MP3 Papier Design

Von

Huang, Yu, und Zhang

Diagram

Details

```
typedef struct packed {
    logic load_latch;
    // IF stage
    logic icache_read;
    logic load_pc;
    // ID stage
    // EX stage
    alu_ops aluop;
    alumux1_sel_t alumux1_sel;
    alumux2_sel_t alumux2_sel;
    branch_funct3_t cmpop;
    cmpmux_sel_t cmpmux_sel;
    arithmux_sel_t arithmux_sel;
    iaddrmux_sel_t iaddrmux_sel;
    // MEM stage
    logic dcache_write;
    logic dcache_read;
    // WB stage
    logic regfilemux_sel;
    logic load_regfile;
} rv32i_control_word;
```

If branch prediction is wrong (we know this in EX stage), we set the structs in IF and ID stage to invalid, and also load PC with the correct target value.

funct3: branch, load, store, arith_reg, arith_imm
funct7: arith_reg (add vs sub, srl vs sra), arith_imm (srli vs srai)

Per stage actions

A stage should not do anything if its struct is invalid.

IF

Read I-cache, then set PC to next value, write the latch. Current pc_out write to IF/ID latch.

ID

Read the REGFILE according to what is specified in the IF/ID latch. Write rs1_out, rs2_out, and other old values in IF/ID latch to ID/EX latch.

EX

Feed aluop and cmpop according to opcode+(funct3,funct7), write alu_out (br_en in case of slt or sltu) and other old values in ID/EX latch to EX/MEM latch.

In branch operation, the CMP and the ALU are working in parallel; if prediction is wrong (in CP1 we do static non-taken), set next IF/ID and ID/EX latches to invalid, and set iaddrmux_sel to load the next instruction from the correct address.

MEM

Read/write memory (D-cache), write dc_out to MEM/WB latch structure. Pass values from EX/MEM latch to MEM/WB latch.

WB

Write to rd in REGFILE with alu_out (alu result or br_en), dc_out (lw, lh, lhu, lb, lbu), u_imm or pc_plus4 according to funct3.

Signal Specification:

CPU & CACHE

signal	width	from	to	specification
icache_rdata	32	instruction cache	cpu	the data read from instruction cache
icache_addr	32	cpu	instruction cache	the address to read from instruction cache, (from PC)
icache_read	1	cpu	instruction cache	read control of instruction cache
dcache_rdata	32	data cache	cpu	the data read from data cache
dcache_addr	32	cpu	data cache	the address to read from data cache
dcache_read	1	cpu	data cache	read control of data cache
dcache_write	1	cpu	data cache	write control of data cache
dcache_byte_en	4	cpu	data cache	byte enable of data cache, generated in datapath with alu_out and funct3 from EX/MEM latch
dcache_wdata	32	cpu	data cache	the data to write to data cache, always from rs2_out of EX/MEM latc

DATAPATH & CONTROL

signal	width	from	to	specification
ctl_word	struct	control	datapath	packed control signal of system
fd_word	struct	datapath	control	state latch between IF/ID stage
de_word	struct	datapath	control	state latch between ID/EX stage

em_word	struct	datapath	control	state latch between EX/MEM stage
mw_word	struct	datapath	control	state latch between MEM/WB stage

INSIDE DATAPATH

signal	width	from	to	specification
alumux1_sel	1	control	alu_src1_mux	select bit of alumux1, 1’b1: alumux1_out = pc_out 1’b0: alumux1_out = sr1
alumux2_sel	3	control	alu_src2_mux	select bits of alumux2, 3'b000: alumux2_out = i_imm 3'b001: alumux2_out = u_imm 3'b010: alumux2_out = b_imm 3'b011: alumux2_out = s_imm 3'b100: alumux2_out = j_imm 3'b101: alumux2_out = rs2
alu_op	3	control	alu	operation of alu, determined by opcode+(funct3+funct7)
cmpmux_sel	1	control	cmp_src2_mux	select bit of comparator, 1’b0: cmpmux_out =i_imm 1’b1: cmpmux_out = rs2
arithmux_sel	1	control	arithmux	select alu_out or br_en to written to EX/MEM latch 1'b0: alu_out 1’b1: cmp_out
iaddrmux_out	1	iaddrmux	pc_in_mux, mar_mux	act as select signal of pc_mux and mar_mux
iaddrmux_sel	2	control	iaddrmux	1'b0: always_pc 1’b1: always_alu br_en: br_en

em_funct3	3	EX/ MEM latch	data cache	generate dcache_byte_en according to the last 2 bits of alu_out from EX/MEM latch
regfilemux_sel	4	control	regfilemux	4'b0000: regfilemux_out = alu_out from MEM/WB latch 4'b0010: regfilemux_out = u_imm from MEM/WB latch 4'b0011: regfilemux_out = lw = dc from MEM/WB latch 4'b0100: regfilemux_out = pc from MEM/WB latch +4 (for the below, choose the correct part of dc according to correct bits from alu_out from MEM/WB latch) 4'b0101: regfilemux_out = lb 4'b0110: regfilemux_out = lbu 4'b0111: regfilemux_out = lh 4'b1000: regfilemux_out = lhu