Microprocessor Systems

Project Report

RISCV SoC Design - GPIO

**Group Members:**

* Moonis Amir Zaman (365753)
* Muhammad Ahmad Saleem (383118)
* Muhammad Bilal Khan (380046)

**Contents:**

* Introduction
* Block Diagram Of Proposed Design
* Simulation Results
* Features Implemented
* Conclusion

**Introduction**

RISC-V is an open and free instruction set architecture (ISA) that can be implemented in various hardware platforms. A system-on-chip (SoC) is a single integrated circuit that contains a processor core and other peripherals such as memory, input/output devices, and accelerators. Designing a RISC-V SoC involves choosing a suitable RISC-V core, selecting the appropriate peripherals and buses, and integrating them into a coherent system. In this report, we present our design of a RISC-V SoC with a general-purpose input/output (GPIO) controller as one of the peripherals. We use Verilog coding in ModelSim to simulate the processor and the GPIO controller, and verify their functionality and performance.

**Block Diagram Of Proposed Design**

A picture containing diagram, technical drawing, plan, schematic

Description automatically generated

* The Datapath block abstracts some of the components of the processor, such as ALU, Register File, Data Memory etc.
* The processor consists of five stages: instruction fetch (IF), instruction decode (ID), execute (EX), memory access (MEM), and write back (WB).
* The IF stage fetches the instruction from the instruction memory using the program counter (PC) and increments the PC by 4 bytes. The fetched instruction is stored in the instruction register (IR) and passed to the ID stage.
* The ID stage decodes the instruction and reads the operands from the register file using the source register addresses (rs1 and rs2). The decoded instruction is also passed to the control unit, which generates the control signals for the rest of the processor.
* The EX stage performs arithmetic or logic operation on the operands using the ALU. The result of the operation is stored in the ALU output register and passed to the MEM stage. The EX stage also computes the branch target address by adding the sign-extended immediate value to the PC.
* The MEM stage accesses the data memory using the ALU output as the address for load and store instructions. The data read from or written to the memory is stored in the memory data register and passed to the WB stage. The MEM stage also checks for branch conditions and updates the PC accordingly.
* The WB stage writes the result of the instruction to the register file using the destination register address (Rd).

**Simulation Results**

The experimental setup was as follows:

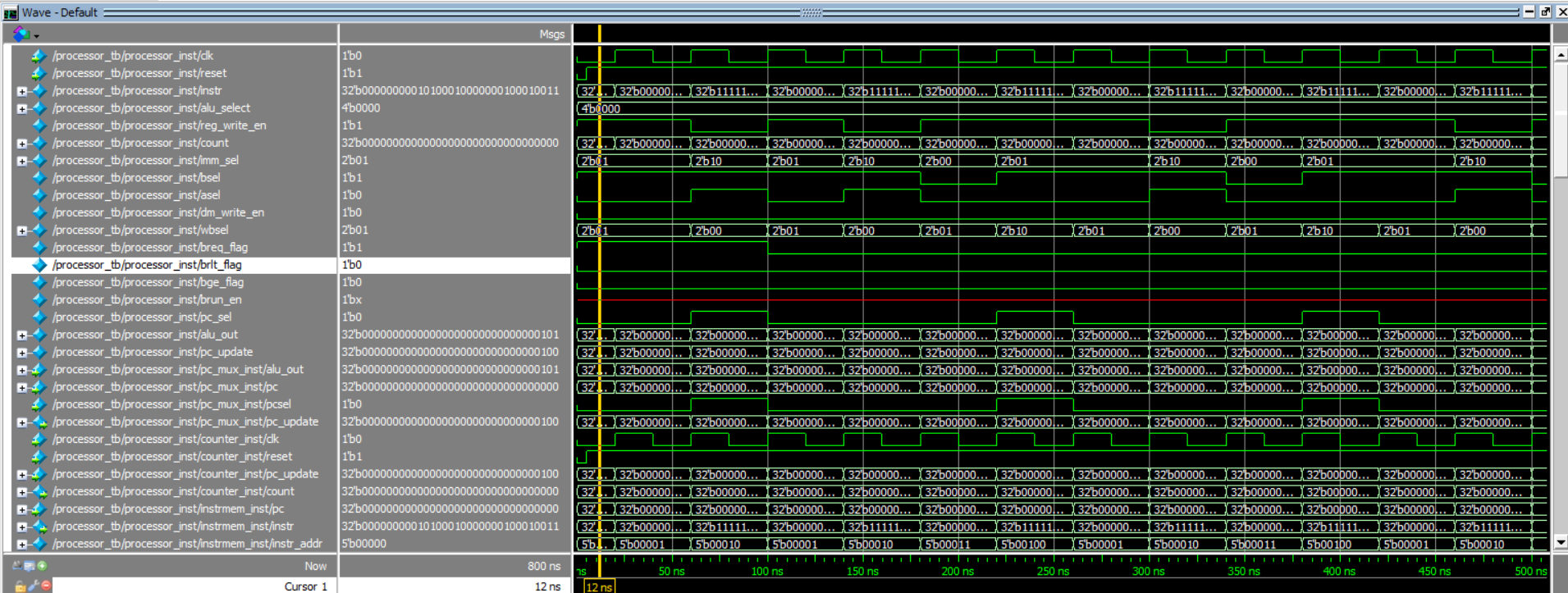
* Testing each module in separate test benches
* Instantiating all modules in single test bench, to create complete processor
* Test Program that uses various instruction formats
* Waveform Simulation in ModelSim

The test program used was:

A picture containing font, text, screenshot, number

Description automatically generated

The waveform produced was as follows:



*The program executed successfully*

**Features Implemented**

* Register, Immediate, Store, Branch, and Jump And Link (JAL) format instructions
* Ability to directly read a program file (binary code)

**Conclusion**

In this project, we have designed and implemented a RISC-V SoC with a GPIO controller and integrated it with a RISC-V processor (RV32I) using Verilog in ModelSim. We have followed the RISC-V SoC design flow as learned in the course. We have verified the functionality of our design by running various test programs to ensure each instruction executes successfully. Our design demonstrates the advantages of using RISC-V as an open-source and customizable ISA for SoC development. We have also learned the basic concepts of RISC-V ISA, Verilog HDL, ModelSim simulation tool, and GPIO controller design.