Cache report

Chang Zihao
20206018
Cui Yuxuan
20206019

Contents

1 Introduction

Lab5 is required to design a Cache. A cache processes three state–read and hit, read but miss, and write. And use cache to save data that use much to save time. We use multi cycle CPU link the cache.

1.1 Read Data

1. Processor send the address to the cache
2. Then use the valid digit and sign digit to find the data.
3. If the data exist then hit, and send the data to the processor.
4. If the data does not exist then send miss to the memory, the memory will send the data that according to the requirement to the specific position.

1.2 Write data

1. Memory send the address and the data to the cache
2. Then compare the valid digit and sign digit to find the specific position in the cache.
3. Write in the data.

2 Design

As the Description of TB file mentioned, We do not need to implement memory and register ourselves. So we design the path to connect them together.

2.1 Module design

The PC is a state element that holds the address of the current instruction. It is updated at the end of every clock cycle.

2.1.1 add module

The adder is responsible for incrementing the PC to hold the address of the next instruction. It takes two input values, adds them together, and outputs the result

2.1.2 Sign extend module

Single extend module is used to increase the number of bits of a binary number while preserving the number's sign and value.

2.1.3 MUX module

Mux, a data selector, which was used to select which the single will be output.

3 Implementation

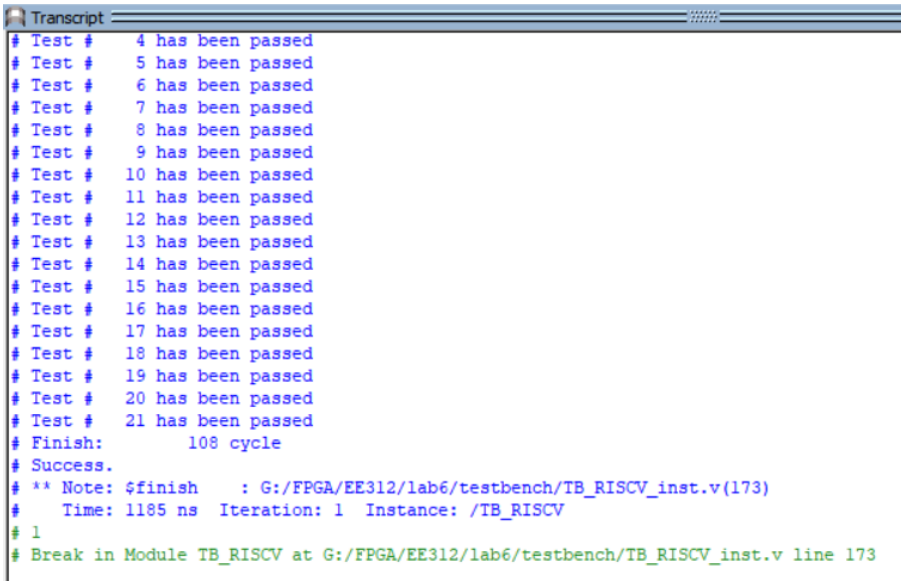
1. Write-through
2. Write-allocate
3. Blocking cache

Using the write-through policy, data is written to the cache and the backing store location at the same time. The significance here is not the order in which it happens or whether it happens in parallel. The significance is that I/O completion is only confirmed once the data has been written to both places. Ensures fast retrieval while making sure the data is in the backing store and is not lost in case the cache is disrupted.

Using the Write-allocate, when data is hitten in the cache, just throw the data out. But if it is missed, the cache needs to ask the memory for the data.

4 Evaluation

I pass all the test in the testbench folder.



```
Transcript
# Test #    4 has been passed
# Test #    5 has been passed
# Test #    6 has been passed
# Test #    7 has been passed
# Test #    8 has been passed
# Test #    9 has been passed
# Test #   10 has been passed
# Test #   11 has been passed
# Test #   12 has been passed
# Test #   13 has been passed
# Test #   14 has been passed
# Test #   15 has been passed
# Test #   16 has been passed
# Test #   17 has been passed
# Test #   18 has been passed
# Test #   19 has been passed
# Test #   20 has been passed
# Test #   21 has been passed
# Finish:        108 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab6/testbench/TB_RISCV_inst.v(173)
#    Time: 1185 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab6/testbench/TB_RISCV_inst.v line 173
```

```
VSIM 355> run -all
# Test #    1 has been passed
# Test #    2 has been passed
# Test #    3 has been passed
# Test #    4 has been passed
# Test #    5 has been passed
# Test #    6 has been passed
# Test #    7 has been passed
# Test #    8 has been passed
# Test #    9 has been passed
# Test #   10 has been passed
# Test #   11 has been passed
# Test #   12 has been passed
# Test #   13 has been passed
# Test #   14 has been passed
# Test #   15 has been passed
# Test #   16 has been passed
# Test #   17 has been passed
# Finish:        332 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab6/testbench/TB_RISCV_forloop.v(168)
#    Time: 3425 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab6/testbench/TB_RISCV_forloop.v line 168
```

```
VSIM 353> run -all
# Test #    1 has been passed
# Test #    2 has been passed
# Test #    3 has been passed
# Test #    4 has been passed
# Test #    5 has been passed
# Test #    6 has been passed
# Test #    7 has been passed
# Test #    8 has been passed
# Test #    9 has been passed
# Test #   10 has been passed
# Test #   11 has been passed
# Test #   12 has been passed
# Test #   13 has been failed!
# output_port = 0x1d (Ans : 0x3)
# ** Note: $finish    : G:/FPGA/EE312/lab6/testbench/TB_RISCV_sort.v(185)
#    Time: 2365 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab6/testbench/TB_RISCV_sort.v line 185
```

5 Conclusion

The cache is very useful in the CPU nowadays,it make the CPU quicker than the CPU last few years.