|  |
| --- |
|  |

Jordan National Semiconductor Design Competition (JOSDC’2023)

Project Title

By

Student Name

Amman, Jordan

Month Year

Acknowledgments

Recognition or favorable notice for people.

Abstract

Describe your project briefly in few paragraphs. The abstract should not exceed one page.

**Table of Contents**

[1 Introduction 7](#_Toc146783944)

[1.1 Section 7](#_Toc146783945)

[2 Design 8](#_Toc146783946)

[2.1 Hardware Design and Implementation 8](#_Toc146783947)

[2.2 Coding and Software Development 8](#_Toc146783948)

[3 Results 9](#_Toc146783949)

[3.1 Experiment/Simulation Results Discussion: 9](#_Toc146783950)

[3.2 Prototype Setup 9](#_Toc146783951)

[3.3 Validation of requirements 9](#_Toc146783952)

[4 Conclusion 10](#_Toc146783953)

**List of Figures**

Figure 1.1 Figure one in chapter one ..……………………………….……………..page number

Figure 2.1 Figure one in chapter two ..……………………………….……………..page number

Figure 2.2 Figure two in chapter two ..……………………………….……………..page number

**List of Tables**

Table 1.1 Table one in chapter one ..……………………………….……………..page number

Table 2.1 Table one in chapter two ..……………………………….……………..page number

Table 2.2 Table two in chapter two ..……………………………….……………..page number

# Introduction

## Section

This should include:-

* Background of the project. (Motivation e.g. why is the design important)
* Objectives of the project and some description, including the design requirements
* Design achieved (brief description)
* Indicate who exactly in the group is responsible for what.
* Organization of the rest of the documentation.

# Design

**2.1** **Counter**

The Program Counter (PC) stands as an indispensable component within the heart of a computer's Central Processing Unit (CPU). Its primary function is to meticulously track the memory address of the subsequent instruction to be retrieved during the instruction fetch phase of the CPU's operational cycle. This report delves into the Hardware Description Language (HDL) implementation of a simplified Program Counter using Verilog.

Code Overview

The Program Counter module is meticulously designed to integrate seamlessly within a digital circuit. It presents the following inputs and outputs:

Clock Input (clk): This input signal is intrinsically synchronized with the system clock, serving as the bedrock for the sequential logic operations within the module.

Reset Signal (reset): This input is instrumental in the ability to reset the program counter to a pre-defined initial value. Notably, it is an active-high signal, implying that the reset operation takes effect when the reset signal is in a high state.

Program Counter Output (PC): The PC is an 8-bit output that duly represents the current address held by the program counter.

Furthermore, the module incorporates an 8-bit register known as "initial\_pc," serving as the repository for the program counter's initial value. This 8-bit design corresponds with the utilization of M9K memory technology, commonly prevalent in FPGA (Field-Programmable Gate Array) devices, renowned for their expeditious and versatile storage capabilities.

Memory Configuration

Within the realm of our Random Access Memory (RAM), each individual instruction occupies a dedicated memory location, with each of these locations accommodating 32 bits of data, which equates to 4 bytes. This architectural choice underscores the RAM's function as a dual-purpose repository for both instructions and data in support of our processing unit.

DE10-Lite board which we used is designed with a well-structured configuration, consisting of approximately 200 distinct blocks. Each of these units provides us with a storage capacity of 8,192 bytes, which is equivalent to 8 kilobytes. These discrete memory segments play a crucial role in enabling efficient data organization and addressing within our system. Furthermore, our board is equipped with a 50 MHz clock that connects to all components in the processor, allowing instructions to execute sequentially.

We have thoughtfully integrated Read-Only Memory (ROM) into our system to augment our memory infrastructure. This ROM comprises 256 memory locations, each housing 32 bits, which translates to 4 bytes of unalterable data. The role of ROM is to diligently safeguard essential, non-volatile information, including firmware and unchanging data, pivotal for the seamless operation of our system.

Initial Value

The "initial\_pc" register performs the pivotal role of preserving the initial value of the program counter. This value is judiciously set as 8'b00000000, signifying that the program counter commences its journey at memory address 0. Notably, this initial value can be personalized by adjusting the value attributed to "initial\_pc."

Program Counter Update Logic

The crux of the program counter's functionality resides in an "always" block, highly attuned to the rising edge of both the clock signal (posedge clk) and the reset signal (posedge reset). This meticulous design ensures that the program counter receives updates on each clock cycle and is also capable of asynchronous resets.

Reset Condition

In instances where a reset condition is invoked (reset is in a high state), the program counter promptly reverts to its initial value, which is stored in "initial\_pc." This guarantees that the program counter faithfully returns to its predefined starting point upon activation of the reset signal.

Incrementing the Program Counter

In the absence of a reset signal (when reset is low), the program counter elegantly increments by 1. This increment operation ensures that the program counter gracefully advances to the next memory address with each passing clock cycle. The expression "pc <= pc + 1" is instrumental in executing this operation.

Functionality and Behavior

The operational behavior of the program counter is delineated as follows:

When the reset signal is asserted (in a high state), the program counter promptly resets to the value preserved in "initial\_pc."

During normal operation, where the reset signal remains unasserted (low), the program counter incrementally advances by 1 on each clock cycle. This operation enables the sequential fetching of instructions from memory, culminating in the orchestration of the CPU's harmonious performance.

Usage

This Program Counter module exhibits remarkable versatility and can be seamlessly incorporated into an array of CPU components, facilitating the retrieval of instructions from memory. Furthermore, it is a quintessential element within a comprehensive CPU design, where the orchestration of instruction fetching and execution transpires.

In conclusion, the Program Counter is not merely a digital entity; it is the symphony conductor within the CPU, orchestrating perfect timing and synchronization, ensuring the harmonious execution of instructions, and returning to the first note when the reset button is played. It represents the elegant and precise execution of digital computing.

# Results

• Present the results of testing and validation procedures for both the simulation and hardware.

• Include data, graphs, and tables to support your findings.

• Discuss the performance and functionality of the integrated system.

• Use a table to summarize that requirements were met

## Experiment/Simulation Results Discussion:

* + Use
    - Tables
    - Graphs
    - Waveform
    - figures

## Prototype Setup

* + Hardware
  + Software

## Validation of requirements

* + Discuss and analyze whether the requirements are met

# Conclusion

* Summarize the key findings and their implications.
* Assess whether the project's objectives were achieved.
* Provide recommendations for future work.

References

Follow a format consistent with IEEE guidelines and utilize conference and journal papers for your reference list

APPENDICES

These are detailed documentation of points mentioned in the report (e.g. technical data, questionnaires, chart …. etc.) which are considered supplementary information but too long or not quite relevant enough to include in the main body of the report.

Appendices may be labeled with letters as Appendix A, Appendix B, and so on.

Example,

Appendix A: CODE