Versatile Computer System with RISCV and FPGA.

Tony Nguyen

*The University of Alabama in Huntsville*

*Huntsville, AL 35899*

mtn0006@uah.edu

***Abstract —* RISCV softcore processor in FPGA is a promising future for embedded system. Such platform provides a software-to-hardware configurable computer device that offers optimal functionality, flexibility, and versatility. This paper conducts an overall study of open-source RISCV in FPGA hardware. It focuses on the process of porting RISCV softcore to FPGA hardware and setting up a development environment for that platform. The paper conducts an experiment of setting up a computer device with SiFive Freedom RISCV softcore running on Altera MAX 10 T-Core FPGA board to demonstrate the theory concept.**

1. INTRODUCTION: RISC-V and FPGA

A basic computer system performance relies on its hardware and processor core. A computer processor defines how data is captured, moved, and manipulated based on its instruction set architecture (ISA). As the industry becomes more competitive, hardware implementation of ISA has become proprietary intellectual property (IP) of semiconductor companies. This motivates a fast-growing ISA known as RISC-V, an open-source ISA designed for scalability and versatility.

RISC-V, rooted in reduced instruction set computer (RISC), has adopted the embedded industry with software support for a wide range of hardware architecture. Its “openness” makes itself quickly becomes a common ISA standard in embedded systems that allows developers the freedom to adopt and customize the processor hardware implementation to fit the desirable design goal. Sharing the same technology trend is the rise of Field Programmable Gate Array (FPGA) – a software-defined integrated circuit (IC) designed for high performance and flexibility, offering hardware with customizability and reusability.

The blending of both technologies introduces a new platform of configurable processor core running on customizable integrated circuits. This paper studies the implementation of RISC-V processor core in FPGA hardware, making up an embedded computer device with functionality, flexibility, and stability. Such computer device allows designers to partition specific workloads within FPGA sub-circuits for better functionality while utilizing the minimal hardware resources. Also, configurable software and customizable hardware provide flexibility in the development process. Both hardware and the processor implementation can be modified in prototyping and maintaining stage. For example, different cache memory layouts (e.g., different cache level, block size, etc.) can be experimented to maximize memory performance. An appropriate number of peripheral (UARTs, SPI, I2C, etc.) can be decided based on different design goals. Such factors result in faster turnaround time while saving the cost for development and maintenance. In addition, not only the popularity of RISC-V and FPGA takes place in open-source developers, but it also attracts the attention of dominant tech companies such as AMD, Intel, etc. As more investment and development take its place, more toolchains and software are made available, guaranteeing a promising future of RISC-V in FPGA computing platform.

1. BACKGROUND
2. *Open-source RISCV softcore for FPGA.*

RISCV softcore and its compatibility of running on FPGA hardware is the main aspect of this study. Due to the variety of RISCV softcore, this paper focuses on two leading FPGA softcore for FPGA: VexRiscv and SiFive Freedom.

VexRiscv is an SpinalHDL implementation of RISCV softcore that is purposely designed for FPGA compatibility. This softcore is highly configurable with multiple variants of pipeline, cache, and peripherals. It allows selection of 2 to 5+ pipeline stages; options for instruction and data cache memory organization; configuration for interrupts and exception handling with different privilege modes; and Linux SoC and RTOS support. VexRiscv fully avoids using vendor specific IP block and primitives to provide optimal FPGA support. The implementation is to simplify the process of configuring and exporting Verilog file that can be later compiled in FPGA software.

SiFive Freedom is the implementation of customizable RISCV System-on-chip (SoC). Within its family, SiFive Freedom E300 is implemented to optimize for highly configurable-based platform such as FPGA. It is purposely for specific-focus performance while reduce development time and effort. SiFive Freedom offers a wider range of options for configuring the RISCV architecture compared to VexRiscv. It features security optimization, instruction customization, and low-power management. In addition, it provides full debugging capabilities as well as add-on acceleration options. However, not only SiFive softcore is purposely designed for to be compatible with software and toolchain that only support for Xilinx – the leading FPGA semiconductor company, but only the base implementation of its RISCV softcore is free for public user.

Overall, both VexRiscv and SiFive Freedom are great choice for FPGA hardware since they both offer a base processor functionality with common peripherals. Both implementation features exporting softcore in Verilog hardware description language, making it versatile to be integrated to FPGA implementation. In comparison, VexRiscv appears to be more appropriate at the intermediate level in which designers can adapt a free RISCV softcore with basic functions. SiFive, on the other hand, is for commercial level since it features more complex architectures and requires additional cost for add-on software and toolchain support. From the practical standpoint of this study in which the experiment is conducted on an Altera MAX 10 FPGA board, SiFive Freedom E300 is chosen since there is publicly available board support package. Fig. 1 shows the top-level block diagram of SiFive Freedom E300 architecture which contains a E31 coreplex, selections of I/O peripherals, memory management, clock generation, etc.

Diagram

Description automatically generated

Fig. 1. Top-Level Block Diagram of the E300 Platform

1. *Porting RISC-V Softcore to FPGA.*

The process of porting RISCV softcore to FPGA hardware can be described in the following diagram. The procedure can be divided into three main steps as shown in Figure. 2:

1. Compile and export RISC-V softcore with RISCV GNU toolchain.
2. Deploy the exported HDL implementation to FPGA hardware.
3. Develop software board support package (BSP).

In step 1, SiFive Freedom utilizes RISC-V GNU toolchain to compile and export the source code into two output files: HDL software implementation in a single Verilog file and a Memory Configuration File (MCS). The output files are then deployed to the MAX 10 T-Core FPGA board in step 2 using Intel Quartus Prime Lite software. Additional hardware design in FPGA fabrication such as top-level design for GPIO and peripherals is also be conducted within this step. The hardware is now configured with a RISCV softcore running in the FPGA fabric. The last step is to add the board support package as to support software-to-hardware communication. The C-program board support package specify entry-point address, memory map, and interrupt vector table to provide a fully functional C-embedded computer device.

Diagram

Description automatically generated

Fig. 2. Procedure diagram of porting RISCV to FPGA.

1. EXPERIMENT

This paper conducts an experiment of setting up a FPGA computer system running RISC-V softcore to explore the practicality and complexity of such procedure. The host machine is a Linux Ubuntu 20.04 that installs Intel Quartus Prime Lite, RISC-V GNU toolchain, OpenOCD, and other software packages that requires to support this experiment.

The experiment builds an RV32I variant of RISC-V which is a 32-bit base integer instruction set. The ISA supports five ISA extension standards:

* M – Extension for Integer Multiplication and Division
* A – Extension for Atomic Instructions
* F – Extension for Single-Precision Floating-Point
* D – Extension for Double-Precision Floating Point
* C – Extension for Compressed Instructions

The extension selection can be built at compile-time with RISC-V GNU toolchain support. The tool chain also allows to build and export the softcore implementation to a single Verilog HDL file. Since the content of the exported HDL file is unsorted and in the arbitrary order, this experiment utilizes a given E300 softcore called T-Core E300, provided by Altera in support running RISC-V in the MAX 10 T-Core FPGA board. The file is an organized and modified version of the exporting file for better readability. The T-Core E300 softcore HDL describes the implementation shown in Fig. 3.

A picture containing chart

Description automatically generated

MAKEFILE:

Define c-level program file

Load board support package (BSP)

Build with RISC-V GNU toolchain

with selected ISA and extension

Build with OpenOCD that allows

debugging through JTAG

Compile the software and upload to

Flash Memory.

Fig. 3. T-Core E300 RISCV softcore architecture

The hardware used in this experiment is an Altera MAX 10 T-Core FPGA development board. The board is a one-for-all solution for getting started with FPGA and RISCV development. It contains MAX10M50DAF484 processor, JTAG modules, USB-Blaster, and extensive of IO peripherals as shown in Fig. 4.

A picture containing text, electronics, circuit

Description automatically generated

Diagram

Description automatically generated

Fig. 4. Altera MAX-10 T-Core FPGA Development Board.

The T-Core is a particular Altera hardware design for RISC-V development. An external QPSI Flash designed to store the BSP for RISCV. The hardware also supports a selectable JTAG masked to the USB Blaster that allows developer to select the end point of host-to-device to FPGA processor or external QSPI Flash. That is, user can deploy the RISCV softcore HDL implementation to the MAX10 FPGA processor and upload the c-program board support package to the external Flash using one single host-to-device connection. In addition, the software supports utilizes multiple toolchains (CMake, RISCV GNU, OpenOCD)to provide a convenience way to compile and upload the C-program BSP. Figure. 6 shows the pseudo content of the Makefile that allows compiling and loading the software easily.

Fig. 6. Makefile pseudo code to compile and upload BSP code.

A simple IO program toggles 4 LEDs is implemented using the BSP to verify the functionality of the RISCV FPGA computer. The program is compiled with different variants of RV32I extension to verify the behaviour. While [M][A][F][C] extensions are successfully built and executed, the RISC-V GNU toolchain issues incorrect memory-file format with the [D] Double-precision Floating Point extension.

1. RESULT AND ANALYSIS

The experiment successfully demonstrates a c-embedded software development environment in a RISC-V FPGA computer device. Although the experiment demonstrated a full hardware setup and verified the software-to-hardware functionality with the blinking LED demo, the software support is very limited. Software development of BSP is outdated and not maintained. Also, portability is board-specific i.e., hardware support might not be the same in other FPGA boards, and BSP is board-dependent. Overall, the process of porting RISC-V softcore to the FPGA hardware appears to be more difficult than expected. Although available software toolchain allows compiling the softcore into a single HDL file, the exported file is almost meaningless without modification to fit the target hardware. In addition, a board support package is required to interact between software and hardware. The BSP implementation can be very complex and demands decent level of c-programming. Therefore, board-specific support provided by the manufacture plays a key role in this process. It reduces majority of low-level software development and provides a starting point for running RISC-V in FPGA.

From a development standpoint, since the experiment utilizes the software supports from the vendor, it is a board-specific workflow. For example, adding cache memory to the RV32I requires modification in register transfer level and software board support package. This adjustment requires major effort and adds time to development process. From a commercial standpoint, both Altera and Xilinx provide extensive software and hardware supports for RISC-V in FPGA computer system that fits customers’ specification. For example, SiFive provides a variety of customable RISC-V E-Series that has comparable performance with ARM M0 to A55. In conclusion, this paper conducts an overall study of RISC-V in FPGA. It gives an overview and analysis of the procedure porting the softcore to the target hardware in practice. The study is a useful material for future research of RISC-V in FPGA – A emerging technology that will play a key role in future computing.

References

1. A. Cilardo, "Memory Encryption Support for an FPGA-based RISC-V Implementation," 2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), Montpellier, France, 2021, pp. 1-5, doi: 10.1109/DTIS53253.2021.9505064.
2. G. Zhang, K. Zhao, B. Wu, Y. Sun, L. Sun and F. Liang, "A RISC-V based hardware accelerator designed for Yolo object detection system," 2019 IEEE International Conference of Intelligent Applied Systems on Engineering (ICIASE), Fuzhou, China, 2019, pp. 9-11, doi: 10.1109/ICIASE45644.2019.9074051.
3. B. W. Mezger, D. A. Santos, L. Dilillo, C. A. Zeferino and D. R. Melo, "A Survey of the RISC-V Architecture Software Support," in IEEE Access, vol. 10, pp. 51394-51411, 2022, doi: 10.1109/ACCESS.2022.3174125.
4. R. Höller, D. Haselberger, D. Ballek, P. Rössler, M. Krapfenbauer and M. Linauer, "Open-Source RISC-V Processor IP Cores for FPGAs — Overview and Evaluation," 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2019, pp. 1-6, doi: 10.1109/MECO.2019.8760205.
5. A. Kamaleldin, S. Hesham and D. Göhringer, "Towards a Modular RISC-V Based Many-Core Architecture for FPGA Accelerators," in IEEE Access, vol. 8, pp. 148812-148826, 2020, doi: 10.1109/ACCESS.2020.3015706.
6. J. Peyron, “Running a RISC-V processor on the arty A7,” Running a RISC-V Processor on the Arty A7 - Digilent Reference. [Online]. Available: https://digilent.com/reference/programmable-logic/arty-a7/arty\_a7\_100\_risc\_v/start. [Accessed: 08-Mar-2023].
7. L. Poli, S. Saha, X. Zhai and K. D. Mcdonald-Maier, "Design and Implementation of a RISC V Processor on FPGA," 2021 17th International Conference on Mobility, Sensing and Networking (MSN), Exeter, United Kingdom, 2021, pp. 161-166, doi: 10.1109/MSN53354.2021.00037.