**DESIGN**

module pipe\_MIPS32 (clk1, clk2);

input clk1, clk2;

reg [31:0] PC, IF\_ID\_IR, IF\_ID\_NPC;

reg [31:0] ID\_EX\_IR, ID\_EX\_NPC, ID\_EX\_A, ID\_EX\_B, ID\_EX\_Imm;

reg [2:0] ID\_EX\_type, EX\_MEM\_type, MEM\_WB\_type;

reg [31:0] EX\_MEM\_IR, EX\_MEM\_ALUOut, EX\_MEM\_B;

reg EX\_MEM\_cond;

reg [31:0] MEM\_WB\_IR, MEM\_WB\_ALUOut, MEM\_WB\_LMD;

reg [31:0] Reg [0:31];

reg [31:0] Mem [0:1023];

parameter ADD = 6'b000000, SUB = 6'b000001, AND = 6'b000010, OR = 6'b000011, SLT = 6'b000100, MUL = 6'b000101, HLT = 6'b111111, LW = 6'b001000, SW = 6'b001001, ADDI = 6'b001010, SUBI = 6'b001011, SLTI = 6'b001100, BNEQZ = 6'b001101, BEQZ = 6'b001110;

parameter RR\_ALU = 3'b000, RM\_ALU = 3'b001, LOAD = 3'b010, STORE = 3'b011, BRANCH = 3'b100, HALT = 3'b101;

reg HALTED;

reg TAKEN\_BRANCH;

always @(posedge clk1) // IF Stage

begin

if (HALTED == 0)

begin

if (((EX\_MEM\_IR[31:26] == BEQZ) && (EX\_MEM\_cond == 1)) || ((EX\_MEM\_IR[31:26] == BNEQZ) && (EX\_MEM\_cond == 0)))

begin

IF\_ID\_IR <= #2 Mem[EX\_MEM\_ALUOut];

TAKEN\_BRANCH <= #2 1'b1;

IF\_ID\_NPC <= #2 EX\_MEM\_ALUOut + 1;

PC <= #2 EX\_MEM\_ALUOut + 1;

end

else

begin

IF\_ID\_IR <= #2 Mem[PC];

IF\_ID\_NPC <= #2 PC + 1;

PC <= #2 PC + 1;

end

end

end

always @(posedge clk2) // ID stage

begin

if (HALTED == 0)

begin

if (IF\_ID\_IR[25:21] == 5'b00000)

ID\_EX\_A <= 0;

else

ID\_EX\_A <= #2 Reg[IF\_ID\_IR[25:21]]; // "rS"

if (IF\_ID\_IR[20:16] == 5'b00000)

ID\_EX\_B <= 0;

else

ID\_EX\_B <= #2 Reg[IF\_ID\_IR[20:16]]; // "rt"

ID\_EX\_NPC <= #2 IF\_ID\_NPC;

ID\_EX\_IR <= #2 IF\_ID\_IR;

ID\_EX\_Imm <= #2 {{16{IF\_ID\_IR[15]}}, {IF\_ID\_IR[15:0]}};

case (IF\_ID\_IR[31:26])

ADD, SUB, AND, OR, SLT, MUL: ID\_EX\_type <= #2 RR\_ALU;

ADDI, SUBI, SLTI: ID\_EX\_type <= #2 RM\_ALU;

LW: ID\_EX\_type <= #2 LOAD;

SW: ID\_EX\_type <= #2 STORE;

BNEQZ, BEQZ: ID\_EX\_type <= #2 BRANCH;

HLT: ID\_EX\_type <= #2 HALT;

default: ID\_EX\_type <= #2 HALT;

endcase

end

end

always @(posedge clk1) // EX Stage

begin

if (HALTED == 0)

begin

EX\_MEM\_type <= #2 ID\_EX\_type;

EX\_MEM\_IR <= #2 ID\_EX\_IR;

TAKEN\_BRANCH <= #2 0;

case (ID\_EX\_type)

RR\_ALU:

begin

case (ID\_EX\_IR[31:26]) // "opcode"

ADD: EX\_MEM\_ALUOut <= #2 ID\_EX\_A + ID\_EX\_B;

SUB: EX\_MEM\_ALUOut <= #2 ID\_EX\_A - ID\_EX\_B;

AND: EX\_MEM\_ALUOut <= #2 ID\_EX\_A & ID\_EX\_B;

OR: EX\_MEM\_ALUOut <= #2 ID\_EX\_A | ID\_EX\_B;

SLT: EX\_MEM\_ALUOut <= #2 ID\_EX\_A < ID\_EX\_B;

MUL: EX\_MEM\_ALUOut <= #2 ID\_EX\_A \* ID\_EX\_B;

default: EX\_MEM\_ALUOut <= #2 32'hxxxxxxxx;

endcase

end

RM\_ALU:

begin

case (ID\_EX\_IR[31:26])

ADDI: EX\_MEM\_ALUOut <= #2 ID\_EX\_A + ID\_EX\_Imm;

SUBI: EX\_MEM\_ALUOut <= #2 ID\_EX\_A - ID\_EX\_Imm;

SLTI: EX\_MEM\_ALUOut <= #2 ID\_EX\_A < ID\_EX\_Imm;

default: EX\_MEM\_ALUOut <= #2 32'hxXXXXXXX;

endcase

end

endcase

end

end

always @(posedge clk2)

begin

if (HALTED == 0)

begin

MEM\_WB\_type <= #2 EX\_MEM\_type;

MEM\_WB\_IR <= #2 EX\_MEM\_IR;

case (EX\_MEM\_type)

RR\_ALU, RM\_ALU: MEM\_WB\_ALUOut <= #2 EX\_MEM\_ALUOut;

LOAD: MEM\_WB\_LMD <= #2 Mem[EX\_MEM\_ALUOut];

STORE: if (TAKEN\_BRANCH == 0) Mem[EX\_MEM\_ALUOut] <= #2 EX\_MEM\_B;

endcase

end

end

always @(posedge clk1)

begin

if (TAKEN\_BRANCH == 0)

begin

case (MEM\_WB\_type)

RR\_ALU: Reg[MEM\_WB\_IR[15:11]] <= #2 MEM\_WB\_ALUOut; // "rd"

RM\_ALU: Reg[MEM\_WB\_IR[20:16]] <= #2 MEM\_WB\_ALUOut; // "rt"

LOAD: Reg[MEM\_WB\_IR[20:16]] <= #2 MEM\_WB\_LMD; // "xt"

HALT: HALTED <= #2 1'b1;

endcase

end

end

endmodule

**TESTBENCH**

module test\_mips32;

reg clk1, clk2;

integer k;

pipe\_MIPS32 mips (clk1, clk2);

initial begin

clk1 = 0; clk2 = 0;

repeat (20) begin

#5 clk1 = 1; #5 clk1 = 0;

#5 clk2 = 1; #5 clk2 = 0;

end

end

initial begin

for (k = 0; k < 32; k = k + 1)

mips.Reg[k] = k;

mips.Mem[0] = 32'h2801000a;

mips.Mem[1] = 32'h28020014;

mips.Mem[2] = 32'h28030019;

mips.Mem[3] = 32'h0ce77800;

mips.Mem[4] = 32'h0ce77800;

mips.Mem[5] = 32'h00222000;

mips.Mem[6] = 32'h0ce77800;

mips.Mem[7] = 32'h00832800; // Fixed typo: 'mipa' should be 'mips'

mips.Mem[8] = 32'hfc000000;

mips.HALTED = 0;

mips.PC = 0;

mips.TAKEN\_BRANCH = 0;

#280;

for (k = 0; k < 6; k = k + 1)

$display("R%1d - %2d", k, mips.Reg[k]);

end

initial begin

$dumpfile("mips.vcd");

$dumpvars(0, test\_mips32);

#300 $finish;

end

endmodule