSMC's 7 nm FinFET process technology, the Versal portfolio stands as the first platform to seamlessly merge software programmability, domain-specific hardware acceleration, and the adaptability required to match the rapid pace of innovation. The portfolio consists of six series of devices uniquely architected for scalability and AI inference capabilities across various markets, including cloud computing, networking, wireless communications, edge computing, and endpoints.

The Versal architecture integrates different engine types with extensive connectivity, communication capabilities, and a network on chip (NoC). This integration enables seamless memory-mapped access to the entire device. Intelligent Engines feature SIMD VLIW AI Engines for adaptive inference and advanced signal processing compute, while DSP(digital signal processing) Engines handle fixed point, floating point, and complex MAC operations. Adaptable Engines combine programmable logic blocks and memory for high-compute density, and Scalar Engines, including Arm Cortex-A72 and Cortex-R5F processors, cater to intensive compute tasks.

The Versal AI Core series showcases breakthrough AI inference acceleration with AI Engines delivering over 100x greater compute performance than current server-class CPUs. This series targets applications in the cloud with dynamic workloads and network scenarios demanding massive bandwidth, all while ensuring advanced safety and security features. The AI Engine's advanced signal processing compute capability makes it well-suited for highly optimized wireless applications, including radio, 5G, backhaul, and other high-performance DSP applications.

AI Engines constitute an array of very-long instruction word (VLIW) processors with single instruction multiple data (SIMD) vector units, finely tuned for compute-intensive applications such as digital signal processing (DSP), 5G wireless applications, and artificial intelligence (AI) technologies like machine learning (ML). These hardened blocks offer multiple levels of parallelism, including instruction-level and data-level parallelism.

The AI Engine-ML (AIE-ML) block within the AI Engines is designed to deliver 2x compute throughput compared to its predecessor. Primarily targeted for machine learning inference applications, the AIE-ML block achieves one of the industry's best performance per Watt for a wide range of inference applications.

Programming the AI Engine array requires a thorough understanding of the algorithm to be implemented, the capabilities of the AI Engines, and the overall data flow between individual functional units. The AI Engine array supports three levels of parallelism:

- **SIMD (Single Instruction, Multiple Data):** Through vector registers that allow multiple elements to be computed in parallel.
- **Instruction Level Parallelism (VLIW Architecture):** Through the VLIW architecture that allows multiple instructions to be executed in a single clock cycle.
- **Multicore Parallelism:** Through the AI Engine array, where many hundreds of AI Engines can execute in parallel.

The AI Engine array consists of a 2D array of AI Engine tiles, where each AI Engine tile contains an AI Engine, memory module, and tile interconnect module [4].

- **AI Engine:** Each AI Engine is a very long instruction word (VLIW) processor containing a scalar unit, a vector unit, two load units, and a single store unit.
- **AI Engine Tile:** An AI Engine tile contains an AI Engine, a local memory module together with several communication paths to facilitate data exchange between tiles.
- **AI Engine Array:** AI Engine array refers to the complete 2D array of AI Engine tiles.
- **AI Engine Program:** The AI Engine program consists of a data flow graph specification written in C/C++. This program is compiled and executed using the AI Engine toolchain.
- **AI Engine Kernels:** Kernels are written in C/C++ using AI Engine vector data types and intrinsic functions. These are the computation functions running on an AI Engine. The kernels form the fundamental building blocks of a data flow graph specification.
- **ADF Graph:** An ADF graph is a network with a single AI Engine kernel or multiple AI Engine kernels connected by data streams. It interacts with the programmable logic, global memory, and processing system with specific constructs like PLIO (port attribute in the graph programming that is used to make stream connections to or from the programmable logic), GMIO (port attribute in the graph programming that is used to make external memory-mapped connections to or from the global memory), and RTP.

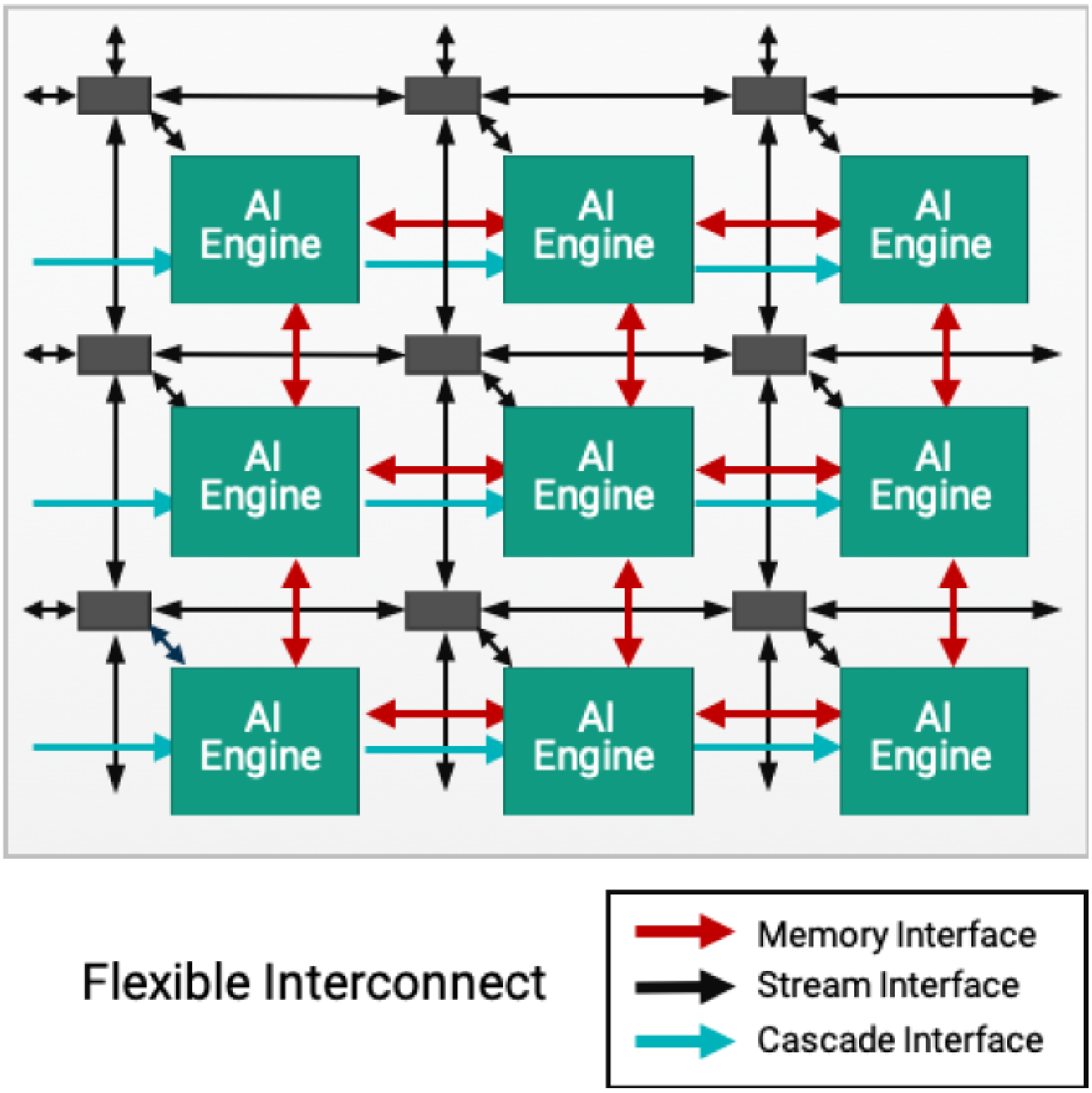


Figure 2.2: AI Engine Architecture

AMD provides two distinct types of AI Engines: AIE (AI Engine) and AIE-ML (AI Engine for Machine Learning). Both engines bring substantial performance enhancements compared to previous-generation FPGAs. AIE is designed to accelerate a well-balanced range of workloads, including ML inference applications and advanced signal processing tasks such as beamforming, radar processing, and other workloads that demand extensive filtering and transforms [5].

The AIE-ML, tailored for machine learning inference applications, delivers superior performance compared to AIE. It incorporates enhanced AI vector extensions and introduces shared memory tiles within the AI Engine array. This enhancement positions AIE-ML as the preferred choice for ML-focused applications. On the other hand, AIE may outperform AIE-ML in certain scenarios involving advanced signal-processing tasks.

2.2.1 AI Engine Tile

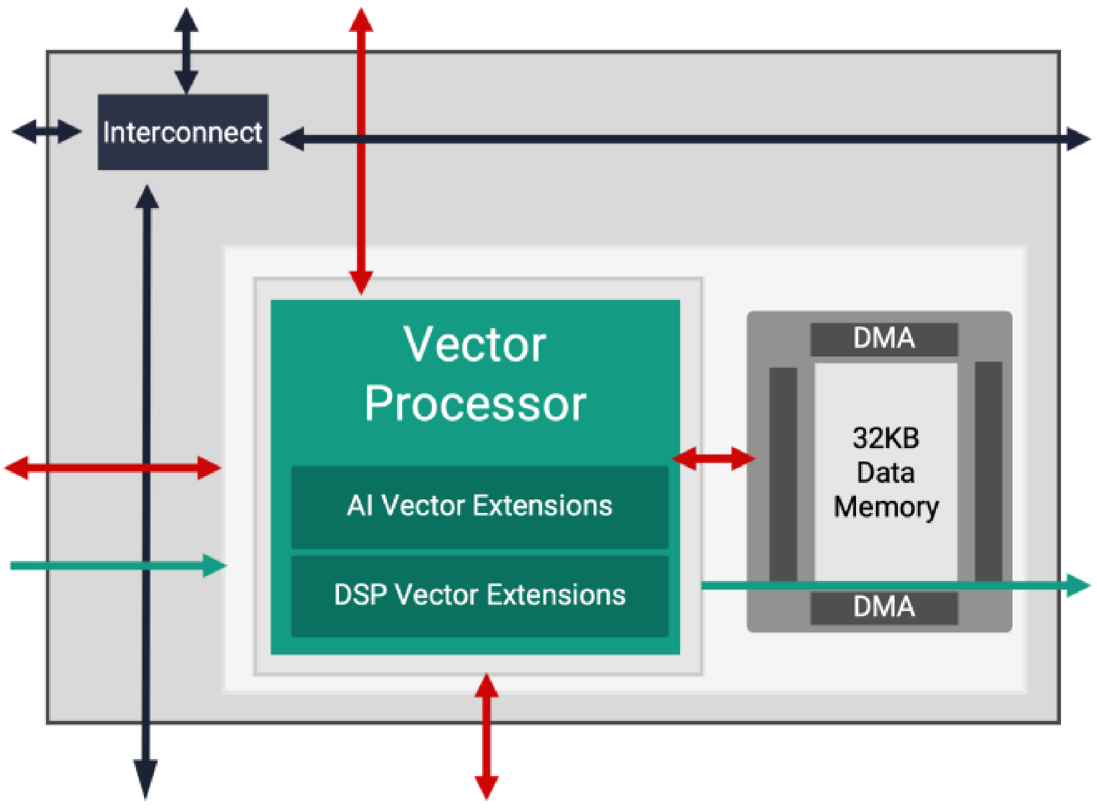


Figure 2.3: AI Engine Tile

AIE is designed to accelerate a diverse range of workloads, encompassing ML inference applications and advanced signal processing tasks such as beamforming, radar processing, FFTs, and filters. Key features include:

- Support for a broad spectrum of workloads and applications.
- Advanced DSP capabilities tailored for communications.
- Efficient processing of video and image data.
- Acceleration of machine learning inference tasks.

- Native support for real, complex, and floating-point data types, including INT8/16 fixed point, CINT16, CINT32 complex fixed point, and FP32 floating-point.
- Dedicated hardware features for optimized FFT and FIR implementations, including 128 INT8 MACs per tile.

2.2.2 AI Engine-ML Tile

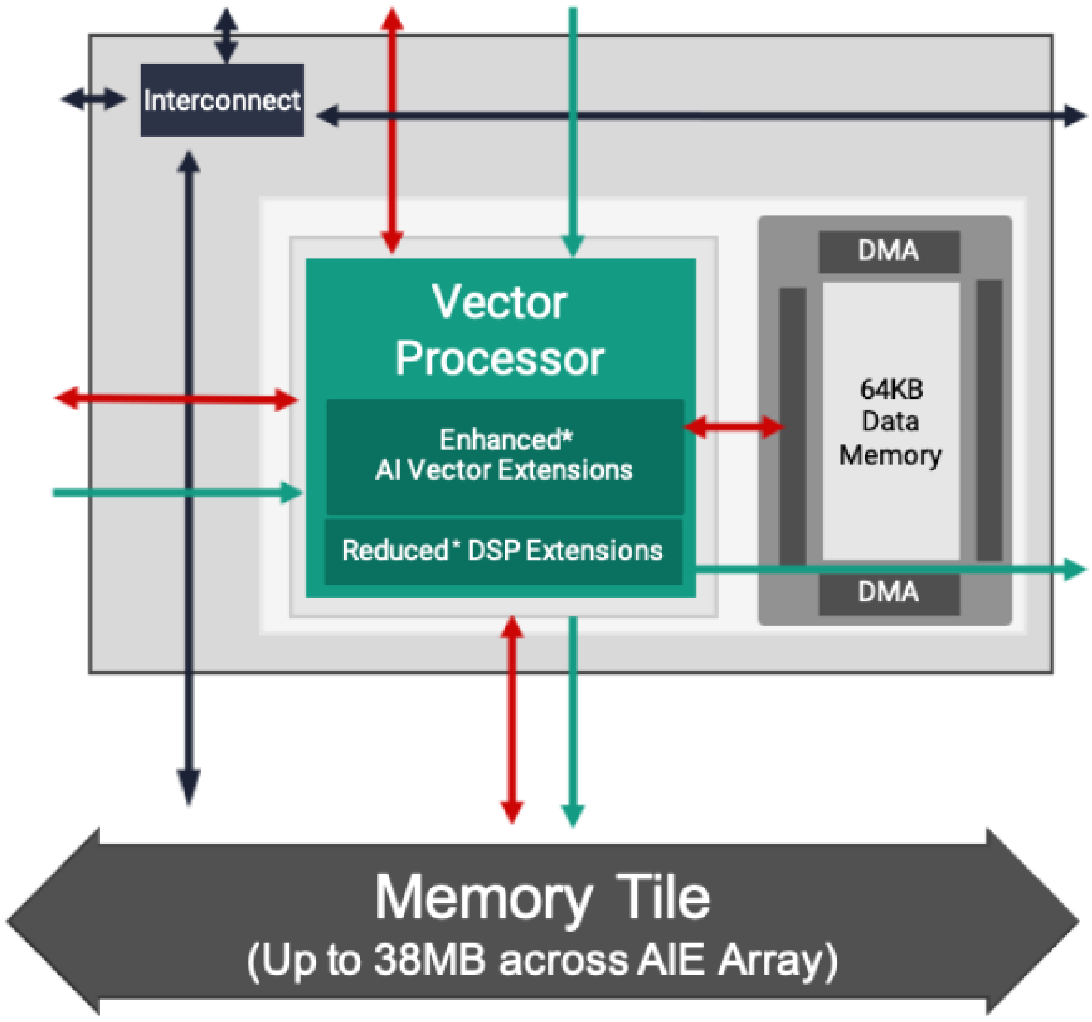


Figure 2.4: AI Engine-ML Tile

The AI Engine-ML architecture undergoes optimization specifically for machine learning, resulting in improvements to both the compute core and memory architecture. While retaining capabilities for both machine learning and advanced signal processing, these optimized tiles shift focus away from INT32 and CINT32 support, commonly utilized in radar processing, to better cater to machine learning applications. Key enhancements include:

- Extended native support for machine learning data types, incorporating INT4 and BFLOAT16.
- Doubled ML compute performance with reduced latency, featuring 512 INT4 MACs per tile and 256 INT8 MACs per tile.
- Increased array memory to localize data, achieved through the doubling of local data memory per tile (64kB) and the introduction of new memory tiles (512kB) for high bandwidth shared memory access.

2.3 Versal Series

The Versal Series represents a family of adaptive compute acceleration platforms by Xilinx, a range of devices tailored to address diverse computing challenges in artificial intelligence, machine learning, networking, and performance testing equipment. Key features across the Versal Series:

- **Adaptive Compute Platform:** The Versal Series introduces an adaptive system-on-chip (SoC) platform, featuring Adaptable Engines and Intelligent Engines for a combination of flexibility and specialized processing capabilities. It seamlessly integrates various engines, including the AI Engine, to deliver powerful heterogeneous acceleration for a wide array of applications.
- **High Bandwidth Memory (HBM):** Leveraging the power of High Bandwidth Memory (HBM), the Versal Series delivers significant memory bandwidth to tackle large datasets efficiently, reducing processing bottlenecks across the entire series.
- **AI/ML Acceleration:** Tailored for AI/ML applications, the Versal Series excels in parallel processing with Adaptable Engines, coupled with the optimization capabilities of the Vitis unified software platform.
- **Compute Pre-Processing:** Versal Series devices efficiently handle large-scale pre-data processing, offering 819 GB/s of HBM bandwidth alongside Adaptable Engines to create potent predictive inputs.

- **Next-Generation Firewall:** Providing unmatched scalability, the Versal HBM series within the series ensures robust multi-layer network security, integrating High-Speed Crypto (HSC) Engines and 32G HBM for enhanced performance.
- **Application Performance Test Equipment:** Catering to the demands of data center networking and cloud providers, the Versal Series features 112G PAM4 transceivers as essential building blocks for adaptive networks and sophisticated test equipment.

Within this family (ACAP), there are several platforms, each designed for a specific purpose:

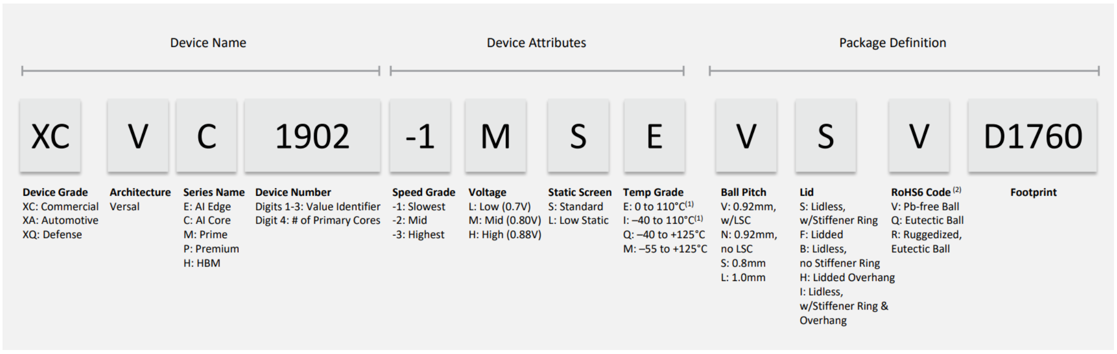
- **Versal HBM Series:** hyper Integration of Fast Memory, Secure Data, and Adaptive Compute.
- **Versal AI Core Series:** it delivers breakthrough AI inference and wireless acceleration with integrated AI engines that deliver outstanding compute performance.
- **Versal AI Edge Series:** delivering Breakthrough AI Performance/Watt For Real-Time Systems.
- **AMD Versal Prime Series:** it provides a diverse set of compute engines, next-generation I/O, and integrated DDR controllers, enabling low-latency acceleration across a wide range of workloads.
- **Versal Premium Series:** engineered for the most demanding compute and data movement applications, now featuring the world's largest adaptive SoC.

When coming to navigate throught Versal Devices it is important to understand the naming convention used for the main FPGA of the platform as described in Figure 2.5.

Among them the project will focus on a new board belonging to the Versal AI Edge Series.

2.4 Versal AI Edge

The Versal AI Edge series stands out by delivering a remarkable 4X improvement in AI performance per watt compared to leading GPUs, making it a prime choice for power and thermally constrained environments at edge nodes. This series is designed to accelerate the entire application spectrum, seamlessly spanning from sensor input to AI processing and real-time control. It boasts the world's most



Device Name	XC	Device Grade	XC: Commercial XA: Automotive XQ: Defense
	V	Architecture	Versal
	C	Series Name	E: AI Edge C: AI Core M: Prime P: Premium H: HBM
	1902	Device Number	Digits 1-3: Value Identifier Digit 4: # of Primary Cores
Device Attributes	-1	Speed Grade	-1: Slowest -2: Mid -3: Highest
	M	Voltage	L: Low (0.7V) M: Mid (0.80V) H: High (0.88V)
	S	Static Screen	S: Standard L: Low Static
	E	Temp Grade	E: 0 to 110°C(1) I: –40 to 110°C(1) Q: –40 to +125°C M: –55 to +125°C
Package Definition	V	Ball Pitch	V: 0.92mm, w/LSC N: 0.92mm, no LSC S: 0.8mm L: 1.0mm
	S	Lid	S: Lidless, w/Stiffener Ring F: Lidded B: Lidless, no Stiffener Ring H: Lidded Overhang I: Lidless, w/Stiffener Ring & Overhang
	V	RoHS6 Code (2)	V: Pb-free Ball Q: Eutectic Ball R: Ruggedized, Eutectic Ball
	D1760	Footprint	

Figure 2.5: Versal Device Ordering Information

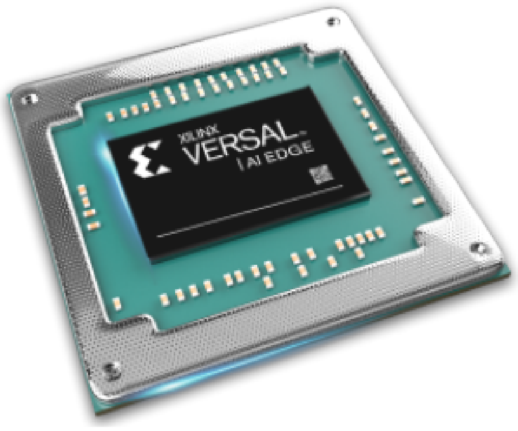


Figure 2.6: Versal AI Edge Chip

scalable portfolio in its class, addressing diverse needs from intelligent sensors to edge computing. The hardware adaptability of the Versal AI Edge series positions it as a dynamic solution that evolves in sync with ongoing AI innovations in real-time systems [2]. Going beyond the realm of AI, the Versal AI Edge series excels in providing high-performance, low-latency AI inference capabilities. Its applications span across various domains, including automated driving, predictive factory operations, healthcare systems, and multi-mission payloads in aerospace and defense. This versatility is coupled with a holistic approach, accelerating the entire application workflow from initial sensor input to AI-driven insights and real-time control. Notably, the Versal AI Edge series adheres to stringent safety and security standards, meeting critical requirements such as ISO 26262 and IEC 61508. This adaptive compute acceleration platform empowers developers to rapidly iterate on sensor fusion and AI algorithms. Moreover, its scalable device portfolio accommodates diverse performance and power profiles, offering a comprehensive solution from edge to endpoint. The most complete and best performing one among all of this family of devices is the VEK280 Evaluation Kit.

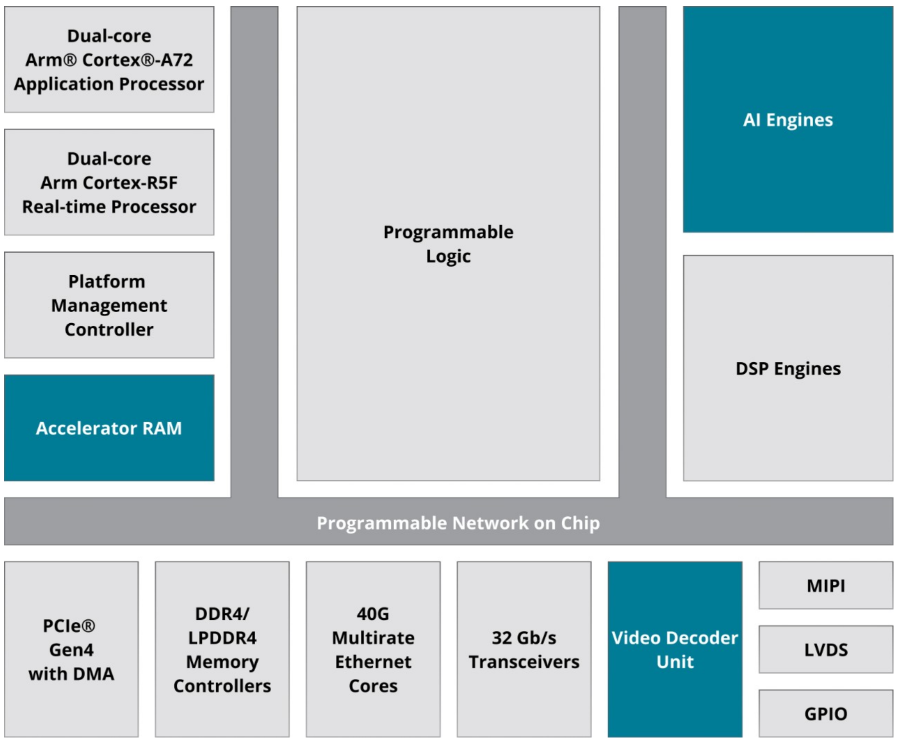

Dual-core Arm® Cortex®-A72 Application Processor
Dual-core Arm Cortex-R5F Real-time Processor
Platform Management Controller
Accelerator RAM
Programmable Logic
AI Engines
DSP Engines
Programmable Network on Chip
PCIe® Gen4 with DMA
DDR4/ LPDDR4 Memory Controllers
40G Multirate Ethernet Cores
32 Gb/s Transceivers
Video Decoder Unit
MIPI
LVDS
GPIO

Figure 2.7: Versal Premium chip diagram

The devices in this family share common processing and connectivity features. They have a dual-core Arm Cortex-A72 Application Processing Unit with 48 KB/32 KB L1 Cache featuring parity and ECC, along with a 1 MB L2 Cache with ECC. Additionally, each device incorporates a Real-Time Processing Unit consisting of a second dual-core Arm Cortex-R5F with 32 KB/32 KB L1 Cache and 256 KB TCM(Tightly Coupled Memory) with ECC(Error-Correcting Code) (Figure 2.7).

For connectivity, the family includes dual Ethernet ports, dual UART ports, dual CAN-FD interfaces, one USB 2.0 port, and dual SPI and I2C interfaces. These shared features across the family provide a consistent foundation for application development, ensuring a common set of processing capabilities and connectivity options.

2.5 The VEK280 Evaluation Kit

The VEK280 Evaluation Kit (Figure 2.8) is a cutting-edge solution powered by the AMD Versal AI Edge VE2802 Adaptive SoC[6]. It distinguishes itself with robust hardware acceleration engines, incorporating AIE-ML and DSP functionalities, and offers a plethora of high-speed connectivity options. Specifically designed for ML inference applications across diverse sectors such as automotive, vision, industrial, scientific, and medical, this exclusive kit boasts advanced capabilities to meet the demanding needs of these industries.

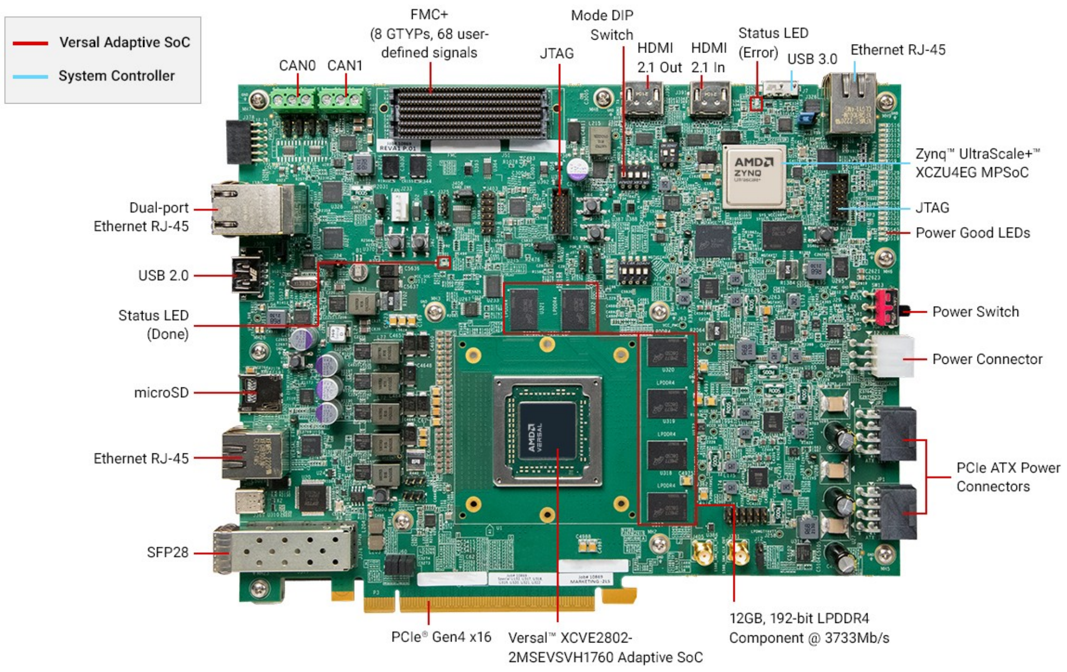


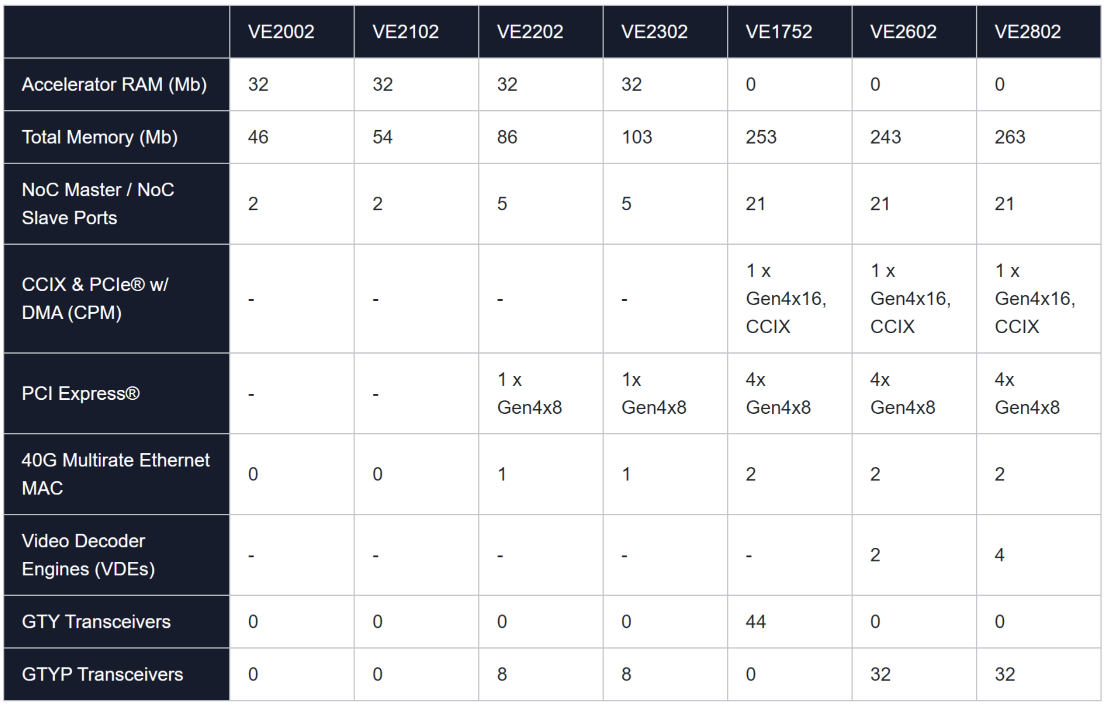
Figure 2.8: Vek 280 description.

However, it's important to note that this exceptional performance does come at a higher cost, exceeding ten thousand dollars. The limited production of only three units, with an additional one reserved for our use, underscores the exclusivity of this kit. This availability suggests that it may not align with the broader market due to its specialized nature and higher cost.

2.6 The new board

The concept behind the creation of the new board was to provide a more budget-friendly device that showcases the capabilities of this chip family. With seven products in this series, selecting one from the middle range allowed us to develop a device aimed at capturing the interest of a broader customer base without imposing a prohibitive price tag (Figure 2.9).

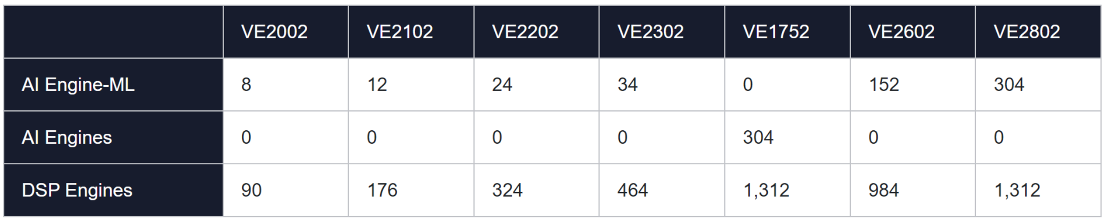
Taking into account the varying quantity of AI Engines across each platform and



	VE2002	VE2102	VE2202	VE2302	VE1752	VE2602	VE2802
Accelerator RAM (Mb)	32	32	32	32	0	0	0
Total Memory (Mb)	46	54	86	103	253	243	263
NoC Master / NoC Slave Ports	2	2	5	5	21	21	21
CCIX & PCIe® w/ DMA (CPM)	-	-	-	-	1 x Gen4x16, CCIX	1 x Gen4x16, CCIX	1 x Gen4x16, CCIX
PCI Express®	-	-	1 x Gen4x8	1x Gen4x8	4x Gen4x8	4x Gen4x8	4x Gen4x8
40G Multirate Ethernet MAC	0	0	1	1	2	2	2
Video Decoder Engines (VDEs)	-	-	-	-	-	2	4
GTY Transceivers	0	0	0	0	44	0	0
GTYP Transceivers	0	0	8	8	0	32	32

Figure 2.9: Versal AI Edge Platform Features

the discernible step that nearly halves the specifications between the two groups of products, the most pragmatic decision was to opt for the top-tier device within the lower range (Figure 2.10). This strategic choice allows us to deliver optimal performance while minimizing both hardware and production costs. In the context of the seven boards, where the first four exhibit less than half the specifications of the superior three, selecting the fourth board from the lower range ensures an efficient balance between performance and cost-effectiveness for this project.



	VE2002	VE2102	VE2202	VE2302	VE1752	VE2602	VE2802
AI Engine-ML	8	12	24	34	0	152	304
AI Engines	0	0	0	0	304	0	0
DSP Engines	90	176	324	464	1,312	984	1,312

Figure 2.10: AI Engine and DSP Engine Features

Chapter 3

The VE2302

The new board should have been meticulously designed, featuring a sophisticated architecture comprising two integral components: the System-on-Module (SOM) and the Carrier Board. This deliberate configuration is engineered to deliver a notably more cost-effective solution when compared to the VEK280, catering to budget considerations without compromising performance.

What sets this board apart is its modular design, a key aspect that optimizes cost and lends itself to unparalleled versatility. The SOM, housing the potent FPGA, forms the computational heart of the system, providing substantial processing power and adaptability. Simultaneously, the Carrier Board plays a pivotal role in customization, allowing users to tailor connectivity and I/O configurations according to the specific requirements of diverse use cases.

This synergy between the SOM and Carrier Board unlocks a spectrum of possibilities for deploying advanced AI and edge computing solutions. Whether in automotive, industrial, scientific, or other sectors, this modular approach ensures adaptability, making it an ideal choice for applications with varying connectivity and processing demands. The flexibility inherent in this design empowers users to harness the full potential of FPGA technology while maintaining the agility to meet their unique needs through tailored connectivity solutions on the Carrier Board.

3.1 Goals

The objectives of the board are summarized below:

- Create a SOM targeting the Versal AI Edge family of devices.
- Create a plan for developing a dense design in a small form factor.

- Create a Carrier Card with the intended mating SOM targeting the Versal AI Edge family of devices.
- Leverage previous design efforts as needed to reduce time to market.
- Incorporate Best Known Methods in design and layout to meet the required performance.
- Develop plans for the Versal AI Edge SOM and for the Versal AI Edge Carrier Card.

3.2 Versal AI Edge SOM

The System-on-Module (SOM) is a compact powerhouse integrating key components, including the robust Versal AI Edge FPGA, Arm Cortex-A72 cores for general-purpose computing, and essential memory resources. Designed to accelerate AI and edge computing workloads, the SOM serves as the core computational engine, ensuring seamless task execution. Its highly integrated and modular nature offers adaptability and easy integration into diverse systems and applications. With specifications tuned for efficiency, the SOM stands as a reliable solution in the realm of AI and edge computing.

3.2.1 Features and block diagram

The Versal AI Edge SOM is offered in commercial and industrial temp with the following feature set (Figure 3.1):

- AMD XCVE2302-1LSESFVA784-E (Pin compatible with the XCVE2202 device)
- LPDDR4 SDRAM (4GB, 2x32)
- PMC OSPI Flash (Octal 64MB up to 256MB)
- PMC eMMC Flash (x8 16GB up to 64 GB)
- PMC USB 2.0 ULPI PHY
- PS Gigabit Ethernet RGMII PHY
- I2C MAC EEPROM
- I2C 8-bit I/O Expander
- 2-channel I2C Switch/Mux

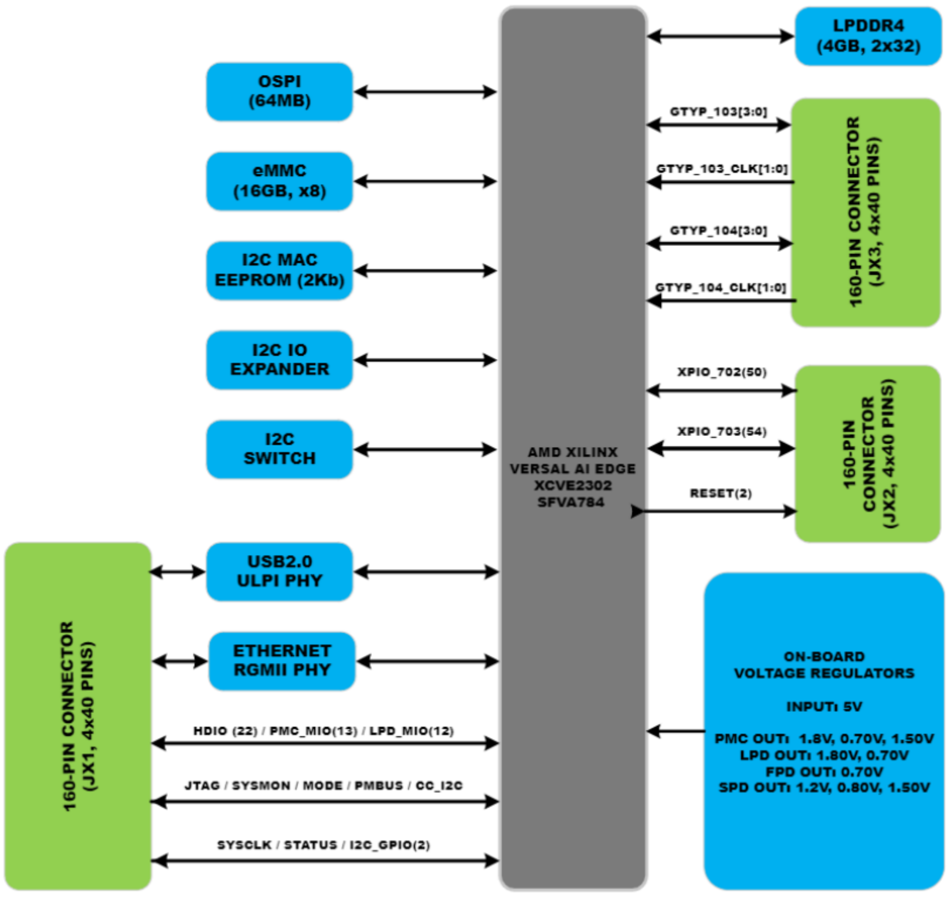

OSPI (64MB)
eMMC (16GB, x8)
I2C MAC EEPROM (2Kb)
I2C IO EXPANDER
I2C SWITCH
USB2.0 ULPI PHY
ETHERNET RGMII PHY
160-PIN CONNECTOR (JX1, 4x40 PINS)
HDIO (22) / PMC_MIO(13) / LPD_MIO(12)
JTAG / SYSMON / MODE / PMBUS / CC_I2C
SYSCLK / STATUS / I2C_GPIO(2)
AMD XILINX VERSAL AI EDGE XCVE2302 SFVA784
LPDDR4 (4GB, 2x32)
GTYP_103[3:0]
GTYP_103_CLK[1:0]
GTYP_104[3:0]
GTYP_104_CLK[1:0]
160-PIN CONNECTOR (JX3, 4x40 PINS)
XPIO_702(50)
XPIO_703(54)
RESET(2)
160-PIN CONNECTOR (JX2, 4x40 PINS)
ON-BOARD VOLTAGE REGULATORS
INPUT: 5V
PMC OUT: 1.8V, 0.70V, 1.50V
LPD OUT: 1.80V, 0.70V
FPD OUT: 0.70V
SPD OUT: 1.2V, 0.80V, 1.50V

Figure 3.1: Versal AI Edge SOM Block Diagram.

- Reference Clock
- Real Time Clock
- On-Board Voltage Regulators
- 3 Micro-Header Connectors. Samtec ADM6/ADF6 family of connectors will be used to implement the Versal AI Edge SOM to the Versal AI Edge Carrier Card connections. These connectors have 0.635mm pitch and are rated at up to 56Gbps data rate with 1.4A/pin current rating:
  - JX1/JX2/JX3 (3x160-pin)
  - Connections to the Carrier Card
  - 104 User XPIO Pins

  - 22 User HDIO Pins
  - 12 User LPD MIO Pins
  - 13 User PMC MIO Pins
  - 4 GTYP Transceivers
  - 4 GTYP Reference Clock Inputs
  - JTAG Interface
  - SYSMON interface
  - USB 2.0 Connector Interface
  - Gigabit Ethernet RJ45 Connector Interface
  - PMBus Interface
  - Carrier Card I2C Interface
  - SOM VCC_BATT Battery Input
  - SOM Reset Input
  - Carrier Card Interrupt Input
  - Carrier Card Reset Output
  - SOM Power Good Output
  - SOM to Carrier Card Ground Pins
  - SOM Input Voltages and Output Sense Pins

3.2.2 Platform Management Controller (PMC) IO Banks

The Platform Management Controller (PMC) contains several I/O Banks that can be connected to various peripherals through MIO port connections. The following sub-sections detail the specific connections to each PMC MIO Bank.

PMC MIO Bank 500

The 26 pins in PMC MIO Bank 500 play a key role in implementing essential interfaces, including a bootable OSPI Flash interface, a USB2.0 interface utilizing a ULPI PHY, and a dedicated MIO pin. Identification of these pins is achieved by creating an example design in Vivado 2022.2, leveraging a BETA version that provides access to the Versal AI Edge device. The physical connector for the USB2.0 interface is present on the Versal AI Edge Carrier Card, necessitating the mapping of the USB2.0 ULPI PHY to the JX Connectors.

- **Octal SPI Flash:** The Versal AI Edge SOM utilizes an OSPI Flash device for primary boot functionality. While specific details such as the manufacturer and model are omitted, it's worth noting that there are options for larger density devices in this footprint, allowing flexibility in OSPI flash configurations.
- **USB2.0 ULPI PHY:** The Versal AI Edge SOM features a USB 2.0 PHY interface using the Microchip USB3321C USB 2.0 ULPI PHY. The USB 2.0 ULPI PHY connector side, connected to the JX connector, facilitates the implementation of a USB 2.0 interface via a single connector on Versal AI Edge Carrier Cards. The USB 2.0 ULPI PHY operates at 1.8V on the Versal AI Edge SOM, and the P1 port of the I2C 8-bit I/O expander is utilized for a soft reset of the USB 2.0 ULPI PHY.
- **PMC General Purpose Input:** The Versal AI Edge SOM includes a general-purpose interrupt IO connected to PMC_MIO_11.

PMC MIO Bank 501

Bank 501 MIO pins on the Versal AI Edge are purposefully designated for implementing diverse interfaces, providing adaptability for various applications. These pins, directed to the JX Connectors on the Versal AI Edge Carrier Card, extend their utility beyond dedicated functions to serve multiple purposes effectively. Within this bank, specific MIO pins are exclusively allocated for establishing an SD3.0 interface on the Versal AI Edge Carrier Card. This versatile SD3.0 interface caters to both BOOT and STORAGE applications, ensuring a flexible and practical solution. Additionally, other MIO pins within the same bank are earmarked for implementing an eMMC interface on the Versal AI Edge SOM, playing a crucial role in supporting STORAGE functions or serving as a pathway for SECONDARY BOOT operations. Furthermore, Bank 501 MIO pins are assigned to implement a PMC I2C interface, serving as the primary I2C interface on the Versal AI Edge SOM. This strategic allocation ensures seamless communication and connectivity within the system. As these physical connections extend to the Versal AI Edge Carrier Card, it becomes essential to map the SD3.0 MIO interfaces to the JX Connectors for optimal integration and functionality.

- **eMMC x8 Flash:** The Versal AI Edge SOM utilizes a Micron MTFC16GAPA LBH-AAT TR eMMC Memory for Secondary Boot and/or storage.
- **JX Connector Interface:** PMC MIO pins for this interface are routed to the JX connectors, allowing flexible implementation on the Versal AI Edge Carrier Card. Customers can tailor the interface to their needs on their Custom Carrier Card design.

- **PMC I2C MAC EEPROM:** The Versal AI Edge SOM features a Microchip AT24MAC402-MAHM I2C EEPROM 2Kbit 1MHz device, housing the unique EUI-48 ethernet MAC address.
- **I2C 8-Bit I/O Expander**
- **I2C 2-Channel Switch/MUX**
- **Carrier Card I2C Interface:** The Versal AI Edge SOM provides a master I2C bus to the Versal AI Edge Carrier Card via the JX connector for seamless communication with I2C devices on both the SOM and Carrier Card.
- **PMBUS I2C Interface:** PMBus is accessible on the Versal AI Edge SOM for programming, controlling, and monitoring on-board PMBus voltage regulators, with access to Carrier Card PMBus signals via the JX connector through the I2C switch/mux.
- **PMC Push Button**

LPD MIO Bank 502

The LPD MIO Bank 502 consists of 26 MIO pins, MIO[25:0]. There are Bank 502 MIO pins that are dedicated to implementing a Gigabit Ethernet interface to an RGMII PHY. The RJ45 ethernet jack will exist on the Versal AI Edge Carrier Card. The rest of the Bank 502 MIO pins are routed to the JX connectors. The generic MIO pins will be used on the Versal AI Edge Carrier Card to implement UART, I2C, CAN, SPI, or general purpose interfaces.

- **Gigabit Ethernet RGMII PHY:** The Versal AI Edge SOM will provide a single Gigabit Ethernet PHY interface using the Microchip KSZ9131RNXU RGMII PHY device in 48-pin QFN package (in industrial temp). The Versal AI Edge SOM Gigabit Ethernet PHY connector side (connected to the JX connector) along with an RJ45 connector located on the Versal AI Edge Carrier Card will be used to implement the Gigabit Ethernet port.
- **JX Connector Interface:** The LPD MIO pins will be trace length matched to allow for proper implementation on the Versal AI Edge Carrier Card of any interfaces that can be targeted to these LPD MIO pins. Although the Versal AI Edge Carrier Card will implement several interfaces, customers utilizing the Versal AI Edge SOM will be able to implement the interface they desire on their Custom Carrier Card design.

PMC CONFIG Bank 503

The PMC Configuration Bank 503 consists of JTAG, RESET, Reference Clock Input, BOOT MODE, RTC Crystal Input, and other associated configuration pins. Some of these signals are implemented on the Versal AI Edge SOM and others are routed to a JX Connector for implementation on the Versal AI Edge Carrier Card.

- **DONE LED:** The Versal AI Edge Carrier Card will implement a BLUE LED indicating the configuration is complete.
- **ERROR OUT LED:** The Versal AI Edge Carrier Card will implement an Error Out LED. The ERROR_OUT LED will be RED.
- **JTAG Interface**
- **BOOT Mode Pins:** The Versal AI Edge Carrier Card will implement a small 4-position DIP switch for the Boot Mode pins.
- **Other stuff:** Real Time Clock, SOM RESET and a Processor Clock

3.2.3 Programmable Logic IO Banks

The Programmable Logic portion of the design consists of several IO Banks that can be connected to various peripherals through the programmable logic GPIO. The following sub-sections detail the specific connections to each PL IO Bank.

- **XPIO Bank 700-701-702 – LPDDR4 Interface:** The Versal AI Edge SOM will provide 4GB of LPDDR4 memory in a 2x32 configuration using 2 Micron MT53E512M32D1ZW-046 IT:B (200-pin BGA package) x32 devices. The LPDDR4 devices are implemented in 512Mb x 32 configuration and supports up to 4266Mbps. The LPDDR4 devices will be connected to the XPIO banks 700, 701, and 702 and be operated at +1.1V at the maximum supported bandwidth available in the PIN EFFICIENT implementation on the Versal AI Edge device.
- **XPIO Bank 702 - JX Connector Interface:** The XPIO Bank 702 consists of 54 XPIO pins. 50 of these pins are routed to a JX connector on the Versal AI Edge SOM. 4 of the Bank 702 XPIO pins are utilized by the LPDDR4 interface.
- **HDIO Bank 302 - JX Connector Interface**
- **Bank 103-104 - GTYP Transceivers**

3.3 Versal AI Edge Carrier Card

The Carrier Card serves as the pivotal interface and expansion platform for the high-speed Versal AI Edge SOM, meticulously crafted for rapid machine learning inference. It is equipped with connectors for seamless power supply, external communication interfaces (including Ethernet and USB), and versatile expansion slots catering to additional peripherals or customized interfaces. In the intricate web of system architecture, the Carrier Card plays a decisive role, effortlessly linking the SOM to the external world, ensuring its integration into a broader system efficiently.

A distinctive feature of the Carrier Card is its unique provision for six MIPI connectors, strategically designed to accommodate up to six cameras. This specialized capability enhances the system's adaptability, enabling it to effortlessly interface with multiple cameras concurrently. This becomes particularly valuable in real-time applications where the Versal AI Edge SOM's exceptional speed in executing machine learning inference is a critical asset.

This configuration not only exemplifies the Carrier Card's versatility but also positions it as a crucial enabler for a wide array of applications. From advanced surveillance systems to intricate computer vision setups, the inclusion of six MIPI connectors aligns perfectly with real-time requirements, making it a key component for applications demanding swift and accurate machine learning inference in real-time scenarios.

3.3.1 Features and block diagram

The Versal AI Edge Carrier Card is offered in commercial and industrial temp with the following feature set (Figure 3.2):

- Versal AI Edge SOM Slot
- 2x SFP28 Interfaces
- HDMI RX/TX Interface
- 6x MIPI-CSI 22-Pin Camera/Display Connectors
- 1x HSIO GTYP / HDIO Connector
- 2x CAN Industrial Header Connectors
- RJ45 Connector
- USB 2.0 Type-A Receptacle
- XPIO Push Buttons

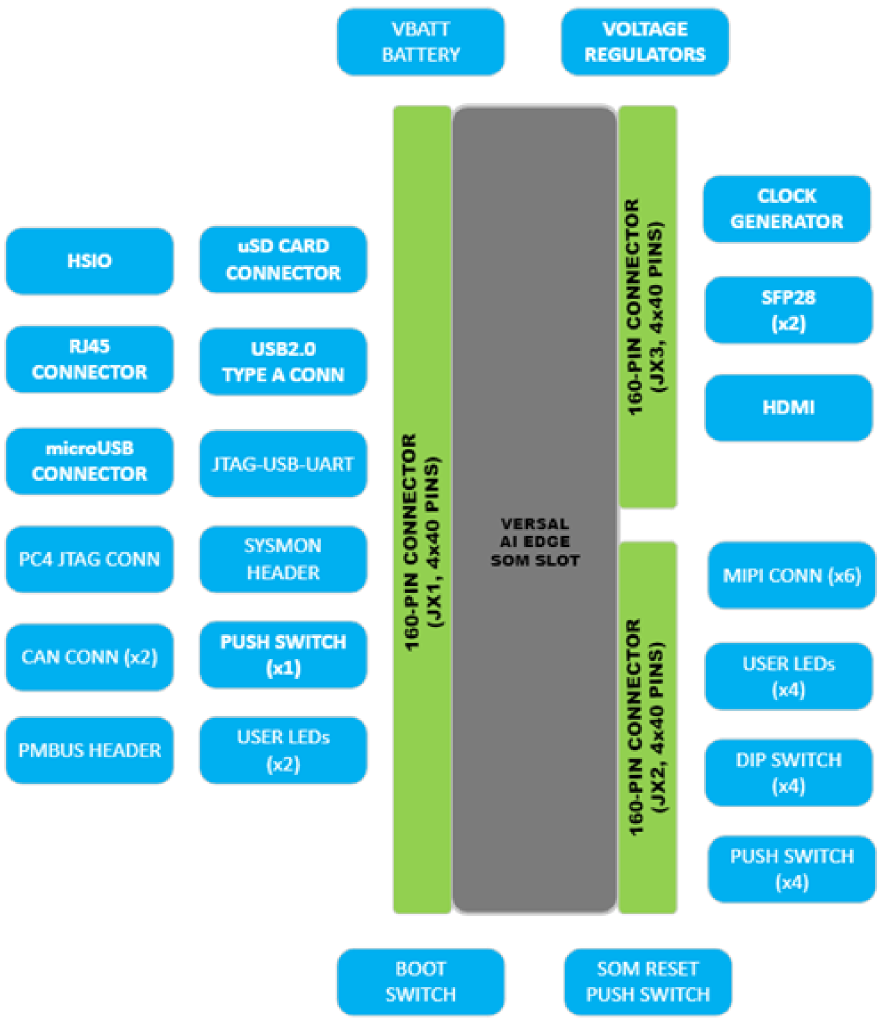


Figure 3.2: Versal AI Edge Carrier Card Block Diagram.

- XPIO LEDs
- PMC Push Button
- I2C GPIO LEDs
- microSD Card Connector
- microUSB Receptable-UART-JTAG Interface
- PC4 JTAG Header

- Differential Clock Generator
- PMBus Header
- VBATT Battery Connector
- SOM Reset Button
- 3x 160-Pin JX Micro-Header Connectors

3.3.2 Functionalities

1. **SFP28 Interfaces:**
   - The Versal AI Edge Carrier Card will provide two SFP28 interfaces. These SFP28 interfaces can be used to implement high-performance Ethernet interfaces as well as a host of other interfaces up to the SFP28 expected line rates.
   - Key components include:
     - `SFP28 I2C I/O Expander`
     - `SFP28 I2C Switch`: The SFP28 I2C interface signals will be generated from an I2C 8-Channel Switch / MUX on the Versal AI Edge Carrier Card.
2. **HDMI TX / RX Interfaces:**
   - Key components include:
     - `HDMI Reference Clocks`: It is expected that several clocks will be necessary for the HDMI TX and HDMI RX interfaces. All of these locks will be mapped to the GTYPs.
     - `HDMI Transceiver Lanes`: The data lanes for the HDMI TX and HDMI RX interfaces should come from the Bank 104 GTYPs. All of these data lanes will be appropriately mapped to the GTYPs.
     - `HDMI Required Programmable IO`: The HDMI RX and TX interface solutions will require control and status pins to be mapped to the Versal AI Edge device.
3. **HSIO TXR2 PL Connector:**
   - This interface is designed to integrate both GTYP (GTY Transceivers) lanes and HDIO (High-Density Input/Output) pins. It combines the capabilities of GTY transceivers and high-density I/O pins to facilitate communication and data transfer, ultimately serving as a crucial component in the overall system architecture.

4. **MIPI-CSI2-DSI-22 Connectors:**

   - The MIPI Camera Serial Interface 2 is a widely adopted, high-speed protocol for the transmission of still and video images from image sensors to application processors.
   - The Versal AI Edge Carrier Card will feature an impressive configuration with six MIPI-CSI2-DSI 22-pin connectors. These connectors will be strategically mapped to the XPIO banks on the Versal AI Edge device, emphasizing the substantial capacity for connecting multiple cameras to the system.

5. **USB2.0 Connector:**

   - The USB 2.0 ULPI PHY connector side (connected to the JX connector) is implemented on the Versal AI Edge Carrier Cards to complete the USB 2.0 interface.

6. **SYSMON Header:**

   - The PMC Bank 500 contains the Versal AI Edge devices SYSMON pins. These pins are routed to a JX connector on the Versal AI Edge SOM. The Versal AI Edge Carrier Card will take the SYSMON signals on the JX connector and provide a header for customers to utilize.

7. **SD3.0 Interface:**

   - The Versal AI Edge SOM and Carrier Card together will provide a microSD card interface.
   - Since microSD cards do not have a Write-Protect (WP) pin, the PMC SD controller WP signal (MIO37) will not be utilized and that MIO will be repurposed as a PMC Push Button.
   - The PMC MIO pins for this interface are routed to the JX connectors. Although the Versal AI Edge Carrier Card will implement the MicroSD Card Interface, customers utilizing the SOM will be able to implement the interface they desire on their Custom Carrier Card design.

8. **Carrier Card I2C Interface:**

   - The Versal AI Edge SOM provides a master I2C bus to the Versal AI Edge Carrier Card via the JX connector so that software can communicate with I2C devices on the Versal AI Edge SOM as well as the slave I2C devices on the Versal AI Edge Carrier Card using a single I2C interface.

- The Versal AI Edge Carrier Card will implement I2C Bus Expanders to implement the various I2C interfaces that exist on the Carrier Card. These I2C interfaces are implemented on the SFP28s, HDMI, HSIO TXR2 PL, and the MIPI connectors.

9. **PMBus I2C Interface:**

   - PMBus may be used on the Versal AI Edge SOM to program/control/-monitor on-board PMBus voltage regulators. The Versal AI Edge SOM will have access to the Carrier Card PMBus signals via the JX connector through the I2C switch/mux.

10. **PMC Push Button:**

    - The Versal AI Edge Carrier Card will provide a PMC active low user PUSH BUTTON. This PUSH BUTTON will be connected to the MIO37 pin on the JX Connector and operated at the proper bank voltage.

11. **I2C Expander LEDs:**

    - The Versal AI Edge Carrier Card will provide active high-user LEDs that are extensions of the I2C GPIO Expander on the Versal AI Edge SOM. These LEDs will be connected to two pins on the JX Connector and operated at 3.3V I/O.

12. **RJ45 Gigabit Ethernet Connector:**

    - The Versal AI Edge SOM Gigabit Ethernet PHY connector side (connected to the JX connector) along with an RJ45 connector located on the Versal AI Edge Carrier Card will be used to implement the Gigabit Ethernet port.

13. **XPIO User Interfaces:**

    - The Versal AI Edge SOM and Versal AI Edge Carrier Card together will provide several interfaces that will be mapped to the remaining XPIO pins. These are routed to the JX connectors.
    - While customers utilizing the Versal AI Edge SOM will be able to implement the interface they desire on their Custom Carrier Card design, it's important to note that the Versal AI Edge Carrier Card will implement several interfaces:
      - `XPIO LEDs`: The Versal AI Edge Carrier Card will provide XPIO active high-user LEDs. These USER LEDs will be connected to 4 XPIO pins on the JX Connector (Figure 4.3).

   - `XPIO Switch`: The Versal AI Edge Carrier Card will provide an XPIO USER SWITCH. The 4-port Dipswitch will be connected to the 4 XPIO pins on the JX Connector
   - `XPIO Push Buttons`: The Versal AI Edge Carrier Card will provide two XPIO PUSH BUTTONS.

14. **LPD User Interfaces:**
   - The Versal AI Edge SOM and Versal AI Edge Carrier Card together will provide several interfaces that will be mapped to the remaining LPD MIO pins. These LPD MIO pins will be routed to the JX connectors.
   - As before customers will be able to implement the interface they desire on their Custom Carrier Card design, but the Versal AI Edge Carrier Card will implement several interfaces:
     - `LPD CAN Interfaces`: The Versal AI Edge Carrier Card will implement two CAN interfaces. The two CAN interfaces will be connected to 6 LPD MIO pins on the JX Connector. The CAN interfaces will terminate at to INDUSTRIAL CONNECTOR TERMINALS. It is expected that the CAN interfaces will be implemented like what exists on the VEK280 design by AMD.
     - `LPD UART Interface`: The Versal AI Edge Carrier Card will implement a UART interface that is shared with the JTAG interface. The UART interface will be part of a USB-JTAG-UART solution.
     - `LPD I2C Interface`: The LPD I2C interface is connected to the SFP28 interface using a x8 I2C Switch to drive the I2C interfaces to the SFP28 modules. It also connects to the SFP28 control interfaces using an I2C GPIO Expander to drive the SFP28 control signals to the SFP28 modules (SFP28 I2C Interfaces). The 8 output ports can be connected to the I2C interfaces on the MIPI camera connectors and the SFP28 I2C interfaces.

15. **Configuration Interfaces:**
   - The PMC Configuration Bank 503 consists of JTAG, RESET, Reference Clock Input, BOOT MODE, RTC Crystal Input, and other associated configuration pins. Some of these signals are implemented on the Versal AI Edge SOM and others are routed to a JX Connector for implementation on the Versal AI Edge Carrier Card.
     - `DONE LED`: The Versal AI Edge Carrier Card will implement a BLUE LED indicating the configuration is complete.
     - `ERROR OUT LED`: There will be an Error Out LED. It will be RED.

  - `USB-JTAG-UART Interface`: The Versal AI Edge Carrier Card will route the JTAG interface and the UART interface from the JX connectors to an FTDI FT2232HL device. An example to use as a reference is on the VEK280. The FTDI USB-JTAG-UART interface will terminate with a micro USB connector.
  - `BOOT Mode Pins`: The Boot Mode pins will be manually controlled by a small 4-position DIP switch.

16. **RESET Structure:**
    - A small push switch will be used to manually assert the SOM_RESET_IN_B signal on the Versal AI Edge Carrier Card and send it to the Versal AI Edge SOM via the JX connector. This will in turn assert the POR_B (Power-On Reset) signal to the Versal AI Edge device and other components on the SOM that are connected to this signal.

17. **Clock Generator:**
    - The Versal AI Edge Carrier Card will provide a Renesas programmable clock source for generating clock inputs for the Versal AI Edge SOM GTYP transceivers and the LPDDR4/MIPI system clock on an XPIO bank.
    - The programming file for the Renesas device is generated by a tool called Timing Commander.

18. **JX Micro Header Connectors:**
    - The Versal AI Edge SOM will utilize 3 micro headers to provide connections to the Versal AI Edge Carrier Card. The Versal AI Edge SOM will more than likely use a 160-pin (4-rows x40-pins) connector for all 3 of the JX connectors (Samtec ADM6/ADF6).

Chapter 4

The creation of the Board Definition Files

The first thing to do to be able to build for the new board was to create the board platform to give to the program as the target for the application. The platform has to be built using Vivado which is the design software for AMD adaptive SoCs and FPGAs. It includes Design Entry, Synthesis, Place and Route, Verification/Simulation tools. When creating a new project in Vivado, you can start from a Xilinx part or predefined board; given that the new board, VE2302, is still in development, no existing Board Definition Files (BDF) were available in either Vivado or the XilinxBoardStore GitHub repository. To address this, the team recommended examining the board definition file of a similar board, the Vek280, along with its schematics (though not publicly available). The objective was to comprehend how the BDF is structured, explore the writing style, and assess the potential for modification or reduction to adapt it to the specifications of the new board, VE2302. The Board Definition File (BDF) will encompass the description of both the System on Module (SOM) and the Carrier Card. This comprehensive file captures the entirety of the integrated system, providing a holistic representation of the combined functionalities and characteristics of the SOM and Carrier Card.

4.1 The schematics

Board schematics serve as intricate graphical blueprints, illustrating the electronic components and their interconnections (example Figure 4.2). These detailed representations offer a visual roadmap of the board's circuitry, providing a comprehensive overview. Utilizing these schematics was crucial for deciphering the necessary information to be included in the BDFs and understanding how to structure them. The complexity of the Vek280 becomes evident when considering its schematic PDF

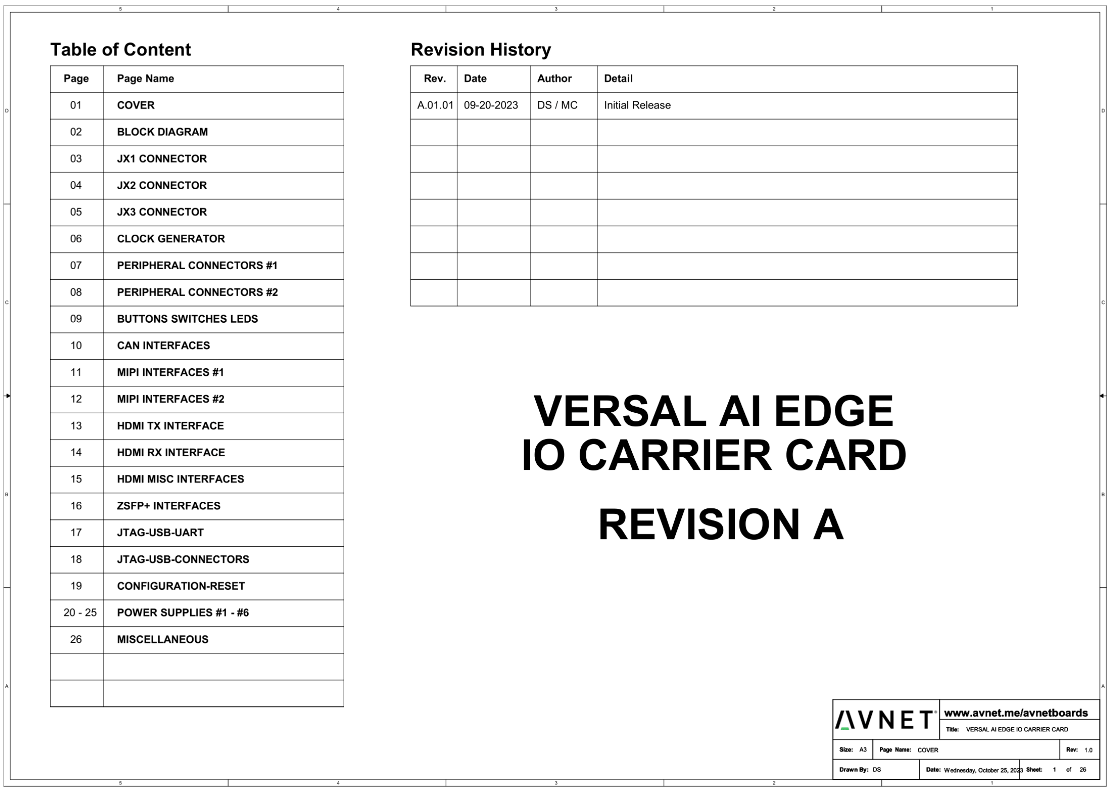


Table of Content

Page	Page Name
01	COVER
02	BLOCK DIAGRAM
03	JX1 CONNECTOR
04	JX2 CONNECTOR
05	JX3 CONNECTOR
06	CLOCK GENERATOR
07	PERIPHERAL CONNECTORS #1
08	PERIPHERAL CONNECTORS #2
09	BUTTONS SWITCHES LEDS
10	CAN INTERFACES
11	MIPI INTERFACES #1
12	MIPI INTERFACES #2
13	HDMI TX INTERFACE
14	HDMI RX INTERFACE
15	HDMI MISC INTERFACES
16	ZSFP+ INTERFACES
17	JTAG-USB-UART
18	JTAG-USB-CONNECTORS
19	CONFIGURATION-RESET
20 - 25	POWER SUPPLIES #1 - #6
26	MISCELLANEOUS

Revision History

Rev.	Date	Author	Detail
A.01.01	09-20-2023	DS / MC	Initial Release

VERSAL AI EDGE
IO CARRIER CARD

REVISION A

AVNET	www.avnet.me/avnetboards		
	Title: VERSAL AI EDGE IO CARRIER CARD		
Size: A3	Page Name: COVER		Rev: 1.0
Drawn By: DS	Date: Wednesday, October 25, 2023	Sheet: 1 of 26	

Figure 4.1: First page of the Carrier Schematics

file, an extensive document spanning 94 pages. This stands in contrast to the 20 pages each for both the SOM and the Carrier of the VE2302 (Figure 4.1).

4.2 Connections

After reviewing the two PDF files for the VE2302, I created a table (Table 4.1) summarizing signals and their functions based on their connections. This table simplifies understanding of the key roles and interactions within the system, offering a quick reference guide to the VE2302's functionality. It's essential to note that not every signal will find detailed mention in the BDF. The table serves as a comprehensive reference, highlighting key connections and their purposes. This selective approach ensures that the BDF focus on pertinent signals crucial to understanding and working with the VE2302, streamlining the documentation process.

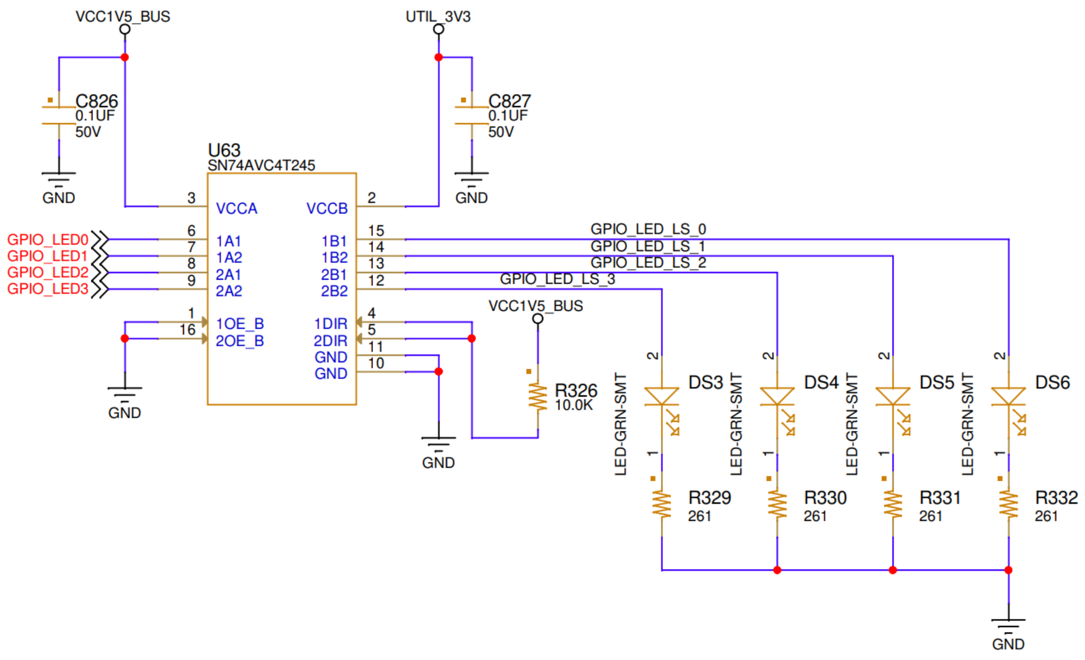


Figure 4.2: Example page from Vek280 schematics: four General Purpose Input Output (GPIO) pins are interfaced with a Logic Level Shifter, which, in turn, connects to four Light Emitting Diodes (LEDs).

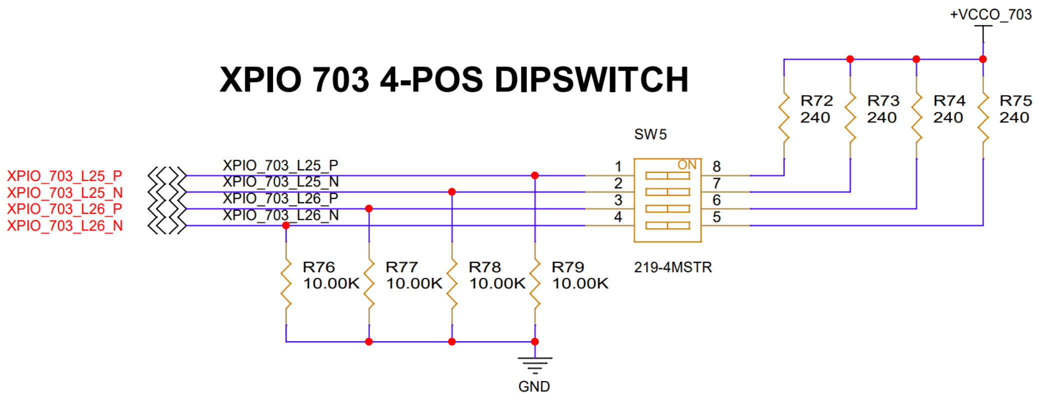


Figure 4.3: Example schematics from VE2302 Carrier Board of the Dipswitch (4-Port Dipswitch).

Purpose	**Signals VE**
SD on Carrier	PMC_MIO38-45
EMMC (Flash Memory)	PMC_MIO26-36 (28 -> I2C RESET)
OSPI (Flash Memory)	PMC_MIO0-11
USB on SOM	PMC_MIO13-25
LEDs on Carrier	PMC_MIO47-48
I2C	PMC_MIO50-51
PMC PUSH BUTTON on Carrier	PMC_MIO37/46
GIGABIT ETHERNET	LPD_MIO0-11 + 24-25
BOOT MODE DIPSWITCH	MODE0-3
REAL TIME CLOCK	RTC_PADI RTC_PADO
RESET VARIOUS	POR_B + specific RST
LPDDR4	BANK 700-701
XPIO JX (1.0V - 1.5 V from Carrier)	BANK 702-703
$\hookrightarrow$	HSIO on Carrier (PL CONNECTOR)
HDIO JX (1.8V - 3.3V from Carrier)	BANK 302
GTYP transceivers	BANK 103-104
$\hookrightarrow$	Various (HDMI, Clock, ...)
Various	BANK 503

Table 4.1: Signal Mapping of VE2302

4.3 The Board Definition Files

Board Definition Files (BDFs) are configuration files that provide a detailed and structured description of a hardware board's components, connections, and constraints. These files are commonly used in electronic design automation (EDA) tools, such as Xilinx Vivado, to define the physical properties and relationships of various elements on a printed circuit board (PCB) or a programmable logic device (PLD) development board. As explained in the tutorial [7]: "A board in Vivado is defined through the board definition files: the picture of our board and three important XML files":

- **board.xml:** This file encompasses fundamental information about the board, including the board name, description, vendor details, and information about various components such as the FPGA part, LEDs, and buttons. It also outlines details about the required interfaces for these components and the preferred IP cores to implement these interfaces (See Section 4.4).
- **preset.xml:** This file defines presets for IP cores specified in the board.xml file. It plays a role in configuring and setting parameters for these IP cores (See Section 4.6).
- **part0_pins.xml:** This file specifically defines the physical pins and I/O standards for the interfaces specified in the board.xml file. It is crucial for mapping out the connectivity and electrical characteristics of the board (See Section 4.5).

The meaning of each element that composes the files is better described in the Vivado Design Suite User Guide: System-Level Design Entry (UG895) [8].

4.4 board.xml

The file begins with an XML tag named `<board>`, wherein we provide fundamental information about the board. Firstly, we define the board file schema version attribute, indicating to the Vivado software how to interpret the data provided in the file. As of Vivado 2023.1, the latest version of the schema is "2.2". Next, we specify the vendor name, board name, and webpage for the board vendor; additionally, we include the name of the preset file, which I will describe later. Following this, we must conclude by writing `</board>` on a new line. All other board information must be defined between these opening and closing tags.

```
<board schema_version="2.2" vendor="avnet.com" name="
  ↪ ve2302_iocc" display_name="Versal VE2302 IOCC
  ↪ Evaluation Platform " url="http://avnet.me/ve2302_iocc
  ↪ " preset_file="preset.xml" supports_ced="true">
  ...
</board>
```

Listing 4.1: Enclosing XML tag of board.xml

An additional crucial piece of information is the image that will be displayed in the selection tool of Vivado when generating a new project as shown in Figure 4.4. This image serves as a visual reference for the Versal VE2302 IOCC Evaluation Platform, assisting users and developers in identifying and comprehending the platform. The *"display_name"* attribute specifies the name that will appear, and the *"sub_type"* attribute indicates its relation to the board. The *"resolution"* attribute is set to "high", implying that the image is of high quality. The *"description"* element provides a brief textual description of the image, mentioning the Versal VE2302 IOCC Evaluation Platform.

```
<images>
  <image name="ve2302_image.jpg" display_name="Versal VE2302
   ↪  IOCC Evaluation Platform" sub_type="board" resolution
   ↪ ="high">
    <description>Versal VE2302 IOCC Evaluation Platform"</
   ↪ description>
  </image>
</images>
```

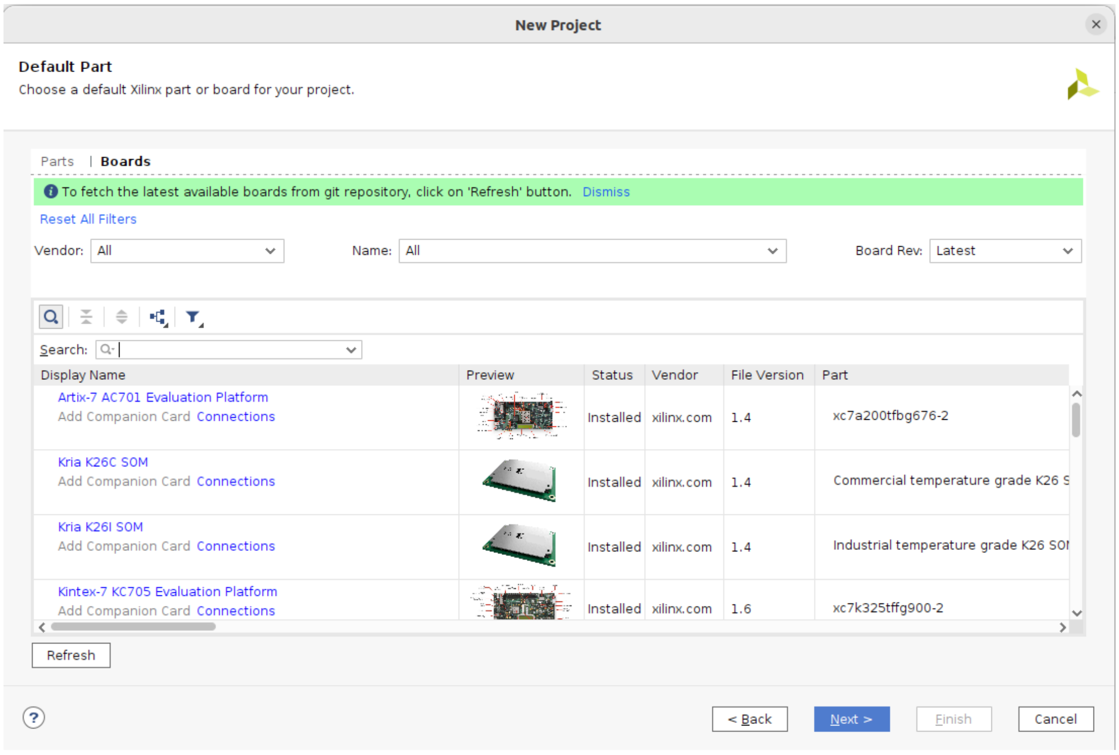
Listing 4.2: <compatible_board_revisions> and <file_version> tags of board.xml

To ensure the proper functioning of board files, it is essential to include the `<file_version>` and `<compatible_board_revisions>` tags:

```
<compatible_board_revisions>
    <revision id="0">Rev A01</revision>
</compatible_board_revisions>
<file_version>1.0</file_version>
```

Listing 4.3: <compatible_board_revisions> and <file_version> tags of board.xml

`<file_version>` tag is used to track the version of board files while the `<compatible_board_revisions>` tag allows us to specify compatible board revisions, such as Revision1, Revision2, and so on. These tags contribute to maintaining compatibility and ensuring that the board files are interpreted correctly by the software. Note that changes to the physical board may also trigger changes in board file, and therefore a new board `<file_version>`. However, revisions to the board file may not require revisions to the physical board; and revisions to the physical board can include changes that do not necessitate an updated board file.



Default Part
Choose a default Xilinx part or board for your project.

To fetch the latest available boards from git repository, click on 'Refresh' button. Dismiss

Reset All Filters

Vendor: All   Name: All   Board Rev: Latest

Search:

Display Name	Preview	Status	Vendor	File Version	Part
Artix-7 AC701 Evaluation Platform Add Companion Card Connections		Installed	xilinx.com	1.4	xc7a200tfbg676-2
Kria K26C SOM Add Companion Card Connections		Installed	xilinx.com	1.4	Commercial temperature grade K26 S
Kria K26I SOM Add Companion Card Connections		Installed	xilinx.com	1.4	Industrial temperature grade K26 SOI
Kintex-7 KC705 Evaluation Platform Add Companion Card Connections		Installed	xilinx.com	1.6	xc7k325tffg900-2

Figure 4.4: Vivado - New Project -> Board selection

Therefore it is possible for a board file to support multiple revisions of a physical board.

The `<parameters>` tag is used to list miscellaneous parameters of the board. It includes one or more nested `<parameter>` tags that define different features or properties of the board. The values of this tag are nearly always identical to those listed below 4.4.

```
<parameters>
    <parameter name="heat_sink_type" value="medium" value_type="string" />
    <parameter name="heat_sink_temperature" value_type="range" value_min="20.0" value_max="30.0" />
</parameters>
```

Listing 4.4: Parameters defined in board.xml

Now that we've covered the fundamental information, we can delve into detailing the components on the board by adding specific information. The `<component>` section forms a very important part of the board file because it defines the components found on the board, as well as different operating modes of the components, and the settings needed to enable these modes. The first one is the FPGA, for this, we also define the vendor and the pin map file. All of the IP cores used to implement interfaces between FPGA and board components must be defined between FPGA `<component>` tags.

As described in Table 4.1 there are a lot of interfaces that need to be mapped. For each one, we must specify the interface mode, name, preferred IP core to implement this interface, and preset name (*preset_proc*) which will link the IP core with predefined configurations in the preset.xml file. The interfaces section provides a listing of all the physical interfaces available on a `<component>`. The `<interfaces>` section contains one or more `<interface>` tags nested within. An interface is defined by multiple ports through the use of the `<port_map>` tag. Interfaces can be defined only inside a `<component>` of *"type=fpga"*. Each interface is further broken down into individual port maps. These port maps serve as a map of a logical port, that is defined in the interface, with a physical port, that relates to a physical package pin on the AMD device. Finally, in the `<pin_map>` section, each physical port is broken down into one or more individual pins depending on the width of the port being mapped. Pins can be shared across different physical ports of the interfaces they are defined in.

```xml
<components>
  <component name="part0" display_name="xcve2302 FPGA" type="fpga" part_name="xcve2302-sfva784-1LP-e-S-es1" pin_map_file="part0_pins.xml" vendor="avnet" spec_url="http://avnet.me/ve2302_iocc">
    <description>XCVE2302 FPGA</description>
    <interfaces>
      ...
      <!-- Push Buttons -->
      <interface mode="master" name="pmc_pb" type="xilinx.com:interface:gpio_rtl:1.0" of_component="pmc_pb" preset_proc="pmc_pb_preset">
        <preferred_ips>
          <preferred_ip vendor="xilinx.com" library="ip" name="axi_gpio" order="0"/>
        </preferred_ips>
        <port_maps>
          <port_map logical_port="TRI_I" physical_port="pmc_pb_tri_i" dir="in" left="1" right="0">
            <pin_maps>
              <pin_map port_index="0" component_pin="pmc_pb_0"/>
              <pin_map port_index="1" component_pin="pmc_pb_1"/>
            </pin_maps>
          </port_map>
        </port_maps>
      </interface>
      ...
    </interfaces>
  </component>
    ...
```

Listing 4.5: FPGA in board.xml with an example of interface declaration of 2 push buttons.

The different interfaces are further specified as components to provide them with all the essential information:

```
<component name="pmc_pb" display_name="PMC PB" type="chip"
   ↪ sub_type="led" major_group="General Purpose Input or
   ↪ Output" part_name="SML-LX0603GW-TR" vendor="LUMEX">
    <description>PMC Push Buttons</description>
</component>
```

Listing 4.6: Component declaration of the 2 PMC Push Buttons (10).

```
<component name="xpio_pb" display_name="XPIO PB" type="chip"
   ↪  sub_type="led" major_group="General Purpose Input or
   ↪ Output" part_name="SML-LX0603GW-TR" vendor="LUMEX">
    <description>XPIO Push Buttons</description>
</component>
```

Listing 4.7: Component declaration of the 2 XPIO Push Buttons (13).

```
<component name="xpio_dp" display_name="XPIO DIP" type="chip
   ↪ " sub_type="led" major_group="General Purpose Input or
   ↪  Output" part_name="SML-LX0603GW-TR" vendor="LUMEX">
    <description>XPIO DIP Switches</description>
</component>
```

Listing 4.8: Component declaration of XPIO 4-port dipswitch (13).

```
<component name="pmc_led" display_name="PMC LED" type="chip"
   ↪  sub_type="led" major_group="General Purpose Input or
   ↪ Output" part_name="SML-LX0603GW-TR" vendor="LUMEX">
    <description>PMC LEDs</description>
</component>
```

Listing 4.9: Component declaration of 2 PMC LEDs (11).

```xml
<component name="xpio_led" display_name="XPIO LED" type="chip" sub_type="led" major_group="General Purpose Input or Output" part_name="SML-LX0603GW-TR" vendor="LUMEX">
    <description>XPIO LEDs</description>
</component>
```

Listing 4.10: Component declaration of 3 XPIO LEDs (13).

Towards the end of the file, there is a section detailing all connections established between the identified system part ("part0") and various previously seen components. This section provides comprehensive information about each connection, including the nature of the relationship, typical delay, and index ranges associated with each component.

```xml
<connections>
    <connection name="part0_lpddr4_clk1" component1="part0" component2="lpddr4_clk1">
        <connection_map name="part0_lpddr4_clk1_1" typical_delay="5" c1_st_index="0" c1_end_index="1" c2_st_index="0" c2_end_index="1" />
    </connection>
    <connection name="part0_pmc_pb" component1="part0" component2="pmc_pb">
        <connection_map name="part0_pmc_pb_1" typical_delay="5" c1_st_index="2" c1_end_index="3" c2_st_index="0" c2_end_index="1" />
    </connection>
    <connection name="part0_xpio_pb" component1="part0" component2="xpio_pb">
        <connection_map name="part0_xpio_pb_1" typical_delay="5" c1_st_index="4" c1_end_index="5" c2_st_index="0" c2_end_index="1" />
    </connection>
    <connection name="part0_xpio_dp" component1="part0" component2="xpio_dp">
        <connection_map name="part0_xpio_dp_1" typical_delay="5" c1_st_index="6" c1_end_index="9" c2_st_index="0" c2_end_index="3" />
    </connection>
    <connection name="part0_pmc_led" component1="part0" component2="pmc_led">
        <connection_map name="part0_pmc_led_1" typical_delay="5" c1_st_index="10" c1_end_index="11" c2_st_index="0" c2_end_index="1" />
    </connection>
    <connection name="part0_xpio_led" component1="part0"
↪ component2="xpio_led">
        <connection_map name="part0_xpio_led_1"
↪ typical_delay="5" c1_st_index="12" c1_end_index="14"
↪ c2_st_index="0" c2_end_index="2" />
    </connection>
    <connection name="part0_LPDDR4_Controller0" component1="
↪ part0" component2="LPDDR4_Controller0">
        <connection_map name="part0_lpddr4_0_1"
↪ typical_delay="5" c1_st_index="300" c1_end_index="433"
↪  c2_st_index="0" c2_end_index="133" />
    </connection>
</connections>
```

Listing 4.11: Connections declarations of the various components (13).

4.5 part0_pins.xml

This file starts with `<part_info> </part_info>` tags in which it is specified the FPGA part used on the board. The new board takes his name from it, "xcve2302-sfva784-1LP-e-S-es1". Between these two tags we will be providing all the pin mapping information.

The name of the IP CORE follows a naming convention that is:

- **Family ("xcve2302"):** This typically refers to the family of FPGAs. For example, "xcve2302" belongs to the Versal Edge family or series.
- **Package ("sfva784"):** The package code specifies the physical package or form factor of the FPGA. It includes details about the size, pin count (e.g. 784), and other physical characteristics.
- **Speed Grade ("1LP"):** This speed grade indicates low-power variants. These FPGAs are optimized for power efficiency, making them suitable for battery-powered or power-sensitive applications.
- **Extended Standard Temperature Range ("e-S"):** The temperature range designation indicates whether the FPGA is suitable for industrial applications and the specific temperature range. The "e-S" suffix indicates that the FPGA is designed for an extended standard temperature range. Extended temperature ranges are useful in applications where the device may experience temperatures higher or lower than those covered by standard temperature ranges.

For comparison purposes, in the Vek280, the FPGA utilized was the "xcve2802-vsvh1760-2MP-e-S-es1", featuring a distinct package with more than double the pin count. This variant operates at the Mainstream Performance capacity of the FPGA, commonly expressed in terms of megahertz (MHz) or gigahertz (GHz).

As you can see in the pictures below, the pin map of the package of the VE2302 (Figure 4.5) is a smaller subset of the VE2802 one (Figure 4.6).

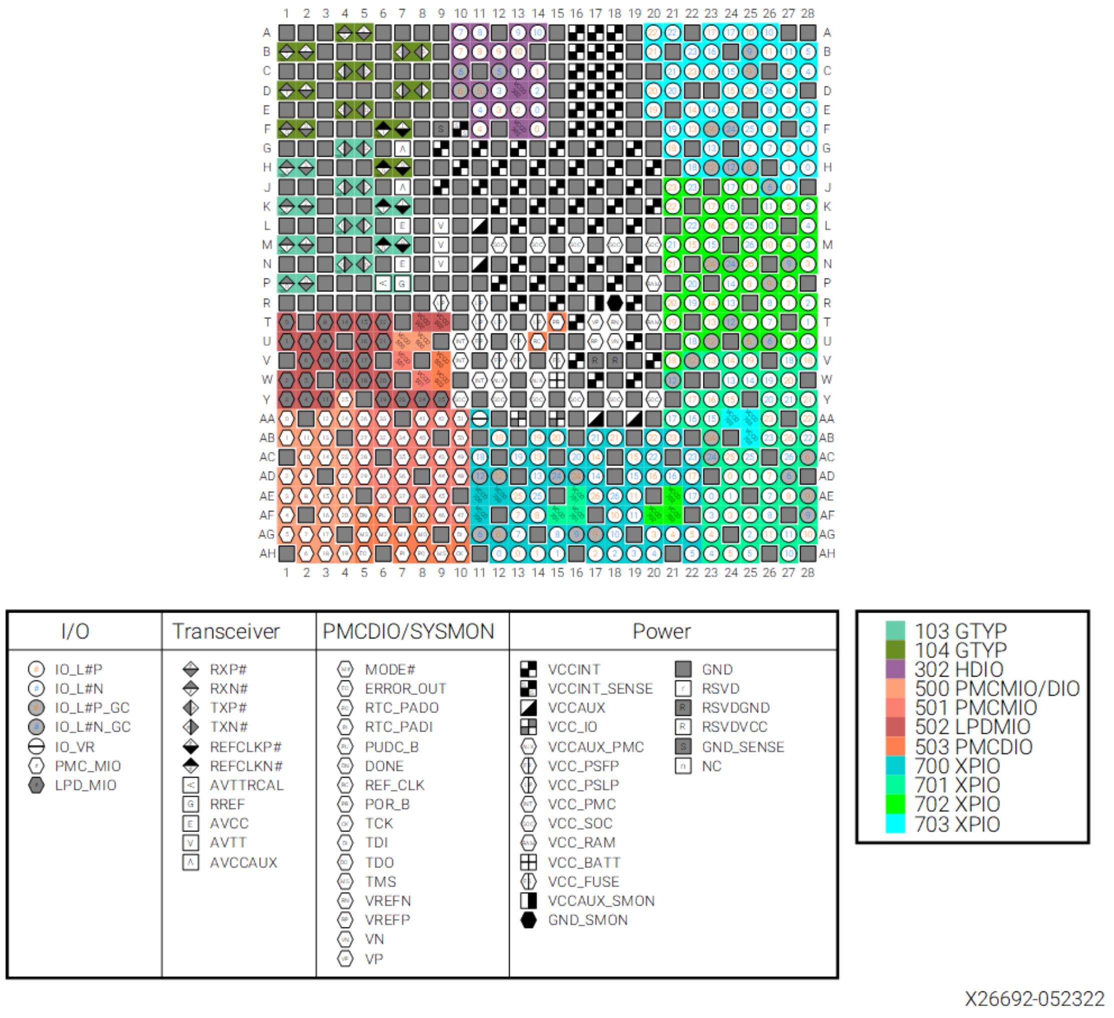


Figure 4.5: SFVA784 Package—VE2302 Pin Map

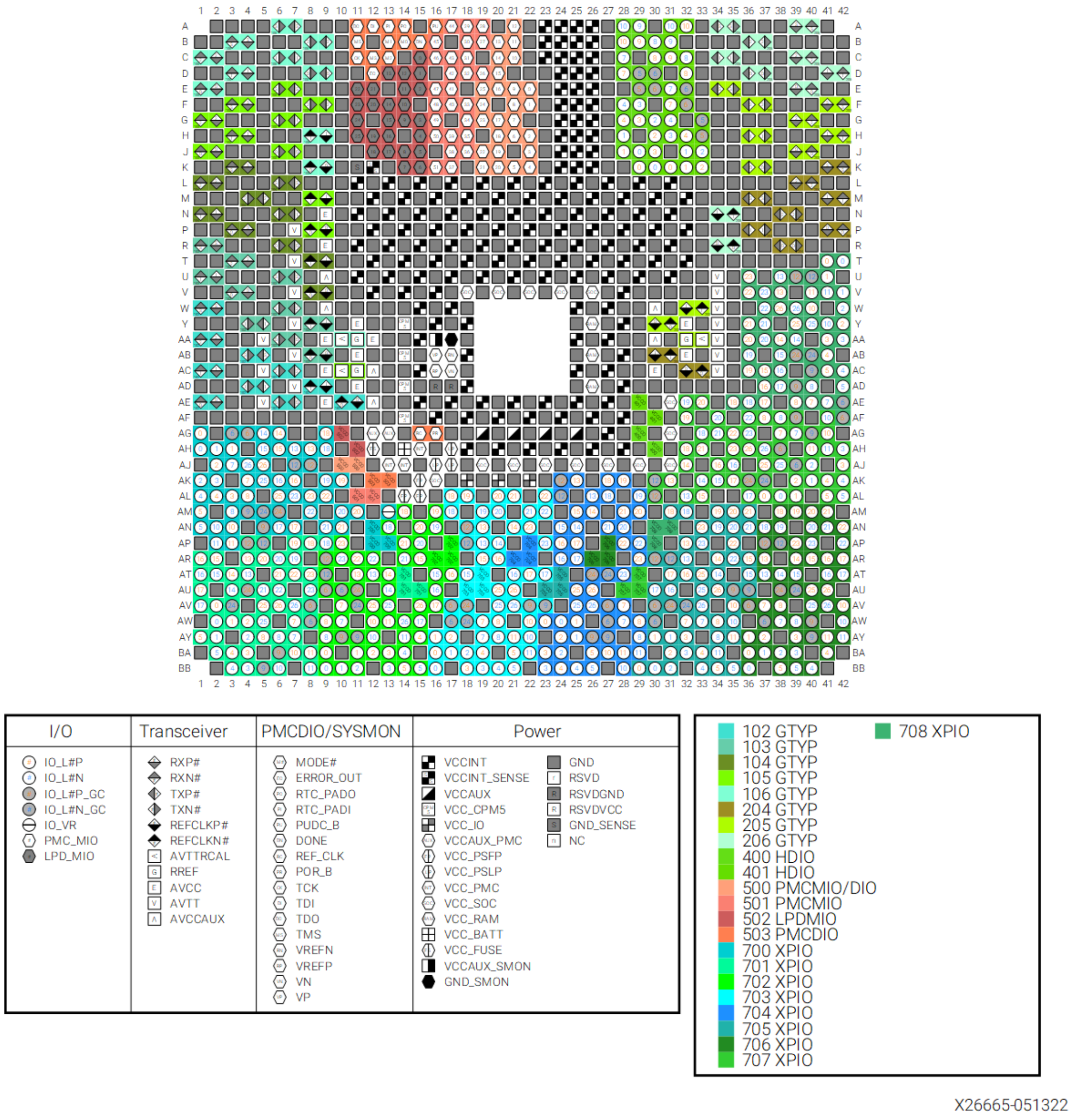


Figure 4.6: VSVH1760 Package—VE2802 Pin Map

The index and name serve as unique identifiers, pins listed here are linked to IP core port pins specified in the board.xml file by the pin name attribute in the board and preset files. The optional *"iostandard"* parameter specifies the programmable I/O Standard used for configuring input, output, or bidirectional ports on the target device (refer to the Xilinx Vivado I/O Standards documentation [9]). It is worth noting that pins in the same bank on the schematics must have compatible iostandards, starting from the same voltage level.

Before diving into the actual file describing how the pin locations are defined (see next section 4.5.2), I have to explain how to find the physical pin location in the schematics.

4.5.1 How to find the pin location

The primary purpose of the part0_pins.xml file is to instruct the program about the physical connections of various signals and components to the main core, designated as "xcve2302-sfva784-1LP-e-S-es1". Once a signal, such as those from the 4-port Dipswitch (Figure 4.3), has been identified, its corresponding point on the JX connector can be determined (See connectors). In this instance, signals should maintain consistent names across all schematics. To confirm, one can trace the signal's path to the connector on the carrier board and verify its name as it exits the pin on the opposite side. For clarity, only the connector on the carrier board is depicted here, as both connectors are identical (Figure 4.7).

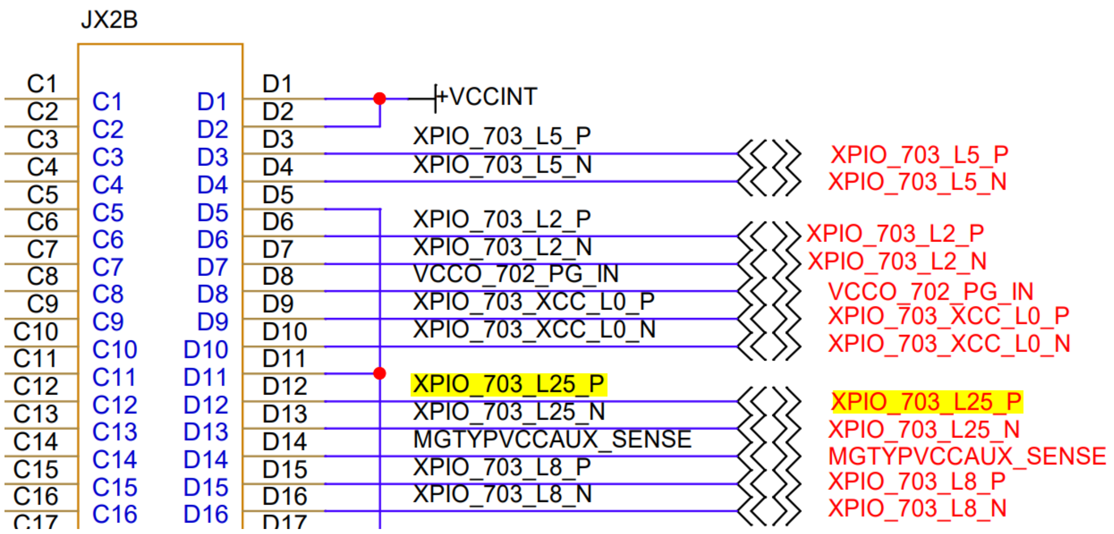


Figure 4.7: JX connector schematics

Once it is known the signal name used on the Som schematic, it remains only to follow it back to the source. In this example, it comes from the BANK 703 and the pin location is **E24** (Figure 4.8).

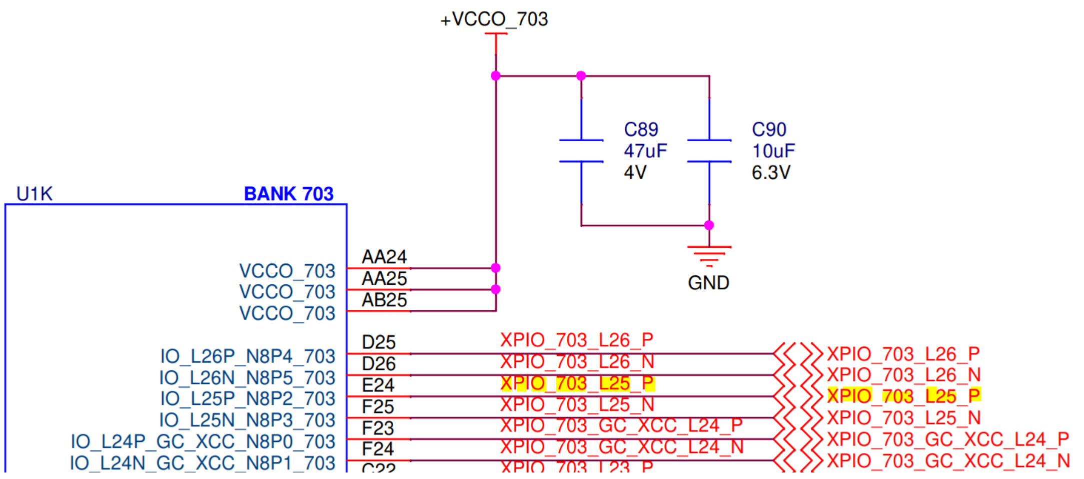


Figure 4.8: Cut of the schematic showing BANK 703 and the signal of the Dipswitch

The name shown on the inside of the component is the name used internally.

4.5.2 The Pins Map

The first elements specified in the `<pins>` tag are the LPDDR4 clocks and the GPIOs. This includes two pins for the clock positive and negative, four pins for the buttons, four pins for the 4-port Dipswitch, and an additional four pins for the LEDs (4.12). It can be seen the correspondence between the signals comments (<!– signal –>) and the signals in the schematics (Figure 4.9).

```
<part_info part_name="xcve2302-sfva784-1LP-e-S-es1">
  <pins>
    <pin index="0" name="lpddr4_clk1_p" iostandard="
    ↪ DIFF_SSTL15" loc="N23"/>
    <pin index="1" name="lpddr4_clk1_n" iostandard="
    ↪ DIFF_SSTL15" loc="N24"/>
    <!-- Push Buttons -->
    <!-- PMC_MIO46 -->
    <pin index="2" name="pmc_pb_0" iostandard="LVCMOS18" loc
    ↪ ="AF9"/>
    <!-- PMC_MIO37 -->
    <pin index="3" name="pmc_pb_1" iostandard="LVCMOS18" loc
    ↪ ="AU34"/>
```

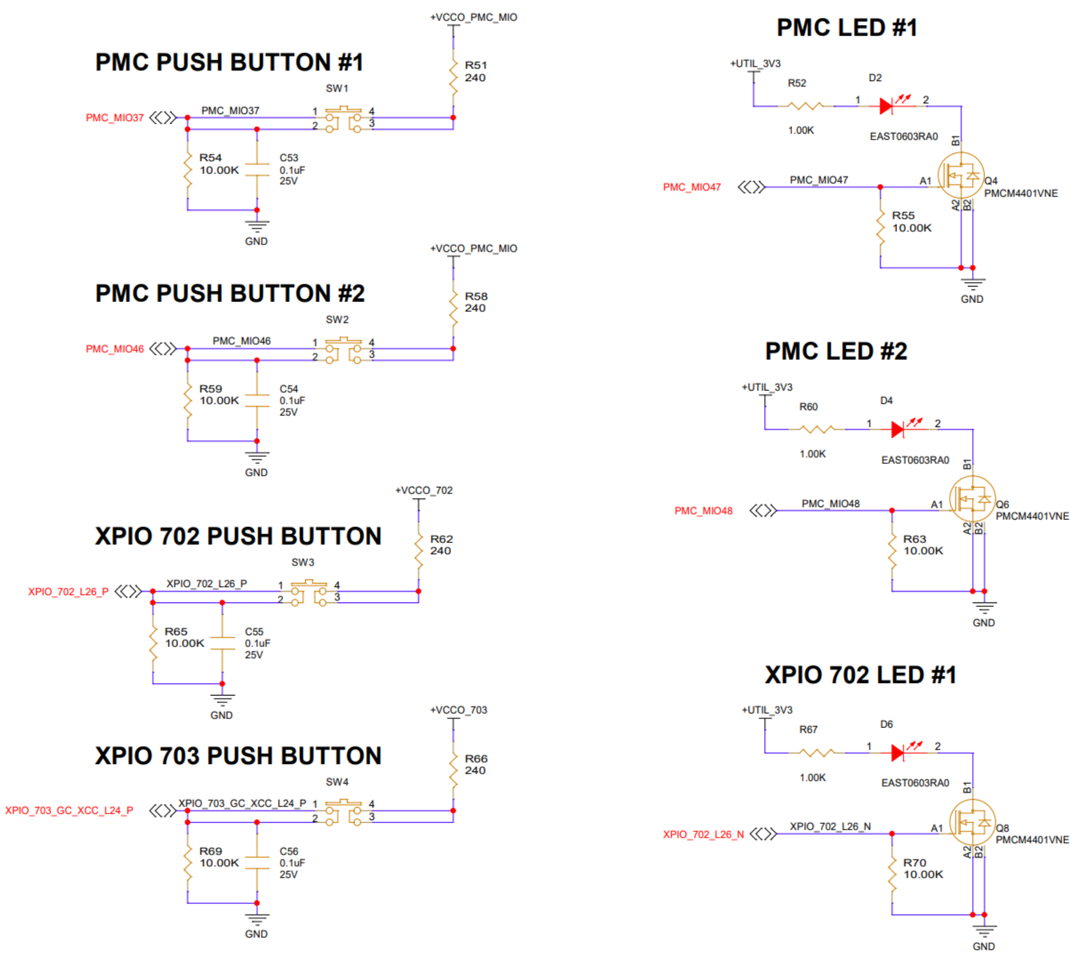


Figure 4.9: GPIO example schematics

```
    <!--  XPIO_702_L26_P  -->
    <pin index="4" name="xpio_pb_0" iostandard="LVCMOS18" loc="N25"/>
    <!-- XPIO_703_GC_XCC_L24_P -->
    <pin index="5" name="xpio_pb_1" iostandard="LVCMOS18" loc="F23"/>
    ...
  </pins>
</part_info>
```

Listing 4.12: First section of part0_pins.xml

Both boards feature LPDDR4 memory, and modifications were made to the physical locations of various pins by referencing the schematics (here the *"loc"* attribute is easier to find in respect as explained in Section 4.5.1 because the memory is directly connected to BANK 700 and 701). Notably, the changes included the removal of pins associated with two additional DDRs, as the new board incorporates a single DDR instead of three, along with the exclusion of LPDDR4 clocks (Figure 4.10).

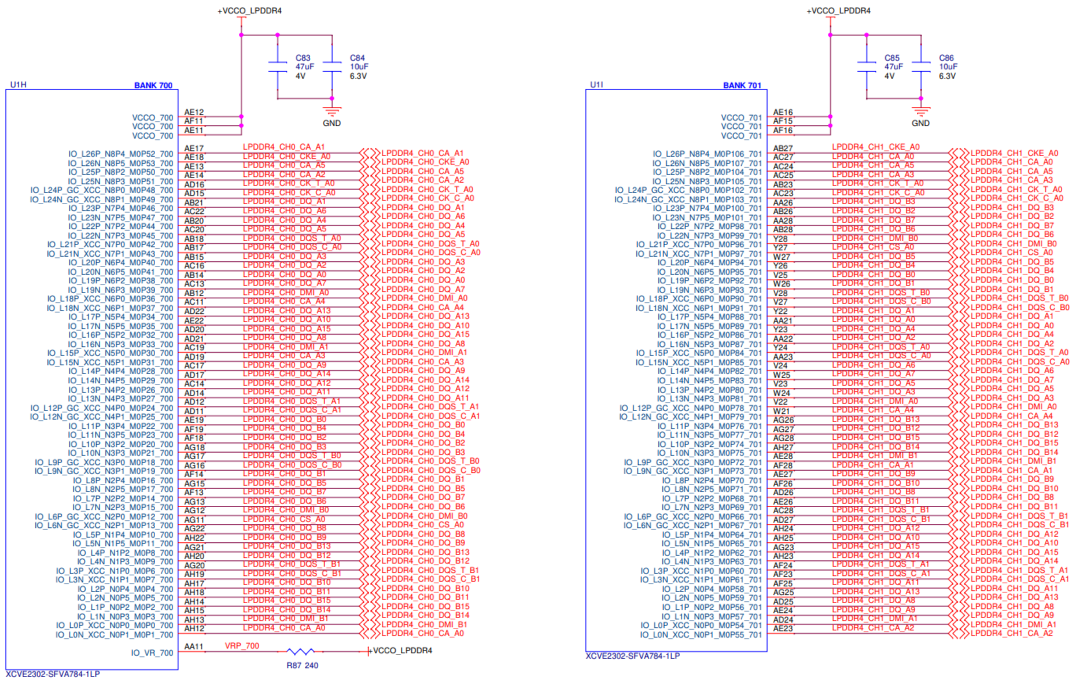


Figure 4.10: Versal AI Edge SOM LPDDR4.

```
<part_info part_name="xcve2302-sfva784-1LP-e-S-es1">
  <pins>
    <!-- LPDDR4_Channel_0 -->
    <pin index="300" name="lpddr4_0_dq0" loc="AB14"/>
    <pin index="301" name="lpddr4_0_dq1" loc="AB21"/>
    <pin index="302" name="lpddr4_0_dq2" loc="AC16"/>
    <pin index="303" name="lpddr4_0_dq3" loc="AB15"/>
    <pin index="304" name="lpddr4_0_dq4" loc="AB20"/>
    <pin index="305" name="lpddr4_0_dq5" loc="AC20"/>
    <pin index="306" name="lpddr4_0_dq6" loc="AC22"/>
    ...
  </pins>
</part_info>
```

Listing 4.13: LPDDR4_Channel_0 of part0_pins.xml

Furthermore, the FMC1 HSPC connectors were removed in the process. Subsequent attention was directed towards the analysis and adjustment of various input and output components on the new board.

Specifically, as seen before (Section 3.3.1), the new board integrates four push buttons, a 4-port dipswitch, and five LEDs (excluding the one managed by the I2C expander interface). These modifications ensure that the new board configuration aligns with the desired design specifications. The ongoing process involves a meticulous review and adjustment of the board's I/O characteristics, ensuring compatibility and adherence to the defined standards outlined in the Xilinx UltraScale I/O and Termination documentation [10].

4.6 preset.xml

Preset file helps customize an IP core in a particular configuration. The preset.xml file starts with an XML tag called `<ip_presets>` in which we must provide this file schema version. The current schema version for the preset file is "1.0":

```
<ip_presets schema = "1.0">
```

Listing 4.14: First tag of preset.xml

Within the `<ip_preset>` the `<ip>` section defines the specific IP that the preset values will apply to. First, we add configurations for the Versal CIPS IP Core. The Control, Interfaces, and Processing Subsystem (CIPS) is common to all Versal designs and contains all of the hardened IP that is common across all Versal devices. These configurations are linked to the board.xml file by the attribute called *"preset_proc_name"*.

```
<ip_preset preset_proc_name="ps_pmc_fixed_io_preset">
  <ip vendor="xilinx.com" library="ip" name="versal_cips"
        version="*" ip_interface="FIXED_IO">
    <user_parameters>
      <user_hier_parameter name="CONFIG.PS_PMC_CONFIG">
        ...
        <user_hier_parameter name="PMC_SD1_PERIPHERAL">
          <user_parameter name="ENABLE" value="1"/>
          <user_hier_parameter name="IO">
           <user_parameter name="PMC_MIO" value="38 .. 45"/>
          </user_hier_parameter>
        </user_hier_parameter>
        ...
      </user_hier_parameter>
    </user_parameters>
  </ip>
```

```
</ip_preset>
```

Listing 4.15: Extract of Versal CIPS definition in preset.xml with declaration of PMC_SD1_PERIPHERAL

Within the `<ip>` section, the `<user_parameters>` and `<user_parameter>` tags define the various configuration presets to apply to the specified IP core. It is important to understand that the *"name"* attribute and the *"value"* range, in this case "38 .. 45", are used to specify the name of the associated signals, here from PMC_MIO38 counting up to PMC_MIO45. This information can be derived from the Table 4.1.

The preset file also serves to configure all parameters of various IPs. The *"name"* attribute designates the name of the pre-configured property for the IP core, while the *value* sets the property. In the case of AMD target reference platforms or evaluation boards, the IP possesses knowledge of the FPGA pins employed on the target boards, known as board awareness. Leveraging this information, the IP integrator's board/connection automation feature can assist in connecting IP interfaces/ports to external ports on the board. IP integrator then generates the necessary physical and other I/O constraints required for the specified I/O port.

The recognition of preconfigured properties is facilitated by board awareness, which automatically selects the appropriate IP using the *"name"* property within the enclosing `<ip>` tag. An example is illustrated in the configuration of the LPDDR4_Controller0_preset (see 4.16).

Examples of board-aware IPs used here (with the full list available in the Vivado Design Suite User Guide [8]) include axi_gpio_v2_0 (utilized for LEDs, push buttons, and switches), axi_noc_v1_0 (employed for memory), clk_wiz_v6_0 (configured for the system clock), and versal_cips_v3_2 (defining interfaces communicating with the FPGA).

```
<ip_preset preset_proc_name="LPDDR4_Controller0_preset">
  <ip vendor="xilinx.com" library="ip" name="axi_noc"
   ↪ version="*">
    <user_parameters>
      <user_parameter name="CONFIG.CONTROLLERTYPE" value="
   ↪ LPDDR4_SDRAM" />
      <user_parameter name="CONFIG.MC_NO_CHANNELS" value="
   ↪ Dual" />
      <user_parameter name="CONFIG.MC_SYSTEM_CLOCK" value="
   ↪ Differential" />
      <user_parameter name="CONFIG.MC_MEMORY_SPEEDGRADE"
   ↪ value="LPDDR4-3733" />
      <user_parameter name="CONFIG.MC_DATAWIDTH" value="32"
   ↪ />
```

```
      <user_parameter name="CONFIG.MC_LP4_PIN_EFFICIENT "
   ↪ value="true" />
      <user_parameter name="CONFIG.MC_FREQ_SEL" value="
   ↪ MEMORY_CLK_FROM_SYS_CLK" />
      <user_parameter name="CONFIG.MC_IP_TIMEPERIOD0_FOR_OP"
   ↪  value="5000" />
      <user_parameter name="CONFIG.MC_OP_TIMEPERIOD0" value=
   ↪ "541" />
      <user_parameter name="CONFIG.MC_LP4_OVERWRITE_IO_PROP"
   ↪  value="true" />
    </user_parameters>
  </ip>
</ip_preset>
```

Listing 4.16: Configuration of LPDDR4 Controller0 in the preset.xml

4.6.1 xitem.json

Another important file not mentioned earlier is xitem.json. While it doesn't play a role in helping Vivado understand the board composition, it retains certain information about the board, its author, and the associated website. In this instance, I updated the search keyword and other values to align with the new ones.

```
{
  "config": {
    "items": [
      {
        "infra": {
          "name": "ve2302_iocc_reva",
          "display": "Versal VE2302 IOCC Evaluation Platform
   ↪ ",
          "revision": "0.1",
          "description": "Versal VE2302 IOCC Evaluation
   ↪ Platform",
          "company": "xilinx.com",
          "company_display": "Xilinx",
          "author": "Radaele",
          "contributors": [
            {
              "group": "Xilinx",
              "url": "www.xilinx.com"
            }
          ],
          "category": "Evaluation Boards",
          "website": "www.xilinx.com/ve2302",
          "search-keywords": [
            "ve2302_iocc_reva",
            "xilinx.com",
            "board",
            "Multiple Parts"
          ]
        }
      }
    ]
  },
  "_major": 1,
  "_minor": 0
}
```

Listing 4.17: xitem.json

Once done with the bdf, I committed all the files to the official GitHub repository of the bdf of Avnet [11] to let my college Mario Bergeron validate the design. Even though the board didn't exist yet, this was crucial to be able to build a DPU-TRD. The DPU-TRD (DPU Targeted Reference Design) is a document or package provided by Xilinx that serves as a reference design for implementing and using the DPU (Deep Learning Processing Unit). It includes information and resources that guide developers in implementing the DPU into their FPGA-based designs. After that, this part was concluded, waiting for the realization of the physical board to test.

4.7 Mario Bergeron

Before delving into the experiments, it is essential to introduce my colleague who has been an integral part of this project. Working closely together, Mario played a pivotal role in the collaborative efforts that shaped this thesis.

Mario Bergeron is a Machine Learning Specialist, specializing in embedded vision and AI at the edge. He has a Bachelor of Science degree in Computer Engineering from Université Laval in Québec City and he accumulated over 30 years of DSP and FPGA-based embedded design experience such as deep learning platforms, reference designs, and customer training [12].

Chapter 5

The benchmarks

In the meantime, courtesy of AMD, I was sent the VEK280 to carry out all the necessary measurements to calculate the theoretical performance of the new VE2302 board. To do that, we did some benchmarks trying various models from Vitis AI [3] on different boards I was able to put my hands on. The primary parameters to be measured are throughput (fps), latency (msec), and power consumption (W). These metrics are crucial for calculating the throughput/power ratio (fps/W), providing a more accurate assessment. While it's evident that the VEK280 outperforms all others in terms of throughput, it's equally important to consider the electrical consumption of each device. All this information will be useful to compute the theoretical performance of the new board.

5.1 The boards

I decided to use four boards, starting from the most used, based on availability (we needed to get the hands-on), cost, and compatibility with the task. These boards are versatile and can be used for a wide range of applications, including embedded systems development, hardware acceleration, prototyping, and testing. They are all built around AMD Xilinx Zynq UltraScale+ and offer an FPGA and various connectivity options, including interfaces such as USB, Ethernet, HDMI, and others, all needed to run prediction models that use USB cameras and monitors to display. The Zynq UltraScale+ MPSoC family is based on the Xilinx UltraScale MPSoC architecture. This family of products integrates a feature-rich 64-bit quad-core or dual-core Arm Cortex-A53 and dual-core Arm Cortex-R5F based processing system (PS) and Xilinx programmable logic (PL) UltraScale architecture in a single device. Also included are on-chip memory, multiport external memory interfaces, and a rich set of peripheral connectivity interfaces.

- **ZUBoard 1CG:** A versatile board featuring the Xilinx Zynq UltraScale+

MPSoC ZU1CG. It has an FPGA, a Dual ARM Cortex-A53 processor, a DDR4 memory, and multiple connectivity options[13].

- **Ultra96-V2:** An ARM-based development board based on the Xilinx Zynq UltraScale+ MPSoC ZU3EG. It features an FPGA, a Quad ARM Cortex-A53, and a Dual Cortex-R5 processors, WiFi/Bluetooth, HDMI, USB, and other interfaces[14]. The EG (and EV) versions feature an Arm Mali-400MP2, a GPU from the Mali series produced by ARM Holdings, one of the most widely shipped mobile GPUs across various platforms. Chosen for its emphasis on minimizing power and bandwidth consumption, the Mali-400 GPU was selected for cost-effective devices.

- **UltraZed-EV:** is a high-performance, full-featured, System-On-Module(SOM) based on the AMD Xilinx Zynq UltraScale+ MPSoC ZU7EV family of devices. It features an FPGA, a Quad ARM Cortex-A9, and a Dual Cortex-A53 processor, a DDR memory, and a variety of connectivity options[15]. In addition to the EG version of the Zynq the EV also has H.264/H.265 Video Codec[16].

- **VEK280:** The VEK280 Evaluation Kit featuring the AMD Versal AI Edge VE2802 Adaptive SoC. Key Features: AIE-ML and DSP hardware acceleration engines, high-speed connectivity, optimized for ML inference applications in automotive, vision, industrial, scientific, and medical sectors[6]. The production is limited, with exclusivity due to high cost. The board is described in detail in Chapter 2.

5.2 The models

The models used to test the performances were chosen among the ones available on the VitisAI GitHub repository [17]. They can be seen in the Table 5.1.

Model	Input	ModelZoo name
Inception V4	229x229	tf_inceptionv4_imagenet_299_299_24.55G
VGG-16	224x224	tf_vgg16_imagenet_224_224_30.96G
ResNet-50	224x224	tf_resnetv1_50_imagenet_224_224_6.97G
Mobilenet-V1	224x224	tf_mobilenetv1_1.0_imagenet_224_224_1.14G
Mobilenet-V2	224x224	tf_mobilenetv2_1.0_imagenet_224_224_0.59G
SSD Mobilenet-V1	300x300	tf_ssdmobilenetv1_coco_300_300_2.47G
SSD Mobilenet-V2	300x300	tf_ssdmobilenetv2_coco_300_300_3.75G

Table 5.1: Models used.

5.2.1 Inception v4

Inception-v4, a convolutional neural network (CNN) architecture belonging to the Inception family, was developed by Google to address limitations observed in its predecessors, including Inception-v1, Inception-v2 (also known as BN-Inception), and Inception-v3. Known for its innovative use of inception modules, which involve parallel convolutional operations with various filter sizes, Inception-v4 focuses on capturing multi-scale features efficiently [18]. Key features of Inception-v4 include the introduction of Inception Modules such as Inception-A, incorporating a 1x1 convolution to enhance computational efficiency, Inception-B with factorized 7x7 convolutions, and Inception-C with parallel branches capturing multi-scale information. The architecture incorporates Reduction Modules to downsample feature map dimensions and a sophisticated Stem Network for richer feature representations. It leverages factorized convolutions (separable convolutions) to reduce parameters and computations, contributing to improved training efficiency. Inception-v4 adopts Global Average Pooling (GAP) before the final fully connected layer, reducing spatial dimensions to a single value per channel. This architecture aims to enhance training efficiency, generalization, and computational performance compared to its predecessors. Notably, it is sometimes referred to as "Inception-ResNet-v2" due to the incorporation of residual connections inspired by ResNet architectures. This integration addresses challenges like the vanishing gradient problem, promoting smoother optimization. Inception-v4 is designed as a potent model applicable to various computer vision tasks, including image classification and object detection.

5.2.2 VGG-16

VGG-16, a convolutional neural network (CNN) architecture belonging to the Visual Geometry Group (VGG) family, was introduced by researchers Karen Simonyan and Andrew Zisserman from the University of Oxford. As documented in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition"[19], VGG-16 is recognized for its simplicity and effectiveness, specifically designed for image classification tasks. The architecture's key characteristics include its significant depth, comprising 16 layers denoted by the "16" in its name. This depth encompasses 13 convolutional layers and 3 fully connected layers, facilitating the learning of hierarchical features of increasing complexity. VGG-16 employs small 3x3 convolutional filters in its convolutional layers, utilizing the rectified linear unit (ReLU) activation function. The use of multiple convolutional layers captures hierarchical representations and allows the network to learn local features. For downsampling spatial dimensions, VGG-16 incorporates max-pooling layers with a 2x2 window and a stride of 2. Three fully connected layers follow the convolutional layers, with the first two layers housing 4,096 neurons each. The third fully connected layer serves as the output layer, containing 1,000 neurons for tasks such as ImageNet classification. Throughout the network, ReLU activation functions introduce non-linearity. VGG-16 distinguishes itself with a relatively large number of parameters, contributing to computational expense and potential higher memory requirements. A distinctive feature is the use of 3x3 convolutional filters throughout the architecture. The absence of batch normalization, a common feature in recent architectures, is notable. Instead, VGG-16 includes dropout in fully connected layers as a regularization technique to prevent overfitting. Although VGG-16 is less prevalent today due to the adoption of more lightweight architectures like ResNet and Inception, its simplicity and efficacy have left a lasting impact on the development of deeper models. It laid the groundwork for subsequent architectures, including VGG-19, influencing the history of deep learning.

5.2.3 ResNet-50

ResNet-50, a variant of the ResNet (Residual Network) architecture, addresses challenges in training deep neural networks. Introduced by Kaiming He and team[20], ResNet-50 features:

- **Residual Blocks:** Key to ResNet, these blocks employ a "shortcut" for gradient flow, enabling training of very deep networks.
- **Depth and Architecture:** With 50 layers, including convolutional layers and residual blocks, ResNet-50 surpasses VGG in depth.
- **Bottleneck Design:** Residual blocks in ResNet-50 use a bottleneck design, reducing computational complexity for efficiency.
- **Global Average Pooling (GAP):** Replacing fully connected layers, GAP reduces spatial dimensions for efficient classification.
- **Shortcut Connections:** Enhance gradient flow during training, critical for very deep architectures.
- **Pre-activation Residual Blocks:** Introduce batch normalization and ReLU activation before convolutional layers for improved training.
- **Pooling and Stride:** Strategic use of max pooling and strided convolutions downsamples spatial dimensions.

Efficient and potent for image classification, ResNet-50's architecture has influenced deeper variants like ResNet-101 and ResNet-152.

5.2.4 MobileNetV1 & MobileNetV2

MobileNetV1, designed by Andrew G. Howard and his team, is a lightweight convolutional neural network (CNN) optimized for efficient image classification on mobile and edge devices. It employs depthwise separable convolutions, introducing width and resolution multipliers for flexible trade-offs between model size, accuracy, and computational efficiency. The unique bottleneck design and ReLU6 activation contribute to its efficiency, making it suitable for real-time applications[21].

Building upon MobileNetV1, MobileNetV2 introduces an inverted residual structure with linear bottlenecks and shortcut connections, inspired by ResNet-50. This evolution enhances feature representation and mitigates vanishing gradient issues. MobileNetV2 retains the width and resolution multipliers, maintaining flexibility. Both models utilize global average pooling for efficient classification[22].

While MobileNetV1 focuses on lightweight efficiency, MobileNetV2 refines the balance between efficiency and performance. These models offer versatile solutions for diverse real-time applications on resource-constrained devices, each addressing specific nuances in accuracy and computational efficiency.

5.2.5 SSD MobileNetV1 & SSD MobileNetV2

SSD MobileNetV1[23] and SSD MobileNetV2[24] are variants of the Single Shot Multibox Detector (SSD) object detection model. They use MobileNetV1 and MobileNetV2(Section 5.2.4) as their base architecture and improve their efficiency by making trade-offs between accuracy and computation. They efficiently detect objects of varying sizes and shapes, making them suitable for real-time applications on resource-constrained devices. SSD MobileNetV2 employing the enhanced MobileNetV2 architecture, with inverted residuals and linear bottlenecks for improved feature extraction, achieves enhanced accuracy while maintaining efficiency.

5.3 The theoretical results

Before talking about our benchmarking of the different models on the selected devices, we need to dive deeper in the technical implementation of the processing unit of the Versal family and to see how they are expected to behave based on their specific product specification. This will help to derive some formulas that will permit us to estimate the future performance of the VE2302 on the same models and tasks.

5.3.1 The DPUCV2DX8G

The AMD Versal Deep Learning Processing Unit (DPUCV2DX8G) stands as a configurable computational engine meticulously designed to optimize convolutional neural networks within Versal Adaptive SoCs devices equipped with AI Engines. Tailored for Versal devices featuring AI Engine-ML tiles, the DPUCV2DX8G offers specialized functionality. Conversely, for Versal devices equipped with AI Engines (excluding AI Engine-ML), the recommended choice is the DPUCVDX8G. The level of parallelism incorporated into the engine is a flexible design parameter, providing adaptability based on the target device and specific application requirements. The DPU is equipped with a comprehensive set of highly optimized instructions and extends support to a wide array of convolutional neural networks, including VGG, ResNet, GoogLeNet, YOLO, SSD, MobileNet, FPN, among others[25].

DPUCV2DX8G Features:

- One AXI4-Lite slave interface for accessing configuration and status registers.
- One AXI4 master interface for DPU instruction fetch.
- Supports a configurable number of AXI4 interfaces for activation and feature map load/store.
- Supports 20 AI Engines-ML tiles per batch handler, where BATCH_N=[1,14].

Key Supported Operators:

- Convolution and transposed convolution.
- Depthwise convolution and depthwise transposed convolution.
- Max pooling.
- Average pooling.
- ReLU, ReLU6, Leaky ReLU, Hard Sigmoid, and Hard Swish.
- Elementwise-Sum and Elementwise-Multiply.
- Dilation.
- Reorg.
- Fully connected layer.
- Concat, Batch Normalization.

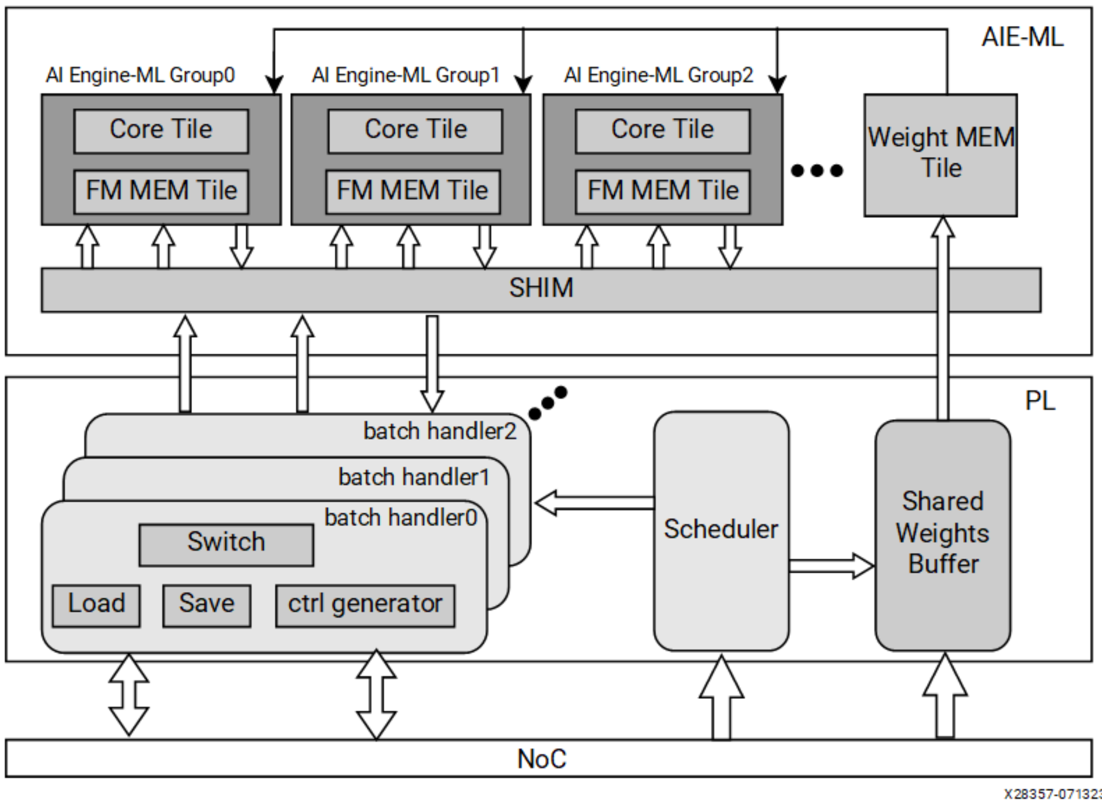


Figure 5.1: DPUCV2DX8G Block Diagram (top-level).

The DPUCV2DX8G is a high-performance, user-configurable convolutional neural network (CNN) processing engine optimized for Versal Adaptive SoCs with AI Engine-ML tiles. It is designed for use in Versal AI Edge and AI Core Adaptable SoCs, as well as AMD Alveo V70 accelerator cards. Utilizing both AI Engine-ML tiles and programmable logic (PL) resources, this IP exposes configurable parameters for AI Engine-ML tiles and PL resource utilization. AI Engine-ML tiles perform convolution and non-convolution operations, and their interface tiles facilitate data transfer between memory tiles and the PL. Each DPU core consists of an AI Engine-ML group, and for multi-batch architectures, each batch handler has a private AI Engine-ML group and memory tile group. The PL component includes a high-level scheduler module and batch handlers for Load and Save, implemented as shared logic for all DPUCV2DX8G batch handlers.

In the Figure 5.2 is shown an example system in which video frames are captured by a Versal device with a connected MIPI camera. The DPUCV2DX8G is integrated into the Versal Adaptive SoCs via the high-performance network on chip (NoC) and SmartConnect to enable deep learning inference tasks, such as image classification, object detection, and semantic segmentation.

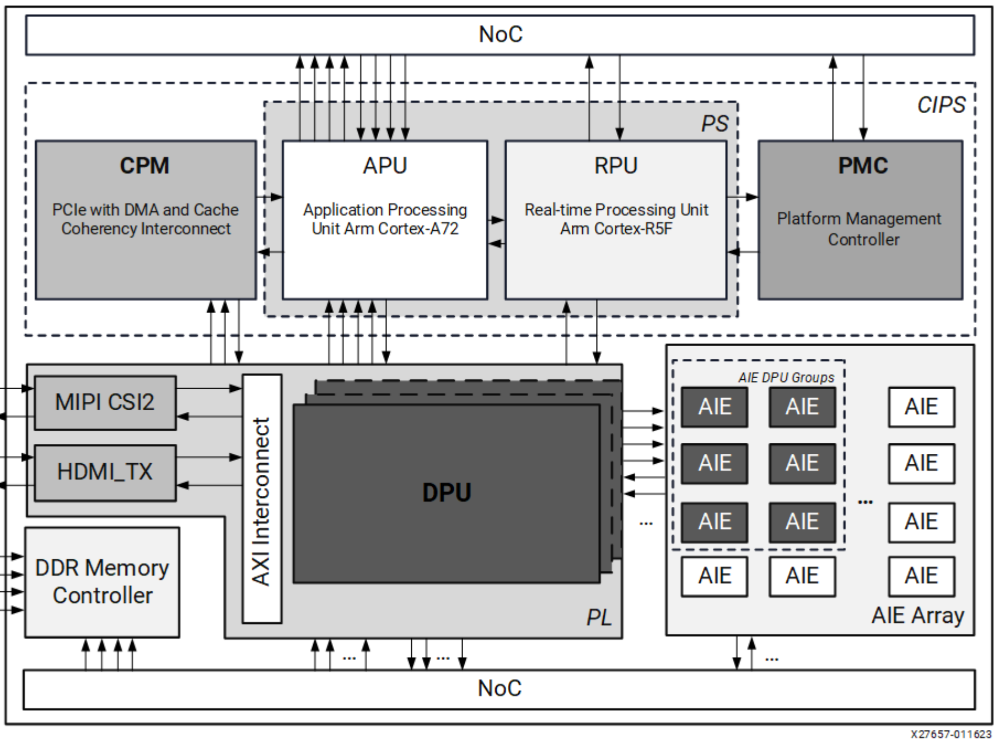

NoC
CIPS
PS
CPM
PCIe with DMA and Cache Coherency Interconnect
APU
Application Processing Unit Arm Cortex-A72
RPU
Real-time Processing Unit Arm Cortex-R5F
PMC
Platform Management Controller
Camera
MIPI CSI2
HDMI
HDMI_TX
DDR4/ LPDDR4
DDR Memory Controller
AXI Interconnect
DPU
PL
AIE DPU Groups
AIE
AIE Array
NoC
X27657-011623

Figure 5.2: Versal Adaptive SoC Integration Example.

5.3.2 Resource Utilization

Architecture	AI Engine Conv Cores	AI Engine Non-Conv Cores	Memory Tiles	LUT	FF	Block RAM	Ultra RAM	DSP	PL NMU
C20B1CU1	16	4	8	42027	51742	12.5	0	78	4
C20B2CU1	32	8	12	53648	63662	14	0	79	5
C20B3CU1	48	12	16	64638	74197	15.5	0	80	6
C20B4CU1	64	16	20	74816	84441	17	0	81	7
C20B5CU1	80	20	24	84612	94807	18.5	0	82	8
C20B6CU1	96	24	28	94744	105087	20	0	83	9
C20B7CU1	112	28	32	104717	115335	21.5	0	84	10
C20B8CU1	128	32	36	114693	125665	23	0	85	11
C20B9CU1	144	36	40	124665	135980	24.5	0	86	12
C20B10CU1	160	40	44	134632	146252	26	0	87	13
C20B11CU1	176	44	48	144476	156663	27.5	0	88	14
C20B12CU1	192	48	52	154120	166893	29	0	89	15
C20B13CU1	208	52	56	164545	177182	30.5	0	90	16
C20B14CU1	224	56	60	174431	187467	32	0	91	17

Table 5.2: Resources Utilization of Different DPUCV2DX8G Architectures

Table 5.2 illustrates the resource utilization of various DPUCV2DX8G architectures. In the example of C20B14CU1, that is the architecture implemented in the VEK280 (2.5), *"C20"* signifies 20 AI Engine-ML Cores per batch handler, *"B14"* indicates 14 batch handlers, and *"CU1"* denotes one compute unit. We must remember that the VE2302 will implement a C20B1CU1 with just one batch handler.

5.3.3 Resource Utilization

The peak number of INT8 operations per clock cycle of a single AI Engine-ML tile is 512. Thus, the total peak theoretical performance is calculated as 512 * TCPB_N * BATCH_N * CU_N * AIE_ACT_FREQ, where:

- **TCPB_N:** is the total number of AI Engine-ML cores per batch handler (for example C20, TCPB_N = 20).
- **BATCH_N:** is the maximum batch size (for example B14, BATCH_N=14).
- **CU_N:** is always 1 in this 1.0 IP release.

Example:
C20B14CU1 (14 AI Engine-ML cores per Batch, 16 cores for Conv, 4 cores for non-Conv)
Peak TOPs = 512 * 20 * 14 * 1 * 1.18 GHz
Peak TOPs = 169.16
The table showing the theoretical results measured in TOPS (Trillions or Tera Operations per Second) can be seen below (Table 5.3).

The table show us that we should expect on the new board a performance 14 times less powerful. This has not to be seen as a negative aspect as we are addressing a different target of market and it needs to be considered that these are theoretical results. Also compared with the other boards considered in the benchmark it outperforms all of them (see Table 5.4). The empirical data will help to understand the actual performance we should expect on the different models.

Architecture	**Peak Theoretical Performance (TOPS)**
C20B1CU1	**12.08**
C20B2CU1	24.17
C20B3CU1	36.25
C20B4CU1	48.33
C20B5CU1	60.42
C20B6CU1	72.5
C20B7CU1	84.58
C20B8CU1	96.67
C20B9CU1	108.75
C20B10CU1	120.83
C20B11CU1	132.92
C20B12CU1	145.0
C20B13CU1	157.08
C20B14CU1	**169.16**

Table 5.3: Peak Theoretical INT8 Performance of the DPUCV2DX8G

	ZUB1CG	**U96V2**	**UZ7EV**	**VE2303**	**VEK280**
Device	1CG	3EG	7EV	VE2302	VE2802
DPU	1-B512	1-B2304	2-B4096	C20B1CU1	C20B14CU1
Frequency	300MHz	200MHz	300MHz	1.18GHz	1.18GHz
Peak TOPS	0.15	0.46	2.46	12.08	169.16

Table 5.4: Resources Utilization of Different DPUCV2DX8G Architectures

5.4 The experiments

The Xilinx Model Zoo contains many pre-built convolutional neural network models. The first thing to do is to create the SD card that will boot on the board that we will measure. To do so Mario Bergeron has created all the pre-built Vitis-AI SD card images that are needed for the chosen boards. These will already contains the models we want to use.

5.4.1 Running the models

To begin, navigate to the classification application examples in the Vitis-AI-Library (5.1).

```
cd ~/Vitis-AI/examples/vai_library/samples/classification
```

Listing 5.1: Example Shell Script

Next, build the models to prepare for execution (5.2).

```
./build.sh
```

Listing 5.2: Example Shell Script

Finally, execute the model using the specified command (5.3).

```
./test_performance_classification <model>
    test_performance_classification.list
    -t <number of threads>
    -s <number of seconds>
    -l <log file name>
```

Listing 5.3: Example Shell Script

5.4.2 Instruments

All the power measurements were performed directly at the source using the Kuman Power Meter (Figure 5.3 left). This instrument supports different modes, the one used here is the **Mode 1:** Time/ Watt/ Cost (Figure 5.3 right).

- **Cumulative Time:** How long you have monitored the device.
- **Watt:** Real-time current Watt of your device.
- **Cost:** How much have you cost (based on the unit price you set). Here set to zero as we don't have prior knowledge about that.

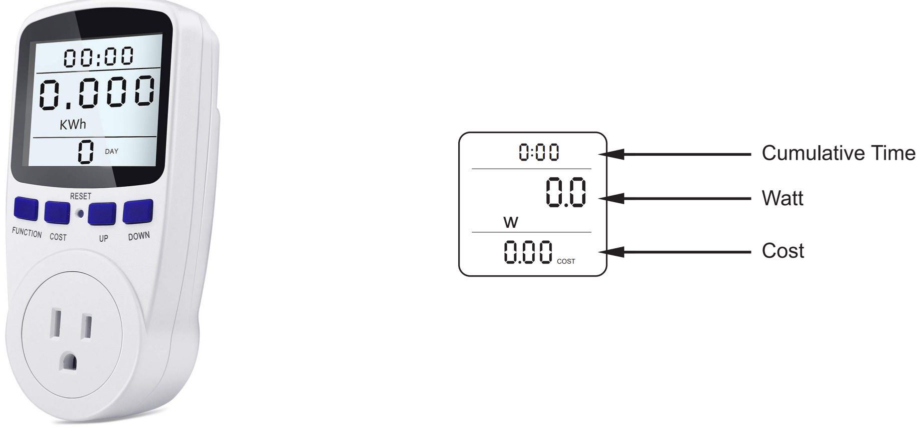


Figure 5.3: Kuman Power Meter (left) and Operating Mode (right)

The usage is very straightforward; you just need to plug it in, power it up, and connect the board. Afterward, we ran each model on every board, taking notes of all the results. By combining the various results obtained from the instrument, a graph illustrating the consumption of the various models can be generated. Here is an example for the VEK280 (C20B14 is the DPU used, as indicated in Table 5.2).

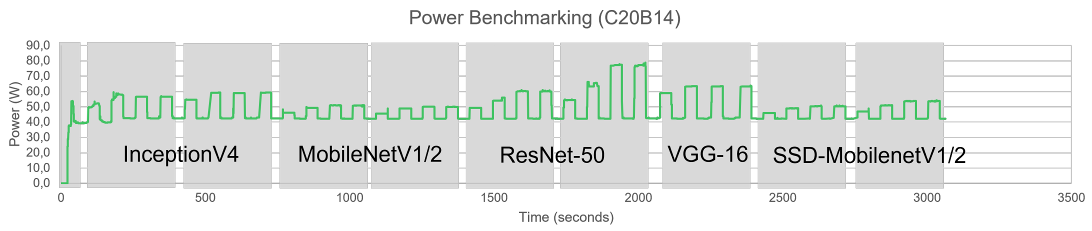


Figure 5.4: Power Graph of VEK280.

5.4.3 Comparison across all boards

Model	ZUB1CG	U96V2	UZ7EV	VEK280
Inception V4	5/5	10/10	30/60/62	467/636/637
VGG-16	5/5	12/12	21/38/39	568/712/712
ResNet-50	17/18	29/30	85/160/172	1801/3296/5099
Mobilenet-V1	81/86	133/145	317/619/799	2381/4648/5200
Mobilenet-V2	89/94	114/123	258/499/611	2298/4472/5242
SSD Mobilenet-V1	35/40	53/65	103/216/319	906/1698/1980
SSD Mobilenet-V2	20/22	32/35	79/158/204	820/1564/1969

Table 5.5: Throughput (fps) - 1/2/N threads (N=4 for UZ4EV, N-16 for VEK280)

Model	ZUB1CG	U96V2	UZ7EV	VEK280
Inception V4	194.64	101.41	32.77	29.90
VGG-16	206.83	82.50	46.75	24.60
ResNet-50	57.76	36.52	11.77	7.76
Mobilenet-V1	12.35	7.78	3.15	5.87
Mobilenet-V2	11.25	9.09	3.86	6.08
SSD Mobilenet-V1	28.81	20.02	9.78	15.44
SSD Mobilenet-V2	49.50	36.32	14.79	17.04

Table 5.6: Latency (msec) - 1 thread

Upon reviewing the various tables (refer to Tables 5.5, 5.6, 5.7, 5.8), it becomes evident that the VEK280, equipped with the latest Versal Adaptive SoCs, surpasses all other boards in performance. The throughput ranges from 6 to 30 times faster compared to the closest competitor. Furthermore, the relative throughput concerning power consumption (fps/watt) is 6 to 9 times higher. This remarkable performance highlights the superiority of the new Versal Adaptive SoCs in the field of Artificial Intelligence inference. It underscores the potential for various real-time applications, emphasizing the technology's prowess in enabling cutting-edge AI solutions. The results showcase the VEK280 as a leading platform for demanding AI workloads, illustrating its capability to drive advancements in AI applications and their real-time implementations.

Model *Idle power*	**ZUB1CG** *7.7*	**U96V2** *11.9*	**UZ7EV** *22.4*	**VEK280** *42.4*
Inception V4	1.1	3.2	6.2	12.4
VGG-16	1.1	4.7	5.5	16.8
ResNet-50	1.2	2.7	5.2	12.3
Mobilenet-V1	1.2	2.4	3.6	4.0
Mobilenet-V2	0.9	1.4	3.2	3.4
SSD Mobilenet-V1	1.1	2.3	3.6	3.8
SSD Mobilenet-V2	1.0	1.8	2.9	4.9

Table 5.7: Power (watts) - 1 thread

Model	**ZUB1CG**	**U96V2**	**UZ7EV**	**VEK280**
Inception V4	4.5	3.1	4.8	37.7
VGG-16	4.5	2.6	3.8	33.8
ResNet-50	14.2	10.7	16.3	146.4
Mobilenet-V1	67.5	55.4	88.1	595.3
Mobilenet-V2	98.9	81.4	80.6	675.9
SSD Mobilenet-V1	31.8	23.0	28.6	238.4
SSD Mobilenet-V2	20.0	17.8	27.2	167.3

Table 5.8: Throughput/delta-Power (fps/W) - 1 thread

5.4.4 Comparison between the VEK280 and VE2302 DPU

Considering the VEK280 as the most comprehensive Versal device, we simulated the integration of the C20B1CU1 DPU into it to project the performance that the VE2302 would attain. The subsequent tables present the results of these computations, providing insights into the throughput, power consumption, and relative throughput in relation to power. The comparison is drawn between the VEK280 (C20B14CU1) and the C20B1CU1, which was implemented in the VEK280 to simulate the VE2302 performance.

Model	**C20B1**	**C20B14**
Inception V4	71 / 74 / 74 / 74	468 / 638 / 633 / 638
VGG-16	53 / 54 / 54 / 54	572 / 719 / 718 / 718
ResNet-50	309 / 354 / 354 / 354	1800 / 3252 / 5082 / 5025
Mobilenet-V1	826 / 1270 / 1256 / 1255	2373 / 4286 / 5279 / 5192
Mobilenet-V2	726 / 1036 / 1035 / 1035	2300 / 4473 / 5259 / 5155
SSD Mobilenet-V1	363 / 587 / 587 / 586	905 / 1650 / 1976 / 1983
SSD Mobilenet-V2	229 / 301 / 301 / 301	821 / 1500 / 1964 / 1966

Table 5.9: Throughput (fps) - 1 / 2 / 14 / 28 threads

Model	**C20B1**	**C20B14**
Idle power	*38.2*	*42.2*
Inception V4	2.5 / 2.6 / 2.6 / 2.6	12.5 / 17.0 / 17.1 / 17.1
VGG-16	2.6 / 2.6 / 2.6 / 2.6	16.8 / 21.2 / 21.2 / 21.2
ResNet-50	3.3 / 3.7 / 3.7 / 3.7	12.2 / 23.4 / 35.3 / 35.1
Mobilenet-V1	2.1 / 3.1 / 3.1 / 3.1	4.0 / 7.0 / 8.7 / 8.6
Mobilenet-V2	1.6 / 2.2 / 2.2 / 2.2	3.4 / 6.7 / 7.8 / 7.7
SSD Mobilenet-V1	2.0 / 3.2 / 3.2 / 3.2	3.7 / 6.8 / 8.2 / 8.2
SSD Mobilenet-V2	1.9 / 2.6 / 2.6 / 2.6	4.7 / 8.8 / 11.6 / 11.6

Table 5.10: Power (watts) - 1 / 2 / 14 / 28 threads

The obtained results indicate a decrease in throughput by factors of 5×, 7×, 8× and 8× for the C20B1 compared to the more powerful C20B14. This outcome aligns with expectations, given the reduced AI Engine-ML cores per Batch, dropping from 14 to just one in the C20B1 architecture. However, an essential observation is that this decrease in performance comes with a more budget-friendly price point. While the exact pricing of the VE2302 (C20B1) is not yet determined, the VEK280 (C20B14) costs over 10 thousand euros.

Interestingly, the results reveal that despite the reduction in performance, the trade-off is justified by the cost efficiency. The fps over watt factor, while

Model	C20B1	C20B14
Inception V4	28 / 28 / 28 / 28	37 / 37 / 37 / 37
VGG-16	20 / 21 / 21 / 21	34 / 34 / 34 / 34
ResNet-50	94 / 96 / 96 / 96	147 / 139 / 144 / 143
Mobilenet-V1	393 / 409 / 405 / 405	593 / 612 / 607 / 603
Mobilenet-V2	454 / 471 / 470 / 470	677 / 667 / 674 / 670
SSD Mobilenet-V1	181 / 183 / 183 / 183	244 / 243 / 241 / 241
SSD Mobilenet-V2	114 / 116 / 116 / 116	174 / 170 / 170 / 170

Table 5.11: Throughput/delta-Power (fps/watt) - 1 / 2 / 14 / 28 threads

experiencing a decline, remains relatively reasonable, only 1.4 times below the superior architecture. This insight suggests that the VE2302 (C20B1) is poised to offer a more economical alternative, making the compromise in performance justifiable, especially when compared to its closest competitor, the UZ7EV.

Chapter 6

Conclusions

In conclusion, the thesis provides a comprehensive exploration of the Versal AI Edge series, focusing on the VE2302 and its integration with the VEK280. The Versal AI Edge series, with its adaptable architecture and advanced features, emerges as a leading solution for artificial intelligence (AI) inference at the edge. The VE2302, being a part of this series, exhibits remarkable capabilities with its optimized tiles, diverse interfaces, and integration possibilities.

The thesis covers various aspects, from the hardware architecture, memory configurations, and interface implementations to the versatile applications enabled by the VE2302. The implementation of the C20B1 DPU into the VEK280, simulating the VE2302's performance, underscores the adaptability and cost-effectiveness of the Versal devices. Despite a reduction in throughput, the trade-off in performance is justified by the more budget-friendly nature of the VE2302, making it a compelling choice for diverse applications.

The detailed explanation of hardware interfaces, MIO banks, and connectors contributes to a thorough understanding of the VE2302's capabilities. The integration with Vitis AI, ML data types support, and extended native support further highlight the platform's prowess in machine learning applications.

The evaluation of performance, power consumption, and throughput over power for various models on different Versal devices, notably the VE2302, provides valuable insights. Despite being a more compact and cost-effective solution compared to the VEK280, the VE2302 showcases commendable performance and demonstrates competitive efficiency in terms of fps over watt. This underscores the versatility and cost-effectiveness of the VE2302, making it an attractive option for a broad range of applications.

In summary, the thesis not only delves into the technical aspects of the Versal AI Edge series but also emphasizes the practical implications and real-world applications. The VE2302 stands out as a versatile and cost-effective solution, making it a strong contender in the landscape of edge AI devices.

In the upcoming months, the plan is to collaborate with Mario Bergeron on the development of a real-time application that effectively demonstrates the capabilities of the forthcoming VE2302 device. The focus will be on integrating the device with real-time input from six cameras. While the specific target market is still under discussion, considerations are leaning towards applications in the automotive and self-driving sectors.

The team is currently engaged in a race against time, striving to deliver a functional prototype by April. The goal is to showcase the prototype at the Embedded World event, a global exhibition held in Nuremberg from April 9 to 11, 2024. This event attracts participants and potential customers from around the world. The team is dedicated to presenting a compelling and innovative solution to capture the attention of a diverse and global audience, aiming to stand out among the offerings of global competitors at the exhibition. The anticipation is high, and the team is committed to making a significant impact in the industry.

The Embedded World Exhibition and Conference [26] serves as a global platform and vital meeting point for the entire embedded community, bringing together leading experts, key players, and industry associations. Its distinctive focus on technologies, processes, and cutting-edge products, coupled with unparalleled expert knowledge, sets it apart on the international stage, making it a must-attend event for the industry. In 2024, the Embedded World Exhibition and Conference will continue to act as a crucial trend barometer, showcasing significant topics directly from the embedded systems industry through its extensive supporting program.

A transversal project that Mario is embarking on involves the integration of the Hailo-8 AI Accelerator [27] into our boards. His initial experiments are currently underway using the ZUBoard [13], the same board employed for our benchmarks. In these experiments, Mario is exploring the integration of the module at the end of the pipeline, following Video Capturing and Image Processing, both carried out in the programmable logic (PL). The preliminary results are already showing incredible promise.

Finally, the thesis successfully demonstrates the remarkable results achievable with the smallest architecture of the new DPU, as integrated into the forthcoming VE2302 board. The limited decrease in performance, coupled with the cost-effectiveness of the VE2302, highlights its potential to cater to a broader market. Notably, with only three physical VEK280 boards in existence, our thesis contributes valuable insights that extend beyond the limited availability of the current hardware. The envisioned accessibility of the VE2302 is poised to target multiple real-time applications, further emphasizing its significance in the evolving landscape of edge AI devices.

Bibliography

[1] Sudeep Srivastava. *AI in Self-Driving Cars – How Autonomous Vehicles are Changing the Industry.* Dec. 2023. URL: https://appinventiv.com/blog/ai-in-self-driving-cars/ (cit. on p. 1).

[2] Advanced Micro Devices, Inc. *Versal AI Edge Series.* 2023. URL: https://www.xilinx.com/products/silicon-devices/acap/versal-ai-edge.html (cit. on pp. 2, 15).

[3] Xilinx AMD. *Vitis AI.* Official Vitis AI Website and Documentation: https://xilinx.github.io/Vitis-AI/3.5/html/index.html. Advanced Micro Devices, Inc. 2024. URL: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html (cit. on pp. 4, 54).

[4] Xilinx. *AI Engine Tools and Flows User Guide.* © 2024 Advanced Micro Devices, Inc. Dec. 2023. URL: https://docs.xilinx.com/r/en-US/ug1076-ai-engine-environment (cit. on p. 9).

[5] Advanced Micro Devices, Inc. *AMD AI Engine Technology.* 2023. URL: https://www.xilinx.com/products/technology/ai-engine.html (cit. on p. 10).

[6] *AMD Versal™ AI Edge Series VEK280 Evaluation Kit.* © 2024 Advanced Micro Devices, Inc. URL: https://www.xilinx.com/products/boards-and-kits/vek280.html (cit. on pp. 16, 55).

[7] Mindaugas. *Creating Xilinx Vivado Board Files for eBAZ4205.* https://www.hackster.io/mindaugas2/creating-xilinx-vivado-board-files-for-ebaz4205-a7b120, © MIT. Apr. 2021 (cit. on p. 37).

[8] Xilinx. *Vivado Design Suite User Guide: System-Level Design Entry.* © 2024 Advanced Micro Devices, Inc. Oct. 2023. URL: https://docs.xilinx.com/r/en-US/ug895-vivado-system-level-design-entry/Defining-Board-Files (cit. on pp. 37, 51).

[9] Advanced Micro Devices, Inc. *Vivado Design Suite Properties Reference Guide.* © 2024 Advanced Micro Devices, Inc. Advanced Micro Devices, Inc. Nov. 2024. URL: https://docs.xilinx.com/r/en-US/ug912-vivado-properties/Vivado-Design-Suite-First-Class-Objects (cit. on p. 46).

[10] Xilinx. *UltraScale Architecture SelectIO Resources User Guide.* © 2024 Advanced Micro Devices, Inc. Aug. 2023. URL: `https://docs.xilinx.com/r/en-US/ug571-ultrascale-selectio/Supported-I/O-Standards-and-Terminations` (cit. on p. 50).

[11] Bryan Fletcher, Tom Curran, Daniel Rozwood, and Tnizan (Witekio). *Avnet bdf.* 2023. URL: `https://github.com/Avnet/bdf/tree/ve2302_iocc-dev` (cit. on p. 53).

[12] M. Bergeron. *MARIO BERGERON.* Copyright © AlbertaBeef 2022. URL: `https://albertabeef.github.io/` (cit. on p. 53).

[13] *ZUBoard 1CG.* Lowest cost ZU+ MPSoC Dev Board, © Copyright 2024 Avnet, Inc. All rights reserved. URL: `https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/zuboard-1cg` (cit. on pp. 55, 71).

[14] *Ultra96-V2.* Arm-based, AMD Xilinx Zynq UltraScale+ MPSoC, © Copyright 2024 Avnet, Inc. All rights reserved. URL: `https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/ultra96-v2` (cit. on p. 55).

[15] *UltraZed-EV.* Proven SOM based on the UltraScale+ MPSoC, © Copyright 2024 Avnet, Inc. All rights reserved. URL: `https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/ultrazed/ultrazed-ev` (cit. on p. 55).

[16] *Zynq™ UltraScale+™ MPSoC.* © 2024 Advanced Micro Devices, Inc. URL: `https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html` (cit. on p. 55).

[17] Xilinx AMD. *GitHub - Vitis AI: Adaptable & Real-Time AI Inference Acceleration.* Vitis AI License: Apache 2.0. Advanced Micro Devices, Inc. 2023 (cit. on p. 55).

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. «Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning». In: *ArXiv* (2016). URL: `https://arxiv.org/abs/1602.07261` (cit. on p. 56).

[19] Karen Simonyan and Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». In: *ArXiv* (2014). URL: `https://arxiv.org/abs/1409.1556` (cit. on p. 56).

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *ArXiv* (2015). URL: `https://arxiv.org/abs/1512.03385` (cit. on p. 57).

[21] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications». In: *ArXiv* (2017). URL: `https://arxiv.org/abs/1704.04861` (cit. on p. 58).

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. «MobileNetV2: Inverted Residuals and Linear Bottlenecks». In: *ArXiv* (2018). URL: `https://arxiv.org/abs/1801.04381` (cit. on p. 58).

[23] B. Sanjana. *SSD MobileNetV1 architecture*. URL: `https://iq.opengenus.org/ssd-mobilenet-v1-architecture/` (cit. on p. 58).

[24] Chen Cheng. «Real-Time Mask Detection Based on SSD-MobileNetV2». In: *ArXiv* (2022). URL: `https://arxiv.org/abs/2208.13333` (cit. on p. 58).

[25] Xilinx. *DPUCV2DX8G for Versal Adaptive SoCs Product Guide*. © 2024 Advanced Micro Devices, Inc. July 2023. URL: `https://docs.xilinx.com/r/en-US/pg425-dpucv2dx8g` (cit. on p. 59).

[26] *Embedded World*. Copyright © 2024 NürnbergMesse GmbH. Apr. 2024. URL: `https://www.embedded-world.de/en` (cit. on p. 71).

[27] Hailo AI. *Hailo TAPPAS*. © All Rights Reserved 2024 HAILO `https://hailo.ai/`. 2024. URL: `https://github.com/hailo-ai/tappas` (cit. on p. 71).