U-Boot and Embedded Linux porting to custom RISC-V based SoC

Mridul M S, Alphin Thomas, Sneha P Roby
Software Engineer, TOSIL SYSTEMS Pvt. Limited
Trivandrum, Kerala, India.

INTRODUCTION

The case study focused on enabling a reliable boot process and operating system support for the Shakti VAJRA SoC, a RISC-V based processor implemented on the Xilinx Arty A7-100T FPGA. The primary requirement was to bring up a flexible bootloader and ensure smooth integration with Embedded Linux for application development.

To accelerate development, QEMU was used alongside the FPGA platform, enabling early validation of U-Boot and Linux integration through full system emulation without hardware dependencies.

Our team successfully ported and customized U-Boot for the VAJRA SoC, addressing challenges such as hardware initialization, memory mapping, and peripheral driver adaptation. This effort enabled stable system bring-up, early console debugging, and Linux kernel booting, laying the foundation for future enhancements such as secure boot and advanced debugging workflows.

This case study is a critical role in accelerating firmware development cycles and establishing a robust platform for further RISC-V based solutions.

THE CUSTOMER

The customer is a research-oriented organization engaged in open-source hardware development, focusing on RISC-V based processors and SoC architectures. Their goal is to enable an ecosystem for prototyping, research, and deployment of custom embedded solutions on FPGA platforms.

THE REQUIREMENT

The customer required a custom U-Boot port for the Shakti VAJRA platform with the following capabilities:

- Boot Linux kernel and bare-metal test applications.
- Provide early UART console for reliable debugging.
- Fit within the resource constraints of an FPGA environment.
- Ensure scalability for future requirements (network boot, secure boot).

Scope of Work: The scope of this work involves porting and deploying the U-Boot bootloader on the Shakti VAJRA SoC (Arty A7-100T, RV64IMAC) to enable reliable system bring-up and software booting. It covers setting up the RISC-V cross-compilation toolchain, configuring U-Boot for the VAJRA architecture, and customizing board-specific initialization. The work extends to implementing and validating UART drivers, defining memory maps, and adapting low-level initialization code for VAJRA's hardware. It includes debugging of boot flow issues and ensuring stable console output. The deployment further encompasses enabling U-Boot scripting support, facilitating the loading of Linux kernels, bare-metal applications, and diagnostics firmware. Incremental validation through serial, flash, and memory operations forms a core part of the effort. The scope also considers optimizing U-Boot for FPGA resource constraints while ensuring flexibility in boot methods. Finally, this work establishes a foundation for advanced boot features like secure boot and network boot, making VAJRA a robust research and development platform.

SOLUTION PROVIDED

A customized U-Boot deployment was implemented on the Shakti VAJRA SoC (Arty A7-100T, RV64IMAC), tailored with minimal configurations, board-specific drivers, and optimized memory mapping. This solution enabled seamless booting of Linux and bare-metal applications, streamlined hardware bring-up, and provided a flexible, robust bootloader environment for future enhancements.

Cross-Compilation Build Setup: The work involved configuring the RISC-V 64-bit cross toolchain and building a minimal U-Boot configuration tailored for the Shakti VAJRA SoC. This lightweight setup was optimized to suit FPGA resource constraints while ensuring essential bootloader functionality.

Board-Specific Customization: The boot flow was tailored to the Shakti VAJRA platform by adapting U-Boot's initialization sequence to its custom memory map and hardware peripherals. This involved ensuring smooth transfer of control from U-Boot to the Linux kernel or bare-metal applications, with reliable UART and QSPI flash support enabling consistent system bring-up.

Boot Flow Development: The deployment ensures proper memory initialization and stack setup, followed by enabling the U-Boot prompt over UART for debugging and interaction. Finally, control is seamlessly handed off to either the Linux kernel or a bare-metal payload, enabling flexible boot options.

Testing & Debugging: The deployment was validated through register-level debugging of the UART to ensure reliable serial console output, along with successful verification of flash

read/write operations. Additionally, U-Boot scripting was implemented to automate repetitive developer workflows, streamlining the bring-up and testing process.

Bare-Metal & Virtualized Testing:

- **Bare-Metal UART on QEMU::** A minimal UART program was executed under QEMU to validate console I/O functionality in a controlled environment before FPGA deployment.

The UART application is shown running on QEMU in Fig. 1.

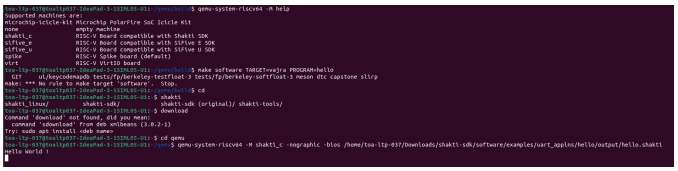

Fig. 1: UART application running on QEMU, demonstrating successful compilation and execution.

Technical Specifications

- **Processor Core:** Shakti VAJRA SoC on Arty A7-100T FPGA
- **Operating Frequency:**100 MHz
- **Memory:** On-chip SRAM: 128 KB
- **External DDR3:** 256 MB
- **Bootloader:** U-Boot
- **Peripherals Supported:** UART, SPI/QSPI Flash, GPIO, Timers
- **Debug Interface:** UART serial console output
- **Toolchain:** RISC-V GCC
- **Operating Systems Tested:** Linux and Bare-metal applications
- **FPGA Platform:** Xilinx Arty A7-100T

Hardware Development

The adaptation of the Shakti VAJRA SoC was limited to FPGA configuration and peripheral mapping. Memory controller initialization was carefully verified to ensure stable DDR3 access. UART electrical signaling was validated to enable reliable serial console communication. QSPI Flash was configured as the primary boot medium for U-Boot and application loading. Minimal modifications were made to U-Boot to align with the VAJRA memory map. Boot flows for both bare-metal applications and Linux were successfully tested.

As shown in Fig. 2, the Arty A7-100T development board is used for loading U-Boot.

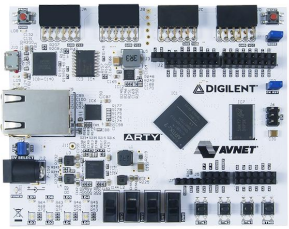

Fig. 2: Artix-7 FPGA Development Board

Software/Firmware Development

The software and firmware development for the Shakti VAJRA SoC on the Arty A7-100T FPGA primarily focused on building a stable bootloader environment, ensuring robust peripheral support, and enabling smooth transitions between bare-metal and operating system workloads.

The development process began with configuring U-Boot for the custom VAJRA architecture. This involved adding board-specific initialization routines such as memory map alignment, clock configuration, and early console setup. These routines ensured that the hardware platform was properly initialized during the earliest stages of the boot process. A custom UART driver was implemented to provide serial communication, enabling reliable debug output and console interaction during boot. The UART integration was essential not only for debugging but also for interaction with U-Boot and higher-level software components.

In addition, SPI Flash support was integrated into U-Boot, allowing programs and kernels to be stored in non-volatile memory and loaded into RAM at runtime. This enabled a persistent and reusable boot process, reducing reliance on external loaders. The bootloader was further extended to support multiple boot flows, covering both bare-metal applications and the Linux kernel. Bare-metal applications were directly loaded into RAM, providing a minimal environment for testing low-level drivers and hardware features. For Linux, U-Boot was configured to handle kernel image loading, device tree passing, and handover of control, ensuring a smooth transition into a fully functional operating system environment.

QEMU was introduced alongside the FPGA-based workflow to accelerate the development cycle. A QEMU-based virtual platform of the VAJRA SoC was configured, enabling developers to validate U-Boot bring-up, memory mappings and UART output. This not only reduced dependency on FPGA resources but also allowed for faster debugging of early boot stages, peripheral drivers, and boot flows. By running parallel validation on both QEMU and the FPGA, issues could be isolated more efficiently and regression testing was automated in the emulator before committing to hardware testing.

To improve the developer experience, U-Boot scripts were configured to automate common tasks, such as memory tests, image loading, and boot sequence selection. These scripts reduced manual intervention and streamlined development and testing cycles. The firmware effort also included validation of logical and structural correctness of configurations, ensuring that both hardware resources and software components interacted without conflicts.
Overall, the firmware development established a reliable booting framework on Shakti VAJRA, bridging the gap between hardware, QEMU-based emulation, bare-metal workloads, and operating system environments. This dual-platform approach laid the foundation for future extensions in both research and product development.

System Integration & Testing

System integration was carried out through an incremental validation approach, beginning with serial output verification to confirm early-stage functionality. Status register behaviors were closely observed, including TX full/empty indicators and RX threshold responses, ensuring proper communication flow. Debugging efforts focused on confirming reliable UART operations, memory access stability, and bootloader execution consistency. Step-by-step validation confirmed the correctness of U-Boot initialization on the Shakti VAJRA SoC. Boot paths were tested for both Flash-based loading and serial download modes, ensuring robustness in payload handoff. The integration process confirmed stable operation under different workloads and maintained consistency across multiple boot cycles. This systematic testing methodology provided confidence in both the hardware-software co-design and the readiness of the platform for further application- level development.

CHALLENGES

The primary challenges during development were centered around UART communication reliability.

Data Mismatch in UART Transmission

Challenge: During early testing, transmitted UART data often showed mismatches, with received characters differing from the expected output.

Solution: The issue was resolved by carefully analyzing UART register configurations, adjusting baud rate settings, and validating timing alignment between transmitter and receiver. These refinements eliminated data corruption and ensured accurate communication.

Multiple Key Press Detection Issue

Challenge: A single key press on the UART console was sometimes interpreted as multiple inputs, causing unexpected behavior.

Fig. 3 shows one of the challenges faced during the UART implementation, namely the multiple key press detection issue.

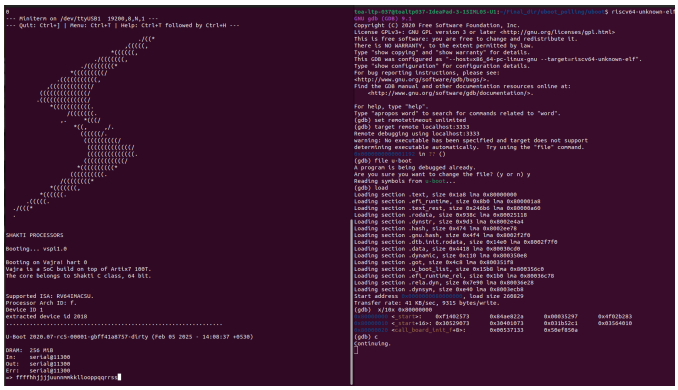

Fig. 3: Console Log of the Multiple Key Press Detection Issue

Solution: Fix the RX-empty check by using the correct 8-bit mask for your status register. Also update uart driver to use the driver's regs pointer instead of hard-coded addresses.

ACHIEVEMENTS

The project successfully enabled the deployment of U-Boot on the Shakti VAJRA SoC. A stable UART console debugging environment was established, which significantly improved system visibility and error tracking during development. The integration validated the custom RISC-V 64-bit porting efforts and confirmed hardware-software compatibility on the FPGA-based platform.
Figure 4 shows the final output of the U-Boot deployment on the Arty A7-100T FPGA development board.

Additionally, the setup laid a strong foundation for future enhancements, including secure boot mechanisms and network- based boot support. The work also demonstrated the feasibility of scaling open-source RISC-V solutions for real-world applications in research and prototyping environments.

CUSTOMER BENEFITS

- **Accelerated Development Cycles with U-Boot Integration:** The integration of U-Boot with the Shakti VAJRA SoC enables rapid application loading and testing, significantly shortening development timelines. Developers benefit from quicker prototyping and validation, which enhances overall productivity.
- **Modular and Scalable Framework for Future Expansion:** The design supports modularity, making it easy to integrate additional peripherals and advanced features. This ensures long-term adaptability, extending the platform's relevance for evolving use cases.
- **Strengthened RISC-V Ecosystem and Reduced Time-to-Market:** Porting U-Boot to the VAJRA SoC

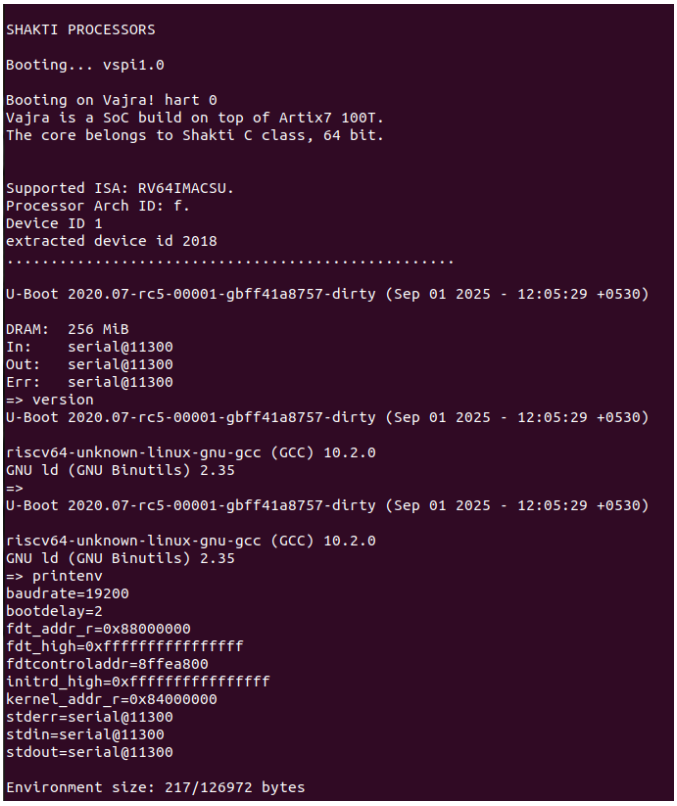

Fig. 4: Serial output log of U-Boot deployment

strengthens the RISC-V ecosystem by providing a robust foundation for development. It accelerates innovation, shortens development cycles, and supports both academic research and industrial applications, enabling faster time-to-market.