```
// unsigned multiplier (with counter)

module mul(clk, reset, start, word1, word2, product, ready);
        input clk, reset, start;
        input [3:0] word1, word2;
        output [7:0] product;
        output ready;

        wire  m0, load, shift, addshift;

        datapath u1 (clk, reset, load, shift, addshift, word1, word2, product, m0);
        controller u2 (clk, reset, start, m0, load, shift, addshift, ready);

endmodule

module datapath (clk, reset, load, shift, addshift, word1, word2, product, m0);
        input clk, reset, load, shift, addshift;
        input [3:0] word1, word2;
        output [7:0] product;
        output m0;

        reg [7:0] product;
        reg [3:0] multiplicand;
        wire[4:0] sum;                  // 5-bit

        assign m0 = product[0];
        assign sum = product[7:4] + multiplicand;

        always @ (posedge clk or posedge reset) begin
                if (reset) begin multiplicand <= 0; product <= 0; end
        else if (load) begin
                        multiplicand <= word1;
                        product <= {4'b0, word2};       // lower 4-bit = multiplier
                end
                else if (shift)                 // shift-right
                        product <= {1'b0, product[7:1]};
                else if (addshift)      // add & shift right
                        product <= {sum, product[3:1]};
        end
endmodule

module controller (clk, reset, start, m0, load, shift, addshift, ready);
        input clk, reset, start, m0;
        output load, shift, addshift, ready;

        reg [0:0] state;
        localparam S0=0, S1=1;
        reg [1:0] count;                        // count four states

        always @ (posedge clk or posedge reset)         // State transitions (with register)
                if (reset) begin state <= S0; count <= 0; end
                else
                        case (state)
                         S0: if (start) begin state <= S1; count <= 3; end
                         S1: if (count==0) state <= S0;
                                  else count <= count - 1;
                        endcase

 // control output logic
        assign load = (state==S0) & start;
        assign shift = (state==S1) & ~m0;
        assign addshift = (state==S1) & m0;
        assign ready = (state==S0) & ~reset;
endmodule

```