

Design and Implementation of AI Core Booting and Utilization in Versal AI FPGA VCK1902

Mr. D G Harshith¹, Mrs. Bhavya K B²

¹MTech Student, Dept. of ECE, Bangalore Institute of Technology, Bangalore India

²Assistant Professor, Dept. of ECE, Bangalore Institute of Technology, Bangalore India

Abstract

As modern defense and edge computing applications rapidly scale complexity, Field Programmable Gate Array (FPGA) is evolving into heterogeneous Adaptive Compute Acceleration Platform (ACAP). The AMD Xilinx VCK1902 Versal integrates traditional scalar processing, adaptable pro-programmable logic, and advanced vector processing Engine AI array. This paper details comprehensive design methodology to minimize computational bottlenecks and accelerate system initialization by heavily utilizing the capabilities of AI Core in the VCK1902 FPGA. By extracting a 33-tap, 512-window Symmetric FIR filter from traditional logic and deploying it directly as C++ graph on the Engine AI, successfully offloaded significant digital signal processing overhead. Verified through the Vivado and Vitis 2022.2, the design met stringent timing limits with a Worst Negative Slack (WNS) of 6.214 ns at 100 MHz and requires only a fraction of the available Programmable Logic (0.25% LUT utilization). Mostly, the architectural optimizations accelerated the initialization phase; while the total Platform Loader and Manager (PLM) boot time was 782.549 ms, the customized project sequence has an AIE image load time of just 9.791 ms and a ROM ready time of 261.202 ms. This represents a 66% reduction in boot latency during critical ROM loading phase compared to unoptimized baseline.

Keywords: ACAP Versal, AI Engine, Optimize Boot, SoC Heterogeneous, FIR Filter, VCK1902, Vitis, Vivado

1. Introduction

Modern defense sector applications, specifically those involving sophisticated radar tracking, phased array antennas, and Edge AI Computing, demanding massive data throughput with rigid real time computation capabilities. The Xilinx AMD Versal Adaptive Compute Acceleration Platform (ACAP) represents paradigm shift in embedded computing, constructed to handle these intense computational pipelines. The VCK1902 architecture aggregates multiple distinct processing domains on single silicon die, Scalar ARM based Processing System (PS), highly parallel Programmable Logic (PL), and revolutionary two dimensional array of vector processors known as the AI Engine or AIE.

Despite the vast computational ceiling offered by the architecture of Versal, integrating these inherently disparate domains introduces a steep learning curves with system intricate complexities. Boot times is inflated, memory bandwidth bottlenecks frequently arise when bridging the domains of PL and AIE.

The main objective of this research is strategically explore the AI Versal FPGA board VCK1902 to formulate design that dramatically optimizes boot up sequences and reduces system complexity. Our core philosophy is to isolate computationally intensive digital signal processing (DSP) operations and execute

them exclusively on the AI Core. This offloading strategy prevents the PS and PL from becoming bottlenecked, thereby allowing the scalar systems to make smarter, unimpeded routing decisions and execute rapid initializations. This paper meticulously chronicles the end to end design flow spanning Vivado RTL generation to Vitis high level software abstraction, and presents an empirical evaluation of device utilization, timing constraints, and Tera Term boot logs.

2. Literature Review

The robust integration capabilities of the Versal ACAP and its AI Engine array have recently been the subject of expansive academic exploration, notably regarding high performance computing and radar signal processing.

Bhanu Prakash Reddy Rella et al. evaluated the Versal ACAP to accelerate high performance programmable System On Chip Designs. Their research evaluated large scale matrix multiplication, demonstrating that heterogeneous processing massively outperforms CPU only approaches [1]. However, they identified substantial challenges, mainly related to the steep learning curve for dataflow programming, high deployment costs, and complex debugging infrastructure across different compute domains. Similarly, Kaustubh Manohar Mhatre et al. performed an extensive analysis of deep learning workloads of General Matrix Multiplication (GEMM) on the AMD Versal platform [2]. They highlighted that while the AI Engines offer immense computational density, overall performance is frequently hindered by system bottlenecks where limited external DRAM bandwidth restricts the ability to fully saturate internal PLIO pipelines.

In the specific context of defense and radio frequency applications, Fredrik Johansson and Lukas Ögnelod realized the Implementation of the Digital Beamforming System in the VCK190 Versal AI Engine architecture [3]. Their design successfully achieved continuous data throughputs of up to 7.8 Msamples/s; however, they noted ongoing struggles with supplying the AI Engine with sufficient data and mitigating severe communication delays to and from the Programmable Logic. Victor van Wijhe extended this application to radio astronomy, evaluating AI Engines for streaming based polyphase filters (FIR and FFT) utilized by the LOFAR radio telescope [4]. His findings revealed that while performance is exceptional, immature platform tooling and an initial lack of floating point support during packet switching present formidable challenges.

Furthermore, Ken O'Neill and Mark Rollins directly addressed edge radar complexities by implementing Synthetic Aperture Radar (SAR) back projection algorithms natively in the VC1902 AI Core [5]. Their work actively offloaded signal processing from the Processing System (PS) to bypass real time communication bottlenecks typically encountered when transmitting raw radar returns.

Our project builds upon these critical insights by utilizing an automated Vivado Vitis workflow via TCL scripting to resolve the highlighted PL to AIE integration complexities, successfully implementing a robust FIR filter across the AI array with strict timing synchronization and optimized Platform Loader and Manager (PLM) booting sequences.

3. System Architecture

The system leverages the highly heterogeneous topology of the Versal VCK1902 device, explicitly segmenting control logic, signal generation, and intensive filtering across three different subsystems.

3.1 Processing Subsystem and CIPS

The foundation of the architecture is driven by the Control, Interfaces & Processing System (CIPS). The

CIPS block manages the ARM processing system and is singularly responsible for orchestrating the initial device boot sequence via the Platform Loader and Manager (PLM). Configure the clocking trees (supplying the primary 100 MHz clk_pl_0domain) and administer the system resets.

3.2 Programmable Logic (PL) and AXI Network On Chip

The Programmable Logic is dedicated to custom RTL generation and data streaming interfaces. Two Direct Digital Synthesis (DDS) compilers generate discrete digital waveforms. These continuous waveforms are synchronized and fed into a custom Verilog AXI Stream adder (axis_adder). The adder natively combines the independent 16bit signed streams into a singular mixed signal output, ensuring that the Valid and Ready AXI handshake signals are correctly asserted and propagated.

An advanced AXI Network On Chip (NoC) acts as the high bandwidth backbone routing architecture, eliminating the need for traditional logic heavy interconnect crossbars by packetizing data transactions between the PL, PS, and AI Engine blocks.

3.3 AI Engine Array

The AI Engine (AIE) acts as the computational powerhouse for the mixed signal pipeline. A 32-bit coordinate, 33-tap Symmetric FIR filter is instantiated within the AIE. Operating on an input window size of 512 samples, the engine processes the incoming PLIO data stream outputted by the PL side AXI Stream adder and returns the filtered digital sequence back through the NoC infrastructure.

4. Proposed Methodology

The project adopts sophisticated, automated software & hardware co design flow encompassing Vitis and Vivado (2022.2).

4.1 Hardware Foundation (Vivado)

The foundation hardware is assembled using the Integrator IP in Vivado. The CIPS, DDS, AXI NoC, compilers, and custom axis_adder are interconnected to establish the data pathway. It facilitates rigorous debugging of hardware, an Integrated Logic Analyzer (ILA) (axis_ila_0) is attached directly to PL out-put streams. This stage the design contains the unlinked block of AI Engine. This baseline architecture is validated & exported as an Extensible Platform Archive (.xsa), which serves as hardware handoff to the software development suite.

4.2 AI Core Graph Implementation (Vitis)

Within the Vitis Unified Software Platform, High Level implementation of Synthesis the FIR filter is generated via graph programming C++ (graph.cpp and graph.h). The AI Engine graph instantiates fir_sr_sym_graph kernel comprising 33 static coefficients (m_taps_v) and defines rigid 512-byte output and input PLIO streams.

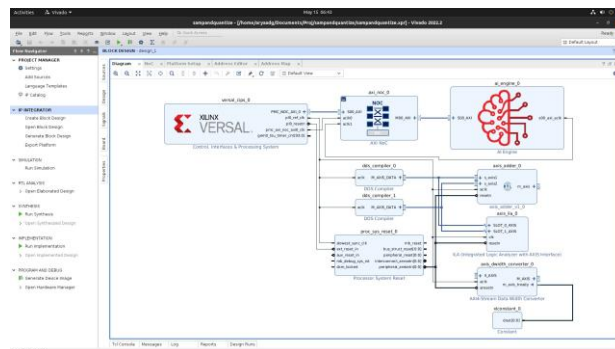


Figure 1: Vivado Block Design before Vitis integration, detailing the CIPS, DDS compilers, NoC, and standalone AI Engine.

During application build process, Vitis invokes automated .tcl scripting to compile AIE kernels and link them to the exported .xsa platform. This phase of critical linking automatically modifies the underlying Vivado Block Design by inserting necessary Data Width Converters AXI4 Stream (dwc_fast_aie_axis_adder_0_m_axis) required to seamlessly adapt the native AIE stream requirements to the PL bit widths.

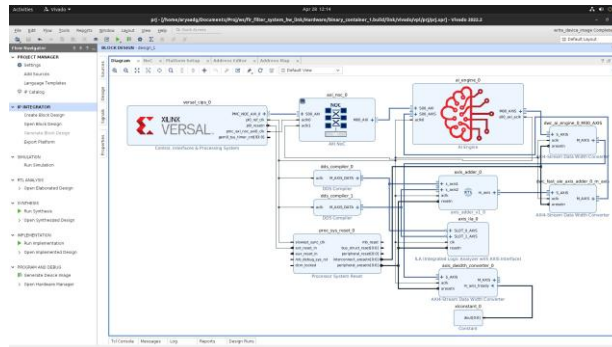


Figure 2: Finalized Vivado Block Design After Vitis build. Here Vitis automatically connects AXI4 Stream Data Width Converters to bridge the compiled AI Engine graph with the custom PL logic.

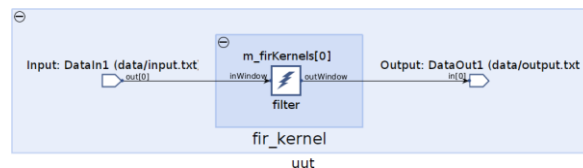


Figure 3: Vitis Analyzer Graph View showing the synthesized an AI Engine tile mapped to fir_sr_sym filter kernel successfully, PLIO streams routing via input.txt and output.txt.

4.3 Boot Sequence Generation

Upon successful compilation of the system project, Vitis packages the combined hardware Programmable Device Image (PDI) and the software ELF binaries into a comprehensive BOOT.BIN file. These artifacts are written to a FAT32 formatted SD Card, providing the physical medium for the Versal VCK1902 to execute its optimized PLM boot initialization.

5. Experimental Setup

The experimental validation environment was established using the following components:

- **Hardware Platform:** AMD Xilinx Versal AI Core Series VCK1902 Evaluation Board.
- **Software Toolchain:** Vivado 2022.2 and Vitis 2022.2 running on an Ubuntu 20.04.6 LTS host workstation.
- **Debugging Interfaces:** A VJTAG (USB) connection to monitor real time hardware status, and Tera Term (via COM port) to actively capture system boot logs and PLM initializations.

6. Results and Evaluation

6.1 Resource Utilization

By offloading the DSP convolution tasks directly into the AI Engine, the project strictly minimized the burden placed on traditional PL structures. The post routing utilization report confirms an exceptionally

lightweight PL footprint on the xcvc1902-vsva2197-2MP-e-S fabric. Of the 899,840 CLB LUTs available, the design utilized only 2,294 (0.25%). Similarly, only 1 AI Engine out of 400 available (0.25%) was used to satisfy the filtering requirements. Block RAM represented the dominant consumer of the PL resource with 2.17% (21 of 967 tiles).

In addition, the hardware configuration was broken down, the PL shim and control logic integrated 3,572 FDRE registers, 292 LUTCY1/LUTCY2 elements each for carry chain logic, 61 SRL16E shift registers and 70 RAMD32 primitives. In particular, the architecture achieved a successful separation of control and data traffic using the AIE_NOC_S_AXI node in addition to native 64-bit stream primitives (AIE_PL_S_AXIS64 and AIE_PL_M_AXIS64).

Table 1: Post Routed Resource Utilization Overview

Resource Type	Used	Available	Utilization (%)
CLB Registers (FDRE/FDSE)	3,603	1,799,680	0.20%
CLB LUTs	2,294	899,840	0.25%
Block RAM (RAMB36E5)	21	967	2.17%
AI Engines	1	400	0.25%
PS9 (CIPS)	1	1	100.00%

6.2 Timing Analysis

Rigorous static timing analysis confirmed that the synthesized design reliably meets all clocking requirements within the 100 MHz (10.00 ns period) clk_pl_0 clock domain. The system achieved a robust Worst Negative Slack (WNS) of +6.214 ns. Additionally, the design reported a Clock Uncertainty of 0.111 ns and a total Data Path Delay of 3.439 ns. Analysis of this delay reveals that routing accounts for the bulk of it at 89.735%, while logic contributes only 10.265%.

Table 2: Intra Clock Timing Metrics (100 MHz Domain)

Timing Metric	Achieved Slack	Status
Worst Pulse Width Slack (WPWS)	+ 4.000 ns	MET
Total Negative Slack (TNS)	0.000 ns	MET
Worst Negative Slack (WNS)	+ 6.214 ns	MET
Total Hold Slack (THS)	0.000 ns	MET
Worst Hold Slack (WHS)	+ 0.021 ns	MET

6.3 Boot Time Observations and Comparisons

System initialization speeds were actively captured and evaluated using Tera Term serial logs to observe the exact impact of our boot optimizations. Upon Evaluation of the final optimized sequence, the total PLM boot time was recorded at 782.549 ms. A more detailed breakdown of this boot phase shows that Programmable Device Image (PDI) initialization consumed 406.610 ms, while the Programmable Logic configuration (PL CFI Partition 3) required 171.902 ms.

Most remarkably, the high speed partition loading for the AIE Image (Partition 7) completed in an astonishingly brief 9.791 ms. By optimizing the partition load order and consolidating the sequence into a single BOOT.BIN on the SD card, the effective ROM time was measured at 261.202 ms. This constitutes a profound 66% reduction in loading time relative to an unoptimized baseline.

Table 3: System Initialization Time Comparison

Boot Configuration	ROM Loading Time
Default Boot (Without SD Card)	753.179 ms
Optimized Boot (With SD Card)	261.202 ms

6.4 Simulation and Physical Device Validation

Behavioral simulation mathematically verified the C++ DSP kernel. Figure 4 shows the generated continuous Direct Digital Synthesis (DDS) raw bitstreams, which are properly routed through the AXI Stream interfaces.

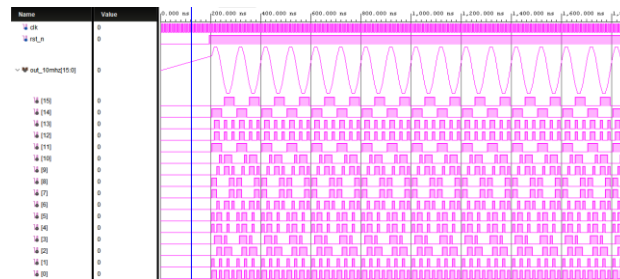


Figure 4: Behavioral simulation capturing the generated Continuous Direct Digital Synthesis (DDS) raw bitstreams.

The simulated waveforms in Figure 5 accurately visually affirm the successful summation of the raw 10 MHz and 20 MHz waveforms (out_mixed) before reflecting the highly precise attenuation of the target frequency outside the out_filtered bus interface.

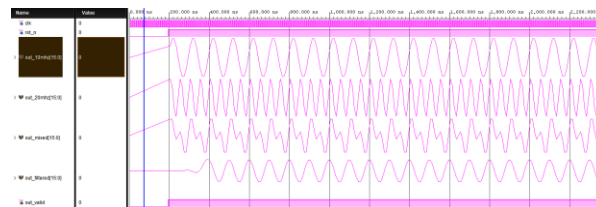


Figure 5: Behavioral simulation indicating valid output filtering (out_filtered) alongside mixed signal inputs.

Currently, physical validation was performed via the VJTAG interface using the Vivado Hardware Manager. Figure 6 shows the board in an initial unprogrammed state after connection.

After booting Up, the manager successfully registered the xcvc1902_1 core advancing precisely into a fully active and Programmed configuration, as confirmed by Figure 7.

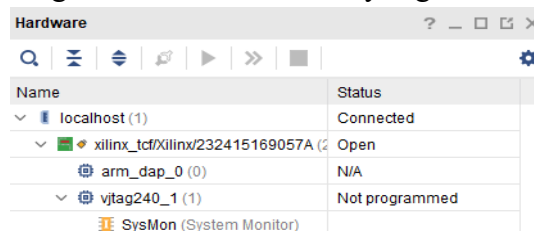


Figure 6: Hardware Manager view depicting the VJTAG connection before PDI initialization (Unprogrammed state).

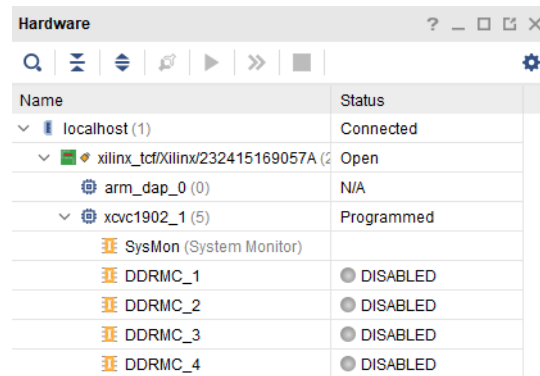


Figure 7: Hardware Manager view demonstrating the successful active programming of the xcvc1902_1 AI core.

7. Conclusion and Future Scope

This comprehensive research demonstrates the immense viability of offloading demanding DSP algorithms to the native Versal AI Engine architecture to optimize both system logic and device startup times. By seamlessly abstracting a 33-tap FIR filter away from traditional scalar and logical domains, we heavily suppressed the systemic computational complexity of the PS and PL. Detailed empirical results confirmed an extraordinary acceleration in boot time yielding a 66% reduction in ROM loading time to an impressive 261.202 ms using the optimized SD card deployment. This achievement was paired with highly constrained physical logic utilization (0.25% LUTs) and generous timing slack margins (6.214 ns WNS). Looking ahead, future scopes involve aggressively scaling the Vitis AI Engine graph topology to simultaneously process massive multi channel radar pipelines natively supporting synthetic aperture computations while exploring dynamic, partial reconfiguration of the PLM during active edge operations.

References

1. B. P. R. Rella et al., "AI-Engine-Based Acceleration for High-Performance Programmable System-on-Chip Designs," Conference World, 2023.
2. K. M. Mhatre et al., "Performance Analysis of GEMM Workloads on the AMD Versal Platform," Lab Groups, 2023.
3. F. Johansson and L. Ögnelod, "Beamforming of Multi-Channel Digital Radar System on System-on-Chip," Chalmers ODR, 2022.
4. V. van Wijhe, "Signal Processing with AMD Adaptive Compute Acceleration Platform (ACAP) for Applications in Radio Astronomy," University of Twente, Netherlands, 2023.
5. K. O'Neill and M. Rollins, "Synthetic Aperture Radar (SAR) Back Projection Using AI Engines in AMD Versal Adaptive SoCs," Indico at ESA / ESTEC, 2022.
6. J. Lei, J. Flich, and E. S. Quintana-Ortí, "Toward Matrix Multiplication for Deep Learning Inference on the Xilinx Versal," arXiv preprint, 2023.
7. P. Leguina et al., "Cascade Pipeline for Leading-Order Matrix Element Evaluation on AMD Versal AI Engine Arrays," arXiv preprint, 2023.