Assignment 1 - Pietro Pennestrì s2382660

1 Circuit Description

The block diagram for the `n3p1` architecture is presented in Figure 1. Separate `VHDL` processes have been used to describe combinational and sequential logic. To avoid overflow during the computations of the sequence the maths operation resolution is set to `2in_data_len` + 4 where `in_data_len` is the input data resolution[1]. This design choice is justified from the graph of Figure 2.

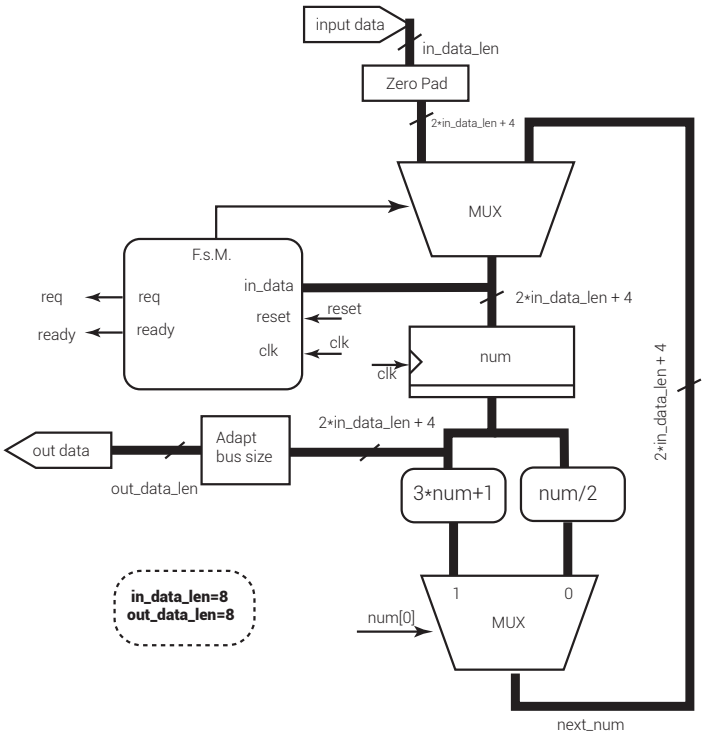


Figure 1: Block diagram for the `n3p1` architecture

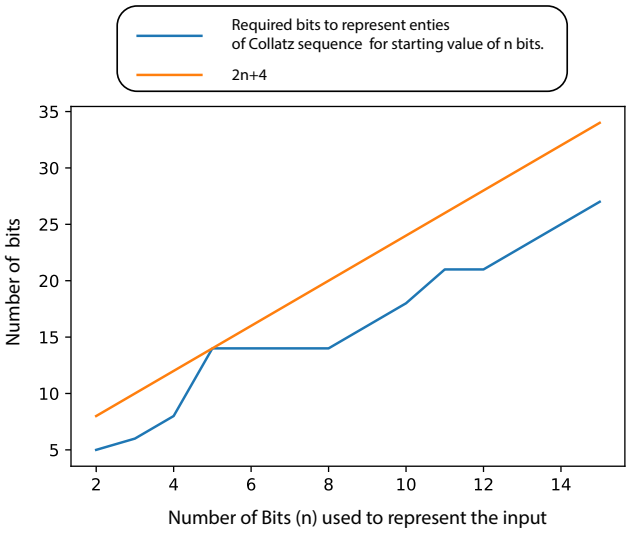


Figure 2: Required bits to represent enties of Collatz sequence for starting value of n bits.

According to the specifications, however the output resolution is set to 8 bits. This means that the content of `num` register is truncated when presented as `out_data`. For example, if the input data is 255_{10} the first element of the Collatz sequence cannot be represented with 8 bits.

A finite state machine (F.s.M.) is designed to control the operations of the `n3p1` architecture. In particular the F.s.M, controls:

- the input of the `num` register and checks whenever the next iteration value is equal to unity;
- output signals `req` and `ready`, whose behaviour is defined in the specifications.

If a zero is given as input an infinite loop is avoided by asking for a new data. It was decided that during this iteration still the `ready` signal would have been raised high and the `out_data` is set to 0_{10}. The user will thus have the possibility to detect that a wrong input data was provided by reading 0_{10} as output.

2 File List

The following `VHDL` files are attached to this design:

- `siso8_ent.vhd` The entity declaration of the siso8 circuit.
- `siso8_n3p1_arch.vhd` The description of the n3p1 architecture
- `tvc_siso8.vhd` The test vector controller for siso8.
- `tb_siso8.vhd` The entities and architectures of the testbench for the siso8 circuit.
- `conf_tb_siso8_n3p1.vhd` Configuration file.

[1] According to the specifications the input and output resolution is set to 8 bits. However, in `vhdl` files this information is set as a *generic* parameter such that it can be easily modified.

Example 1

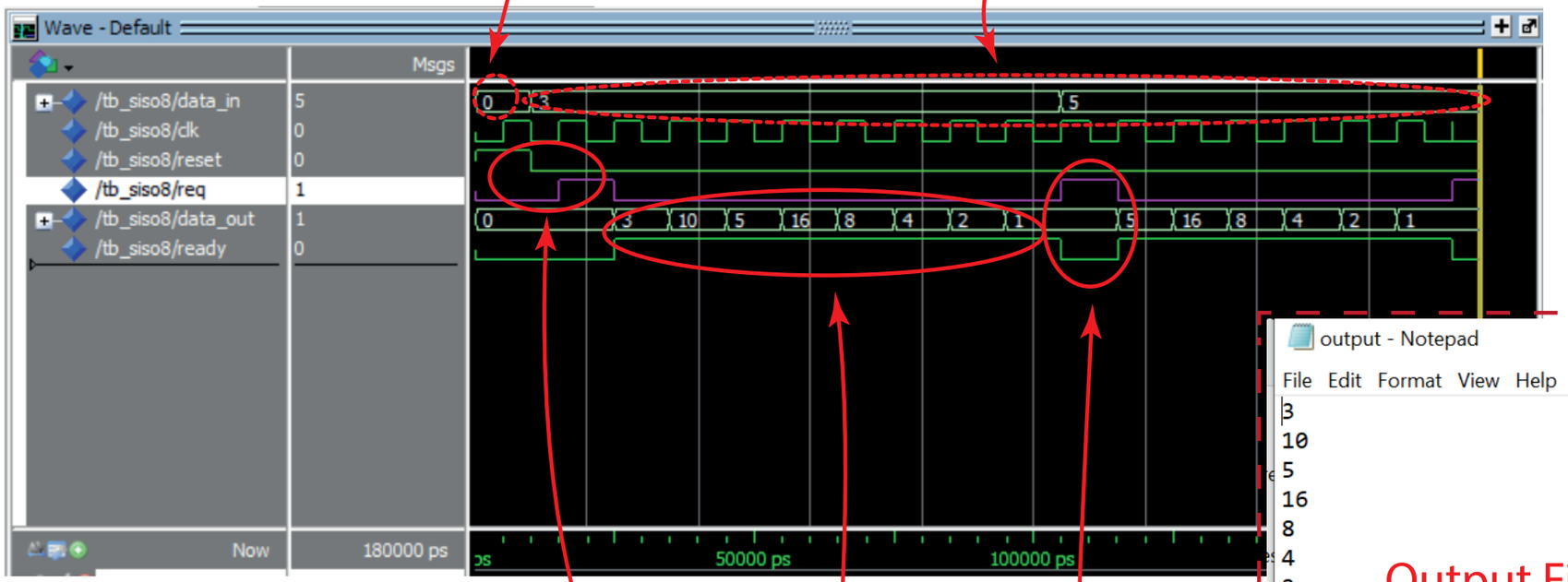
Before reading data from the input file
data_in is set to 0.
(check tvc_siso8.vhd)

Input taken from input file
(check tvc_siso8.vhd)

input - Notepad

File Edit Format View

3
5



output - Notepad

File Edit Format View Help

3
10
5
16
8
4
2
1
5
16
8
4
2
1

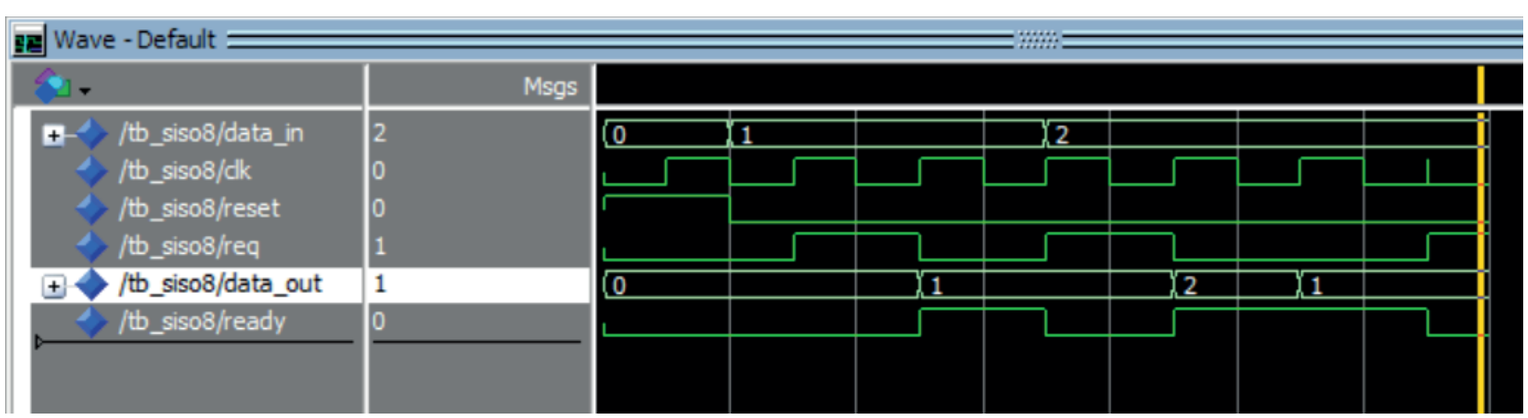
Output File

After the reset signal
a data request is made.

One clock cycle after the data request computation starts.
Ready signal is high during this process. Output data is
written to a file

Ready signal is low
during data request.

Example 2



```vhdl
-----------------------------------------------
-- File Description: Configuration file.
-- Author: Pietro Pennestri
-- Student No: s2382660
-- mail: pietro.pennestri@gmail.com
-----------------------------------------------

configuration conf_tb_siso8_n3p1 of tb_siso8 is
    for structure
        for dut: siso8 use entity work.siso8(n3p1);
        end for;
        for tvc: tvc_siso8 use entity work.tvc_siso8(behavior);
        end for;
    end for;
end conf_tb_siso8_n3p1;
```

```vhdl
----------------------------------------------
-- File Description: The entity declaration of the siso8 circuit.
-- Author: Pietro Pennestri
-- Student No: s2382660
-- mail: pietro.pennestri@gmail.com
----------------------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity siso8 is
    generic(
        data_in_len : integer;
        data_out_len : integer
      );
    port (
        data_in: in std_logic_vector(data_in_len-1 downto 0);
        clk: in std_logic;
        reset: in std_logic;
        req: out std_logic;
        data_out: out std_logic_vector(data_out_len-1 downto 0);
        ready: out std_logic
        );
end siso8;
```

```vhdl
-----------------------------------------------
-- File Description: The description of the n3p1 architecture
-- Author: Pietro Pennestri
-- Student No: s2382660
-- mail: pietro.pennestri@gmail.com
-----------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


architecture n3p1 of siso8 is

     constant math_resolution : integer := 2*data_in'length + 4;
     signal num : std_logic_vector(math_resolution-1 downto 0);
     signal next_num : std_logic_vector(math_resolution-1 downto 0);

begin
    seq : process( clk, reset ) -- this is a sequential process (The F.s.M.)
        type state_type is (at_rst, waiting_for_input, accepting_input ,processing, data_ready);
        constant unity : std_logic_vector(num'length-1 downto 0) := std_logic_vector(to_unsigned(1, num'length));
        variable state : state_type := waiting_for_input;
        variable next_state : state_type := waiting_for_input;
    begin
        if (reset='1') then
            -- reset actions
            ready <= '0';
            req <= '0';
            state := at_rst;
            next_state := waiting_for_input;
            num <= (others=>'0');
        elsif (rising_edge(clk)) then
            state := next_state;
            case(state) is
                when waiting_for_input =>
                    -- in this state the fsm waits for an
                    -- input from the user 'req' raised HIGH
                    ready <= '0';
                    req <= '1';
                    next_state := accepting_input;

                when accepting_input =>
                    -- The input from the user is assigned to
                    -- num and the 'ready' signal is raised HIGH.
                    num(data_in'length-1 downto 0) <= data_in ;
                num(num'length-1 downto data_in'length ) <= (others=>'0');
                    ready <= '1';
                    req <= '0';
                    if( data_in(data_in'length-1 downto 1)=unity(data_in'length-1 downto 1) ) then
                        -- check if input data (in decimal) is 0 or 1. In this case asks for new input.
                        next_state := waiting_for_input;
                    else
                        next_state := processing;
                    end if;
                when processing =>
```

```
57                      num <= next_num;
58                      if (next_num = unity) then
59                          next_state := waiting_for_input;
60                      end if ;
61
62                  when others =>
63                      ready <= '0';
64                      req <= '0';
65                      next_state := waiting_for_input;
66              end case ;
67
68          end if ;
69      end process ; -- seq process
70
71      next_val : process(num) -- combinational process
72      begin
73
74          -- num is adapted to be assigned to data_out. Note that according to the size
    of data_out, num can be
75          -- either truncated or zero-padded. To avvoid truncation the recommended size
    for data_out is
76          -- 2*data_in'length + 4. However if know that len_data_out < len_num then an
    easier definition of
77          -- data_out is  data_out <= num(data_out'length-1 downto 0);
78          data_out <= std_logic_vector(to_unsigned( to_integer(unsigned(num)) ,
    data_out'length)) ;
79
80          if (num(0)= '1') then
81              -- action to perform when number is odd
82              -- multiply by three + 1
83              next_num  <= std_logic_vector( unsigned(num) + unsigned((num(num'length-2
    downto 0) & "1"))  )  ;
84          else
85              -- action to perform when number is even
86              -- divide by two
87              next_num <= "0" & num(num'length-1 downto 1);
88          end if ;
89      end process ; -- next_val process
90
91  end n3p1 ; -- arch
```

```vhdl
----------------------------------------------
-- File Description: The entities and architectures of the testbench for the siso8 circuit.
-- Author: Pietro Pennestri
-- Student No: s2382660
-- mail: pietro.pennestri@gmail.com
----------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity tb_siso8 is
    generic(
        data_in_len : integer := 8;
        data_out_len : integer := 8
      );
end tb_siso8;

architecture structure of tb_siso8 is

    component siso8
        generic(
            data_in_len : integer;
            data_out_len : integer
        );
        port (data_in  : in std_logic_vector (data_in_len-1 downto 0);
                clk      : in std_logic;
                reset    : in std_logic;
                req      : out std_logic;
                data_out : out std_logic_vector (data_out_len-1 downto 0);
                ready    : out std_logic);
    end component;

    component tvc_siso8 is
        generic(
            data_in_len : integer;
            data_out_len : integer
        );
        port (
          siso8_data_in : out std_logic_vector(data_in_len -1 downto 0);
          siso8_clk : out std_logic;
          siso8_reset : out std_logic;
          siso8_req: in std_logic;
          siso8_data_out: in std_logic_vector(data_out_len -1 downto 0);
          siso8_ready: in std_logic
        );
    end component;

    signal data_in  : std_logic_vector (data_in_len-1 downto 0);
    signal clk      : std_logic;
    signal reset    : std_logic;
    signal req      : std_logic;
    signal data_out : std_logic_vector (data_out_len-1 downto 0);
    signal ready    : std_logic;



begin
```

```
60
61      dut : siso8
62      generic map (data_in_len => data_in_len,
63                   data_out_len => data_out_len
64      )
65      port map (  data_in  => data_in,
66                  clk      => clk,
67                  reset    => reset,
68                  req      => req,
69                  data_out => data_out,
70                  ready    => ready);
71
72      tvc : tvc_siso8
73      generic map (data_in_len => data_in_len,
74                   data_out_len => data_out_len
75      )
76      port map (  siso8_data_in  => data_in,
77                  siso8_clk      => clk,
78                  siso8_reset    => reset,
79                  siso8_req      => req,
80                  siso8_data_out => data_out,
81                  siso8_ready    => ready);
82
83  end structure;
```

```vhdl
----------------------------------------------
-- File Description: The test vector controller for siso8.
-- Author: Pietro Pennestri
-- Student No: s2382660
-- mail: pietro.pennestri@gmail.com
----------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use STD.textio.all;
use ieee.std_logic_textio.all;

entity tvc_siso8 is
  generic(
    data_in_len : integer;
    data_out_len : integer
  );
  port (
    siso8_data_in : out std_logic_vector(data_in_len -1 downto 0);
    siso8_clk : out std_logic;
    siso8_reset : out std_logic;
    siso8_req: in std_logic;
    siso8_data_out: in std_logic_vector(data_out_len -1 downto 0);
    siso8_ready: in std_logic
  );
end tvc_siso8;

architecture behavior of tvc_siso8 is
    -- Signals for clock
    constant TbPeriod : time := 10 ns;
    signal TbClock : std_logic := '0';
    signal TbSimEnded : std_logic := '0';

    -- Signals for files
    file file_in : text;
    file file_out : text;

begin -- begin architecture
    -- Clock generation
    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';
    siso8_clk <= TbClock;
    -- The stimuli process sets the inputs for siso8
    stimuli : process
      variable iline : line;
      -- variable input_data : std_logic_vector(siso8_data_in'length-1 downto 0); -- use this definition if in the input files data are in binary
      variable input_data : integer; -- use this definition if in the input files data are in decimal
    begin -- process stimuli
      -- the input file should be located in the same directory of
      -- the modelsim project
      file_open(file_in,"input.txt",read_mode); -- open input.txt in read mode
      siso8_reset <= '1'; -- reset before starting to test
      siso8_data_in <= (others=>'0');
      wait for TbPeriod;
      siso8_reset <= '0';
      -- read input data line by line from the input file
      while not endfile(file_in) loop
        readline(file_in, iline);
```

```
59          read(iline, input_data);
60          --siso8_data_in <= input_data; -- use this definition if in the input files
data are in binary
61          siso8_data_in <= std_logic_vector(to_unsigned(input_data,
siso8_data_in'length)); -- use this definition if in the input files data are in
decimal
62          wait until siso8_req = '0';
63          wait until siso8_ready = '0';
64        end loop;
65
66        file_close(file_in); -- close input.txt
67        -- Stop the clock and hence terminate the simulation
68        TbSimEnded <= '1';
69        wait;
70
71      end process ; -- stimuli
72
73      writing : process
74          variable oline : line;
75          variable int_data_out : integer;
76      begin
77         file_open(file_out, "output.txt", write_mode);
78         while not TbSimEnded = '1' loop
79            if(siso8_ready='1') then
80              int_data_out := to_integer(unsigned(siso8_data_out));
81              write(oline, integer'image(int_data_out));
82              --write(oline, siso8_data_out , right, siso8_data_out'length);
83              writeline(file_out, oline);
84            end if;
85            wait for TbPeriod;
86        end loop ;
87        file_close(file_out);
88        wait;
89      end process ; -- writing
90
91  end behavior ; -- behavior
```