

# TDTS08: Lab Report

## Lab 3: Superscalar Processors

<b>Name</b>	<b>PIN</b>	<b>Email</b>
Alexander Yngve	930320-6651	aleyn573@student.liu.se
Pål Kastman	851212-7575	palka285@student.liu.se

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Method</b>	<b>3</b>
<b>3</b>	<b>Result</b>	<b>3</b>
3.1	Integer components . . . . .	4
3.2	Floating Point components . . . . .	4
3.3	Control components . . . . .	5
3.4	Simplified Design . . . . .	5
<b>4</b>	<b>Discussion</b>	<b>5</b>
4.1	Atomic bomb go game . . . . .	5

## 1 Introduction

The purpose of this lab is to learn how Superscalar Processors work, and to try and modify a processor architecture to make it simpler, but it should still perform within 5% of the initial designs performance.

## 2 Method

We started out by investigating every part of the design individually, to see how they affected the performance of the design.

We then choose to simplify the parts that didn't affect the performance. We determined what parts we couldn't simplify due to that the performance would go further than 5% from the initial performance.

Now we looked at the parts of the design that we could modify, and at their traces.

## 3 Result

In order to establish a baseline performance the simulator was run with the default arguments and a trace was created, with the command shown in listing 1.

**Listing 1** – Simulator command.

```
sim-outorder -config superscalar.cfg -ptrace trace.trc
100000:+30 go.ss 3 8
```

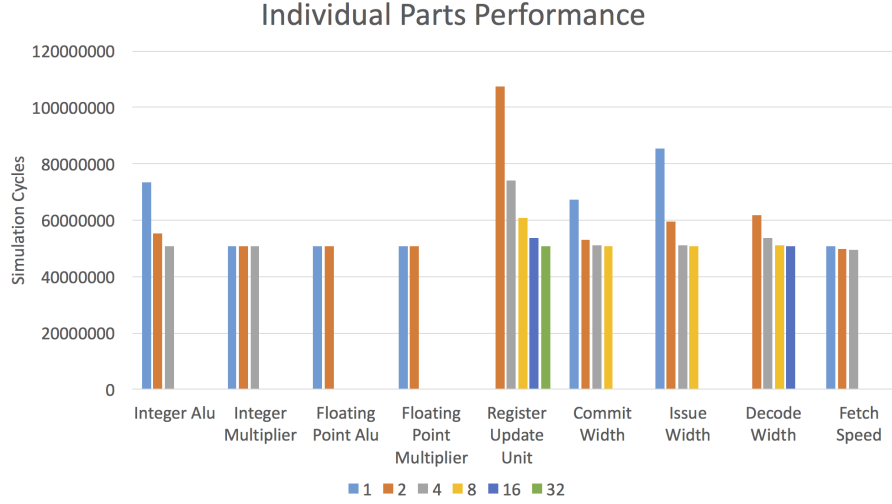
The config file "superscalar.cfg" is supplied with the lab and contains all default settings. The most interesting of these are shown in table 1.

**Table 1** – Default settings.

Setting	Value
res:ialu	4
res:imult	4
res:fpalu	2
res:fpmult	2
ruu:size	32
commit:width	8
issue:width	8
decode:width	16
fetch:speed	1

Running the simulator with the default settings gave us a simulation time of **50868222** cycles, with the 5% requirement from section 1 this results in a maximum of **53411633** simulation cycles.

### 3.1 Integer components



**Figure 1** – How the individual parts perform.

In figure 1 we can see that reducing the number of integer multipliers to only one, no noticeable change in performance could be seen. Therefore we decided to reduce the amount to only one.

**Table 2** – Integer settings.

Setting	Value	Simulation cycles
res:ialu	2	55360220
res:ialu	1	73570279
res:imult	2	50868222
res:imult	1	50868609

When observing the Integer alu:s, we instead see in Table ??, that the performance will reduce drastically when reducing the amount from 4, thus we decided not to change this in the design, as the performance will drop beneath 5% of the standard design performance.

### 3.2 Floating Point components

In figure 1 we can see that by changing the floating point alu & multiplier, the system didn't perform any worse, thus we decided to simplify these as much as possible.

**Table 3** – Floating point settings.

Setting	Value	Simulation cycles
res:fpalu	1	50868222
res:fpmult	1	50868222

### 3.3 Control components

The speed of the system was already at the lowest possible, which means by changing this value will increase the complexity of the design, and therefore we decided not to.

When changing the Register Update Unit we can see in Table ??, that when reducing only one step to 16 instead of 32 we get such a result that by changing this we would not longer be within 5% of the standard designs performance.

**Table 4** – Control components settings.

Setting	Value	Simulation cycles
ruu:size	16	53807041
ruu:size	8	60781481
ruu:size	4	74098784
ruu:size	2	107565825
commit:width	4	51076751
commit:width	2	53040135
commit:width	1	67227355
issue:width	4	51187526
issue:width	2	59378979
issue:width	1	85571238
decode:width	8	51165249
decode:width	4	53609877
decode:width	2	61683477
fetch:speed	4	49804290
fetch:speed	2	50868222

### 3.4 Simplified Design

## 4 Discussion

For floating point alu & multiplier we believe that these parts doesn't affect the performance due to that go.ss only uses integer values.

### 4.1 Atomic bomb go game

In Hiroshima, Japan on August 6, 1945, a game of Go was being played. Meanwhile high above the clouds, a lonely B29 Superfortress was soaring peacefully

like an Albatross. It was carrying a very special payload, a product of a joint venture between science and military interests. The game was about to enter its third day of play when the B29 dropped an atomic bomb at 8:15am. After the boom the game was continued outside the city due recommendation from the police.