Hardware design of a Sign-Magnitude sequential multiplier

Proposed problems

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

December 9, 2023

Problem 1

Implement register Q of the Sign-Magnitude sequential multiplier architecture.

Start from the template below (available ▸ here):

```verilog
module reg_q(
  input clk, rst_b, clr_lsb, ld_ibus, ld_obus, sh_r,
  input sh_i, [7:0] ibus,
  output reg [7:0] obus, [7:0] q
);
  always @ (posedge clk, negedge rst_b)
    //treat inputs rst_b, clr_lsb, ld_ibus, sh_r here

  always @ (*) //write content to obus when ld_obus==1
    obus = (ld_obus) ? q : 8'bz;
endmodule
```

Problem 2

Construct register A of the Sign-Magnitude sequential multiplier architecture.

Start from the template below (available ▸ here) knowing that content's writing to the Outbus is constructed in a similar manner to register Q's.

```
module reg_a(
  input clk, rst_b, clr, sh_r, ld_sgn, ld_obus, ld_sum,
  input sh_i, sgn, [7:0] sum,
  output reg [7:0] obus, [7:0] q
);
  //implementation here

  //write content to obus
endmodule
```

Problem 3

Build Sign-Magnitude sequential multiplier architecture's control unit, starting from the template below (available ▸ here).

```
module ctrl_u(
  input clk, rst_b, bgn, q_0, cnt_is_7,
  output c0, c1, c2, c3, c4, c5, c6, fin
);
  //implementation here
endmodule
```

Problem 4

Complete the Sign-Magnitude sequential multiplier architecture starting from the template below (available ▸ here), which includes a testbench for multiplying operands $-23 * 2^{-7}$ and $-3 * 2^{-7}$.

```verilog
module sm_unit (
  input clk, rst_b, bgn, [7:0] ibus,
  output fin, [7:0] obus
);
  //implementation here
endmodule

module sm_unit_tb;
    reg clk, rst_b, bgn; reg [7:0] ibus; wire fin; wire [7:0] obus;

    sm_unit test (.clk(clk), .rst_b(rst_b), .bgn(bgn), .ibus(ibus),
        .fin(fin), .obus(obus));
    localparam CLK_PERIOD=100, CLK_CYCLES=17, RST_PULSE=25;
    localparam X=8'b10010111/*=-23*2^(-7)*/, Y=8'b10000011;/*=-3*2^(-7)*/
    initial begin clk=1'd0; repeat (CLK_CYCLES*2) #(CLK_PERIOD/2) clk=~clk; end
    initial begin rst_b=1'd0; #(RST_PULSE); rst_b=1'd1; end
    initial begin bgn=1'd1; #200; bgn=1'd0; end
    initial begin ibus=0; #100 ibus=X; #100 ibus=Y; end
endmodule
```