Assignment 2 - Pietro Pennestrì s2382660

Contents

1 Circuit Description **1**

2 Simulations **2**

3 Pareto optimal points **3**

Uploaded Files

The following files are attached to the report:

- `siso_gen_gcdOneComp_arch.pdf` The pdf print of the proposed architecture.
- `gcd19.xlsx` : time-area trade-off graph.

1 Circuit Description

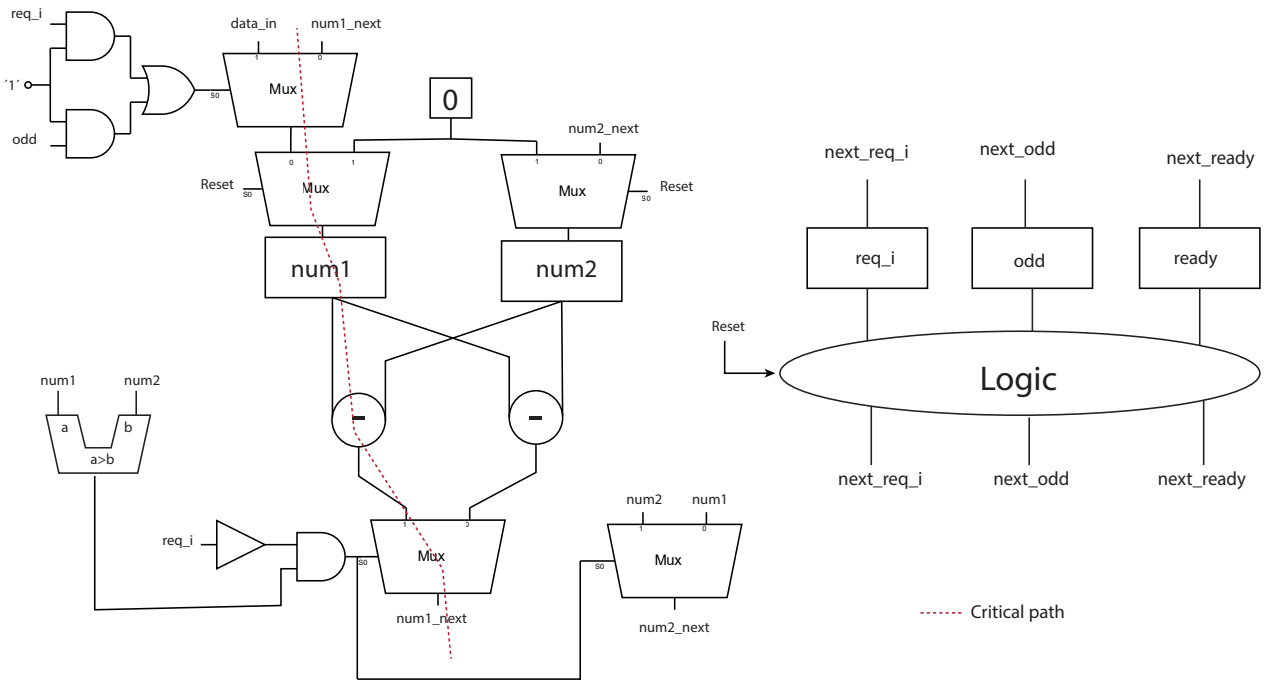

Figure 1: a

The following modifications have been made to improve the `gcd` architecture of Assignment 1:

- One of the two comparator was removed to reduce the area of the architecture.
- The logic for `num2_next` was designed in such a way that always `num2 <= num2_next` (if `reset` signal is not raised high). In particular during the the input loading phase we will have the following behaviour:
  - `req_i = 1 and odd = 0` : then `num1 = data_in` and `num2_next = num_1`
  - `req_i = 0 and odd = 1` : then `num1 = data_in` and `num2 = num2_next`

The presence of two subtractors has a negative effect on the area specifications, but the slack time of the architecture is increased.
The proposed architecture, as well as the original one, works for input pairs `(a,b)` such that `a!0 and b!=0` . If this condition is violated an infinite loop occurs.

2 Simulations

The simulation waveform for the pre-synthesis simulation is depicted in Figure 2.

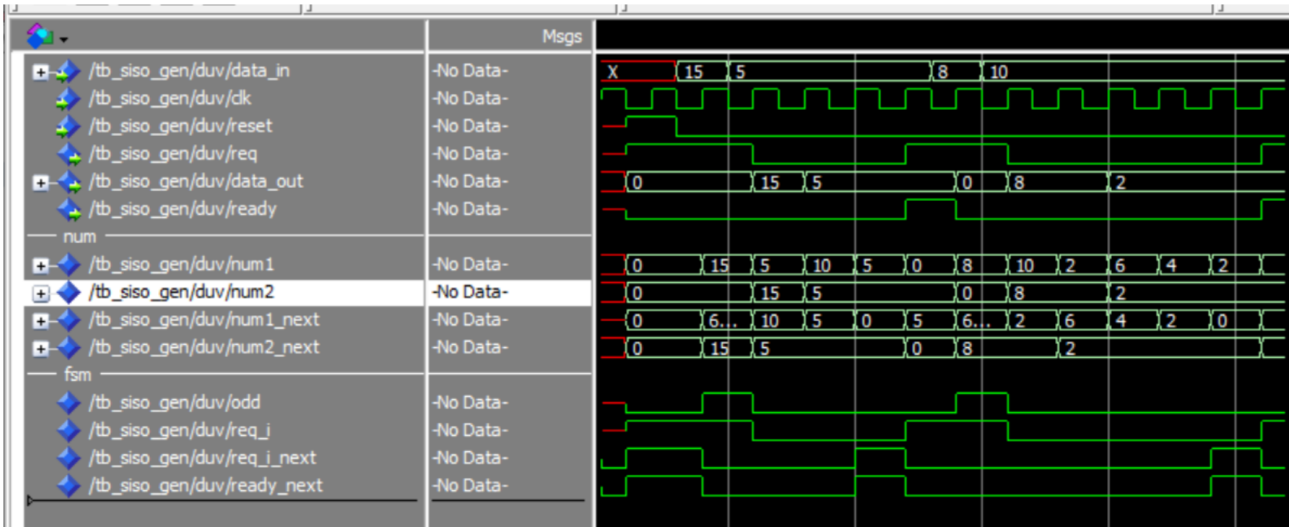

Figure 2: `@clock=5ns` Pre-synthesis simulation

The simulation waveform for the post-synthesis simulation is depicted in Figure 3.

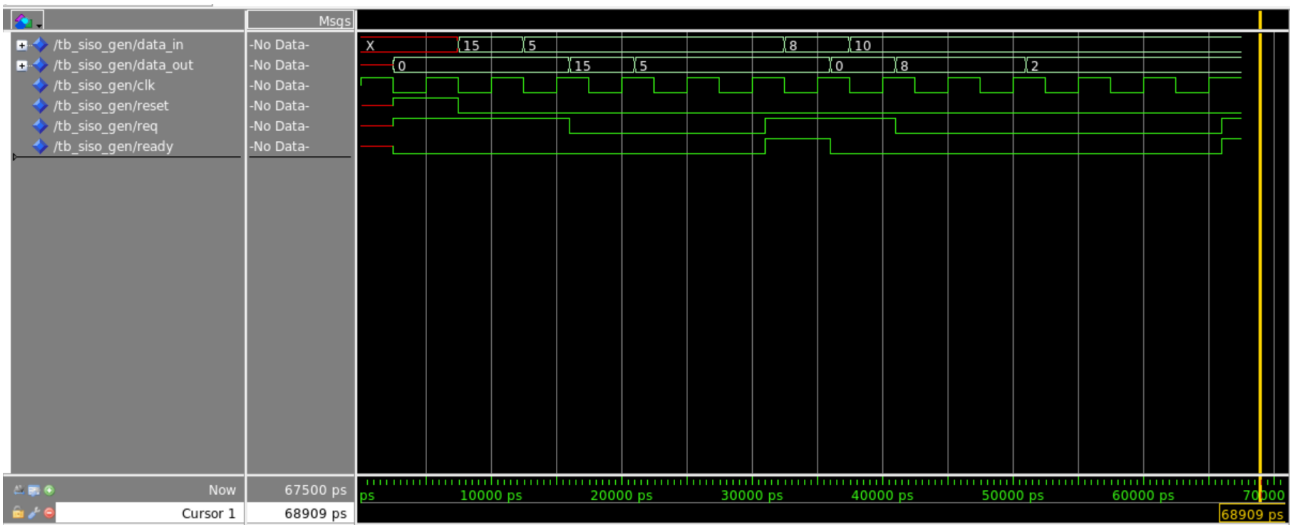

Figure 3: `@clock=5ns` Post-synthesis simulation

3 Pareto optimal points

The modified `excell` file is depicted in Figure 4

constraint	slack	gcd time	gcd area
1,50	0,01	1,49	10010
2,00	0,03	1,97	9566
2,50	0,15	2,35	9233
3,00	0,04	2,96	8598
3,50	0,53	2,97	8570
4,00	1,03	2,97	8570
5,00	2,03	2,97	8570

constraint	slack	clev. time	clev. area
1,50	0,01	1,49	8867
2,00	0,07	1,93	8119
2,50	0,16	2,34	7773
3,00	0,17	2,83	7436
3,50	0,01	3,49	6854
4,00	0,13	3,87	6818
5,00	1,13	3,87	6818

Constraint	Slack proposed	Time Proposed	Area
1,5	0,03	1,47	8887
2,00	0,01	1,99	8338
2,50	0,02	2,48	7838
3,00	0,06	2,94	7285
3,50	0,56	2,94	7244
4,00	1,03	2,97	7244
5,00	2,06	2,94	7244

Pareto time	Pareto area
1,49	8867
1,93	8119
2,34	7773
2,83	7436
3,49	6854
3,87	6818

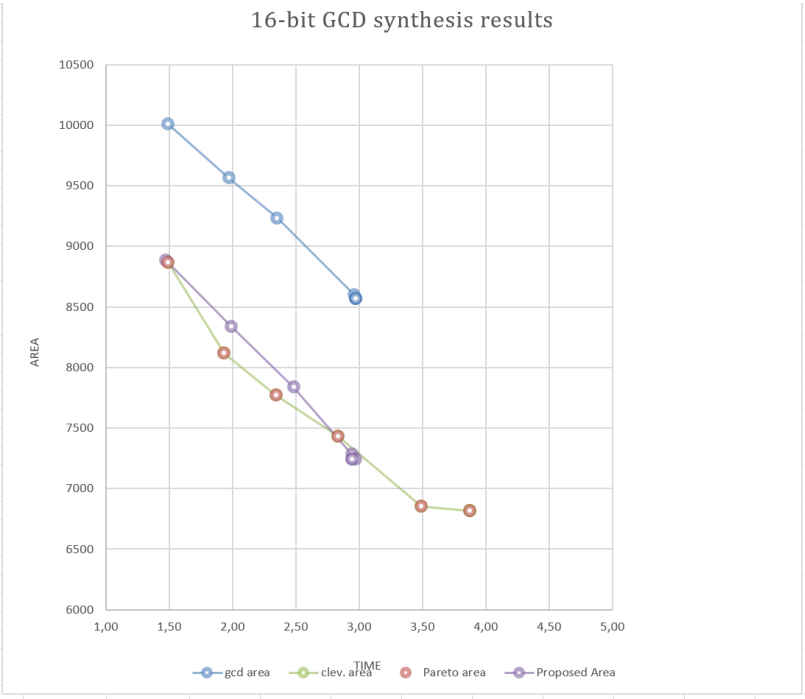

Figure 4: Solutions Comparison

Regarding area, in the worst case my solution is 5.88 % larger then the *clever solution*.

Since it was decided to maintain two subtractors, this downgrading, was expected from the beginning of the design because. However, regarding slack time, significant increases are observed in the proposed solution. The observed improvement, is due to the removal of one multiplexer in the combinational path of `next_num2`.
The following changes can be implemented in future architectures:

- Remove one of the two subtractors, without affecting too much on the slack specifications;
- Remove the infinite loop caused by inputs of the form `(0,a)`, `(a,0)` and `(0,0)`
- Add to the current architecture the following rules to compute the `gcd`:

$$\begin{aligned} \gcd(2a, 2b+1) &= \gcd(a, 2b+1) \\ \gcd(2a, 2b) &= 2\gcd(a, b) \\ \gcd(a, 0) &= a \end{aligned}$$

Of course all these additional rules decrease area performance index, but decrease the clock cycles number to obtain the result.

```vhdl
-- this architecture needs arithmetic functions
library ieee;
use ieee.numeric_std.all;

architecture gcdOneComp of siso_gen is
  -- registers
  signal num1, num2: unsigned(word_length-1 downto 0);
  signal odd, req_i: std_logic;
  -- wires
  signal num1_next, num2_next: unsigned(word_length-1 downto 0);
  signal req_i_next, ready_next: std_logic;

begin
  -- the next process is sequential and only sensitive to clk and reset
  seq: process(clk, reset)
  begin
    if (reset = '1')
    then
      num1 <= (others => '0');
      num2 <= (others => '0');
      odd <= '0';
      req_i <= '1'; -- the system is ready to receive data after reset
      ready <= '0';
    elsif rising_edge(clk)
    then

        if ((req_i = '1') or (odd = '1') )  then
            num1 <= unsigned(data_in);
        else
            num1 <= num1_next;
        end if;

        num2 <= num2_next;

        if ((req_i = '1') and (odd = '0'))
        then
            req_i <= '1';
        else
            req_i <= req_i_next;
        end if;


        if ((req_i = '1') and (odd = '0'))
        then
            odd <= '1';
        else
            odd <= '0';
        end if;

        if ((req_i = '1') ) then
            ready <= '0';
        else
            ready <= ready_next;
        end if;


    end if; -- (reset = '1')
  end process seq;

  -- combinational process
```

```
61    next_val: process(num1, num2)
62    begin
63
64      if (num1 > num2  and req_i = '0')
65      then
66          num1_next <= num1 - num2;
67          num2_next <=  num2 ;
68      else
69          num1_next <= num2 - num1 ;
70          num2_next <=  num1 ;
71      end if;
72
73
74      if (num1 = num2)
75      then
76        ready_next <= '1';
77        req_i_next <= '1';
78      else
79        ready_next <= '0';
80        req_i_next <= '0';
81      end if;
82
83
84    end process next_val;
85
86    -- output register read only from num2 !
87    data_out <= std_logic_vector(num2);
88
89    -- req wires to req_i
90    req <= req_i;
91  end gcdOneComp;
92
```