Microelectronic Systems

DLX Microprocessor: Design & Development

Final Project Report

Master degree in Computer Engineering

Master degree in Electronics Engineering

Referents: Prof. Mariagrazia Graziano, Giovanna Turvani

Authors: group_16

Battilana Matteo, La Greca Salvatore Gabriele, Pollo Giovanni

June 29, 2021

Feature

- Frequency - Slack - Area - Ecc

Grandes nacelles :

- Nacelle A318 PW
- Inverseur A320 CFM
- Inverseur A340 CFM
- Nacelle A340 TRENT
- Inverseur A330 TRENT
- Nacelles A380 TRENT900
- Nacelles A380 GP7200

Petites nacelles :

- Nacelle SAAB2000
- Inverseur DC8
- Inverseur CF34-8
- Inverseur BR710
- Nacelle F7X

Contents

Listings

CHAPTER 1

Introduction

1.1 Abstract

1.2 Workflow

- Workflow used - git / github / pair programming

CHAPTER 2

Hardware Architecture

2.1 Overview

2.2 Pipeline Stages

2.3 Control Unit

2.4 Memories Interface

2.5 Instruction Set

CHAPTER 3

Fetch Stage

3.1 Instruction Register

3.2 Program Counter

3.3 Jump and Branch Management

CHAPTER 4

Decode Stage

4.1 Instruction Decode

4.2 Register File and Windowing

4.3 Hazard Control

4.4 Comparator

- Unsigned things

4.5 Jump and Branch decision

4.6 Next Program Counter computation

CHAPTER 5

Execute Stage

5.1 ALU: Arithmetic Logic Unit

5.1.1 Adder

5.1.2 Multiplier

5.1.3 Logic Operands

The basic and most simple implementation of a logic unit is based on single logic gates on N bits whose outputs are muxed, in order to generate the correct output. The problem with this solution is that the number of input signals to the multiplexer is extremely high; this implementation does not only suffer from the point of view of the delay but, since each logic function is implemented with a specific gate, the total area is huge.

In order to overcome the problems highlighted before, a more compact implementation has been chosen: the T2 logic unit.

This logic unit allows to perform AND, NAND, OR, NOR, XOR and XNOR using only 5 NAND gates, on two levels, and 4 selection signals. The schematic is the one in figure 5.1.

In order to compute one of the logical instructions, the select signals are properly activated as follow:

S_0	S_1	S_2	S_3	operation
0	0	0	1	AND
1	1	1	0	$NAND$
0	1	1	1	OR
1	0	0	0	NOR
0	1	1	0	XOR
1	0	0	1	$NXOR$

For example, in order to generate the AND logical operation, we have to select $S_3 = 1$, so that $out = R_1 \cdot R_2$; on the other hand, if we need NAND $S_0 = S_1 = S_2 = 1$ and $S_3 = 0$, so that $out = \overline{R_1} \cdot \overline{R_2} + \overline{R_1} \cdot R_2 + R_1 \cdot \overline{R_2} = \overline{R_1} \cdot \overline{R_2}$ that using the De Morgan law $out = \overline{R_1 \cdot R_2}$. This allows to obtain the best performances also because all paths work in parallel, compacting the area and the delay.

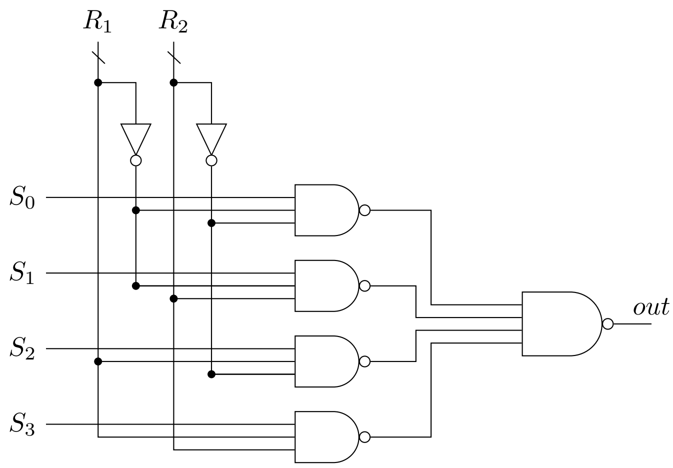

Figure 5.1: Logic unit

5.1.4 Shifting

5.2 Set-Like Operations unit

- setcmp

CHAPTER 6

Memory Stage

6.1 Load-Store Unit

- Unsigned things

6.2 Address Mask Unit

CHAPTER 7

Write Back Stage

Mux selects from Memory Output (LoadStore Unit) or ALU output.

Signal to enable register file write. Registers to delay the write register address

CHAPTER 8

Testing and Verification

8.1 Test Benches

8.2 Simulation

8.3 Post Synthesis Simulation

CHAPTER 9

Physical Design

9.1 Synthesis

9.2 Place and Route

CHAPTER 10

Conclusions