**My MIPS\_16 DOCUMENT**

Author: Fan ZhenYa, [fzywork@163.com](mailto:fzywork@163.com) 0:33:34, Feb 16, 2012

# Overview

## Technical brief:

1. 16-bit data width

2. classic 5-stage static pipeline, 1 branch delay slot, theoretical CPI is 1.0

3. pipeline is able to detect and prevent RAW hazards, no forwarding logic

4. 8 general purpose register (reg 0 is special, according to mips architecture)

5. up to now supports 13 instructions, see ./doc/instruction\_set.txt for details

6. Maximum clk Frequency: 82.688MHz on Xilinx 3s1000fg320-5 device (XST).

# Architecture



# Implementation

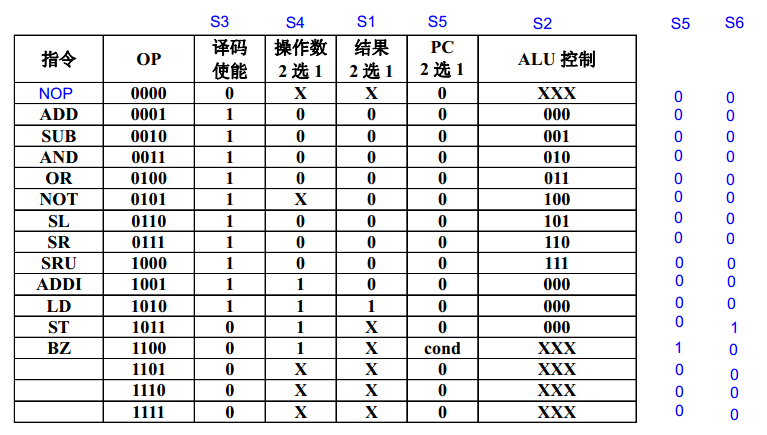
## IF\_stage:

PC, and a asynchronized instruction memory(a ROM for simulation). Instruction memory further can be an I-cache or an external instruction memory interface.

## ID\_stage:

Central control logic and Register read

Truth table of Central Control Logic:



## EX\_stage:

ALU

## MEM\_stage:

A ram for now, further can be a D-cache or an external memory interface.

## WB\_stage:

Write back stage

## Register File:

a 8-entry 16-bit register file, with 1 synchronized write port and 2 synchronized read port. For Register 0, read data from it will always be 0, and write operations will also be discarded.

## Hazard Detection Unit:

Data Hazard detection. if there is a RAW hazard, it will stall the pipeline.

Method: It compare the source register of the instruction in ID\_stage and it's previous 3 instructions' destination register. If the source register is equal to any of the three destination regs and not equals to zero, the Hazard Detection Unit will assert pipeline\_stall signal. That signal will freeze the IF & ID stage, and insert bubbles into EX stage. When the hazard instruction was flushed out of the pipeline, pipeline\_stall signal will be canceled.

## Testbenches:

Every module in this project has its complete testbench and corresponding modelsim simulation scripts, located in ./bench/<modelname>.

# Software tools

## FPGA Synthesis & Implement tool:

Xilinx ISE, work directory: ./backend/Xilinx

**Synthesis report:**

Device utilization summary:

---------------------------

Selected Device : 3s1000fg320-5

Number of Slices: 456 out of 7680 5%

Number of Slice Flip Flops: 290 out of 15360 1%

Number of 4 input LUTs: 883 out of 15360 5%

Number used as logic: 627

Number used as RAMs: 256

Number of IOs: 10

Number of bonded IOBs: 10 out of 221 4%

Number of GCLKs: 1 out of 8 12%

Timing Summary:

---------------

Speed Grade: -5

Minimum period: 12.094ns (Maximum Frequency: 82.688MHz)

Minimum input arrival time before clock: 6.115ns

Maximum output required time after clock: 6.678ns

Maximum combinational path delay: No path found

## RTL Simulation tool:

Modelsim SE 10.0a, work directory: ./sim

## Assembler:

A assembler implemented by JAVA, Java version: 1.6.0\_30. In directory: ./sw

**Useage:**

java mips\_16\_assembler [Options]

Options:

<source\_code\_path> <dest\_path>:

Assemble source code to dest. For exaple .\bin\test1.asm .\bin\test1.prog

<source\_code\_path>:

Assemble source code to dest file a.prog

-h(or --help):

Show this help

# Sample programs

## Test1:

**Purpose:** test basic instrctions, and RAW hazard protection

**Source code:** (\bench\mips\_16\_core\_top\test1.asm)

L1: ADDI R1,R0,8

ADDI R2,R1,8

ADDI R3,R2,8

ADD R4,R2,R3

ST R4,R1,2

LD R5,R1,2

SUB R6,R4,R5

BZ R6,L1

ADDI R7,R7,1

**Modelsim simulation result:**

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# mips\_16 core test

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

# rom load successfully

#

# running test1

#

# current pc: 0 ,instruction: 9208

# current pc: 1 ,instruction: 9448

# current pc: 2 ,instruction: 9688

# current pc: 3 ,instruction: 1898

# current pc: 4 ,instruction: b842

# current pc: 5 ,instruction: aa42

# current pc: 6 ,instruction: 2d28

# current pc: 7 ,instruction: c1b8

# current pc: 8 ,instruction: 9fc1

# current pc: 0 ,instruction: 9208

# current pc: 1 ,instruction: 9448

# current pc: 2 ,instruction: 9688

# current pc: 3 ,instruction: 1898

# current pc: 4 ,instruction: b842

# current pc: 5 ,instruction: aa42

# current pc: 6 ,instruction: 2d28

# current pc: 7 ,instruction: c1b8

# current pc: 8 ,instruction: 9fc1

# current pc: 0 ,instruction: 9208

# current pc: 1 ,instruction: 9448

…….

# display\_all\_regs:

# ------------------------------

# R0 R1 R2 R3 R4 R5 R6 R7

# 0 8 16 24 40 40 0 3

# ------------------------------

# ram[10] = 40

## Test2:

**Purpose:** Multiply R3=R1\*R2 using add and shift instructions

**Source code:** (\bench\mips\_16\_core\_top\test2.asm)

**Modelsim simulation result:**

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# mips\_16 core test

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

# rom load successfully

#

# running test2

#

# multiply R3=R1\*R2

#

# display\_all\_regs:

# ------------------------------

# R0 R1 R2 R3 R4 R5 R6 R7

# 0 0 0 0 0 0 0 0

# ------------------------------

# running test2

#

# current pc: 0 ,instruction: 921c

# current pc: 1 ,instruction: 9411

# current pc: 2 ,instruction: 9c01

# current pc: 3 ,instruction: 9e00

# current pc: 4 ,instruction: 9a10

# current pc: 5 ,instruction: 9fc1

# current pc : 6

# display\_all\_regs:

# ------------------------------

# R0 R1 R2 R3 R4 R5 R6 R7

# 0 28 17 0 0 0 0 0

# ------------------------------

# current pc: 6 ,instruction: 3990

# current pc: 7 ,instruction: c102

# current pc: 8 ,instruction: 0000

# current pc: 9 ,instruction: 16c8

# current pc: 10 ,instruction: 6270

# current pc: 11 ,instruction: 84b0

# current pc: 12 ,instruction: 2978

# current pc: 13 ,instruction: c103

# current pc: 14 ,instruction: 0000

# current pc: 15 ,instruction: c035

# current pc: 16 ,instruction: 0000

# current pc: 5 ,instruction: 9fc1

# current pc : 6

# display\_all\_regs:

# ------------------------------

# R0 R1 R2 R3 R4 R5 R6 R7

# 0 56 8 28 15 16 1 1

# ------------------------------

# current pc: 6 ,instruction: 3990

# current pc: 7 ,instruction: c102

# current pc: 8 ,instruction: 0000

# current pc: 10 ,instruction: 6270

# current pc: 11 ,instruction: 84b0

……..

# current pc: 14 ,instruction: 0000

# current pc: 15 ,instruction: c035

# current pc: 16 ,instruction: 0000

# current pc: 5 ,instruction: 9fc1

# current pc : 6

# display\_all\_regs:

# ------------------------------

# R0 R1 R2 R3 R4 R5 R6 R7

# 0 0 0 476 2 16 1 14

# ------------------------------

# current pc: 6 ,instruction: 3990

# current pc: 7 ,instruction: c102

# current pc: 8 ,instruction: 0000

# current pc: 10 ,instruction: 6270

# current pc: 11 ,instruction: 84b0

# current pc: 12 ,instruction: 2978

# current pc: 13 ,instruction: c103

# current pc: 14 ,instruction: 0000

# current pc: 15 ,instruction: c035

# current pc: 16 ,instruction: 0000

# current pc: 5 ,instruction: 9fc1

# current pc : 6

# display\_all\_regs:

# ------------------------------

# R0 R1 R2 R3 R4 R5 R6 R7

# 0 0 0 476 1 16 1 15

# ------------------------------

# current pc: 6 ,instruction: 3990

# current pc: 7 ,instruction: c102

# current pc: 8 ,instruction: 0000

# current pc: 10 ,instruction: 6270

# current pc: 11 ,instruction: 84b0

# current pc: 12 ,instruction: 2978

# current pc: 13 ,instruction: c103

# current pc: 14 ,instruction: 0000

# current pc: 17 ,instruction: c03f

# current pc: 18 ,instruction: 0000

# current pc: 17 ,instruction: c03f

# current pc: 18 ,instruction: 0000

…….

# current pc: 17 ,instruction: c03f

# current pc: 18 ,instruction: 0000

# display\_all\_regs:

# ------------------------------

# R0 R1 R2 R3 R4 R5 R6 R7

# 0 0 0 476 0 16 1 16

# ------------------------------

# Reference

胡伟武，计算机体系结构，清华大学出版社，2011