Porting and FPGA Implementation of LXDE Desktop Environment Based on RISC-V

Xiaofeng Zou
Inspur Artificial Intelligence Research Institute and Shandong Yunhai Guochuang Cloud Computing Equipment Innovation Center Co., Ltd., Jinan,, China,
zouxf@inspur.com

Tuo Li
Inspur Electronic Information Industry Co., Ltd. and Inspur Artificial Intelligence Research Institute, Beijing,, China,
lituo@inspur.com

Rengang Li
Inspur Electronic Information Industry Co., Ltd. and Inspur Artificial Intelligence Research Institute, Beijing,, China,
lirg@inspur.com

Linlin Yang
Shandong Yunhai Guochuang Cloud Computing Equipment Innovation Center Co., Ltd., Jinan,, China,
yanglinlin@inspur.com

Xiankun Wang
Shandong Yunhai Guochuang Cloud Computing Equipment Innovation Center Co., Ltd., Jinan,, China,
wangxiankun@inspur.com

Changhong Wang
Inspur Electronic Information Industry Co., Ltd. and Inspur Artificial Intelligence Research Institute, Beijing,, China,
wangchh01@inspur.com

ABSTRACT

As a new generation of open and reduced instruction set architecture, RISC-V has the characteristics of concise instruction set, modularity and scalability, and has been quickly and widely used due to its advantages of agile development mode and complete tool chains, but there are few desktop systems based on RISC-V at present. Based on the open source RISC-V BOOM core, this paper constructs a desktop system which can start the LXDE graphical interface on Xilinx VU440 FPGA. Specifically, we implement a variety of peripheral modules such as bus conversion and VGA on the boom core, and port the open source bootloader, Linux 4.15 kernel and LXDE desktop environment. Under the desktop system, standard test sets, such as CoreMark can be run normally, and the test results show that the performance of the system is at a high level. By extending peripherals, this RISC-V prototype system can be used to realize SoC in many special areas.

CCS CONCEPTS

• **Computer systems organization**; • **Embedded and cyber-physical systems**; • **System on a chip**;

KEYWORDS

RISC-V, BOOM, SoC, Desktop environment

ACM Reference Format:
Xiaofeng Zou, Tuo Li, Rengang Li, Linlin Yang, Xiankun Wang, and Changhong Wang. 2021. Porting and FPGA Implementation of LXDE Desktop Environment Based on RISC-V. In *The 5th International Conference on Computer Science and Application Engineering (CSAE 2021), October 19–21, 2021, Sanya, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487075.3487151

1 INTRODUCTION

In 2010, David A. Patterson et al. [1] proposed the fifth-generation reduced instruction set architecture RISC-V. The instruction set adopts the BSD open source protocol and has the advantages of concise instruction, modularization and scalability [2]. At the same time, Chisel language can realize agile chip development, and the complete tool chain makes RISC-V get rapid promotion and application, and therefore a large number of open source RISC-V cores have appeared.

In recent years, the research of RISC-V mainly focuses on the improvement of instruction set standards and ecosystem. The SoC application systems built with RISC-V as the core, especially RISC-V platforms which can start the graphical interface of the operating system are relatively rare. Based on the open source RISC-V out-of-order execution core-BOOM, this paper implements a variety of peripherals such as VGA, DDR, USB, IIC, SD, QSPI and UART through system bus conversion, ports the open source bootloader, Linux 4.15 kernel and LXDE desktop environment, and constructs a desktop system that can start LXDE. The prototype system is implemented with Xilinx VU440 FPGA, and standard test sets such as CoreMark, Dhrystone and NBench can be run in the system desktop environment.

2 BOOM

BOOM (Berkeley Out of Order Machine) is an out-of-order execution processor core with RISC-V instruction set architecture, released by Christopher Celio [3] in 2015. The second version of the architecture BOOM v2 [4] was released in 2017, and the third version BOOM v3 [5] was released in 2020. BOOM adopts RV64G instruction set, and it is a superscalar processor with out of order execution. Supporting out of order execution can improve the instruction level parallelism in code, realize fine-grained data prefetching, and improve the efficiency and performance of system instruction execution. BOOM can support virtual machine mode, boot Linux

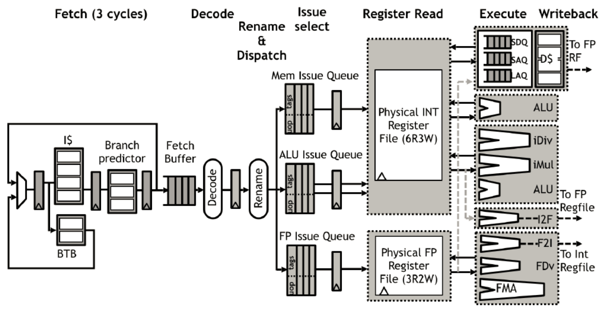

Figure 1: Design of BOOM Instruction Pipeline.

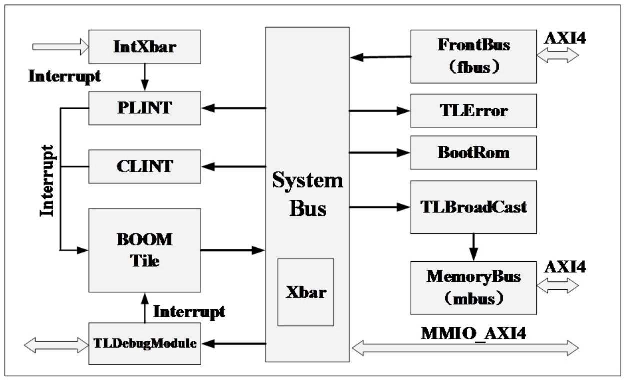

Figure 2: BOOM SoC Structure.

mode and multi-core mode. Its implementation only uses 9000 lines of Chisel hardware construction language code, and through highly parameterized generator synthesizable RTL code can be generated, the generation environment can be obtained through GitHub [6].

The instruction pipeline of BOOM V2 is shown in Figure 1 [7]. According to the pipeline function, the BOOM pipeline is divided into 10 stages: fetch, decode, register rename, dispatch, issue, register read, execute, memory, write back and commit. In fact, many stages are combined in the implementation, the actual pipeline mainly consists of the six stages in the Figure 1. The BOOM instruction pipeline has the following characteristics: the front end of the instruction pipeline adopts a three-station design, including a large group-associated branch target buffer (BTB) and a pipelined register renaming station, with the floating point and integer registers separated; A special floating-point calculation pipeline is designed; The instruction launch windows of floating-point, integer and memory access micro-operations are separated; A separate pipeline station is used for launch selection and register read operations.

The SoC shown in Figure 2 can be generated using the RISC-V BOOM environment of GitHub. The number of BOOM Tiles is configurable, and the value is at least one, and each BOOM Tile contains a BOOM core; The system bus contains an Xbar module, the main function of Xbar is to realize the conversion between buses, and BOOM Tile can access all other SoC modules through System Bus; Memory Bus can access external storage, and the output is 64-bit AXI4 bus; MMIO interface is a 64-bit AXI4 bus; IntXbar is an interrupt crossbar, which collects interrupts from internal devices

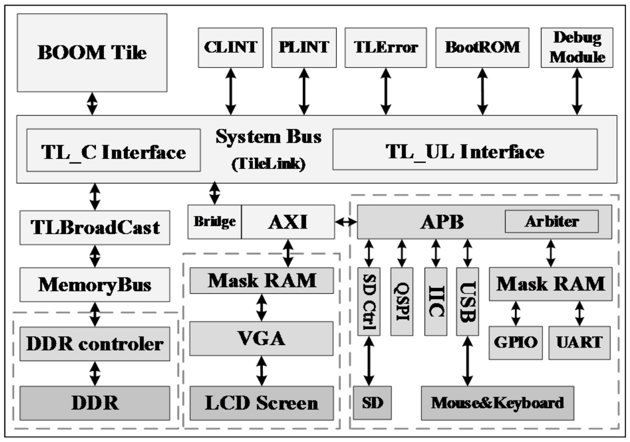

Figure 3: Desktop System Structure.

and external devices and transmits them to PLINT; PLINT is an external interrupt to receive the interrupt signal from IntXbar, and the number of interrupts can be configured; CLINT is an internal interrupt, with two basic output interrupts int0 and int1, int0 is a software interrupt, and int1 is a timer interrupt.

BOOM Core is the core module of the BOOM processor, which implements the whole process from instruction decoding to write back. In this paper, the generated BOOM core is the second version, and by modifying the parameter configuration in the BOOM generator, the dual core processor of BOOM v2 is generated.

3 HARDWARE ARCHITECTURE DESIGN

3.1 System Structure and Bus

Jonathan Balkind et al. [8] built an open-source, SMP linux-booting RISC-V system scaling from one to many cores based on OpenPiton and Ariane. In this paper, we expanded the system bus based on the generated BOOM SoC basic framework and reserved bus interface. Through the system memory bus, using the DDR controller on the FPGA, we connect DRAM as the memory, and design external modules. We connect the display monitor through the VGA display module, and connect the mouse and keyboard through the USB module.

The SoC system bus is the main channel for the core to extend external devices. On the BOOM System bus, we extend the AXI bus and APB bus by designing bridge conversion modules, arbiters and peripheral bus buffers. The structure of the desktop system is shown in Figure 3. Among them, BOOM Tile is the core module containing the BOOM processor, System Bus is the generated system bus, the processor and BOOM's own peripherals are mounted on the system bus through the TL_C and TL_UL interfaces of TileLink, and the peripherals we added are mounted through the AXI/APB bus. The whole SoC system uses three kinds of buses, namely TileLink, AXI and APB.

Table 1: SoC system Bus Address Space Division

Module	Base Address	Size	Specification
Debug	0x0000_0000	0x0000_1000	Debug
Mode	0x0000_1000	0x0000_1000	Peripherals on chip
Error	0x0000_3000	0x0000_1000	
BootROM	0x0001_0000	0x0001_0000	
Clint	0x0200_0000	0x0001_0000	
Plic	0x0c00_0000	0x0400_0000	
...	...	...	
MMIO	0x6000_0000	0x0100_0000	Mask RAM
	0x6100_0000	0x0000_3000	USB, UART, GPIO,IIC
	0x6100_3000	0x0001_0000	SD, QSPI
	0x6200_0000	0x0001_0000	VGA
	...	...	...
Mbus	0x8000_0000	0x1000_0000	DDR

3.2 Address Space and Peripherals

Based on the original system address space of the BOOM core SoC, this paper extends the peripheral module on the MMIO address space. The main address space of the system is shown in Table 1

After the implementation of the above peripheral modules, we design or modify the drivers of the corresponding peripheral devices such as Framebuffer and USB in Linux, and port the drivers based on the Linux kernel.

4 FIRMWARE AND SOFTWARE DESIGN

4.1 Bootloader

In this paper, the bootloader of the BOOM SoC system refers to the zeroth stage bootloader (ZSBL), first stage bootloader (FSBL) and Berkeley bootloader (BBL) design methods in the RISC-V community [9]. Among them, ZSBL can be programmed in the on-chip BootROM, or it can be externally downloaded to the BootROM via UART. FSBL and BBL are stored in SD card or QSPI flash according to the actual boot mode.

The RISC-V system starts from the programmed code in the on-chip Reset Vector. First, it judges the start mode and interrupt cause, and jumps to the on-chip ZSBL, executes the ZSBL code, sets the interrupt and exception vector table, configures related registers, and floating-point, branch prediction, MMU, etc. After completing the initial settings, perform the initialization of the system and the C environment, and then jump to the main function for execution, obtain the FSBL position through the GUID, and move the FSBL to the L2 LIM and continue execution. FSBL is responsible for transporting the boot program and kernel from SD card to DDR, BBL is responsible for parsing DTS, initializing individual peripheral devices, initializing the configuration of the CPU, and booting the kernel. The firmware startup process is shown in Figure 4

4.2 Linux Kernel

The Linux kernel used in this paper is version 4.15. RISC-V tool chain is used to complete the compilation of the Linux kernel and RISC-V library files, and realize the porting of Linux version 4.15 on the RISC-V prototype platform. And this paper implement the memory-based root file system on the RISC-V prototype platform, which realizing the data storage, hierarchical organization, access and acquisition, etc. The root file system can be successfully mounted after the kernel starts. The kernel code image file is stored in the root file system, and the system boot program will load some basic initialization scripts and services into the memory after the root file system is mounted.

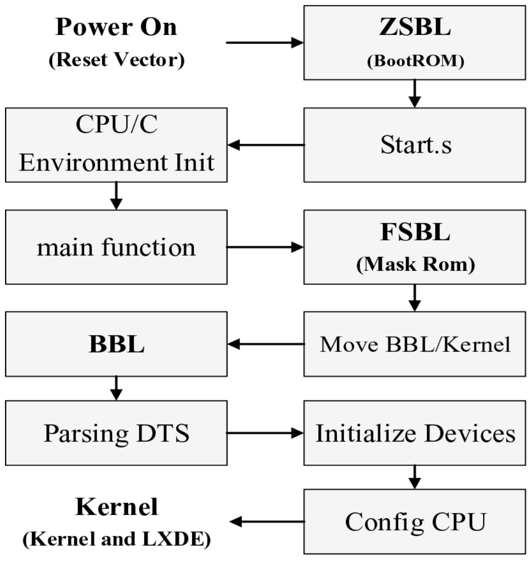

Figure 4: Firmware Startup Process.

5 DESKTOP ENVIRONMENT

In this paper, we first ported Xorg as the display manager and window manager. X window is composed of X server and X client, and the communication between X server and X client is through X protocol. As an X server, Xorg provides a basic display interface for X client, and also reflects user's operations to X client. It is an

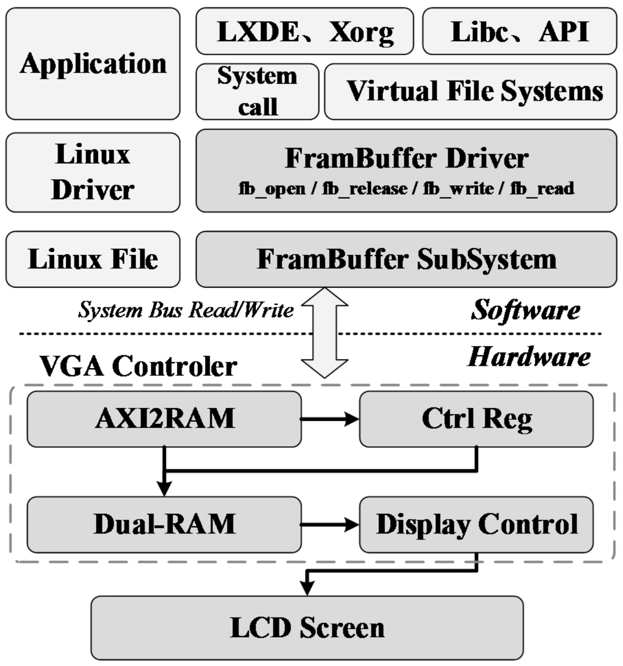

Figure 5: Desktop Environment Framework.

intermediate layer between X client and hardware, responsible for the underlying operations.

LXDE [10] is a beautiful and international embedded desktop environment based on GTK2. Compared with KDE and GNOME, LXDE takes up less resources and is suitable for working on embedded platforms. LXDE is a lightweight and fast desktop environment, which is designed to be user-friendly and occupy less resources, while maintaining low resource utilization. LXDE uses less memory and CPU to present a feature-rich desktop environment as much as possible. Unlike other desktop environments, LXDE strives to be a modular desktop environment, so each component can be used independently. This makes it easier to port LXDE to different distributions and platforms.

Frame Buffer is an interface provided by the Linux kernel for display devices. Upper-level applications can directly read and write to the display buffer through Frame Buffer, and write operations will be directly reflected on the display device. We first modified the Frame Buffer driver to realize the interaction between the desktop environment LXDE and the Linux kernel. Then we designed the VGA display controller to realize the interaction between the Linux kernel and the display device through the Frame Buffer, and open up the graphical display link. The desktop environment framework is shown in Figure 5

Use the Buildroot tool to write configuration files and Makefile scripts for the various components of the Xorg and LXDE desktop environments, including lxde-common, lxsession, lxterminal, etc. Then, based on the RISC-V gcc tool chain, cross-compile to generate a root file system image file containing the desktop environment. Finally, it is embedded in the Linux kernel and cross-compiled to generate the Linux kernel image file. The transplantation of Xorg and LXDE on the RISC-V prototype platform was completed.

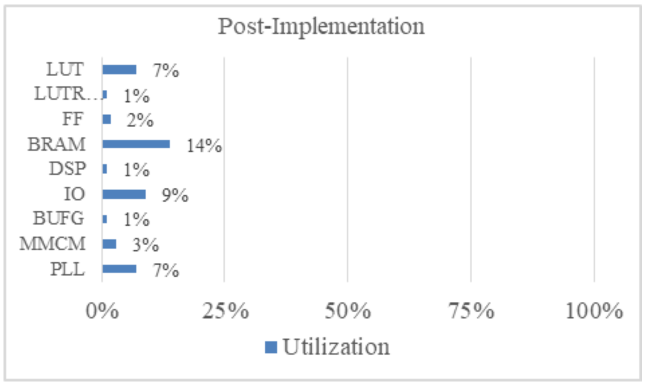

Figure 6: FPGA Resource Utilization of Desktop System.

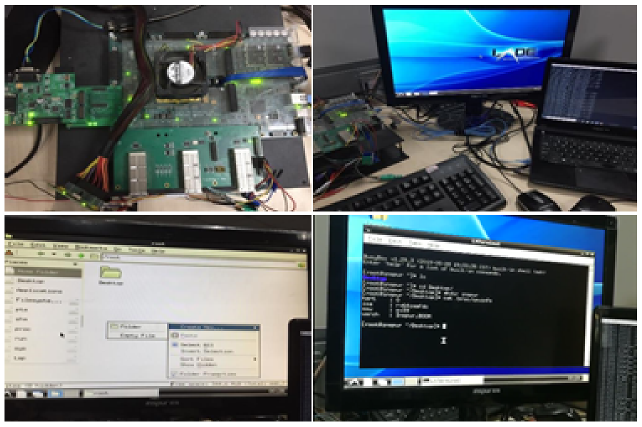

Figure 7: LXDE Desktop Environment.

6 FPGA IMPLEMENTATION AND STARTUP

We implemented the SoC Logic on Xilinx Virtex UltraScale 440 FPGA, and the core logic frequency is 50MHz. and the logic utilization rate is shown in Figure 6. The Linux image and desktop environment program are stored in the SD card.

When the system starts, it is loaded into DDR step by step through the firmware. Finally, the LXDE desktop environment runs, it is successfully displayed on the LCD Screen through the VGA display interface, as shown in Figure 7. The startup process information is displayed on the serial port.

In the desktop environment, the mouse and keyboard input is normal, and the desktop program operations such as window switching and video playback can be performed. Standard test sets such as CoreMark, Dhrystone and NBench can be run in the system desktop environment.

7 CONCLUSION

Through the research and practice of the RISC-V development process, this paper constructs a desktop system based on the open source RISC-V BOOM core, which can start the LXDE graphical interface on Xilinx VU440 FPGA. Specifically, we implement a variety of peripheral modules on the BOOM core, and port the open source boot loader, Linux 4.15 kernel and LXDE desktop environment. Under the desktop system, various applications can run normally, such as playing video, window control, etc. Standard test sets, such as CoreMark can be run normally.

The porting of the desktop environment will help to improve the RISC-V ecosystem and provide research basis and reference for the subsequent specific practical applications based on RISC-V.

ACKNOWLEDGMENTS

The work of this paper is partially supported by the Major Innovative Project of Shandong Province (2019TSLH0125), The porting work of the desktop environment and another work [11] are carried out at the same time, based on a common research foundation. The research findings, conclusions and reference designs used in this article, only represent the author's own views and have nothing to do with commercial activities.

REFERENCES

[1] Andrew Waterman, Yunsup Lee, David A. Patterson, and Krste Asanovic (2011). The RISC-V instruction set manual, Volume I: Base user-level ISA. Technical Report No.UCB/EECS-2011-62, EECS Department, University of California, Berkeley, May 2011, 7(9):475.

[2] Krste Asanovic, David Patterson (2014). The Case for Open Instruction Sets. MICROPROCESSOR report, August 2014(8):1-7.

[3] Christopher Celio, David Patterson, Krste Asanovic (2015). The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor. Technical Report No.UCB/EECS-2015-167, EECS Department, University of California, Berkeley, 2015.

[4] Christopher Celio, Pi-Feng Chiu, Borivoje Nikolic, David A. Patterson, Krste Asanovic (2017), BOOM v2: an open-source out-of-order RISC-V core. Technical Report No.UCB/ EECS-2017-157, EECS Department, University of California, Berkeley, 2017.

[5] Jerry Zhao, Ben Korpan, Abraham Gonzalez, Krste Asanovic (2020). SonicBOOM: The 3rd Generation Berkeley Out-of-Order Machine. Fourth Workshop on Computer Architecture Research with RISC-V (CARRV). 2020.

[6] University of California. https://github.com/riscv-boom/riscv-boom. . July 30, 2021.

[7] Christopher Celio, Pi-Feng Chiu, Krste Asanović, David Patterson (2018). An open-source out-of-order processor with resilient low-voltage operation in 28nm CMOS. Hot Chips 2018.

[8] Jonathan Balkind, Katie Lim, Fei Gao, Jinzheng Tu, David Wentzlaff, Michael Schaffner, Florian Zaruba, Luca Benini (2019). OpenPiton+Ariane: The First Open-Source, SMP Linux-booting RISC-V System Scaling From One to Many Cores. Third Workshop on Computer Architecture Research with RISC-V (CARRV). 2019.

[9] SiFive, Inc. https://github.com/sifive/freedom-u540-c000-bootloader. July 30, 2021.

[10] https://sourceforge.net/projects/lxde/files/. July 30, 2021.

[11] LI Tuo, ZOU Xiao-Feng, LIN Ning-Ya, ZHANG Lu, LIU Tong-Qiang, ZHOU Yu-Long, LI Ren-Gang. Design of FPGA Prototype of Server Management Controller Based on RISC-V (2021). Computer Systems Applications, 30(7), 136-141.