// complex multiplier
// accepts complex inputs a and b and outputs prod p
// inputs are in fixed-point 2’s complement (4 bits before and 12
// after decimal point)
// output has twice the bits (8 bits before and 24 after decimal
// point)
module cmult(
    output logic signed [7:-24] p_r, p_i,
    input signed [3:-12] a_r, a_i, b_r, b_i,
    input clk, reset, go );

    // DATAPATH
    // datapath consists of wires, multiplier, adder, and registers
    logic a_sel, b_sel, pp1_en, pp2_en, sub, p_r_en, p_i_en;
    logic signed [3:-12] a_operand, b_operand;
    logic signed [7:-24] pp, sum;
    logic signed [7:-24] pp1, pp2;

    // multiplier
    assign a_operand = ~a_sel ? a_r : a_i;
    assign b_operand = ~b_sel ? b_r : b_i;
    assign pp = a_operand * b_operand;

    // partial product regs
    always @(posedge clk)
        if(pp1_en) pp1 <= pp;
    always @(posedge clk)
        if(pp2_en) pp2 <= pp;

    // adder
    assign sum = ~sub ? pp1 + pp2 : pp1 - pp2;

    // product regs
    always @(posedge clk)
        if(p_r_en) p_r <= sum;
    always @(posedge clk)
        if(p_i_en) p_i <= sum;

    // CONTROL
    // control is implemented with an FSM

    // fsm states
    enum bit [2:0] { step0, step1, step2, step3, step4 } curr_state, next_state;

    // fsm transitions
    always @(posedge clk or posedge reset)
        if (reset) curr_state <= step0;
        else curr_state <= next_state;
always @* begin
  case(curr_state)
    step0: if(~go) next_state = step0; else next_state = step1;
    step1: next_state = step2;
    step2: next_state = step3;
    step3: next_state = step4;
    step4: next_state = step0;
  endcase
end

// fsm outputs
always @(curr_state,reset) begin
  a_sel = 0; b_sel = 0; pp1_en = 0; pp2_en = 0;
  sub = 0; p_r_en = 0; p_i_en = 0;
  case(curr_state)
    step0: begin pp1_en = 1; end
    step1: begin a_sel = 1; b_sel = 1; pp2_en = 1; end
    step2: begin b_sel = 1; pp1_en = 1; sub = 1; p_r_en = 1; end
    step3: begin a_sel = 1; pp2_en = 1; end
    step4: begin p_i_en = 1; end
  endcase
end
endmodule