University of Burgundy

Real-Time Imaging Project

Kimberly Grace Sevillano Colina, Zhilling Qian, Yanyin Yao,
Abdullahi Atanda

Contents

1 Introduction

1.1 Background

Field-Programmable Gate Arrays (FPGAs) are versatile and powerful tools in the field of digital design and embedded systems. They offer the flexibility to program and reprogram the hardware for a wide range of applications. Very High-Speed Integrated Circuit Hardware Description Language (VHDL) is a key language used for describing the behavior and structure of electronic systems, particularly in FPGA development. The combination of FPGAs and VHDL allows for significant experimentation and innovation in digital design.

In this project, the Nexys4 DDR FPGA board is utilized as the primary development platform. This board is widely recognized in educational and prototyping environments for its robust feature set and Xilinx Artix-7 FPGA, making it an ideal choice for implementing and testing digital designs.

1.2 Project Significance

Understanding and applying VHDL in microprocessor design is crucial for anyone entering the field of digital electronics and embedded systems. This project not only provides hands-on experience with VHDL but also offers insight into the fundamental concepts of CPU architecture and design. It bridges the gap between theoretical knowledge and practical application, using the Nexys4 DDR board as a testbed.

1.3 Objectives

The primary objectives of this project include:

- Designing a simple microprocessor using VHDL, with implementation on the Nexys4 DDR FPGA board.
- Implementing core functionalities such as an accumulator, program counter, and basic instructions (LDA, STA, ADD, etc.).
- Exploring potential extensions and applications of the processor design within the constraints of the Nexys4 DDR platform.
- Addressing the challenges encountered in FPGA-based microprocessor design and proposing viable solutions.

2 Materials and Methods

2.1 Nexys4 DDR FPGA Board

The Nexys4 DDR board is a pivotal component in this project. It's a comprehensive digital circuit development platform based on the Xilinx Artix-7™ FPGA, known for its high capacity and performance. The board's extensive external memories, variety of ports, and built-in peripherals like sensors and a speaker amplifier, make it suitable for a wide range of digital designs.

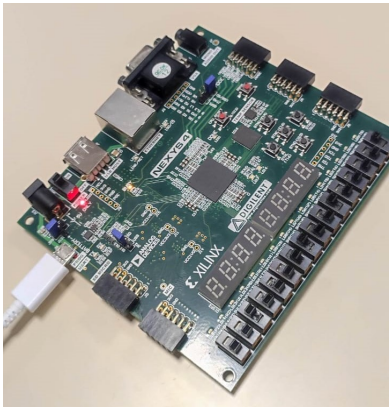

Figure 1: Nexys4 DDR board.

2.2 VHDL Development Environment

The VHDL development environment was set up to design and simulate the microprocessor. This included configuring the necessary software tools and preparing the development workflow for efficient design and testing.

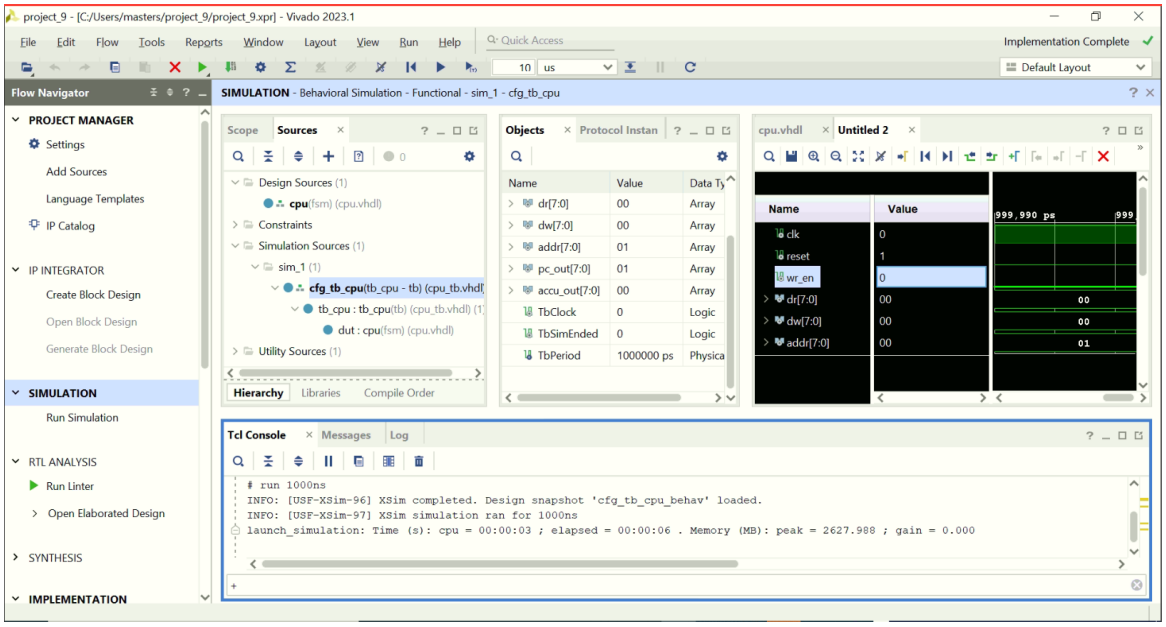

Figure 2: Integrated development and simulation environment used for the CPU design.

2.3 Initial CPU Design Approach

The initial approach to designing the CPU involved defining the basic structure and functionality in VHDL. This included setting up the entity and architecture of the CPU, focusing on the interface, op-codes, and the finite state machine (FSM) logic. The specifics of the architecture and implementation are detailed in Section 3.

3 Designing a Simple Microprocessor in VHDL

3.1 Overview of the Microprocessor Design

The design of the microprocessor aimed to create a simple yet functional CPU using VHDL. The focus was on achieving basic computational capabilities, suitable for educational and prototyping purposes. The design process involved defining the architecture, implementing the instruction set, and ensuring compatibility with the Nexys4 DDR FPGA board.

3.2 CPU Architecture

The architecture of the CPU was designed to be straightforward yet capable of performing fundamental operations. It includes an accumulator for arithmetic operations, a program counter for instruction sequencing, and a memory interface for data storage and retrieval.

3.2.1 Accumulator and Program Counter

The accumulator is an 8-bit register used for arithmetic and logic operations. It temporarily stores the results of these operations. The program counter (PC) is another crucial 8-bit register that keeps track of the memory address of the next instruction to be executed. The PC ensures the sequential execution of instructions, incrementing after each operation unless altered by specific instructions like jumps.

3.2.2 Memory Interface and Internal Registers

The CPU interfaces with memory through an 8-bit address bus and separate 8-bit input and output data buses. This design allows for efficient data transfer between the CPU and memory. Internal registers, including the accumulator and program counter, facilitate the CPU's operations, holding data and addresses relevant to the current instruction.

3.2.3 Implementation of Basic Instructions

The CPU supports a basic set of instructions, each encoded with a unique op-code:

- **LDA (Load Accumulator)**: Loads data from a specified memory address into the accumulator.
- **STA (Store Accumulator)**: Stores the content of the accumulator at a specified memory address.
- **ADD**: Adds data from a specified memory address to the accumulator.
- **JNC (Jump if No Carry)**: Jumps to a specified address if the carry flag is not set.
- **JMP (Jump)**: Unconditionally jumps to a specified address.

These instructions form the basis of the CPU's functionality, enabling it to perform basic data processing tasks.

The VHDL implementation of the CPU reflects this architecture. The finite state machine (FSM) within the VHDL code controls the operation of the CPU, transitioning between states based on the current instruction and its operands. This FSM approach allows for clear and manageable CPU operation sequencing.

3.3 Testing and Validation

The CPU design was rigorously tested using simulation tools and test benches. These tests ensured that each instruction was executed correctly and that the CPU behaved as expected in various scenarios.

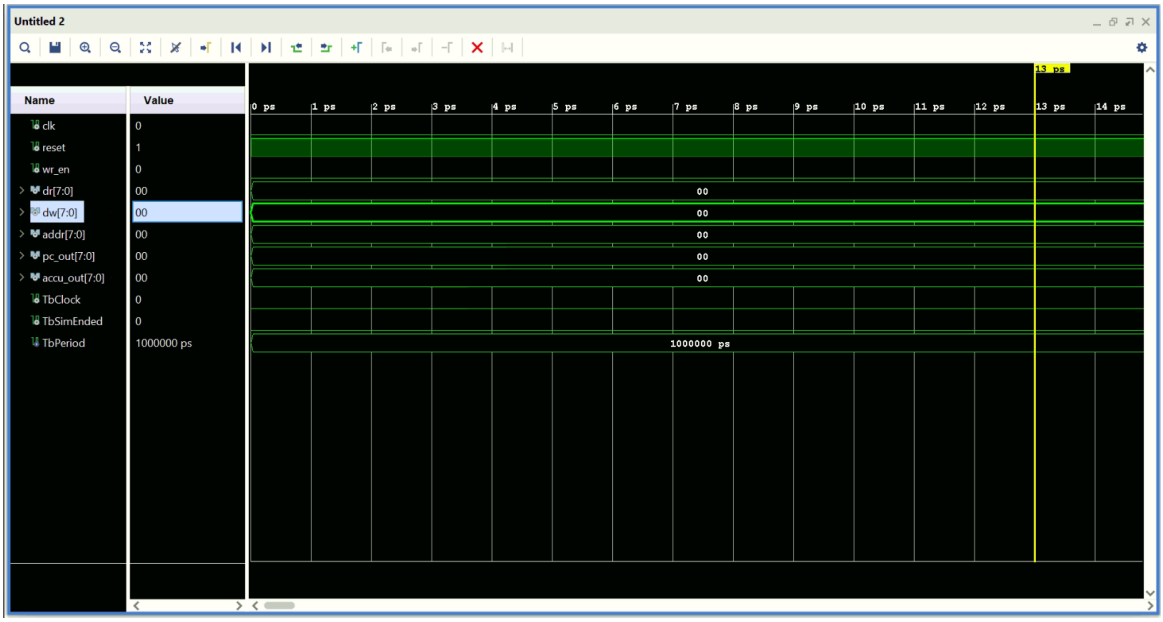

Figure 3: Waveform simulation of the basic CPU design showing the timing of various signals during operation.

3.4 CPU Design

3.4.1 CPU Top-Level Entity (cpu_top)

- Ports

clk (Input): System clock, synchronizes all CPU operations.

reset (Input): Resets the CPU, establishing a known initial state.

seg (Output): Drives a 7-segment display to show data. an (Output): Activates certain parts of the 7-segment display.

led (Output): LED for general visual indication. led_inicial (Output): Additional LED for specific purposes.

dr (Input): Represents data read from memory. op_code_out (Output): Displays the operation code executed by the CPU.

Internal Components

CPU Component: The processing core that implements CPU operation logic.

Internal Registers: Includes wr_en, dw, addr, pc_out, accu_out, etc., handling various aspects of CPU processing and control.

Display Processing (display_controller) Controls what and how information is displayed on the 7-segment display.

3.4.2 CPU Entity (cpu)

Op-Codes LDA (Load Accumulator): Loads a value into the accumulator.

STA (Store Accumulator): Stores the value of the accumulator in a memory address.

ADD (Add): Adds a memory value to the accumulator.

JNC (Jump if No Carry): Jumps to an address if there is no carry.

JMP (Jump): Unconditionally jumps to an address.

SUB (Subtract): Subtracts a memory value from the accumulator.

PRNG (Pseudo Random Number Generator): Generates a pseudo-random number.

Registers and Signals

Accumulator (accu): Stores the result of operations. Program Counter (pc): Indicates the current position in the program.

Address (addr): Specifies a memory address for operations.

Data Read (dr): Represents data read from memory.

Data Write (dw): Represents data to be written to memory.

Processes clk_divider: Divides the system clock for various operations.

anode_control: Controls the display on the 7-segment display.

seven_seg_decoder: Decodes digits for display.

fsm_proc: Implements the finite state machine for CPU operation control.

XDC File (Constraints) Defines the physical pin mapping in the FPGA for all signals used in the design.

Includes configurations for clk, seg, an, led, led_inicial, dr, op_code_out, etc., specifying their corresponding pins and voltage standards.

This design presents a basic CPU capable of performing simple operations such as loading, storing, adding, subtracting, and conditional or unconditional jumps, with a visual interface through a 7-segment display and LEDs.

4 Project Management and Author Contribution

Each team member's role was integral to the project, combining technical skills, research, strategic planning, and project management to achieve the successful design and implementation of the microprocessor.