Homework DAT.

Pietro Pennestrì - s2382660

Files attached to report

1) `cmp_add_ctrl_mygcd_arch`
Architecture for `mygcd`.

2) `conf_tb_siso_gen_dpctrl`
Configuration for pre syn. sim.

3) `conf_tb_siso_gen_dpctrlPost`
Configuration for post syn. sim.

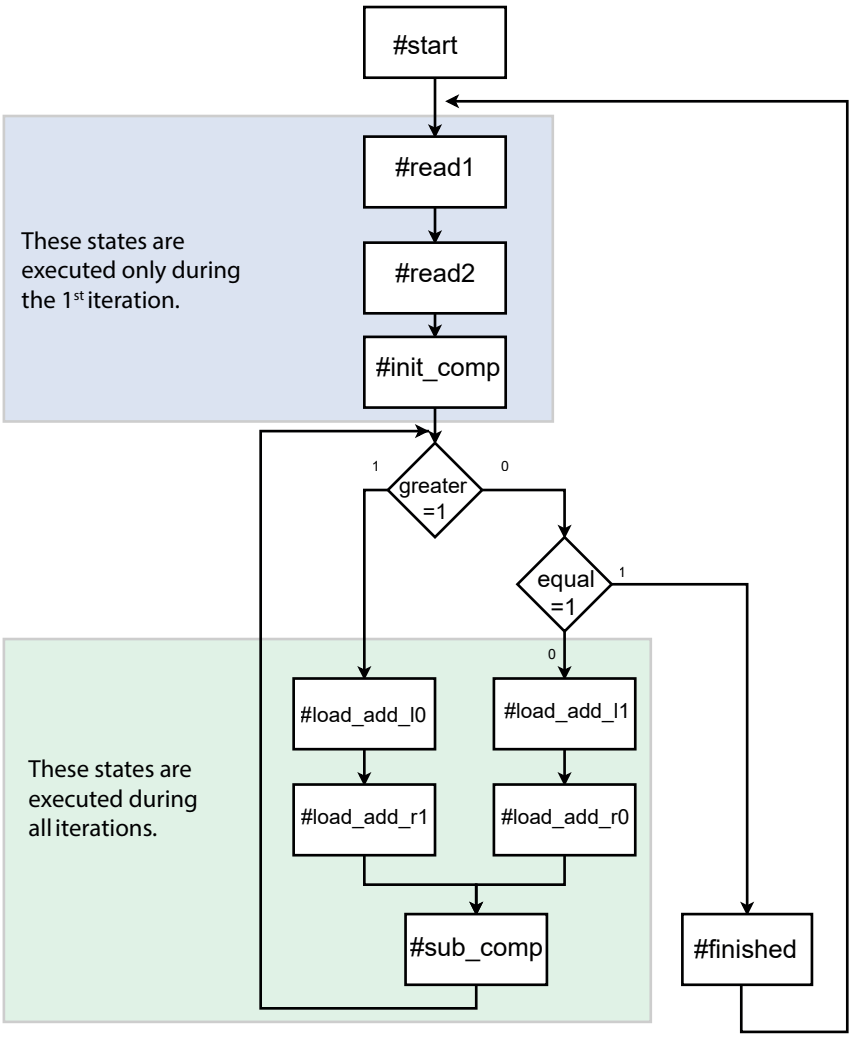


Figure 1: `mygcd` architecture

#0	data_in(0) or min(cmp_l, cmp_r)
#1	data_in(1) or subtraction result
#2	NOT USED
#3	NOT USED

Table 1: Register file organization

	Original	mygcd
1st iteration	7 clock cycles	6 clock cycles
All other iterations	5 clock cycles	3 clock cycles

Table 2: Clock cycles optimization

	Original	mygcd
Slack (ns)	4.71	4.70
Area (nm^2)	21026	20929

Table 3: Synthesis comparison for a clock period of 10ns

The original control unit does not take advantage that the register file, in the data path can be read and written at the same time. The new control unit architecture (`mygcd`), depicted in Figure 1, exploits this specification to reduce the number of clock cycles per iteration. Table 2 summarizes the improvements obtained with the new architecture.

The behaviour of the states in `mygcd` are the followings:

-) `#start`: Initialization of the system.
-) `#read1`: The input data is written in `reg(0)`.
-) `#read2` : The input data is written in `reg(1)`. The content of reg(0) loaded in `cmp_l`.
-) `#init_comp`: The content of `reg(1)` is loaded in `cmp_r`.
-) `#load_add_l0`: The content of `reg(0)` is loaded into `add_l`.
-) `#load_add_l1`: The content of `reg(1)` is loaded into `add_l`.
-) `#load_add_r0`: The content of `reg(0)` is loaded into `add_r` and `cmp_l`.
-) `#load_add_r1`: The content of `reg(1)` is loaded into `add_r` and `cmp_l`. `reg(0)` is set equal to `reg(1)`.
-) `#sub_comp`: The result of the subtraction is stored in `reg(1)` and `cmp_r`.
-) `#finished`: This state is visited when the computation is completed.

Simulations. The system has been tested **successfully**, both in pre and post synthesis, with inputs from `gcd16_large.txt` input file. However, for graphical purposes, in the following we will show only the computation of the gcd for 16 and 4.
Slack and area for the synthesis of `mygcd` are reported in Table 3.

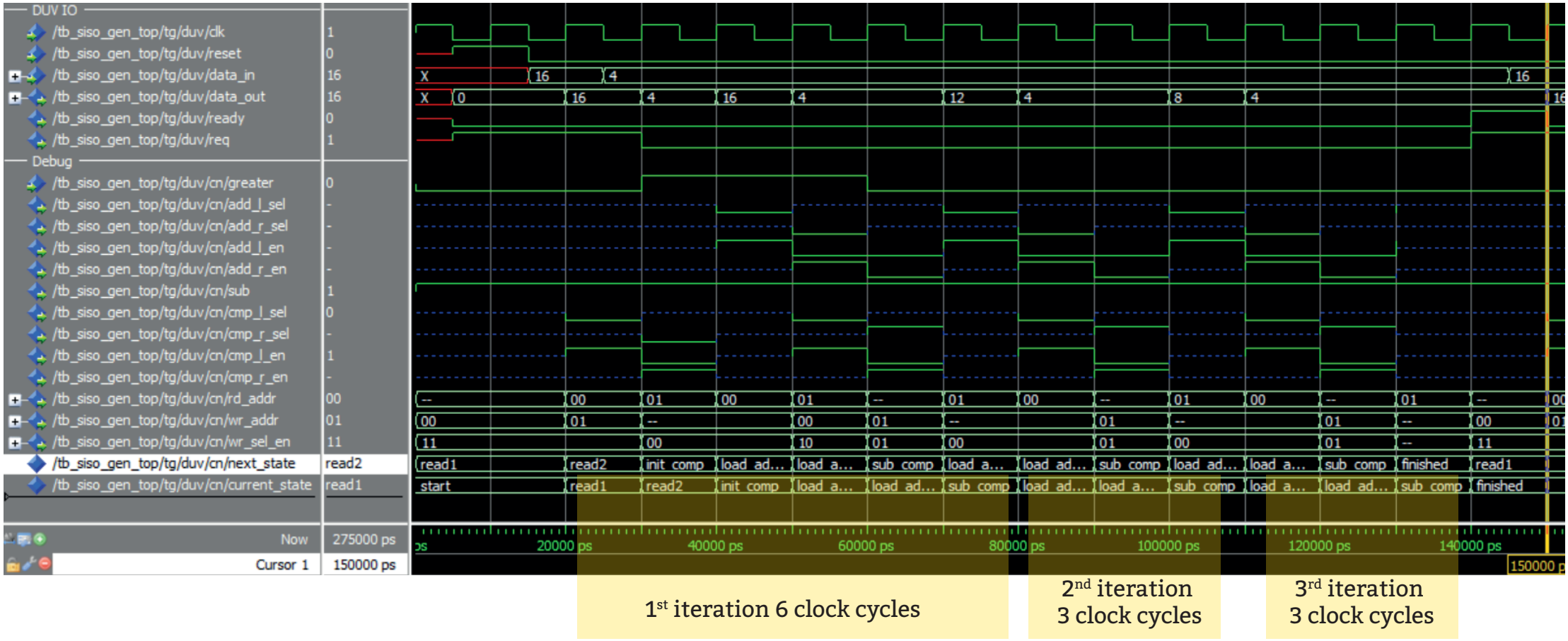
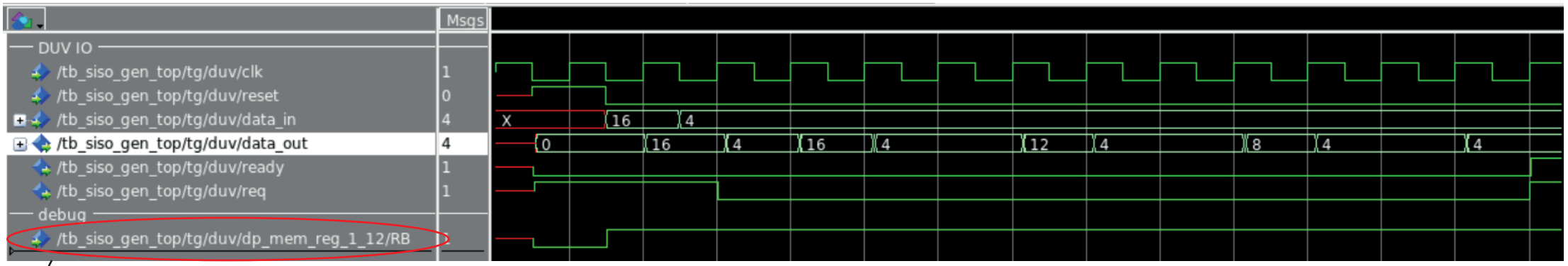


Figure 2: Pre-Synthesis Simulation Input: 16,4 clock period 10ns



Flip Flop clear signal cannot be accessed from pre-synthesis simulation.

Figure 3: Post-Synthesis Simulation Input: 16,4 clock period 10ns

```vhdl
architecture mygcd of cmp_add_ctrl is

    -- definition of type state
    type state is (
        start,read1, read2, init_comp, load_add_l0,
        load_add_l1, load_add_r0, load_add_r1,sub_comp, finished
    );
    signal current_state, next_state: state;

begin -- begin architecture

    seq : process(clk, reset)
    begin
        if reset='1' then
            -- reset action
            current_state <= start;
            req <= '1';
            ready <= '0';
        elsif(rising_edge(clk)) then
            current_state <= next_state;

            if (next_state = read1) or (next_state = finished) then
              req <= '1';
            else
              req <= '0';
            end if;

            if next_state = finished then
              ready <= '1';
            else
              ready <= '0';
            end if;

        end if ;
    end process ; -- end process seq


    new_state : process(current_state, equal, greater)
    begin
        case(current_state) is
            when start => next_state <= read1;
            when read1 => next_state <= read2;
            when read2 => next_state <= init_comp;
            when init_comp =>
                if equal= '1' then
                    next_state <= finished;
                elsif greater = '1' then
                    next_state <= load_add_l0;
                else
                    next_state <= load_add_l1;
                end if ;

            when load_add_l0 => next_state <= load_add_r1;
            when load_add_l1 => next_state <= load_add_r0;

            when load_add_r0 => next_state <= sub_comp;
            when load_add_r1 =>  next_state <= sub_comp;

            when sub_comp =>
                if equal= '1' then
```

```
61                  next_state <= finished;
62              elsif greater = '1' then
63                  next_state <= load_add_l0;
64              else
65                  next_state <= load_add_l1;
66              end if ;
67          when finished => next_state <= read1;
68      end case ;
69  end process ; -- end process new state
70
71  outputs : process(next_state)
72  begin -- process outputs
73      sub <= '1';
74      case(next_state) is
75
76          when start =>
77              -- nothing happens. It can be considered an init state of the system
78              -- adder configuration
79              add_l_sel <= '-'; add_l_en <= '-';
80              add_r_sel <= '-'; add_r_en <= '-';
81
82              -- cmp configuration
83              cmp_l_sel <= '-'; cmp_l_en <= '-';
84              cmp_r_sel <= '-'; cmp_r_en <= '-';
85              -- reg file configuration
86              rd_addr <= "--"; wr_addr <= "--";
87              wr_sel_en <= "--";
88          when read1 =>
89              -- read data from data_in port and writing
90              -- the content of reg[0] = data_in
91              -- adder configuration
92              add_l_sel <= '-'; add_l_en <= '-';
93              add_r_sel <= '-'; add_r_en <= '-';
94              -- cmp configuration
95              cmp_l_sel <= '-'; cmp_l_en <= '-';
96               cmp_r_sel <= '-'; cmp_r_en <= '-';
97              -- reg file configuration
98              rd_addr <= "--"; wr_addr <= "00";
99              wr_sel_en <= "11";
100
101         when read2 =>
102             -- read data from data_in port and writing
103             -- the content of reg[1] = data_in
104             -- load the left comparator register with reg[0]
105
106             -- adder configuration
107             add_l_sel <= '-'; add_l_en <= '-';
108             add_r_sel <= '-'; add_r_en <= '-';
109
110             -- cmp configuration
111             cmp_l_sel <= '0'; cmp_l_en <= '1';
112             cmp_r_sel <= '-'; cmp_r_en <= '-';
113             -- reg file configuration
114             rd_addr <= "00"; wr_addr <= "01";
115             wr_sel_en <= "11";
116
117         when init_comp =>
118             -- load the rigth comparator register with reg[1]
119
120             -- adder configuration
```

```
121                 add_l_sel <= '-'; add_l_en <= '-';
122                 add_r_sel <= '-'; add_r_en <= '-';
123
124                 -- cmp configuration
125                 cmp_l_sel <= '-'; cmp_l_en <= '0';
126                 cmp_r_sel <= '0'; cmp_r_en <= '1';
127                 -- reg file configuration
128                 rd_addr <= "01"; wr_addr <= "--";
129                 wr_sel_en <= "00";
130
131             when load_add_l0 =>
132                 -- load left register of adder with reg[0]
133
134                 -- adder configuration
135                 add_l_sel <= '0'; add_l_en <= '1';
136                 add_r_sel <= '-'; add_r_en <= '-';
137
138                 -- cmp configuration
139                 cmp_l_sel <= '-'; cmp_l_en <= '-';
140                 cmp_r_sel <= '-'; cmp_r_en <= '-';
141                 -- reg file configuration
142                 rd_addr <= "00"; wr_addr <= "--";
143                 wr_sel_en <= "00";
144
145             when load_add_l1 =>
146                 -- load left register of adder with reg[1]
147
148                 -- adder configuration
149                 add_l_sel <= '0'; add_l_en <= '1';
150                 add_r_sel <= '-'; add_r_en <= '-';
151
152                 -- cmp configuration
153                 cmp_l_sel <= '-'; cmp_l_en <= '-';
154                 cmp_r_sel <= '-'; cmp_r_en <= '-';
155                 -- reg file configuration
156                 rd_addr <= "01"; wr_addr <= "--";
157                 wr_sel_en <= "00";
158
159
160             when load_add_r0 =>
161                 -- load rigth register of adder and comp left register with reg[0]
162
163                 -- adder configuration
164                 add_l_sel <= '-'; add_l_en <= '0';
165                 add_r_sel <= '0'; add_r_en <= '1';
166
167                 -- cmp configuration
168                 cmp_l_sel <= '0'; cmp_l_en <= '1';
169                 cmp_r_sel <= '-'; cmp_r_en <= '-';
170                 -- reg file configuration
171                 rd_addr <= "00"; wr_addr <= "--";
172                 wr_sel_en <= "00";
173
174
175             when load_add_r1 =>
176                 -- load rigth register of adder and comp lef register with reg[1]
177                 -- set reg(0) = reg(1)
178
179                 -- adder configuration
180                 add_l_sel <= '-'; add_l_en <= '0';
```

```
181                 add_r_sel <= '0'; add_r_en <= '1';
182 
183                 -- cmp configuration
184                 cmp_l_sel <= '0'; cmp_l_en <= '1';
185                 cmp_r_sel <= '-'; cmp_r_en <= '-';
186                 -- reg file configuration
187                 rd_addr <= "01"; wr_addr <= "00";
188                 wr_sel_en <= "10";
189 
190             when sub_comp =>
191                 -- load rigth register of comp and reg[1] with sub result
192 
193                 -- adder configuration
194                 add_l_sel <= '-'; add_l_en <= '0';
195                 add_r_sel <= '-'; add_r_en <= '0';
196 
197                 -- cmp configuration
198                 cmp_l_sel <= '-'; cmp_l_en <= '0';
199                 cmp_r_sel <= '1'; cmp_r_en <= '1';
200                 -- reg file configuration
201                 rd_addr <= "--"; wr_addr <= "01";
202                 wr_sel_en <= "01";
203 
204 
205             when finished =>
206 
207                 -- adder configuration
208                 add_l_sel <= '-'; add_l_en <= '-';
209                 add_r_sel <= '-'; add_r_en <= '-';
210 
211                 -- cmp configuration
212                 cmp_l_sel <= '-'; cmp_l_en <= '-';
213                 cmp_r_sel <= '-'; cmp_r_en <= '-';
214                 -- reg file configuration
215                 rd_addr <= "01"; wr_addr <= "--";
216                 wr_sel_en <= "--";
217 
218 
219         end case ;
220 
221     end process; -- outputs
222 
223 end mygcd ; -- mygcd arch
```

```vhdl
configuration conf_tb_siso_gen_dpctrl of tb_siso_gen_top is
  for top
    for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
            generic map (word_length => 16);
      for structure
        for duv: siso_gen use entity work.siso_gen(dpctrl);
          for dpctrl
            for dp: cmp_add_dp use entity work.cmp_add_dp(behavioral);
            end for;
            for cn: cmp_add_ctrl use entity work.cmp_add_ctrl(mygcd);
            end for;
          end for;
        end for;
        for tvc: tvc_siso_gen use entity work.tvc_siso_gen(file_io)
            generic map (word_length => 16,
                         half_clock_period => 5 ns,
                         in_file_name => "gcd16Small.in",
                         out_file_name => "gcd16.out");
        end for;
      end for;
    end for;
  end for;
end conf_tb_siso_gen_dpctrl;

```

```
1
2  configuration conf_tb_siso_gen_dpctrl of tb_siso_gen_top is
3    for top
4      for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
5              generic map (word_length => 16);
6        for structure
7          for duv: siso_gen use entity work.siso_gen(flat_dpctrl_gcd_16_10);
8          end for;
9          for tvc: tvc_siso_gen use entity work.tvc_siso_gen(file_io)
10             generic map (word_length => 16,
11                          half_clock_period => 5 ns,
12                          in_file_name => "gcd16Small.in",
13                          out_file_name => "gcd16.out");
14         end for;
15       end for;
16     end for;
17   end for;
18 end conf_tb_siso_gen_dpctrl;
19
```