module p7(otpt, alu1, alu2, carry, zero\_case, data\_in, mux\_cntrl, clk, write\_enable,

read\_addr1, read\_addr2, write\_addr, sel);

output reg [31:0] otpt, alu1, alu2;

output reg carry, zero\_case;

input[31:0] data\_in;

input[2:0] sel;

input mux\_cntrl, clk, write\_enable;

input[5:0] read\_addr1, read\_addr2, write\_addr;

reg[31:0] mux\_out;

reg [31:0] register [31:0];

always @(posedge clk)

begin

mux\_out <= data\_in \* ~mux\_cntrl + otpt \* mux\_cntrl;

alu1 <= register[read\_addr1];

alu2 <= register[read\_addr2];

if(write\_enable)

register[write\_addr] <= data\_in;

case(sel)

3'b000 : {carry ,otpt} <= alu1 + alu2;

3'b001 : {carry ,otpt} <= alu1 - alu2;

3'b010 : otpt <= alu1 ^ alu2;

3'b011 : otpt <= alu1 & alu2;

3'b100 : otpt <= alu1 + 1'b1;

3'b101 : otpt <= alu1 | alu2;

3'b110 : otpt <= alu1 << 1;

3'b111 : otpt <= alu1 >> 1;

endcase

if(otpt == 0);

zero\_case <= 1;

end

endmodule

module p7\_tb();

wire[31:0] otpt;

wire carry, zero\_case;

reg[31:0] data\_in, final\_otpt;

reg[2:0] sel;

reg mux\_cntrl, clk, write\_enable;

reg[5:0] read\_addr1, read\_addr2, write\_addr;

wire[31:0] alu1, alu2;

integer i;

initial

begin

clk <= 1'b0;

read\_addr1 <= 4'b0001;

read\_addr2 <= 4'b0010;

sel = 3'b000;

data\_in = 0;

write\_addr = 26;

write\_enable <= 1;

final\_otpt <= 0;

mux\_cntrl <= 0;

#200

for(i = 0; i < 25; i = i + 1)

begin

data\_in = $urandom%(64);

write\_enable = 1;

mux\_cntrl <= 0;

write\_addr <= i;

#200;

end

for(i = 0; i < 25; i = i + 1)

begin

write\_enable = 0;

mux\_cntrl <= 1;

read\_addr1 <= i;

read\_addr2 <= 26;

final\_otpt <= final\_otpt + otpt;

#200;

end

write\_enable = 1;

write\_addr = 31;

data\_in = final\_otpt;

#200;

end

always #100 clk = ~clk;

p7 uut(otpt, alu1, alu2, carry, zero\_case, data\_in, mux\_cntrl, clk, write\_enable,

read\_addr1, read\_addr2, write\_addr, sel);

endmodule

