SOC RISC-V Processor with UART Integration

System-on-Chip Implementation and Verification

Ahmad Mukhtar

National University of Sciences and Technology (NUST)
Chip Design Centre (NCDC), Islamabad, Pakistan

September 28, 2024

Abstract

This report presents the design, implementation, and verification of a System-on-Chip (SoC) that integrates a RISC-V processor with a UART transmitter. The goal was to create a custom SoC that could transmit data through UART whenever the ALU result matches a specific condition. In this project, we have successfully integrated the RISC-V processor core and a UART transmitter module to enable data transmission and status feedback. Key features of the SoC, the methodology, test results, and analysis are provided in this report.

Contents

Chapter 1

Introduction

1.1 Overview

The purpose of this project was to integrate a RISC-V processor with a UART transmitter to create a functional System-on-Chip (SoC). The goal was to enable the processor to interact with a peripheral—specifically, a UART module—using standard load (‘lw‘) and store (‘sw‘) instructions. The system was designed to transmit the least significant byte (LSB) of the value stored in register ‘x6‘ whenever a specific memory-mapped address was accessed.

1.2 Objectives

The main objectives of this project are as follows: • Design and implement a System-on-Chip (SoC) that integrates a RISC-V processor with a UART transmitter.

• Enable the processor to communicate with the UART peripheral using standard RISC-V load (‘lw‘) and store (‘sw‘) instructions.

• Trigger data transmission over UART based on specific ALU output (‘ALU_OUT‘) conditions.

• Monitor the UART transmitter’s status and store this status in register ‘x7‘.

Chapter 2

Methodology

2.1 System Architecture

The SoC is composed of two primary modules:

1. **RISC-V Processor Module** - This module contains the main data path, instruction memory, register file, and ALU. It is responsible for executing instructions, managing registers, and generating output data for the UART module.

2. **UART Transmitter Module** - This module transmits the data provided by the processor over a serial interface. The status of the UART is reported back to the processor, which is stored in the register ‘x7‘.

2.2 Implementation

2.2.1 RISC-V Processor Integration

The processor was programmed to monitor specific addresses. When a store (‘sw‘) instruction targets address ‘1600‘, the value in register ‘x6‘ is sent to the UART transmitter module for transmission. Similarly, when a load (‘lw‘) instruction is issued with address ‘1604‘, the UART’s current status is read back into the ‘x7‘ register.

2.2.2 UART Transmission Control

The UART transmitter module is activated whenever the ALU output (‘ALU_OUT‘) matches ‘1600‘, and the ‘signal‘ from the processor indicates that the UART should begin transmission. The transmitter then sends the LSB of the data from register ‘x6‘. The status of the UART (busy or ready) is stored in register ‘x7‘.

2.2.3 Control Flow

The main control logic is implemented as follows:
• When the ALU generates ‘1600‘ as the output, the ‘load_signal‘ is set high, and the UART is loaded with the data from register ‘x6.

• The UART status is monitored and reflected in the ‘status‘ logic, which indicates if the UART is free (‘1‘) or busy (‘0‘).

Chapter 3

Results and Analysis

3.1 Successful UART Transmission

The following figure shows the successful transmission of data through the UART transmitter:

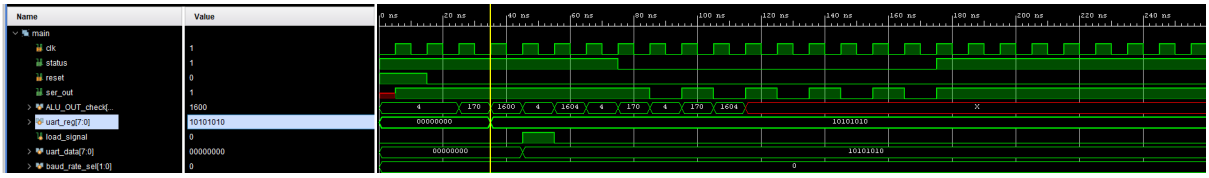

Figure 3.1: Successful Transmission through UART

- This waveform illustrates that whenever the `ALU_OUT` reached `1600`, the UART transmitted the value from register `x6`.
- The `load_signal` was correctly asserted when the condition was met.
- The `ser_out` signal toggled as expected during the transmission, indicating successful UART operation.

3.2 Data and Status Monitoring

The following figure shows the status of the UART and the corresponding updates in the processor registers:

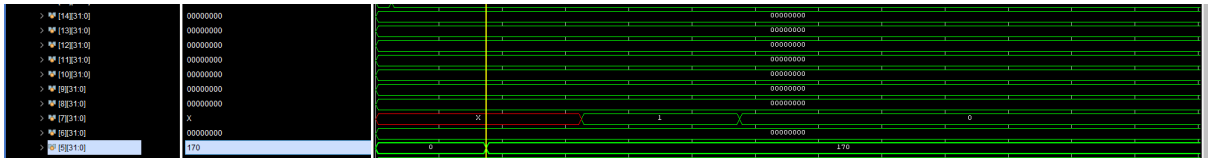

Figure 3.2: Register Values: Transmitted Data in x6 and UART Status in x7

The above waveform confirms the following:

- Register ‘x6‘ contained the data (‘0xAA‘ or ‘170‘ in decimal) to be transmitted.

- Register ‘x7‘ correctly reflected the status of the UART: ‘0‘ when busy and ‘1‘ when ready for the next transmission.

Chapter 4

Conclusion

4.1 Summary

This project successfully demonstrated the integration of a RISC-V processor and a UART transmitter to form a System-on-Chip (SoC). The integration allowed for effective communication between the processor and the UART peripheral using standard RISC-V instructions. Data transmission via UART and peripheral status feedback were verified, as shown in the results.

4.2 Challenges and Future Work

During the development, one of the major challenges was ensuring proper timing between the ALU operations and UART state updates. Future work will involve extending this SoC to include more peripherals and incorporating an interrupt-based mechanism for UART communication to further improve system efficiency.

Chapter 5

References

- Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann.
- Datasheets and technical manuals of the UART IP used in the design.