VSRVES01 and VSRVES02 Demonstrators

Document History

Revision	Date	Author/Org.	Description
V1.0	2026-04-10	TK	Original release
V1.1	2026-04-17	TK	Minor changes after internal review
V1.2	2026-05-21	TK	Status added and updated

TRISTAN has received funding from the Key Digital Technologies Joint Undertaking (KDT JU) under grant agreement nr. 101095947. The KDT JU receives support from the European Union's Horizon Europe's research and innovation programme and Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey

Table of Contents

Figures

Tables

1 Introduction

1.1 Background

VLSI Solution Oy has over 30 years of experience in developing audio integrated circuits based on proprietary DSP architectures. These systems are highly optimized for real-time signal processing, where deterministic behavior and efficiency are critical.

In modern applications, there is an increasing need to integrate connectivity features such as Ethernet and to support complex software stacks including networking protocols (e.g., TELNET, HTTP, DHCP) and operating systems. These requirements shift part of the system functionality from dedicated hardware and DSP-based processing toward general-purpose software execution.

A key architectural principle in the VSRVES01 and VSRVES02 demonstrators is the separation of roles between processing units:

- The **RISC-V processor acts as the system master**, running a standard Linux operating system and handling networking, control, and high-level software tasks.
- The **VSDSP6 processor remains the main computational workhorse**, executing real-time audio processing and other performance-critical algorithms.

This division allows each processor to operate in the domain where it is most efficient: Linux provides flexibility and ecosystem support, while the DSP ensures high-performance deterministic real-time processing.

Another important consideration is memory architecture. Running a full Linux operating system requires external high-capacity memory, typically LPDDR, which introduces additional cost compared to traditional microcontroller-based systems using small on-chip or external SRAM.

However, in modern use cases, particularly those involving advanced audio processing, machine learning, or AI-assisted algorithms, the memory requirements of the DSP system itself increase significantly. In such systems, external LPDDR memory is required regardless of the operating system.

By allowing both the RISC-V processor and the DSP to **share the same LPDDR memory**, the incremental cost of supporting Linux instead of a lightweight RTOS (such as FreeRTOS) becomes negligible. This enables the use of a full-featured Linux environment without a significant cost penalty, while still maintaining high-performance DSP-based processing.

This architectural approach provides a scalable and future-proof platform for combining real-time signal processing with modern software-driven features.

1.2 Purpose

The purpose of this document is to describe the architecture, functionality, and usage of the VSRVES01 and VSRVES02 demonstrator integrated circuits and their associated evaluation platforms.

The document provides:

- A system-level overview of the demonstrator concept
- A detailed description of the VSRVES01 and VSRVES02 integrated circuits
- Information about the demonstrator printed circuit boards (PCBs), including the VSRVES01 CAT Board
- Description of system operation, including boot flow and software environment

1.3 Scope

This document covers the following three components:

- **VSRVES01 Demonstrator IC** [1]
  - First-generation demonstrator based on the VSRV1 RISC-V core [2]
  - Integrated with VSDSP6 for audio processing and system control
  - Includes LPDDR2 memory multiplexing in boot but no sharing in full operation
- **VSRVES02 Demonstrator IC** [3]
  - Second-generation demonstrator based on the enhanced VSRV2 RISC-V core [4]
  - Includes LPDDR2 memory sharing of both processors in full operation
  - Includes architectural improvements such as increased memory bandwidth, DMA support, and additional ISA extensions
- **Demonstrator Evaluation Boards**
  - VSRVES01 CAT Board and related hardware platforms
  - Interfaces, peripherals, and typical system configurations

The document focuses on functional and architectural aspects of the demonstrators and their use as development and evaluation platforms. Detailed electrical specifications and production-level characterization are outside the scope of this document.

1.4 Acronyms and Definitions

ACRONYM	DESCRIPTION
ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
AXI	Advanced eXtensible Interface (AMBA bus protocol)
BGA	Ball Grid Array
CAT Board	Customer Application Test Board (also referred to as the VSRVES01 CAT Board)
CPU	Central Processing Unit
DAC	Digital-to-Analog Converter
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DSP	Digital Signal Processor
Ethernet	Wired networking interface
GPIO	General Purpose Input/Output
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
I/O	Input / Output
ISA	Instruction Set Architecture
LPDDR	Low Power Double Data Rate memory in general
LPDDR2	(Low Power Double Data Rate 2) is a high-speed, energy-efficient synchronous DRAM standard designed by JEDEC
MAC	Media Access Controller
MMU	Memory Management Unit
PCB	Printed Circuit Board
PHY	Physical Layer (Ethernet transceiver)
PLL	Phase-Locked Loop
QFN	Quad Flat No-leads package
RISC-V	Open standard Instruction Set Architecture
ROM	Read-Only Memory
RTC	Real-Time Clock
RTP	Real-time Transport Protocol
SD card	Secure Digital memory card
SoC	System-on-Chip
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCP/IP	Transmission Control Protocol / Internet Protocol
TELNET	Network protocol for remote terminal access
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VLC	VideoLAN Client

ACRONYM	DESCRIPTION
VRI	VLSI image format for Linux binaries
VSBUS	VLSI Solution internal peripheral bus
VSOS	VSDSP Operating System

TERM	DEFINITION
Dual-core architecture	System containing two processors (RISC-V and DSP) with distinct roles
DSP subsystem	Processing domain optimized for real-time signal processing tasks
RISC-V subsystem	Processing domain running Linux and handling control and networking
Shared memory	External LPDDR memory accessible by both processors (fully shared in VSRVES02)
Multiplexed memory	Memory access model where only one processor accesses LPDDR at a time (VSRVES01)
DMA-based transfer	Data movement between memory and peripherals without CPU intervention
Dual MAC architecture	Two independent Ethernet MACs connected to a single PHY (VSRVES02)
DSP-assisted boot	Boot process where DSP initializes system and starts RISC-V
Real-time processing	Deterministic, low-latency computation (handled by DSP)
Linux-capable core	Processor supporting full Linux OS including MMU and required ISA extensions
Edge processing	Data processing performed close to the data source
IoT	Internet of Things — connected embedded devices
Demonstrator IC	Prototype integrated circuit used for validation
Evaluation board	Hardware platform used for testing and development

2 System Overview

2.1 Dual-Core Concept

The VSRVES demonstrator systems are based on a heterogeneous dual-core architecture combining a RISC-V processor and a DSP digital signal processor. The two processors have clearly defined and complementary roles, enabling an efficient balance between software flexibility and real-time performance.

2.1.1 Role of the RISC-V Processor

The RISC-V processor functions as the system master and runs a standard Linux operating system. Its primary responsibilities include:

- Execution of networking protocols (e.g., TCP/IP stack, HTTP server)
- System-level control and configuration
- System-level file system management and user applications
- High-level software integration and updates

By using an unmodified upstream Linux kernel, the system benefits from a mature software ecosystem, long-term maintainability, and access to existing tools and libraries. In addition, Linux provides a continuously maintained security framework, enabling the system to receive upstream security updates and reducing the long-term maintenance burden.

2.1.2 Role of the DSP Processor

The DSP processor acts as the main computational workhorse of the system. It is optimized for deterministic, real-time signal processing tasks, including:

- Audio decoding and encoding
- Signal processing of sensor data
- Time-critical control functions

The DSP runs a lightweight real-time operating system (VSOS) and is designed to handle workloads where low latency and predictable execution are essential. This makes it particularly suitable for modern embedded applications such as IoT and edge processing, where real-time data handling and efficient signal processing are required close to the data source.

2.1.3 Division of Responsibilities

The dual-core architecture enables a clear separation between:

- **Software-intensive tasks,** handled by the RISC-V processor under Linux
- **Real-time and performance-critical tasks**, handled by the DSP

This separation improves system efficiency by allowing each processor to operate in its optimal domain, avoiding the compromises that would arise from using a single processor for both roles.

2.1.4 Shared Memory Architecture

Both processors utilize external LPDDR2 memory as the main system memory; however, the memory access architecture differs between the VSRVES01 and VSRVES02 demonstrators.

VSRVES01 Memory Access

In VSRVES01, the LPDDR2 memory is multiplexed between the DSP and the RISC-V processor during system operation:

- During system initialization, the DSP has control of the memory interface and is responsible for loading the Linux image into LPDDR2 memory
- After initialization, control of the memory is effectively handed over to the RISC-V processor running Linux
- The DSP and RISC-V processor do not access the LPDDR2 concurrently in normal operation

This approach simplifies the memory architecture but limits simultaneous data exchange between the processors.

VSRVES02 Memory Access

In VSRVES02, the memory architecture is enhanced to support true shared access to LPDDR2 memory:

- Multiple Direct Memory Access (DMA) channels are introduced on the memory interface
- The DSP, SD card interface, and memory-to-memory operations can access LPDDR2 via DMA through the shared AXI interconnect, independently of direct CPU intervention
- The RISC-V processor and DSP can operate concurrently, with data transfers handled efficiently via DMA, and both processors have access to the same LPDDR2 address space without hardware-based partitioning

This enables:

- Parallel operation of Linux and DSP workloads
- Efficient data movement without CPU intervention
- Improved system performance and scalability

System Architectural Implications

The evolution from multiplexed to shared memory access provides significant system-level benefits:

- Reduced latency in data exchange between processors
- Lower CPU overhead due to DMA-based transfers
- Better support for data-intensive workloads such as streaming, audio processing, and edge/AI applications

As discussed earlier, the use of LPDDR2 memory is primarily driven by DSP and system-level requirements. The ability to share this memory efficiently further strengthens the overall system architecture without introducing additional cost.

2.1.5 System Initialization

During system startup, the DSP processor plays an active role in system initialization:

- The DSP boots first from internal ROM
- The DSP's ROM code polls available boot interfaces
- When SPI-connected non-volatile memory (SPI NOR Flash) responds, the DSP will continue to boot from there
- The code in SPI-connected non-volatile memory loads the Linux image from the non-volatile storage (e.g., SD card or SPI NAND flash) into LPDDR2 memory
- The DSP releases the RISC-V processor from reset
- The RISC-V processor begins executing the Linux kernel from LPDDR2 memory

This boot approach leverages the deterministic behavior of the DSP and simplifies system bring-up without requiring a complex boot infrastructure in the RISC-V subsystem.

2.2 Demonstrator System Architecture

The VSRVES01 and VSRVES02 demonstrator ICs implement a complete system-on-chip (SoC) architecture integrating processing cores, memory interfaces, and peripheral subsystems. The generic architecture is designed to support Linux-based networking alongside real-time signal processing in a compact and efficient embedded platform. A simplified high-level block diagram is shown in Figure 1.

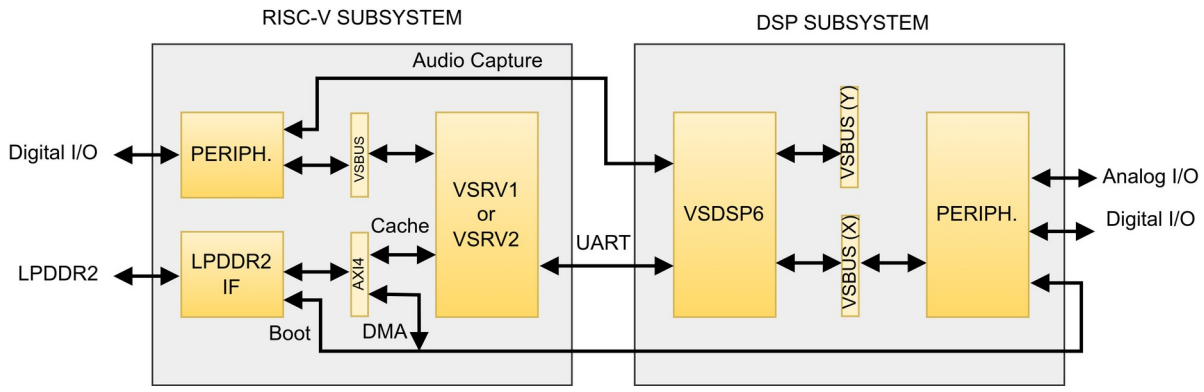

Figure 1: Demonstrator high level block diagram

2.2.1 Overview

At a high level, the system consists of the following main components:

- RISC-V processor subsystem (VSRV1 or VSRV2)
- DSP processor subsystem (VSDSP6)
- External LPDDR2 memory interface (LPDDR2)
- Peripheral interconnect buses (VSBUS)
- High-speed interface (AXI-based)

The architecture separates high-bandwidth memory access and peripheral communication to optimize both performance and latency.

2.2.2 Processing Subsystems

The system contains four primary subsystems:

RISC-V Subsystem

The RISC-V subsystem includes:

- CPU core (VSRV1 or VSRV2)
- Instruction and data caches
- Memory Management Unit (MMU)
- VSBUS for peripherals
- RISC-V peripherals
- AXI interface to external memory
- LPDDR2 interface to external memory
- Optionally: DMA channels, and boot ROM

This subsystem runs a standard Linux operating system and is responsible for system control, networking, and high-level software execution.

DSP Subsystem

The DSP subsystem is dedicated to real-time processing and includes:

- DSP core (VSDSP6)
- Harvard Architecture (Instruction bus, X- and Y-databus)
- Local memory and peripherals
- Interfaces to external memory and peripherals
- Interface to analog subsystem
- Optionally: DMA interface to AXI4 bus of RISC-V subsystem

It runs a real-time operating system and handles audio processing and other time-critical tasks.

RTC Subsystem

- Keeps real-time clock
- Can generate alarms
- Has registers to store status information
- Has battery backup

Analog subsystem

- Has all analog interfaces and PLLs

2.2.3 External Memory Architecture

The system uses external LPDDR2 memory as the main memory for both processors.

- The RISC-V subsystem accesses LPDDR2 through an AXI-based memory interface
- In boot, when RISC-V is in reset state, the DSP subsystem can write directly to LPDDR2 memory through a dedicated 32-bit boot interface

As described in Section 2.1.4, the memory access model differs between demonstrators:

- **VSRVES01**: multiplexed access between DSP and RISC-V
- **VSRVES02**: shared access enabled via AXI interconnect with multiple masters (RISC-V caches and DMA channels)

This architecture allows efficient handling of large data sets required by Linux, audio processing, and emerging edge/AI workloads.

2.2.4 Peripheral Architecture

Peripherals are connected using a custom lightweight bus (VSBUS):

- Designed for low-latency, single-cycle access
- Compatible with existing VSDSP-based peripherals
- Used for standard system peripherals such as:
  - UART
  - Timer
  - Interrupt controller
  - SPI

The VSBUS enables simple integration and minimizes hardware overhead compared to more complex interconnects.

2.2.5 High-Speed Interfaces

High-bandwidth components are connected via the AXI-based memory system:

- External LPDDR2 memory
- SD card interface (optional)
- VSBUS of RISC-V (optional)
- VSBUS (Y-bus) of VSDSP (optional)

These interfaces provide block data transfers to accelerate LPDDR2 access. DMA bridges to SD card and VSBUS peripherals, where implemented.

2.2.6 Data Flow and System Operation

The architecture supports efficient data flow between subsystems:

- The RISC-V processor handles networking and top-level control
- The DSP processes data streams such as audio
- Low-latency peripherals: VSBUS provides simple and fast access to peripherals to meet real-time requirements. Data is exchanged via a high-speed UART connected to both subsystems. In VSRVES02, this serves as an optional data link between the subsystems.
- VSRVES02: Data is also exchanged through shared memory and DMA mechanism

This enables use cases such as:

- Audio streaming over Ethernet
- Simultaneous networking and signal processing

- Real-time data handling with Linux-based control
- Real-time IoT sensor interfacing with low-latency interrupts
- Real-time edge processing with low computational load, such as speech processing

2.2.7 Design Rationale

The system architecture is based on the following design principles:

- **Separation of concerns:** Linux and real-time processing are handled by different processors and clock domains
- **Low-latency peripherals:** VSBUS provides fast and efficient access to peripherals to meet real-time requirements.
- **Efficient memory usage:** shared LPDDR2 memory in VSRVES02 avoids duplication, makes data sharing and partitioning flexible, and reduces system cost
- **Scalability:** enhanced memory architecture and DMA support in VSRVES02 enable higher performance and more complex applications

This approach results in a flexible and efficient platform for embedded systems combining real-time processing with modern software capabilities.

3 VSRVES01 SoC Description

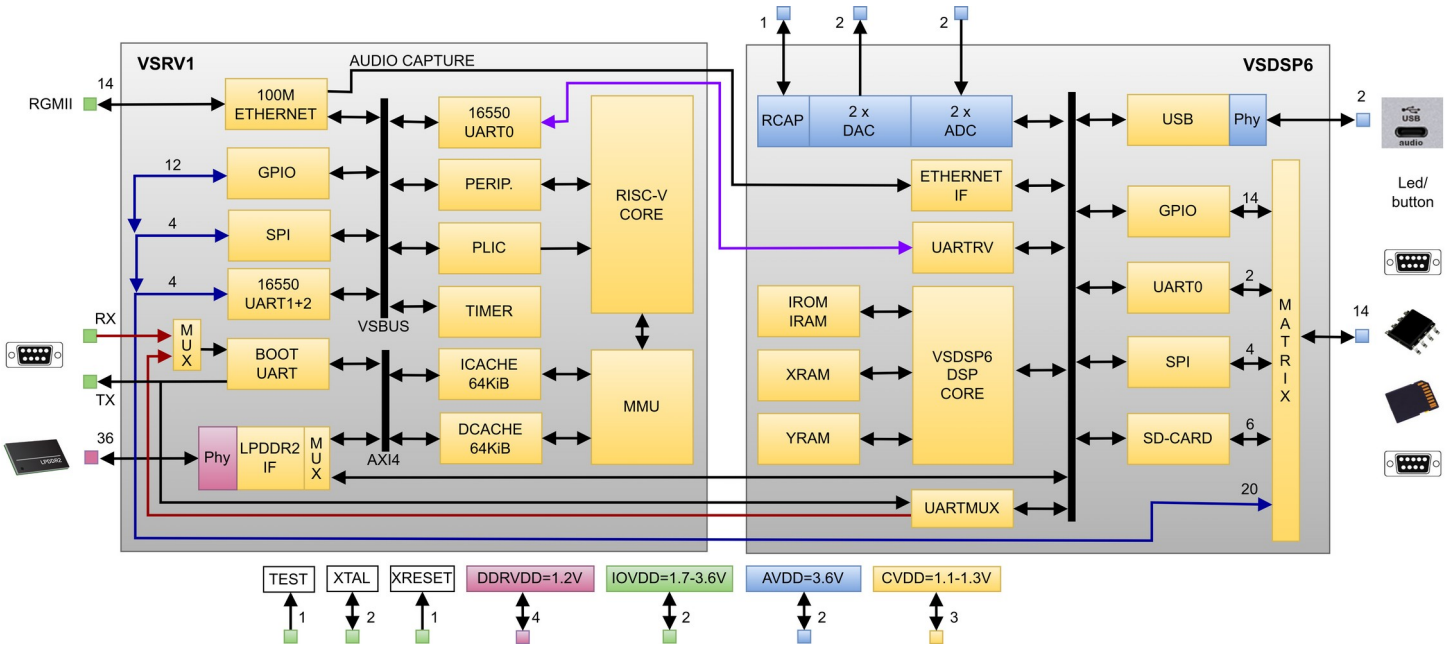

Figure 2: Block diagram of VSRVES01 SoC

3.1 Introduction

VSRVES01 is the first silicon demonstrator implementing the generic system architecture described in Chapter 2. The device validates the integration of a Linux-capable RISC-V processor with a real-time DSP subsystem in a single system-on-chip.

Rather than repeating the general architecture, this chapter focuses on the implementation-specific characteristics of VSRVES01 shown in Figure 2.

3.2 Implementation Overview

The VSRVES01 SoC integrates:

- One VSRV1 RISC-V processor subsystem running Linux
- One VSDSP6 DSP subsystem for real-time processing
- External LPDDR2 memory interface
- Ethernet, SD card, and peripheral interfaces

The implementation emphasizes:

- Low hardware complexity

- Fast development and validation
- Functional verification of Linux + DSP cooperation

Several architectural simplifications were made, particularly in memory access and subsystem interaction, which directly affect system behavior.

3.3 Building Blocks and Status

The main building blocks of the VSRVES01 demonstrator are shown in Table 1.

Building block	Description	Status
VSRV1 subsystem	32-bit Linux-capable RISC-V processor	finished
VSDSP6 subsystem	DSP for real-time processing	finished
LPDDR interface	Multiplexed external memory access	finished
Ethernet MAC	Single MAC with filtering and buffering	finished
UART IPC	Inter-processor communication	finished
Linux software stack	Networking, OS services	finished
DSP software (VSOS)	Real-time processing environment	finished
Demonstration software	Audio streaming + web server	finished
Evaluation hardware	VSRVES01 CAT board	finished

Table 1: VSRVES01 building blocks and their status

3.4 Memory Access Model

A defining characteristic of VSRVES01 is the **multiplexed LPDDR2 memory architecture**.

- During system initialization, the DSP has exclusive access to LPDDR2
- After boot, the RISC-V processor becomes the primary user of the memory.
- Memory ownership is multiplexed, and if LPDDR2 is allocated to the DSP, the RISC-V processor must be held in reset
- Concurrent access by both processors is not supported

Implications:

- Processor interaction is not possible through shared memory
- Inter-processor data exchange through shared memory is not possible
- System operation is effectively divided into phases

This approach simplifies the hardware implementation but restricts scalability for data-intensive and low-latency inter-processor communication.

3.5 Boot Dependency

In VSRVES01, the RISC-V processor does not operate independently at system startup.

- There is no boot ROM in the RISC-V subsystem
- All initialization is performed by the DSP

Consequently:

- The RISC-V subsystem is not autonomous at startup
- Boot behavior is deterministic and controlled by the DSP
- System flexibility is high, but independence of subsystems is limited

3.6 Network Interface Architecture

VSRVES01 implements a single Ethernet MAC shared between the RISC-V processor and the DSP subsystem.

To support both processing domains, the MAC includes:

- Multiple buffering mechanisms
- IP-based filtering

Operation

- Network traffic is received through a single physical interface
- Packets are routed based on filtering rules:
  - Linux-related traffic → RISC-V processor
  - Selected data streams → DSP subsystem

Limitations

- The MAC is a **shared resource**, introducing contention
- Data paths are not fully independent
- Data handling and configuration often involves the RISC-V processor

As a result, latency and throughput are limited compared to architectures with dedicated interfaces. This limitation is addressed in VSRVES02.

3.7 Inter-Processor Communication

VSRVES01 includes a dedicated high-speed UART interface between the RISC-V processor and the DSP subsystem.

Characteristics

- Direct point-to-point asynchronous communication between processors
- Low-latency control channel

Role in the System

The fast UART is used for:

- Control signaling between processors
- Data exchange
- System coordination during runtime

Limitations

- Limited bandwidth compared to memory-based communication
- Not suitable for high-throughput data transfer

This reflects a trade-off between simplicity and performance in the first demonstrator.

3.8 System-Level Behavior

The VSRVES01 demonstrator confirms that:

- A minimal RISC-V core can run a full Linux operating system
- Linux-based networking and DSP-based real-time processing integration is possible and has clear advantages in real-time systems
- Selective routing of audio packets from the Ethernet MAC to the DSP via filtering significantly reduces audio latency and eliminates the requirement to have fast shared memory for the audio streaming
- The system can perform simultaneous tasks such as:
  - Audio streaming over Ethernet
  - Web server operation

However, due to architectural constraints:

- Interaction between processors is limited to MAC data capture and UART
- Real-time data exchange is not fully optimized without shared memory
- LPDDR2 memory cannot be allocated to DSP after boot

3.9 Demonstrator Integration Steps

VSRVES01 demonstrator integration steps and their status are shown in Table 2.

Milestone	Description	Status	Date
M1	Integration of VSRV1 core and DSP subsystem (RTL)	finished	Sep 2025
M2	FPGA-based validation	finished	Oct 2025
M3	Silicon prototype (VSRVES01)	finished	Feb 2026
M4	Linux bring-up and networking stack integration	finished	May 2026
M5	Audio streaming and web server demonstration	finished	May 2026
M6	Evaluation board (CAT board) availability	finished	May 2026

Table 2: Integration steps of VSRVES01 demonstrator

3.10 Measured Performance

The VSRVES01 system has been validated both on FPGA platforms and in silicon.

Key observed characteristics include:

- Linux boot time of approximately 2.3 seconds
- CoreMark/MHz of 1.72
- Stable operation at approximately 110 MHz clock frequency in 110 nm technology
- Reliable execution of combined networking and audio workloads

The system demonstrates stable long-term operation and confirms the feasibility of the proposed architecture.

The measured performance also reflects the limitations of the multiplexed memory architecture, particularly in scenarios requiring continuous interaction between processing subsystems.

3.11 Conclusions

The VSRVES01 demonstrator highlights several important design trade-offs:

Advantages

- Simple and compact real-time heterogeneous architecture
- Low gate count vs. performance
- Fast development and validation cycle

- Successful Linux integration

Limitations

- No concurrent shared memory access
- Shared Ethernet interface is enough for audio but not flexible enough in general
- UART has limited bandwidth for communication between the processors
- RISC-V subsystem depends on DSP for boot

Lessons Learned

The development of VSRVES01 led to several key improvements for future designs:

- Need for RISC-V extensions to decrease the boot time
- Need for sharing the LPDDR2 for both subsystems
- Importance of DMA-based data movement for data intensive applications
- Independent boot capability of RISC-V subsystem for better autonomy

These findings directly guided the architectural enhancements implemented in the VSRVES02 demonstrator.

4 VSRVES02 SoC Description

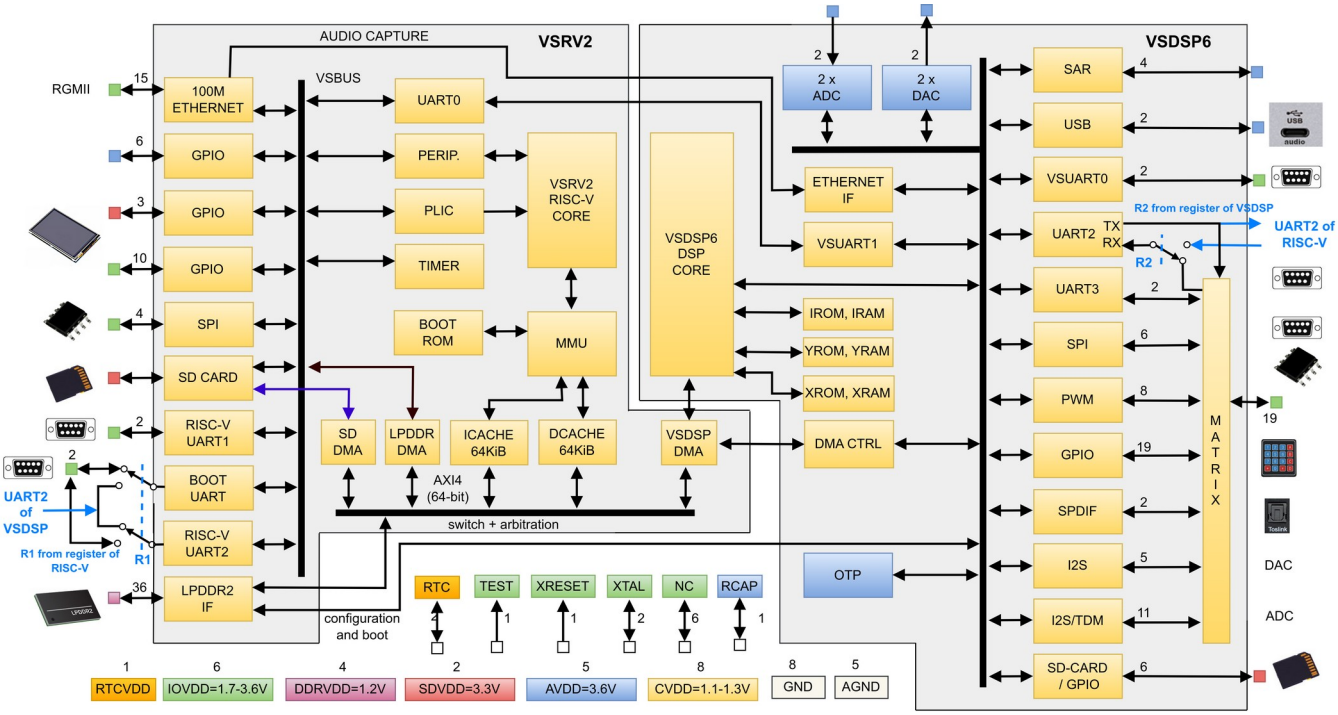

Figure 3: Block diagram of VSRVES02 SoC

4.1 Introduction

VSRVES02 is the second-generation demonstrator implementing the VSRV2 RISC-V core and an enhanced system architecture. It builds directly on the experience gained from VSRVES01 and addresses the key limitations identified in the first demonstrator. The block diagram is shown in Figure 3.

The primary goal of VSRVES02 is to enable:

- Faster Linux boot
- Concurrent operation of processing subsystems
- Efficient data movement between system components
- Allocation of part of the LPDDR2 memory to the DSP
- More flexible Ethernet interface

This is achieved through architectural enhancements in memory access, interconnect, and network interface design.

4.2 Key Architectural Improvements

Compared to VSRVES01, VSRVES02 introduces the following major improvements:

- Enhanced RISC-V core (VSRV2) with additional ISA extensions
- Wider memory interface (64-bit)
- One SD card interface dedicated to RISC-V subsystem, with DMA to LPDDR2
- Concurrent access to shared LPDDR2 memory by both processors via AXI and DMA
- Dual Ethernet MAC with shared PHY architecture

These changes transform the system from a sequential, loosely coupled architecture into a fully concurrent multi-master architecture with shared memory.

4.3 Building Blocks and Status

The main building blocks of the VSRVES02 demonstrator are shown in Table 3.

Building block	Description	Status
VSRV2 subsystem	Enhanced RISC-V processor	complete
VSDSP6 subsystem	DSP for real-time processing	complete
AXI interconnect	Multi-master shared memory system	complete
DMA subsystem	Multi-channel data movement	complete
Dual Ethernet MAC	Independent network paths	complete
Boot ROM	RISC-V autonomous boot support	complete
Linux software stack	OS and networking support	On-going, target completion date June 2026
DSP software (VSOS)	Real-time processing environment	On-going, target completion date June 2026
Evaluation hardware	VSRVES02 CAT board	On-going, target completion date June 2026

Table 3: VSRVES02 building blocks and their status

4.4 Memory Architecture

One of the most significant improvements in VSRVES02 is the transition to a true shared memory architecture.

- LPDDR2 memory can be accessed by multiple system components through a shared AXI interconnect.
- Three DMA channels enable independent data transfers:
  - SD card interface
  - DSP subsystem

- Memory-to-memory operations (peripherals of RISC-V to/from LPDDR2)

All high-bandwidth data paths, including DMA channels and the RISC-V processor caches, are connected to LPDDR2 memory through a shared AXI-based interconnect. In this architecture, LPDDR2 acts as the central shared memory resource. All DMA transfers use LPDDR2 as the source and/or destination; direct peripheral-to-peripheral DMA transfers are not supported in this architecture.

All data transfers on the AXI bus are performed using a 64-bit data width. The LPDDR2 interface converts this internal 64-bit data path to the 16-bit external memory bus used by the LPDDR2 device. The AXI interconnect serializes access to LPDDR2 based on the defined priority scheme.

Characteristics

- During system initialization, the DSP has exclusive access to LPDDR2
- After boot, memory is shared between RISC-V and DSP processors
- Multiple bus masters (RISC-V data cache, instruction cache, and DMA channels) can access LPDDR2
- Parallel access requests are handled through AXI-based arbitration
- Efficient data movement without CPU intervention using DMA
- Both processors can access the same LPDDR2 address space without hardware-based partitioning, with memory usage managed at the software level. Software coordination is therefore required to avoid conflicting accesses to shared buffers.

AXI Arbitration Model

The AXI interconnect operates as a multi-master bus, with arbitration handled in the AXI wrapper. Arbitration priority is defined as follows (highest to lowest):

- RISC-V data cache
- RISC-V instruction cache
- SD card DMA
- Memory-to-memory DMA (VSBUS peripherals)
- DSP subsystem DMA

The arbitration scheme is non-preemptive. Once a transfer has been granted access to the memory interface, it is completed without interruption. Higher-priority masters cannot preempt an ongoing transfer and are serviced when the interface becomes available.

Implications

- Continuous interaction between RISC-V and DSP enabled by shared memory
- Reduced latency in data exchange compared to VSRVES01
- Significantly improved system throughput in data-intensive tasks
- Improved memory allocation scalability for various applications
- Predictable memory access behavior, with worst-case latency bounded by the duration of an ongoing transfer

4.5 Network Interface Architecture

VSRVES02 introduces a dual Ethernet MAC architecture, replacing the shared MAC approach used in VSRVES01. The two MACs provide separate logical Ethernet data paths while sharing the same physical Ethernet interface.

Architecture

- Two independent MAC units connected to a single physical Ethernet interface
- Separate logical data paths for:
  - RISC-V processor (Linux networking stack)
  - DSP subsystem (real-time data processing)

Benefits

- More scalable packet-buffer allocation
- Using two Ethernet MAC addresses on a single physical interface reduces cost and increases flexibility.
- Networking and real-time processing can operate independently

This architecture significantly improves system flexibility in streaming and real-time applications.

4.6 Inter-Processor Communication

In VSRVES02, inter-processor communication is primarily based on:

- Shared LPDDR2 memory
- DMA-based data transfers to LPDDR2

These are documented in Chapter 4.4.

UART-based communication used in VSRVES01 still exists, but it is largely replaced in practice by shared-memory and DMA-based communication.

Characteristics

- High-bandwidth data exchange
- Reduced CPU overhead
- Efficient bulk data transfer

Implications

- Improved scalability for data-intensive workloads
- Better support for continuous data streams from non-Ethernet sources (such as a companion Wi-Fi chip)
- Simplified software architecture for data sharing

4.7 Boot Architecture

VSRVES02 introduces an enhanced boot mechanism compared to VSRVES01.

- An experimental boot ROM is included in the RISC-V subsystem
- The DSP can still assist in system initialization if required

Characteristics

- More flexible boot configurations
- Reduced dependency on DSP for initialization
- Foundation for future autonomous secure boot capability

4.8 System-Level Behavior

The architectural improvements in VSRVES02 enable:

- True parallel operation of RISC-V and DSP subsystems
- Continuous data exchange between processing domains
- Reduced latency in real-time data paths originating from non-Ethernet sources (e.g., Wi-Fi, Bluetooth)
- Efficient handling of high-throughput workloads

The system is better suited for applications such as:

- Low-latency audio streaming extended to cover Wi-Fi and Bluetooth devices
- Edge processing and AI-assisted workloads
- Networking-intensive embedded systems

4.9 Expected Performance

The VSRVES02 figures are estimates based on simulation and FPGA validation until silicon measurements are available.

Based on architectural improvements and simulation results:

- Reduced Linux boot time (from 2.3 to 1.4 seconds)
- CoreMark/MHz of about 2.0
- Improved memory throughput due to 64-bit interface
- Lower CPU load due to DMA-based data movement
- Reduced end-to-end latency in streaming applications

These improvements directly address the limitations observed in VSRVES01.

4.10 Package Design

VSRVES02 introduces the need for a dedicated package design optimized for the requirements of the second-generation demonstrator. We developed a custom package to support increased system complexity, higher data bandwidth, and number of peripheral interfaces compared to VSRVES01.

Design Objectives

The package design was driven by the following requirements:

- Support for higher memory bandwidth of 64-bit LPDDR2 interface
- Support of expanded peripheral connectivity, approximately 130 I/O pins
- Efficient power distribution of multiple subsystems and power domains, 6 power domains, 2 ground domains
- Small package outline to enable small battery-powered IoT applications
- Use existing tooling of the package vendor to minimize the tooling NRE
- Moderate ball pitch to enable low-cost PCB soldering processes
- Location of the pins to enable good PCB routing layout from the I/O pins to the other components, especially to LPDDR2
- Allow optional second die on top (SPI Flash)

Design flow

The package design starts from the target application requirements. It defines the required interfaces, including the number and placement of I/O pins, to support a good PCB layout. Analog pins (such as crystal oscillator, ADC and DAC) and high frequency digital (such as LPDDR2) are priority

pins and their routing is designed first. Power routing also has an important role: the die should have low-resistance paths to the PCB power planes. The planning is done using a PCB design tool.

When pins have been drafted based on PCB requirements, next step is to divide them evenly on all four sides of the die without significantly changing the order and side that was earlier optimized for the PCB. Placing ground or power pins adjacent to sensitive signals significantly reduces unwanted coupling. If the pad ring does not contain all power domains, then I/Os need to be grouped by power domain, and clamp cells must be placed for each domain. This phase can be done with a spreadsheet assuming that dimensions of I/O cells are known.

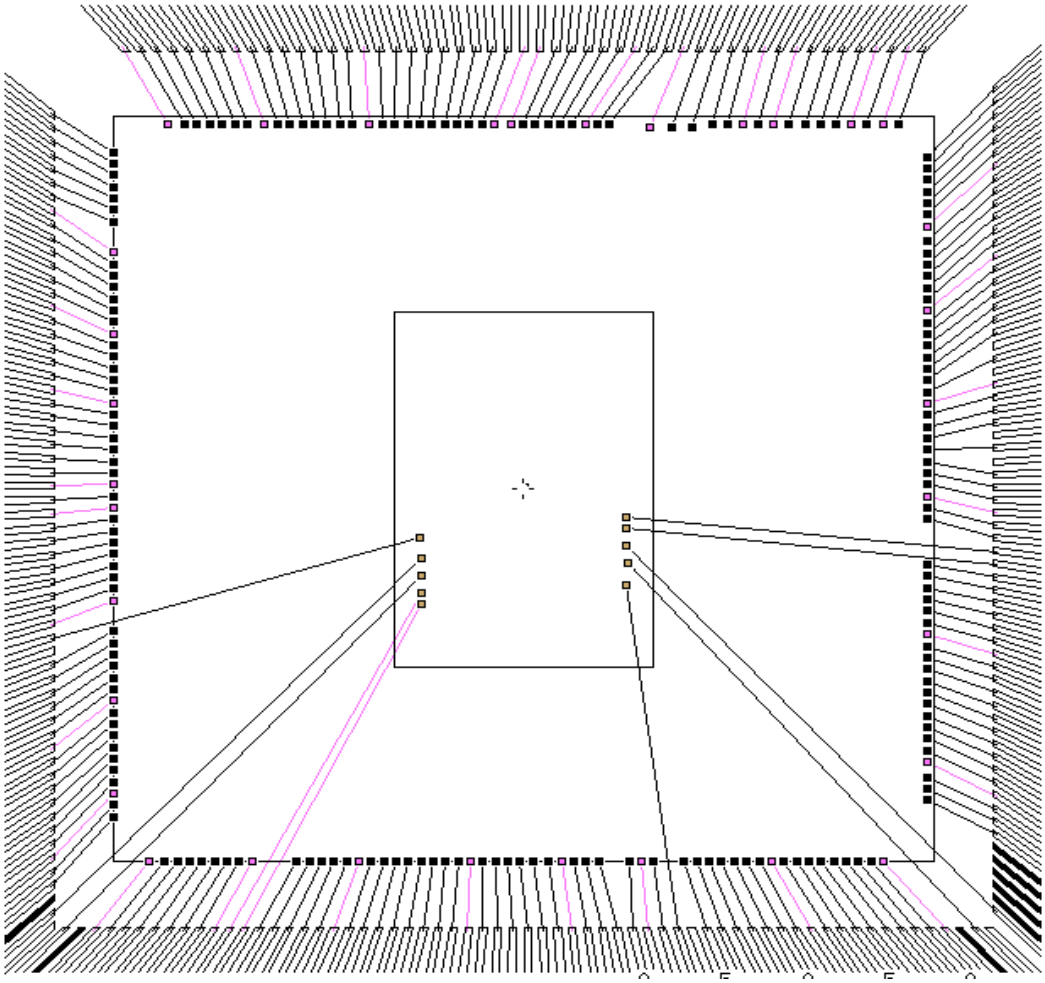

Figure 4: Draft bonding of the VSRVES02 die and optional Flash die

The next phase is to estimate the chip size and create the first draft bonding. This is done by using the IC layout tool. We generated the bonding pads of the target chip using a layout generator, with each bonding pad having a corresponding terminal at its center. Then we generated the landing pad coordinates with a package bonding layout generator and then we connected the terminals to the

landing leads of the package. Resulting layout is shown in Figure 4. Red bonding wires of the figure indicate ground. This phase is done using a silicon layout tool with support for layout generators (layout by programming).

The last step is to transfer the coordinates of the landing bonding pins from the silicon layout tool to the package design tool. We simply export the terminal names and coordinates from the layout tool in text format to the file. This file is then imported into the package design tool to automatically generate the landing pins for the package design. For package design we used low-cost commercial version PCB design tool (Eagle) such that tool used 1000x scaling i.e. micrometers were represented as millimeters in the design tool. In PCB design tool the bonding landing pins are routed to the package pins, in this case balls of the TFBGA package. The resulting 4-layer package layout is shown in Figure 5.

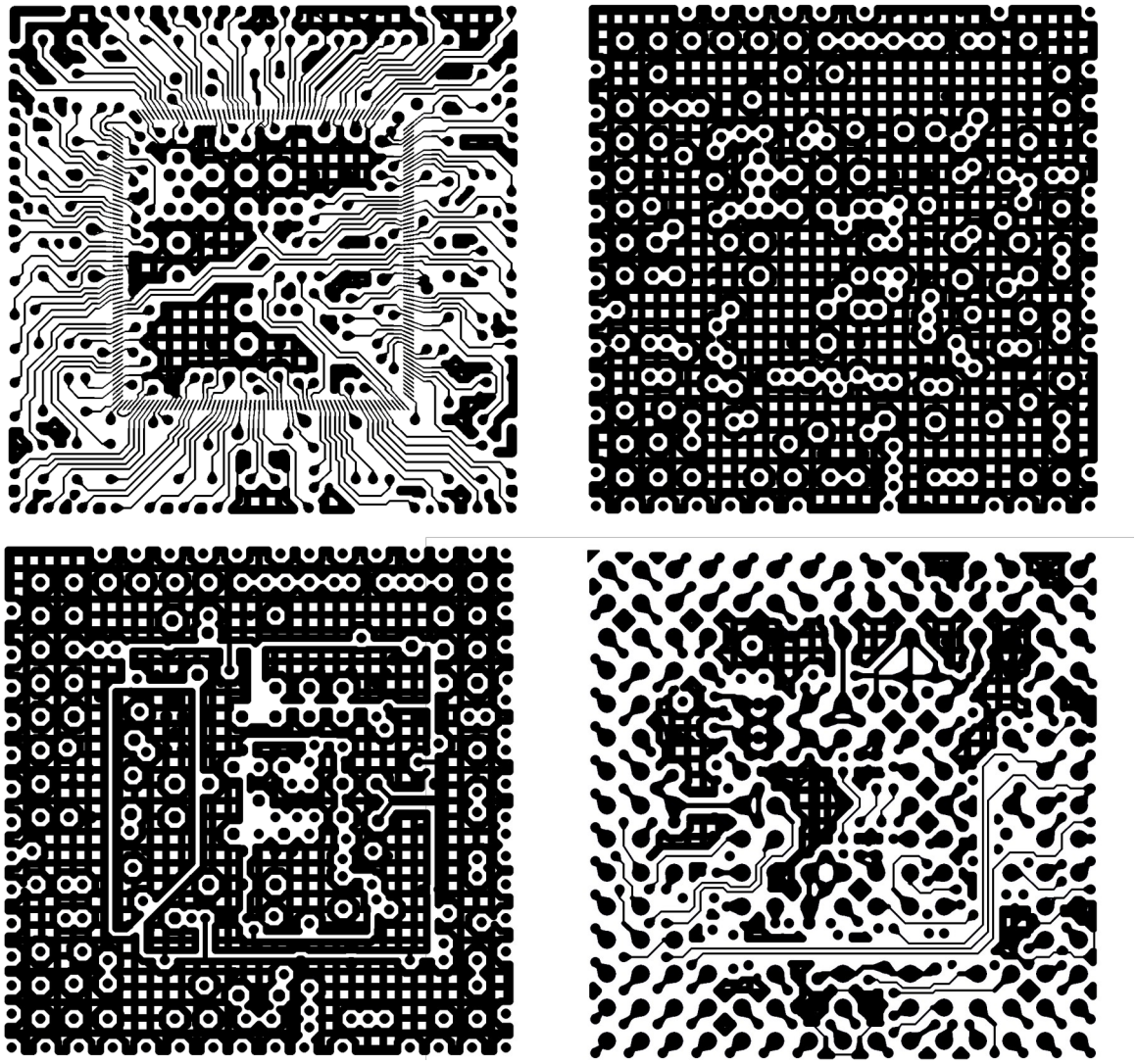

Figure 5: Substrate layers of VSRVES02 package

Package summary

After multiple iterations the best compromise was achieved by using TFBGA package with the following characteristics:

- Package outline 10 x 10 x 0.8 mm
- Ball size: 0.30 mm
- Ball pitch: 0.65 mm
- Number of balls: 169
- Number of layers: 4
- Bonding wire: CuAuPd
- Wire diameter: 0.7 mil

The ball layout of the package is shown in Figure 6.

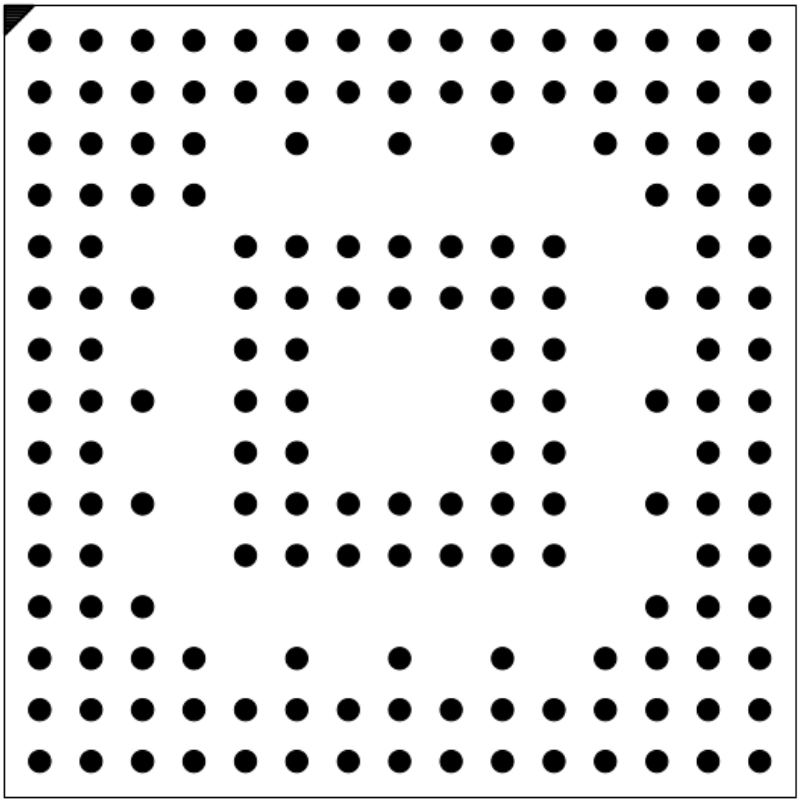

Figure 6: Top view of the ball locations

Final bonding diagram

The bonding diagram in Figure 7 shows VSRVES02 die on the bottom, and optional Flash die on the top (above VSRVES02).

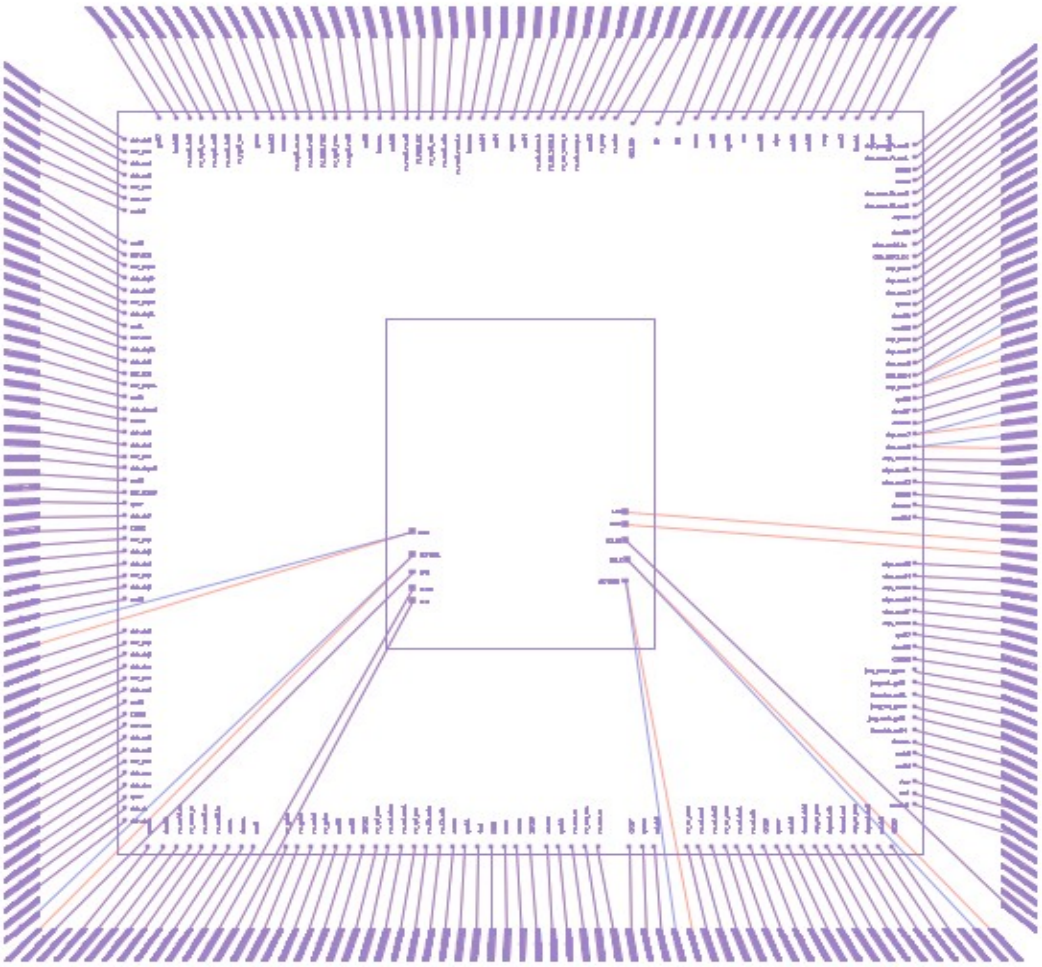

Figure 7: Bonding diagram showing VSRVES02 and optional flash die

4.11 Demonstrator Integration Steps

VSRVES02 demonstrator integration steps and their status are shown in Table 4.

Milestone	Description	Status	Date
M1	VSRV2 architecture definition	finished	Aug 2025
M2	Simulation and architectural validation (ExactStep)	finished	Sep 2025
M3	RTL implementation	finished	March 2026
M4	FPGA-based validation	finished	April 2026
M5	SoC-level integration	finished	May 2026
M6	Silicon prototype (VSRVES02)	Tapeout completed; samples expected after packaging	May 2026
M7	Evaluation board (CAT board) availability	started/on-going	April 2026

Table 4: VSRVES02 demonstrator integration steps

Remaining bring-up and characterization activities are planned after the TRISTAN project period.

4.12 Design Evolution Summary

The transition from VSRVES01 to VSRVES02 shown in Table 5 represents a shift from a simplified validation platform to a more complete and scalable system architecture.

Feature	VSRVES01	VSRVES02
RISC-V core architecture	VSRV1: RV32IMSU_Zicsr_Zifencei	VSRV2: RV32IMA_Zicsr_Zifencei_Zba_Zbb_Zbs_Zbkb_Zicond
Package	QFN-88 10x10x0.8mm	TFBGA-169 10x10x0.8mm
LPDDR2 memory visibility	Multiplexed (DSP during boot, RISC-V during operation)	Multiplexed in boot, shared during operation
AXI / internal memory data bus	32-bit	64-bit
Inter-processor communication	Fast UART	Shared memory + DMA
Ethernet interface	Single MAC (shared & filtered)	Dual MAC (independent paths)
Boot	DSP-dependent	Boot ROM + DSP support
Parallelism	No DMA	Three DMAs

Table 5: Design evolution of the processors

4.13 Key Performance Indicators (KPIs)

Key performance indicators of the demonstrators are shown in Table 6.

KPI Category	VSRVES01 (VSRV1) Measured	VSRVES02 (VSRV2) Estimated
Memory Access	Multiplexed	Shared (AXI + DMA)
Linux Boot Time	2.3 s	1.4 s
Nbench memory	0.718	1.024
Nbench integer	0.640	0.710
Nbench Floating-point	0.734	0.733
CoreMark/MHz	1.72	2.0
Clock MHz	110	110
Inter-Processor Comm.	UART	Shared memory + DMA
Memory bandwidth	220 MHz x 16 bits	440 MHz x 16 bits
CPU utilization for data movement tasks	High, no DMA	Low, due to DMAs
Audio data communication through Ethernet	Receive only	Receive and transmit
Minimum audio packet	10 ms	0.020 ms
Analog-to-analog audio delay between two devices (TX->RX)	Not possible	5–6 ms, 3–4 ms with minimum-phase software filters
Audio streaming stability from Ethernet to headphone	Days of continuous Ethernet audio streaming without interruptions	TBD after silicon bring-up
Service responsiveness	HTTP and TELNET responsiveness during concurrent workloads	TBD after silicon bring-up
Concurrent workload handling	Simultaneous execution of: Audio streaming Web server operation Control interface activity	TBD after silicon bring-up

Table 6: KPIs of VSRVES01 and VSRVES02 demonstrator SoCs

Note that VSRVES01 results are currently based on the SoC and demo PCB, while VSRVES02 is not yet available and the data were estimated using simulation and FPGA validation.

4.14 Conclusions

VSRVES02 demonstrates a significantly improved system architecture compared to the first demonstrator.

By enabling concurrent memory access, introducing independent network data paths, and improving inter-processor communication, the system achieves:

- Low latency across all supported audio streaming interfaces
- Higher throughput
- Better scalability

These enhancements validate the architectural direction and provide a solid foundation for future product development.

5 VSRVES01 Demonstrator PCB Description

5.1 Introduction

The VSRVES01 CAT Board shown in Figure 8 is the primary evaluation platform for the VSRVES01 demonstrator SoC. It provides all necessary components and interfaces to operate, evaluate, and develop applications using the integrated RISC-V and DSP subsystems.

The board enables full system functionality, including:

- Linux-based networking
- Audio processing
- External memory operation
- Peripheral interfacing

The VSRVES01 CAT Board specific support page is located in

https://www.vlsi.fi/en/support/evaluationboards/vsrves01catboard.html

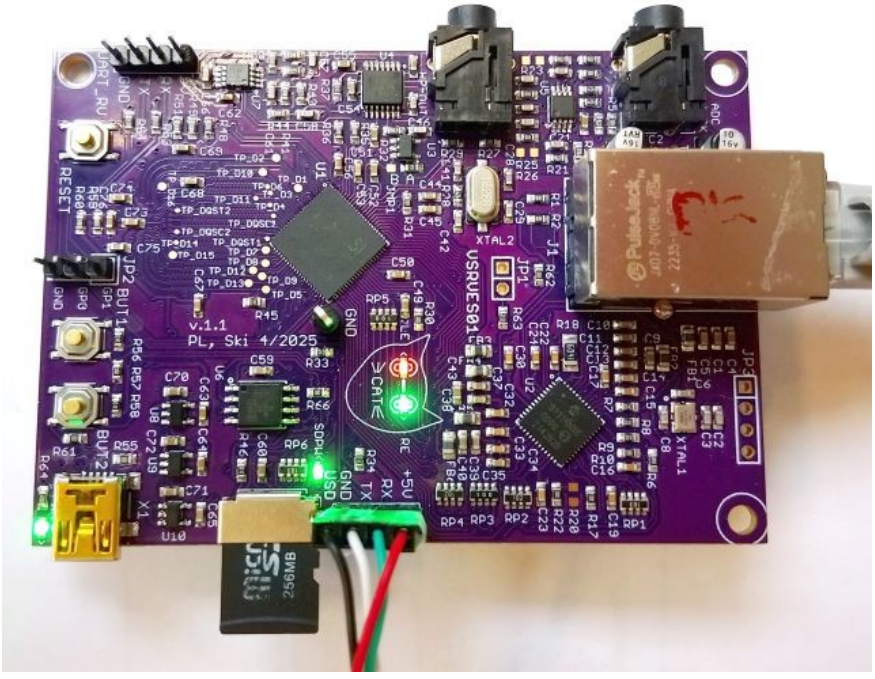

Figure 8: VSRVES01 CAT Board

5.2 Board Overview

The block diagram of the VSRVES01 CAT board is shown in Figure 9. The board integrates:

- VSRVES01 SoC
- LPDDR2 memory
- Ethernet PHY and connector
- SD card interface containing the Linux image and VSOS dynamic libraries
- Audio input and output interfaces
- SPI Flash for VSOS boot
- UART interfaces for debugging and control
- Power supply and clocking circuitry

The board is designed as a complete standalone system, requiring only external power and standard interfaces (e.g., Ethernet, USB/UART, SD card) for operation.

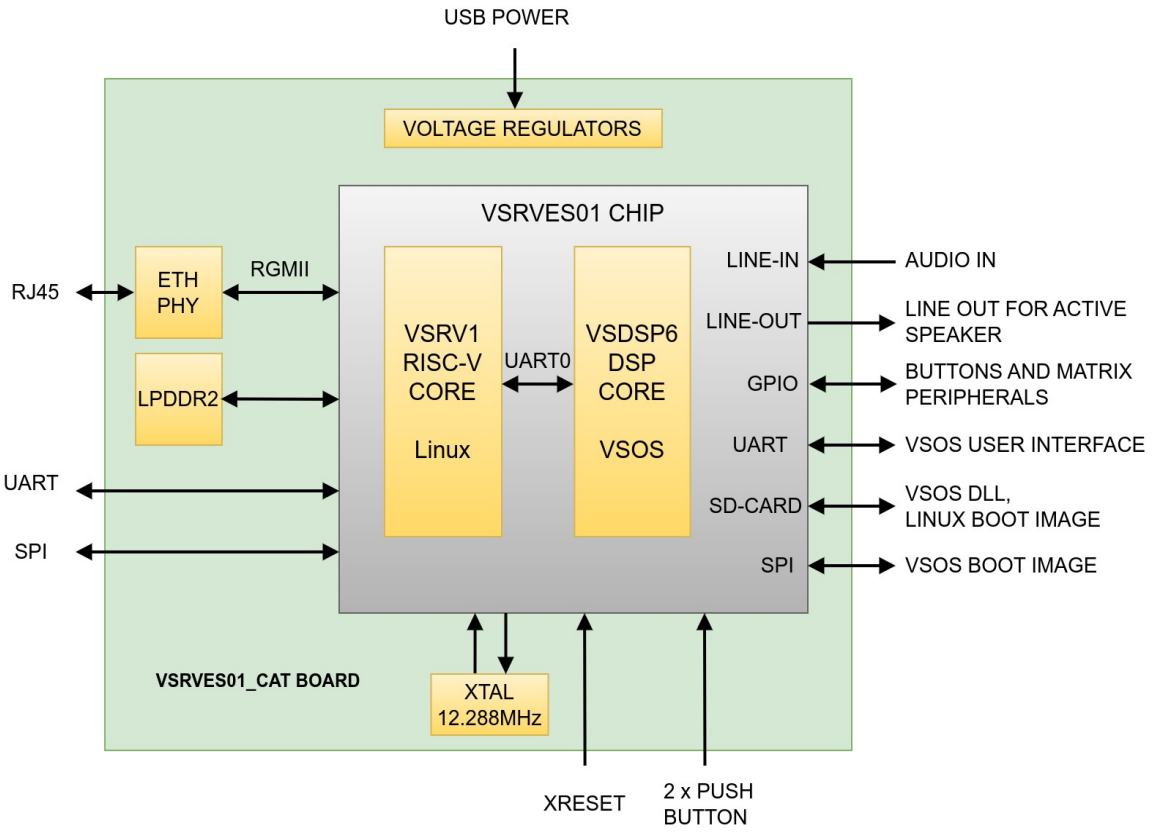

Figure 9: Block diagram of VSRVES01 CAT board

5.3 Boot Configuration

The board supports the DSP assisted boot mechanism implemented in VSRVES01.

Typical boot flow on the board:

- DSP boots from SPI Flash
- DSP initializes system peripherals and LPDDR2 interface
- DSP reads the Linux image from the SD card and copies it to the LPDDR2 memory using a dedicated 32-bit boot bus to the LPDDR2 interface
- LPDDR2 interface is disconnected from DSP and allocated to RISC-V
- RISC-V processor is released from reset and Linux boots
- RISC-V and DSP subsystems are running independently and can communicate by using UART0

The use of external storage (SD card) allows flexible software development and easy system updates. The VSOS dynamic libraries are usually moved to SPI flash when application is released to production.

5.4 Interfaces and Connectivity

The board provides multiple interfaces for system interaction:

DSP Subsystem Interfaces

- Stereo ADC and DAC
- UART communication from/to VSOS. This is the main debug console
- UART, user configurable and application specific
- SPI containing VSOS boot image, data, and optionally VSOS dynamic libraries
- SD card containing Linux image, data, and VSOS dynamic libraries in development phase
- GPIO, for buttons, status LED, etc.
- 2 x push-buttons connected to the GP0 and GP1 pins
- Ethernet (shared access with RISC-V)

RISC-V Subsystem Interfaces

- SPI, for matrix display etc
- UART as Linux console
- UART, user configurable and application specific
- GPIO, for buttons, status LED, etc.
- Ethernet (shared with DSP)

5.5 Typical Use Cases

The VSRVES01 CAT Board supports a variety of application scenarios, including:

- Audio streaming over Ethernet
- Embedded web server with audio functionality
- Real-time audio processing controlled by Linux
- Development and testing of Linux + DSP cooperative applications

5.6 Development Environment

Required hardware

- VSRVES01 CAT Board (https://webstore.vlsi.fi/vsrves01-cat-board)
- USB → UART cable connected between VSRVES01 CAT Board and PC (included in the CAT Board shipment)
- Micro-SD card that is formatted to FAT32 (included in the CAT Board shipment)
- Personal computer (Laptop)

Required terminal emulation software

- A UART terminal emulation program, such as:
  - microcom (https://github.com/geo-tp/MicroCOM)
  - PuTTY (https://putty.org/index.html)
  - TeraTerm (https://teratermproject.github.io/index-en.html)

Software development toolchain

The VSRVES01 CAT Board is delivered with all required software. SPI flash has the boot image for DSP and SD card has Linux image, Linux configuration, and VSOS dynamic libraries. Software download is needed only if the user wants to modify or update the software. Toolchain related instructions, such as:

- Running the CAT board
- Running demo programs on the CAT board
- Installing and updating VSOS Kernel
- Developing VSDSP Applications and drivers
- Building RISC-V/Linux toolchain for VSRV
- Building RISC-V side of VSRV
- Converting binaries to VRI format for SD card

are available in the VSRV User's Guide [5].

5.7 Demo software

The VSRVES01 CAT Board is available at https://webstore.vlsi.fi/vsrves01-cat-board, and detailed instructions to run the demo are provided in the VSRV User's Guide [5]. Only a summary is provided here.

The VSRVES01 CAT Board comes with hardware and software. Additionally, the following are required:

- PC (laptop)
- Ethernet cable to connect the VSRVES01 CAT Board to PC (via router)
- headphones connected to the 3.5mm headphone connector

The block diagram of the demo software is shown in Figure 10.

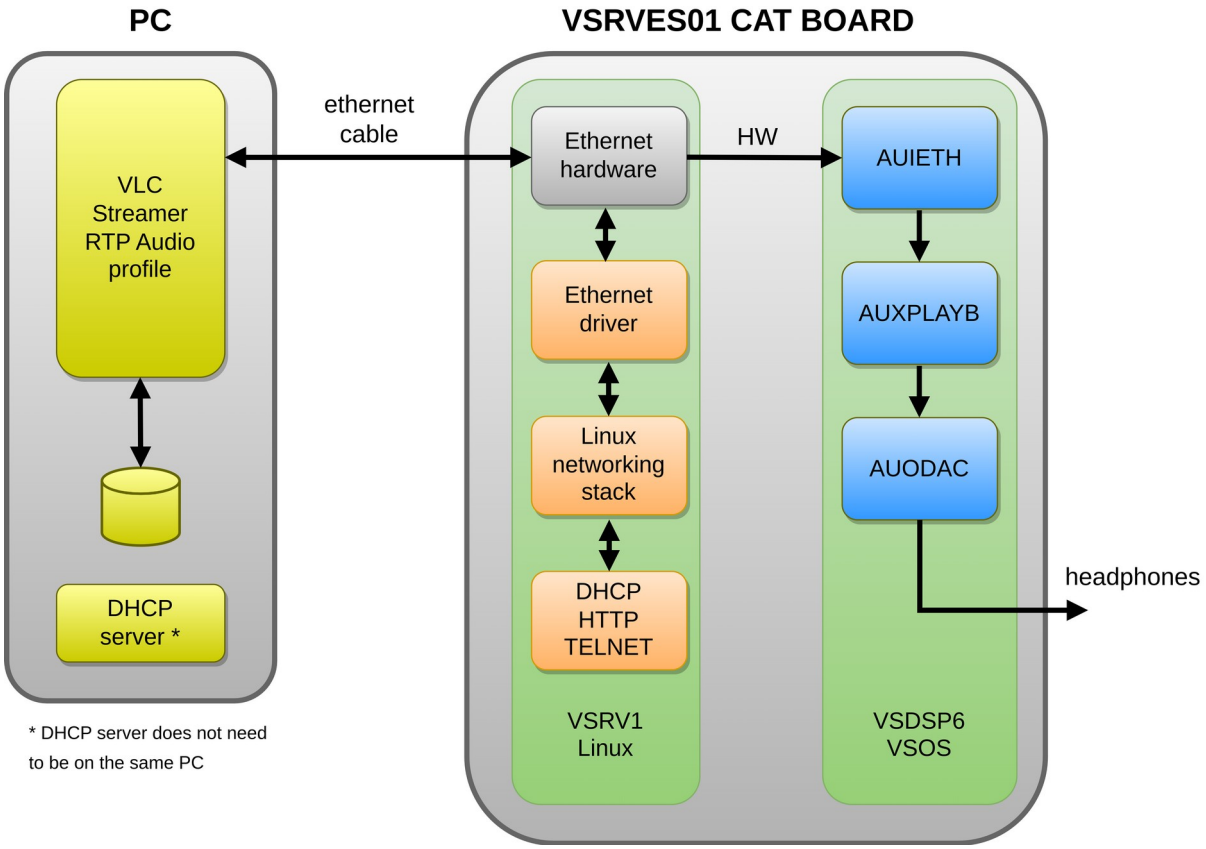

Figure 10: Block diagram of the demo software

The demonstration follows the standard DSP-assisted boot process explained in Chapter 3.5 . After boot, the system is accessible through standard network interfaces.

5.7.1 Network Services

Once Linux is running, the system provides multiple network-based services:

- DHCP client operation for automatic network configuration
- TELNET access for remote command-line interaction
- HTTP server for web-based control and monitoring

These services demonstrate that the RISC-V processor can run a full Linux networking stack and standard applications.

5.7.2 Audio Streaming Demonstration

A key part of the demonstration is audio streaming over Ethernet, combined with real-time DSP processing.

Two main use cases are demonstrated:

Streaming from Network

- Audio data is streamed from a host computer (e.g., using VLC)
- The CAT Board receives the stream via Ethernet
- The DSP processes and outputs audio in real-time
- The received audio is played back to line-out or headphones

Local Audio Processing

- Audio input from line-in can be processed by the DSP
- The processed signal is output to line-out or headphones

The DSP handles audio decoding and processing, while Linux manages the network connection and data flow.

The system can be accessed from a host computer over the local network using standard tools such as a web browser or telnet client.

Access of the audio processing parameters

- The RISC-V subsystem runs a web server that allows audio parameters to be changed on the fly while receiving or transmitting audio
- The interface is dynamically loaded and runs on the client laptop

5.7.3 Combined Operation

The demonstration shows that the system can perform **multiple tasks simultaneously**, including:

- Serving web pages via HTTP
- Accepting remote connections via TELNET
- Streaming and processing audio

5.7.4 Observations and Significance

The demonstration confirms the key architectural principles of the VSRVES01 system in a practical environment.

The system successfully demonstrates that:

- A compact RISC-V core can run a full Linux operating system with networking support
- The DSP subsystem can perform real-time audio processing without interruption
- Networking and audio processing can operate concurrently within the same system

The observed behavior highlights the effectiveness of the architectural division:

- The RISC-V processor manages networking, system control, and user interaction
- The DSP subsystem handles time-critical audio processing

This separation allows reliable real-time performance while maintaining the flexibility of a Linux-based software environment.

The demonstration also validates the feasibility of using the VSRVES01 architecture in practical applications, such as:

- Network-connected audio devices
- IoT systems combining connectivity and real-time processing
- Embedded platforms requiring both Linux-based control and deterministic signal processing

Overall, the demonstration provides strong evidence that the VSRV-based architecture enables an efficient combination of software-driven functionality and high-performance DSP processing in a single system.

6 References

[1] https://www.vlsi.fi/en/products/vsrves01.html "VSRVES01 – Linux-capable RISC-V and VSDSP System-on-a-Chip"

[2] https://www.vlsi.fi/en/vsrv/vsrv1.html "VSRV1 - 32-bit Linux-capable RISC-V Core" repository

[3] https://www.vlsi.fi/en/products/vsrves02.html "VSRVES02 – Linux-capable RISC-V and VSDSP System-on-a-Chip"

[4] https://www.vlsi.fi/en/vsrv/vsrv2.html "VSRV2 - 32-bit Linux-capable RISC-V Core" repository

[5] https://www.vlsi.fi/fileadmin/products/vsrv/vsrv_guide.pdf "VSRV User's Guide - How to use and develop with VSRV"

[6] https://www.vsdsp-forum.com/phpbb/viewforum.php?f=17 VSRVES01 discussion thread

[7] https://www.vlsi.fi/en/support/evaluationboards/vsrves01catboard.html VSRVES01 Cat Board top page

[8] https://www.vsdsp-forum.com/phpbb/viewtopic.php?t=3234 VSRVES01 Cat Board discussion thread