Politecnico di Torino

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

July 1, 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

5.1.4 Shifting

The implemented shifter allows to perform shift right, logical/arithmetical shift left and left/right rotate using the full operand `A` on 32 bits and 6 bits from the second one `B` and three *control signals*. Differently from the T2 version, it uses and addition signal in order to be able to manage also the rotate instruction. Our implementation takes three inputs:

- `A`: the operand to be shifted/rotated;
- `B`: only the 5 LSB [4,3,2,1,0] are used to select first the mask to be used and then the starting point from that mask;
- `SEL`: it encodes the operation type; the second bit is used to select among arithmetic and logic, the third bit is used to select the direction of the shift/rotate (left/right) and the first one is used only if the operation is a rotate. This is the encoding:

`SEL`	**Operation**
000	Shift logic right
001	Shift logic left
010	Shift arith right
011	Shift arith left
100	Rotate right
101	Shift right

The unit perform the requested operation in three stages:

1. The first consist in preparing 4 possible "masks", each already shifted of 0, 8, 16, 32 left or right depending on the configuration. This allows to shift for all 32 bits. Basically it copies the input `A` into the 4 masks that will be used by the next stage. Being in 32 bits, the generated masks are in $32 + 8 = 40$ bits. The only different between this implementation and the T2 one, is that,

in case of rotate, the additional 8 bits of the masks are filled with the corresponding 8 bits that are going "out" during the rotation.

2. The second level perform a coarse grain shift, that is basically consist on selecting one mask among the 4 possible masks generated in the previous stage. This selection is done by using the bits 4, 3 of `B`.

3. The third level, using the bits 2, 1, 0 of `B` and the selected mask, preform a fine grain refinement. The 3 bits allows to select the starting index from the mask, in fact it allows to select among 8 positions.

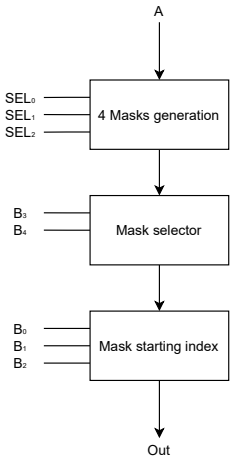

Figure 5.1: Schematic of the Shifter/Rotate Unit

Examples
For example, if we need to perform a left left of 9 bits `A`, where `A=18`, the corresponding `B` value will be 1001; this means that the second masks will be taken and the output result will from the bit at position $40 - 1 = 39$ to the one at $39 - 32 = 7$ included.

$$\text{MASK 2: } 0\underbrace{\mathbf{0000000\ 00000000\ 00000000\ 00010010\ 0}}_{\text{shifted A}}0000000$$

On the other hand, if we need to perform a right shift the masks are generated in the opposite way, so the zeros are put in the MSB of the mask, shifted by 0, 8 ... positions. In this case we need also to distinguish between the an arithmetic and a logic shift; in the first case, instead of filling the "empty" bits with zero, the operand sign is used. For example, if we want to shift `A=-18` of `B=3` bits, the first mask is used:

$$\text{MASK 1: } 11111\underbrace{\mathbf{111\ 11111111\ 11111111\ 11111111\ 11101}}_{\text{shifted A}}110$$

In the last case, let's suppose to rotate right `A=1255` (=10011100111) by 5 position:

$$\text{MASK 1: } 11\underbrace{\mathbf{100111\ 00000000\ 00000000\ 0000100\ 111}}_{\text{rotated A}}00111$$

As you can see, in case of MASK 1 for the right rotation, the 8 LSB of `A` are copied into the 8 MSB of the mask.

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