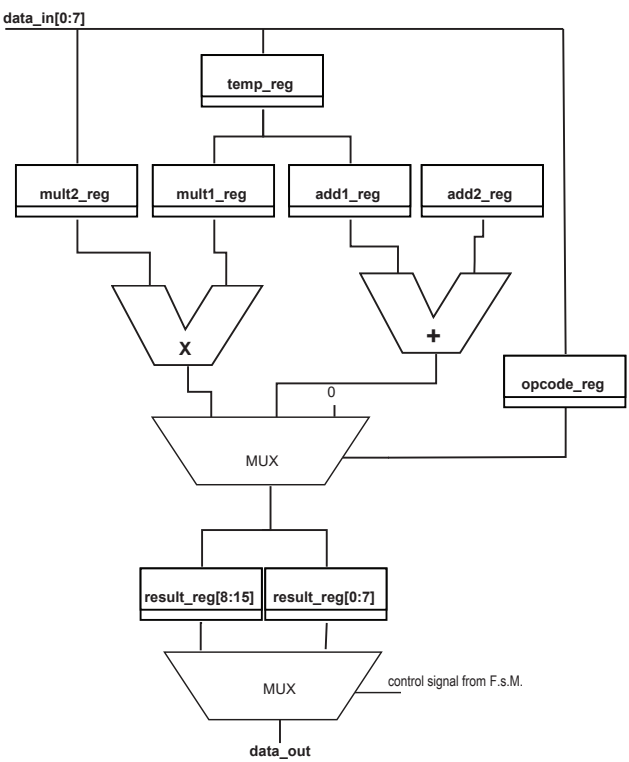


POW Assignment

Pietro Pennestrì s2382660

File List

- siso_gen_calc5_arch:
new architecture for siso8
- conf_tb_siso_gen_calc5_8:
configuration file for pre-synthesis simulation
- conf_tb_siso_gen_calc5_8_post :
configuration file for the post-synthesis simulation



From the simulation of Figure 2 the following behaviour of calc architecture can be observed:

- 1 Both the arithmetic logic units are always on. Power can be saved by disabling the unused unit.
- 2 Both the arithmetic units are on during the input loading phase. Such behavior will cause an unwanted operation because left_in_reg is loaded with the new data, while right_in_reg contains data from the previous cycle.

From previous considerations, a new architecture calc5, depicted in Figure 1 was developed. In particular, the following features can be recognized:

- all the arithmetic units have their dedicated register.
- temp_reg is a temporary register that allows loading simultaneously the arithmetic units registers. This modification overcomes the problem described in 2.

Figure 1. calc5 architecture

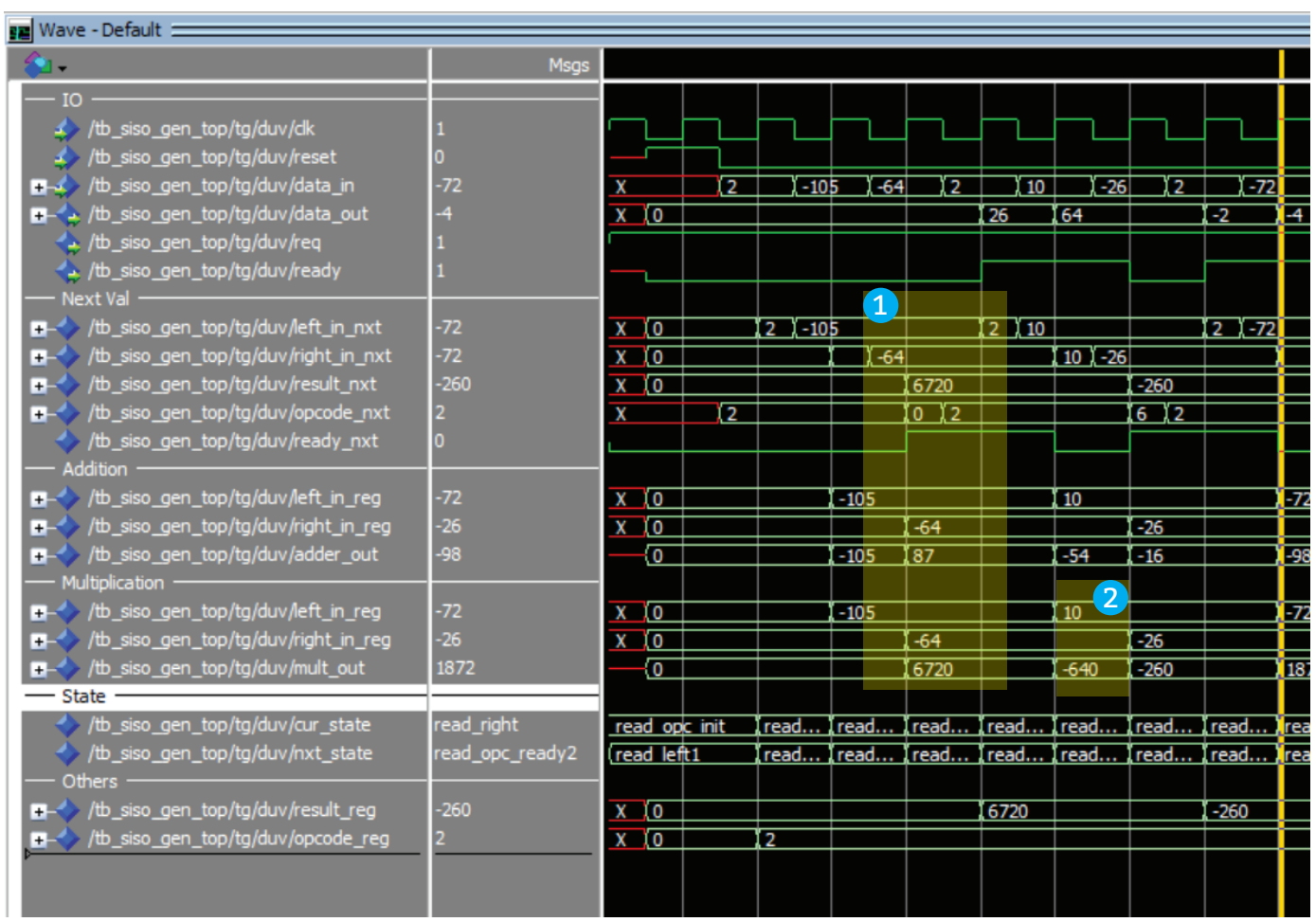


Figure 2. Simulation of the calc architecture.

Despite the data path modification, the F.s.M. structure was not modified. However, different actions are performed in the states. A description of the F.s.M. is depicted in Figure 9.

Simulations Results

calc5 architecture with or without clock gating has been **successfully** tested for all the provided input sets.

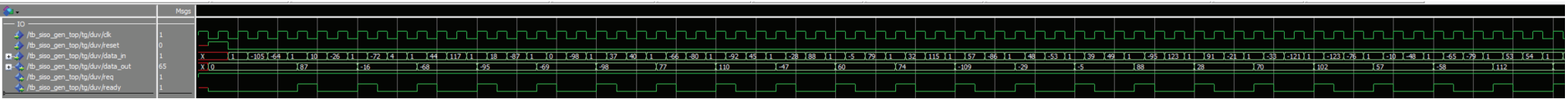


Figure 3. Pre synthesis simulations calc5 @clock_period=5ns, input=add8

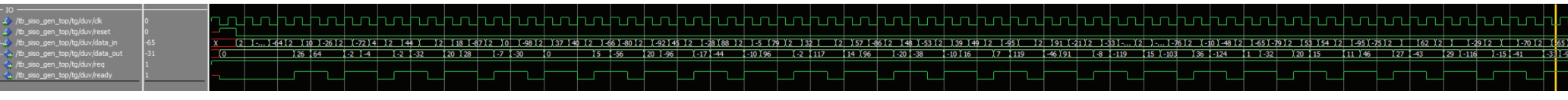


Figure 4. Pre synthesis simulations calc5 @clock_period=5ns, input= mul8

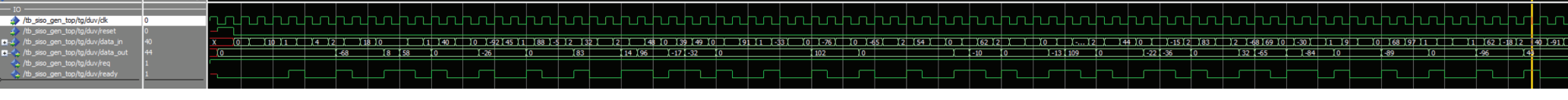


Figure 5. Pre synthesis simulations calc5 @clock_period=5ns, input= mix8

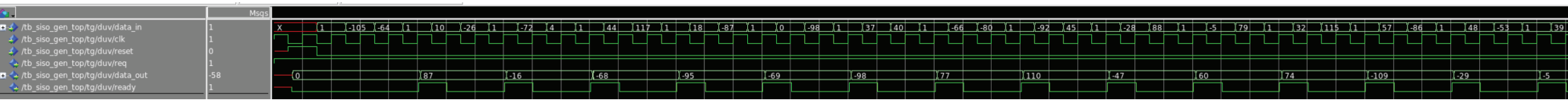


Figure 6. Post synthesis simulations calc5 with clock gating @clock_period=5ns, input=add8

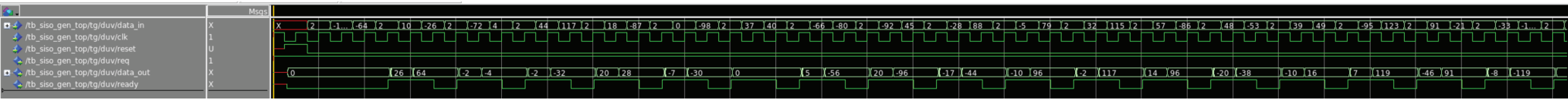


Figure 7. Post synthesis simulations calc5 with clock gating @clock_period=5ns, input= mul8

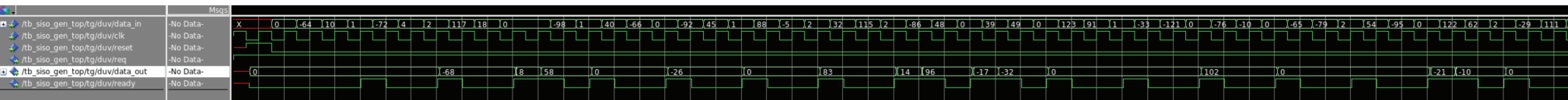


Figure 8. Post synthesis simulations calc5 with clock gating @clock_period=5ns, input= mix8

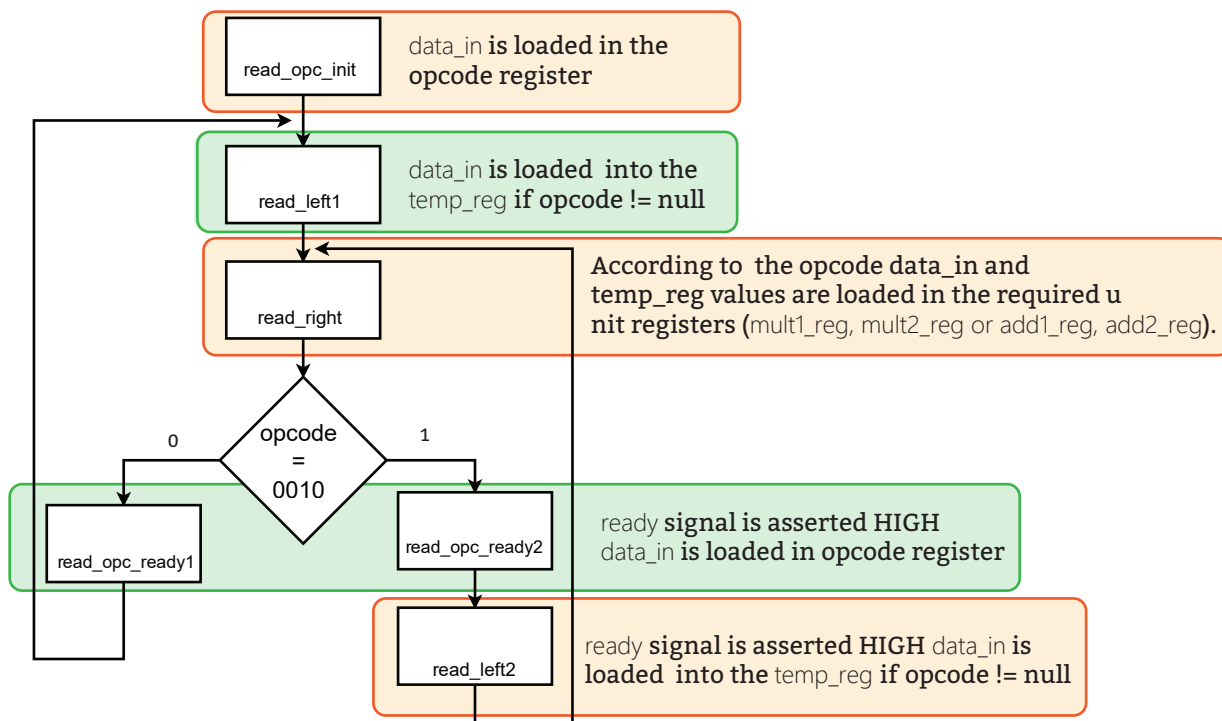


Figure 9. Finite States Machine

Synthesis & Power Analysis Results

Input	Internal Power [mW]		Switching Power [mW]		Leakage Power [pW]		Total Power [mW]		Clock Gating
	calc5	calc	calc5	calc	calc5	calc	calc5	calc	
null8	0.5992	1.4830	0.1100	0.5409	3.5388 e+04	3.0715 e+04	0.7092	2.0239	Yes
null8	1.6131	1.9539	9.1906e-02	0.5551	3.6086 e+04	3.0202 e+04	1.7550	2.5090	No
mul8	1.4760	1.7134	0.4555	0.5717	3.5388 e+04	3.0715 e+04	1.9315	2.2851	Yes
mul8	2.6173	2.2300	0.5007	0.6636	3.6086 e+04	3.0202 e+04	3.1180	2.8939	No
add8	0.8988	1.5704	0.1571	0.5477	3.5388 e+04	3.0715 e+04	1.0560	2.1182	Yes
add8	1.9527	2.0476	0.1652	0.5874	3.6086 e+04	3.0202 e+04	2.1180	2.6351	No
mix8	1.0133	1.6001	0.2429	0.5516	3.5388 e+04	3.0715 e+04	1.2563	2.1517	Yes
mix8	2.0947	2.1243	0.2462	0.6084	3.6086 e+04	3.0202 e+04	2.3410	2.7327	No

Table 1: Power Summary for calc5 & calc architecture

	calc	calc5	calc CG	calc5 CG
Area [nm ²]	9643	12086	9212	11318
Slack [ns]	1.60	1.68	1.61	1.67

Table 2: Area and Slack specifications of calc5 & calc architecture

Table 1 shows that the modification introduced in calc5 architecture, together with clock gating, can reduce the total power in most of the cases near to 50%. Only the null input case benefits from a power reduction of up to 66% because, in the original architecture, unnecessary arithmetic operations were performed. In the case of mul8 input, the calc5 architecture without clock gating has the worst power specification then original architecture. This increase is due to the extra registers present in calc5. However, if clock gating is applied, this pitfall is overcome. In particular, the calc5 architecture with clock gating and mul8 as input has the total power reduced of about 26% with respect to the original calc architecture with clock gating.

Table 2 shows the effect on area and slack by the modification introduced in calc5 architecture. As expected, calc5 has the worst area performances if compared to calc architecture. In particular in the clock gating version, we observe an increase in area of about 19%.

```
1  -- File Author: Pietro Pennestri'
2  -- Description: Pre synthesis simulation configuration file for calc5
3
4  configuration conf_tb_siso_gen_calc5_8_mult of tb_siso_gen_top is
5    for top
6      for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
7        generic map (word_length => 8);
8      for structure
9        for duv: siso_gen use entity work.siso_gen(calc5) ;
10       end for;
11      for tv: tvc_siso_gen use entity work.tvc_siso_gen(file_io)
12        generic map (word_length => 8, -- code does not compile w/o this
13                    half_clock_period => 2.5 ns,
14                    in_file_name => "mul8.in", -- Input Files: add8.in,
mul8.in, nul8.in
15                    out_file_name => "mul8.out");
16      end for;
17    end for;
18  end for;
19 end conf_tb_siso_gen_calc5_8_mult;
20
21
22
23
24
25 configuration conf_tb_siso_gen_calc5_8_addition of tb_siso_gen_top is
26   for top
27     for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
28       generic map (word_length => 8);
29     for structure
30       for duv: siso_gen use entity work.siso_gen(calc5) ;
31       end for;
32       for tv: tvc_siso_gen use entity work.tvc_siso_gen(file_io)
33         generic map (word_length => 8, -- code does not compile w/o this
34                     half_clock_period => 2.5 ns,
35                     in_file_name => "add8.in", -- Input Files: add8.in,
mul8.in, nul8.in
36                     out_file_name => "add8.out");
37     end for;
38   end for;
39 end for;
40 end conf_tb_siso_gen_calc5_8_addition;
41
42
43
44
45
46 configuration conf_tb_siso_gen_calc5_8_mix of tb_siso_gen_top is
47   for top
48     for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
49       generic map (word_length => 8);
50     for structure
51       for duv: siso_gen use entity work.siso_gen(calc5);
52       end for;
53       for tv: tvc_siso_gen use entity work.tvc_siso_gen(file_io)
54         generic map (word_length => 8, -- code does not compile w/o this
55                     half_clock_period => 2.5 ns,
56                     in_file_name => "mix8.in", -- Input Files: add8.in,
mul8.in, nul8.in
57                     out_file_name => "mix8.out");
```

```
58         end for;  
59     end for;  
60 end for;  
61 end for;  
62 end conf_tb_asiso_gen_calc5_8_mix;  
63
```

```
1  -- File Author: Pietro Pennestri'
2  -- Description: Post synthesis simulation configuration file for calc5
3
4  configuration conf_tb_siso_gen_calc5_8_mix_post of tb_siso_gen_top is
5  for top
6    for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
7      generic map (word_length => 8);
8    for structure
9      for duv: siso_gen use entity work.siso_gen(flat_calc5_8_5);
10     end for;
11    for tv: tv_siso_gen use entity work.tv_siso_gen(file_io)
12      generic map (word_length => 8, -- code does not compile w/o this
13                  half_clock_period => 2.5 ns,
14                  in_file_name => "mix8.in",
15                  out_file_name => "mix8.out");
16    end for;
17  end for;
18 end for;
19 end for;
20 end conf_tb_siso_gen_calc5_8_mix_post;
21
22 configuration conf_tb_siso_gen_calc5_8_nul_post of tb_siso_gen_top is
23 for top
24   for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
25     generic map (word_length => 8);
26   for structure
27     for duv: siso_gen use entity work.siso_gen(flat_calc5_8_5);
28     end for;
29     for tv: tv_siso_gen use entity work.tv_siso_gen(file_io)
30       generic map (word_length => 8, -- code does not compile w/o this
31                   half_clock_period => 2.5 ns,
32                   in_file_name => "nul8.in",
33                   out_file_name => "nul8.out");
34     end for;
35   end for;
36 end for;
37 end for;
38 end conf_tb_siso_gen_calc5_8_nul_post;
39
40 configuration conf_tb_siso_gen_calc5_8_add_post of tb_siso_gen_top is
41 for top
42   for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
43     generic map (word_length => 8);
44   for structure
45     for duv: siso_gen use entity work.siso_gen(flat_calc5_8_5);
46     end for;
47     for tv: tv_siso_gen use entity work.tv_siso_gen(file_io)
48       generic map (word_length => 8, -- code does not compile w/o this
49                   half_clock_period => 2.5 ns,
50                   in_file_name => "add8.in",
51                   out_file_name => "add8.out");
52     end for;
53   end for;
54 end for;
55 end for;
56 end conf_tb_siso_gen_calc5_8_add_post;
57
58 configuration conf_tb_siso_gen_calc5_8_mul_post of tb_siso_gen_top is
59 for top
60   for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
```

```

61     generic map (word_length => 8);
62   for structure
63     for duv: siso_gen use entity work.siso_gen(flat_calc5_8_5);
64     end for;
65     for tvcl: tvcl_siso_gen use entity work.tvcl_siso_gen(file_io)
66       generic map (word_length => 8, -- code does not compile w/o this
67         half_clock_period => 2.5 ns,
68         in_file_name => "mul8.in",
69         out_file_name => "mul8.out");
70     end for;
71   end for;
72 end for;
73 end for;
74 end conf_tb_siso_gen_calc5_8_mul_post;
75
76 configuration conf_tb_siso_gen_calc5_gc_8_mix_post of tb_siso_gen_top is
77   for top
78     for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
79       generic map (word_length => 8);
80     for structure
81       for duv: siso_gen use entity work.siso_gen(flat_calc5_gc_8_5);
82       end for;
83       for tvcl: tvcl_siso_gen use entity work.tvcl_siso_gen(file_io)
84         generic map (word_length => 8, -- code does not compile w/o this
85           half_clock_period => 2.5 ns,
86           in_file_name => "mix8.in",
87           out_file_name => "mix8.out");
88       end for;
89     end for;
90   end for;
91 end for;
92 end conf_tb_siso_gen_calc5_gc_8_mix_post;
93
94 configuration conf_tb_siso_gen_calc5_gc_8_nul_post of tb_siso_gen_top is
95   for top
96     for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
97       generic map (word_length => 8);
98     for structure
99       for duv: siso_gen use entity work.siso_gen(flat_calc5_gc_8_5);
100      end for;
101      for tvcl: tvcl_siso_gen use entity work.tvcl_siso_gen(file_io)
102        generic map (word_length => 8, -- code does not compile w/o this
103          half_clock_period => 2.5 ns,
104          in_file_name => "nul8.in",
105          out_file_name => "nul8.out");
106        end for;
107      end for;
108    end for;
109  end for;
110 end conf_tb_siso_gen_calc5_gc_8_nul_post;
111
112 configuration conf_tb_siso_gen_calc5_gc_8_add_post of tb_siso_gen_top is
113   for top
114     for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
115       generic map (word_length => 8);
116     for structure
117       for duv: siso_gen use entity work.siso_gen(flat_calc5_gc_8_5);
118       end for;
119       for tvcl: tvcl_siso_gen use entity work.tvcl_siso_gen(file_io)
120         generic map (word_length => 8, -- code does not compile w/o this

```

```
121         half_clock_period => 2.5 ns,
122         in_file_name => "add8.in",
123         out_file_name => "add8.out");
124     end for;
125 end for;
126 end for;
127 end for;
128 end conf_tb_siso_gen_calc5_gc_8_add_post;
129
130 configuration conf_tb_siso_gen_calc5_gc_8_mul_post of tb_siso_gen_top is
131     for top
132         for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
133             generic map (word_length => 8);
134             for structure
135                 for duv: siso_gen use entity work.siso_gen(flat_calc5_gc_8_5);
136                 end for;
137                 for tvk: tvk_siso_gen use entity work.tvk_siso_gen(file_io)
138                     generic map (word_length => 8, -- code does not compile w/o this
139                                 half_clock_period => 2.5 ns,
140                                 in_file_name => "mul8.in",
141                                 out_file_name => "mul8.out");
142                 end for;
143             end for;
144         end for;
145     end for;
146 end conf_tb_siso_gen_calc5_gc_8_mul_post;
147
148
```

```

1  -- this architecture needs arithmetic functions
2  library ieee;
3  use ieee.numeric_std.all;
4  architecture calc5 of siso_gen is
5    -- state for ALU
6    -- two "read opcode (opc)" states, the first does not raise "ready"
7    type state is (read_opc_init, read_opc_ready1, read_opc_ready2,
8                  read_left1,   read_left2,   read_right);
9    signal cur_state: state;
10   signal nxt_state: state;
11
12   -- internal registers for left and right ALU input and ALU
13   -- output, all having the same width
14
15   signal add1_reg:   signed(word_length-1 downto 0);
16   signal add2_reg:   signed(word_length-1 downto 0);
17   signal mult1_reg:  signed(word_length-1 downto 0);
18   signal mult2_reg:  signed(word_length-1 downto 0);
19
20   -- temp register
21   signal temp_reg:   signed(word_length-1 downto 0);
22
23   signal result_reg: signed(2*word_length-1 downto 0);
24   -- at most 16 different operations are supported by this ALU
25   signal opcode_reg: std_logic_vector(3 downto 0);
26
27   -- and their next values
28   signal add1_reg_nxt:   signed(word_length-1 downto 0);
29   signal add2_reg_nxt:   signed(word_length-1 downto 0);
30   signal mult1_reg_nxt:  signed(word_length-1 downto 0);
31   signal mult2_reg_nxt:  signed(word_length-1 downto 0);
32   signal temp_reg_nxt:   signed(word_length-1 downto 0);
33
34
35   signal result_nxt:   signed(2*word_length-1 downto 0);
36   signal opcode_nxt:   std_logic_vector(3 downto 0);
37
38   -- output of adder
39   signal adder_out: signed(word_length-1 downto 0);
40
41   -- output of multiplier
42   signal mult_out: signed(2*word_length-1 downto 0);
43
44   -- next value for ready
45   signal ready_nxt: std_logic;
46
47 begin
48   -- the next process is sequential and only sensitive to clk and reset
49   seq: process(clk, reset)
50   begin
51     if (reset = '1')
52     then
53       add1_reg    <= (others => '0');
54       add2_reg    <= (others => '0');
55       mult1_reg   <= (others => '0');
56       mult2_reg   <= (others => '0');
57       opcode_reg  <= (others => '0');
58       result_reg  <= (others => '0');
59       temp_reg    <= (others => '0');
60       ready       <= '0';

```

```

61     cur_state    <= read_opc_init;
62     elsif rising_edge(clk)
63     then
64         add1_reg    <= add1_reg_nxt;
65         add2_reg    <= add2_reg_nxt;
66         mult1_reg   <= mult1_reg_nxt;
67         mult2_reg   <= mult2_reg_nxt;
68         temp_reg    <= temp_reg_nxt;
69         opcode_reg  <= opcode_nxt;
70         result_reg  <= result_nxt;
71         ready       <= ready_nxt;
72         cur_state   <= nxt_state;
73     end if;
74 end process seq;
75
76 ----- combinational next-value process -----
-----
77 nxt: process(cur_state)
78 begin
79     case cur_state is
80     when read_opc_init =>
81         nxt_state    <= read_left1;
82     when read_left1 | read_left2 =>
83         nxt_state    <= read_right;
84     when read_right =>
85         -- multiplication needs two cycles to output result
86         if (opcode_reg = "0010")
87         then
88             nxt_state    <= read_opc_ready2;
89         else
90             nxt_state    <= read_opc_ready1;
91         end if;
92     when read_opc_ready1 =>
93         nxt_state    <= read_left1;
94     when read_opc_ready2 =>
95         nxt_state    <= read_left2;
96     end case;
97 end process nxt;
98 -----
-----
99
100 ----- In Registers Process -----
-----
101 inRegistersProcess : process(data_in, cur_state, add1_reg,add2_reg, mult1_reg,
mult2_reg, temp_reg, opcode_reg)
102 begin
103     case(cur_state) is
104     when read_left1 | read_left2 =>
105         case opcode_reg is
106         when "0001" | "0010" => -- addition
107             add1_reg_nxt    <= add1_reg;
108             add2_reg_nxt    <= add2_reg;
109             mult1_reg_nxt   <= mult1_reg;
110             mult2_reg_nxt   <= mult2_reg;
111             temp_reg_nxt    <= signed(data_in);
112         when others => -- non-implemented codes behave like null command
113             add1_reg_nxt    <= add1_reg;
114             add2_reg_nxt    <= add2_reg;
115             mult1_reg_nxt   <= mult1_reg;
116             mult2_reg_nxt   <= mult2_reg;

```

```

117         temp_reg_nxt    <= temp_reg;
118     end case;
119 when read_right =>
120     case opcode_reg is
121     when "0001" => -- addition
122         add1_reg_nxt    <= temp_reg;
123         add2_reg_nxt    <= signed(data_in);
124         mult1_reg_nxt   <= mult1_reg;
125         mult2_reg_nxt   <= mult2_reg;
126         temp_reg_nxt    <= temp_reg;
127     when "0010" => -- multiplication
128         add1_reg_nxt    <= add1_reg;
129         add2_reg_nxt    <= add2_reg;
130         mult1_reg_nxt   <= temp_reg;
131         mult2_reg_nxt   <= signed(data_in) ;
132         temp_reg_nxt    <= temp_reg;
133     when others => -- non-implemented codes behave like null command
134         add1_reg_nxt    <= add1_reg;
135         add2_reg_nxt    <= add2_reg;
136         mult1_reg_nxt   <= mult1_reg;
137         mult2_reg_nxt   <= mult2_reg;
138         temp_reg_nxt    <= temp_reg;
139     end case;
140 when others =>
141     add1_reg_nxt    <= add1_reg;
142     add2_reg_nxt    <= add2_reg;
143     mult1_reg_nxt   <= mult1_reg;
144     mult2_reg_nxt   <= mult2_reg;
145     temp_reg_nxt    <= temp_reg;
146 end case ;
147 end process ; -- inRegistersProcess
148 -----
149
150 ----- Opcode Process -----
151 opcodeProcess : process(cur_state, data_in)
152 begin
153     case(cur_state) is
154     when read_opc_ready1 | read_opc_ready2 | read_opc_init =>
155         opcode_nxt    <= data_in(3 downto 0);
156
157     when others =>
158         opcode_nxt    <= opcode_reg;
159     end case ;
160 end process ; -- opcodeProcess
161 -----
162
163 ----- Result Process -----
164 resultProcess : process(cur_state,opcode_reg , adder_out, mult_out)
165 begin
166     case(cur_state) is
167     when read_opc_ready1 | read_opc_ready2 =>
168         case opcode_reg is
169         when "0000" => -- the null result
170             result_nxt <= (others => '0');
171         when "0001" => -- addition
172             result_nxt(word_length - 1 downto 0) <= adder_out;
173             result_nxt(2*word_length - 1 downto word_length) <= (others => '0');
174         when "0010" => -- multiplication
175             result_nxt <= mult_out;
176         when others => -- non-implemented codes behave like null command

```

```
177         result_nxt <= (others => '0');
178     end case;
179     when others =>
180         result_nxt <= result_reg;
181     end case ;
182 end process ; -- resultProcess
183 -----
184
185 ----- Ready Process -----
186 readyProcess : process(cur_state)
187 begin
188     case(cur_state) is
189         when read_left2 | read_opc_ready1 | read_opc_ready2 =>
190             ready_nxt <= '1';
191         when others =>
192             ready_nxt <= '0';
193     end case ;
194 end process ; -- readyProcess
195 -----
196 -- adder, wrap around in case of overflow, so discard carry
197 adder_out <= add1_reg + add2_reg;
198
199 -- multiplier
200 mult_out <= mult1_reg * mult2_reg;
201
202 -- output register is lowest half of result_reg, except when second
203 -- part of multiplication result needs to be output
204 data_out <= std_logic_vector(result_reg(2*word_length - 1 downto word_length))
205         when cur_state = read_left2
206         else std_logic_vector(result_reg(word_length - 1 downto 0));
207
208 -- this block should receive data in every clock cycle
209 req <= '1';
210 end calc5;
```