My MIPS_16 DOCUMENT

Author: Fan ZhenYa, fzywork@163.com

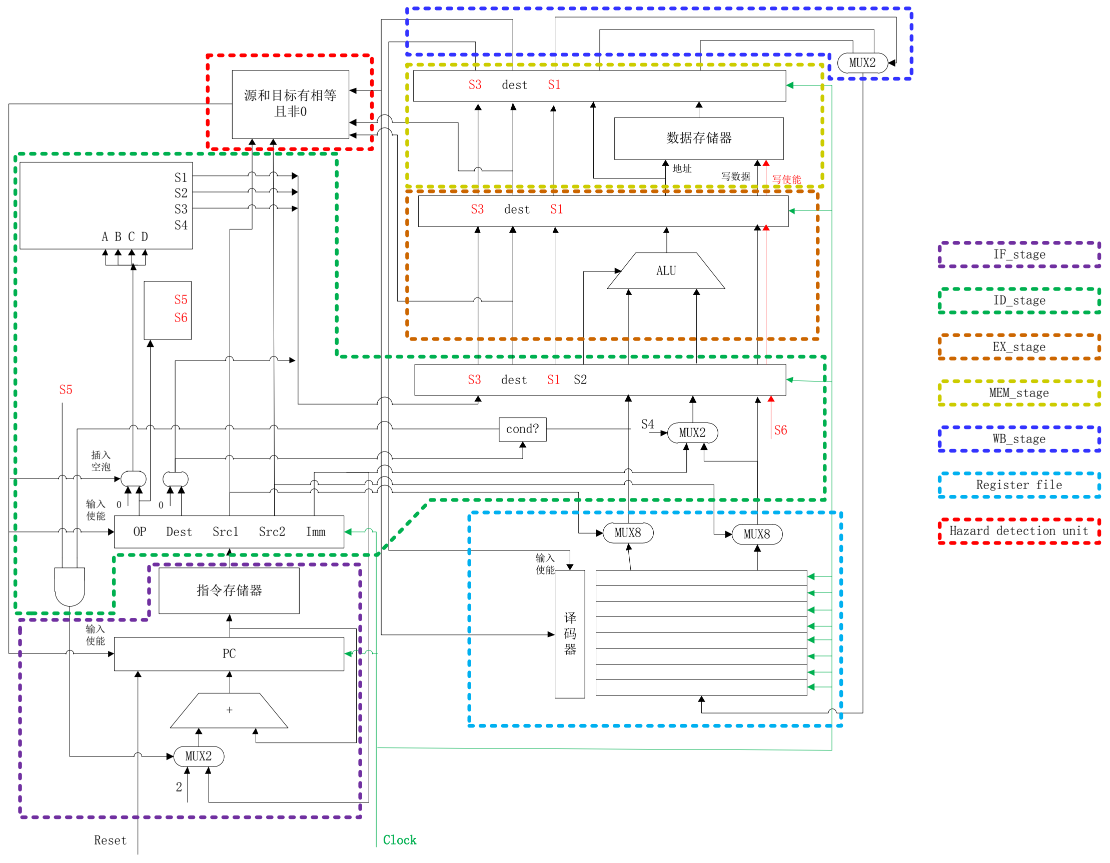
1. Overview

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).

2. Architecture

My MIPS_16 architecture



3. 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:

		S3	S4	S1	S5	S2	S5	S6
指令	OP	译码使能	操作数 2 选 1	结果 2 选 1	PC 2 选 1	ALU 控制		
NOP	0000	0	X	X	0	XXX	0	0
ADD	0001	1	0	0	0	000	0	0
SUB	0010	1	0	0	0	001	0	0
AND	0011	1	0	0	0	010	0	0
OR	0100	1	0	0	0	011	0	0
NOT	0101	1	X	0	0	100	0	0
SL	0110	1	0	0	0	101	0	0
SR	0111	1	0	0	0	110	0	0
SRU	1000	1	0	0	0	111	0	0
ADDI	1001	1	1	0	0	000	0	0
LD	1010	1	1	1	0	000	0	0
ST	1011	0	1	X	0	000	0	1
BZ	1100	0	1	X	cond	XXX	1	0
	1101	0	X	X	0	XXX	0	0
	1110	0	X	X	0	XXX	0	0
	1111	0	X	X	0	XXX	0	0

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>.

4. 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

<u>Useage:</u>

```
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
```

5. Sample programs

Test1:

<u>Purpose:</u> test basic instrctions, and RAW hazard protection

<u>Source code:</u> (\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
# ------------------------------
```

6. Reference

胡伟武，计算机体系结构，清华大学出版社，2011