Management and analysis of physics datasets, Part. 1

Seventh Laboratory

Stefano Pavinato
20/12/2018

Istituto Nazionale di Fisica Nucleare
Laboratori Nazionali di Legnaro

Outline

Outline

Goals

- Practice with VHDL Moore FSMs.

VHDL naming convention

Signals/components	Name
Clock	*clk*
Reset	*rst*
Input Port	*port_in*
Output Port	*port_out*
VHDL file name	*entityname.vhd*
Test bench file name	*tb_entityname.vhd*
Signal between 2 comps	*sign_cmp1_cmp2*
Process name	*p_name*
state name	*s_name*
...	...

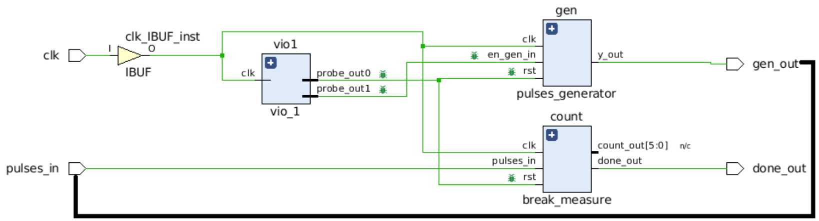
Outline

Presentation

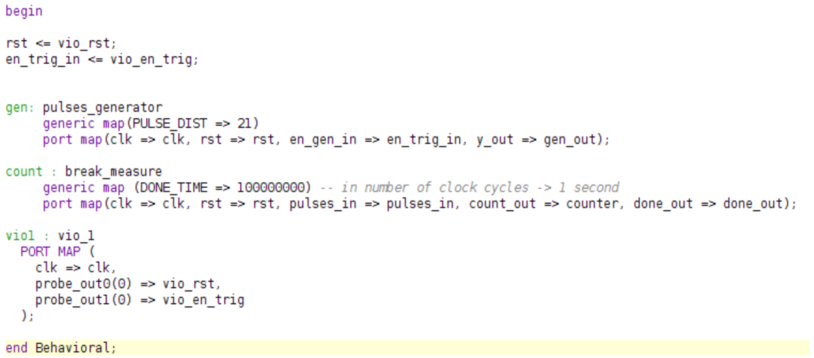
- You have three design source files (*.vhd*) and one constraint file (*.xdc*).
- There is a top module (*top.vhd*), a two pulses generator (*pulses_generator.vhd*) and a third file (*break_measure.vhd*).
- You have to write code only in the *break_measure.vhd* file.

Circuit description

The VIO core was used in order to generate a reset (*rst*) signal and an *en_gen* signal. The last signal enables the generation of two pulses. You can substitute the two signals with two switches and/or buttons or you have to generate the VIO core.



Circuit description - VHDL

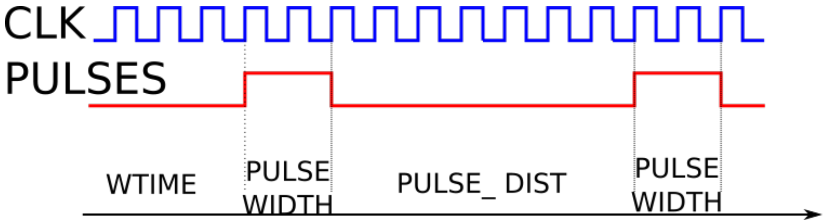


```vhdl
begin

rst <= vio_rst;
en_trig_in <= vio_en_trig;


gen: pulses_generator
     generic map(PULSE_DIST => 21)
     port map(clk => clk, rst => rst, en_gen_in => en_trig_in, y_out => gen_out);

count : break_measure
     generic map (DONE_TIME => 100000000) -- in number of clock cycles -> 1 second
     port map(clk => clk, rst => rst, pulses_in => pulses_in, count_out => counter, done_out => done_out);

viol : vio_1
  PORT MAP (
    clk => clk,
    probe_out0(0) => vio_rst,
    probe_out1(0) => vio_en_trig
  );

end Behavioral;
```

Two pulses generator (1)

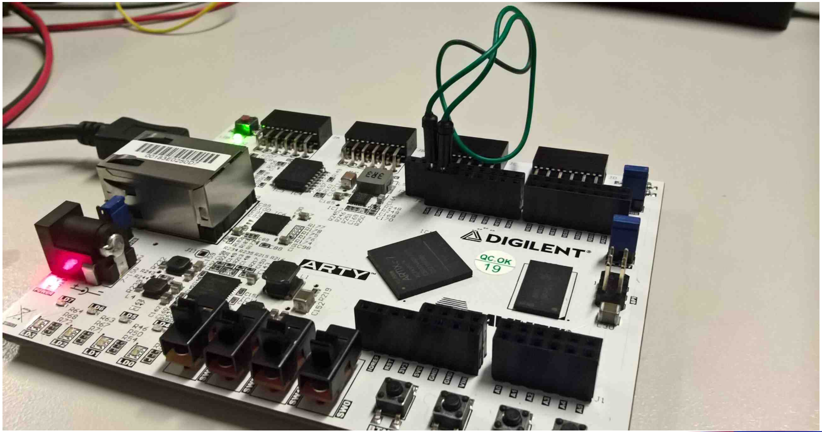
- It was implemented as a Moore FSMs.
- It has three inputs: the clock, the reset and the enable generator *en_gen* signals. When the reset is done $0 \rightarrow 1 \rightarrow 0$ and there is a rising edge of the *en_gen* signal this core generates two pulses.
- This component is configurable with three parameters (*generics*):
  1. WTIME: after a time $\mathrm{WTIME} * T_{clock}$ the first pulse in generated;
  2. PULSE_WIDTH: the duration of each pulse is $\mathrm{PULSE\_WIDTH} * T_{clock}$
  3. PULSE_DIST: the second pulse is generated after $\mathrm{PULSE\_DIST} * T_{clock}$ from the falling edge of the first pulse.

Two pulses generator (2)



Circuit description - Board

The pulses at the output of the generator component are the input of the *break_measure* component. So, if you do not modify the constraint file, you have to short *IO*41 with *IO*40.

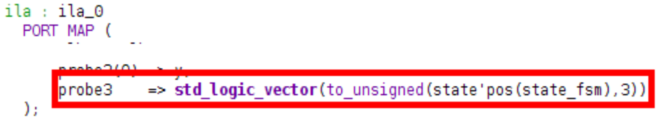


Measure of the time between the pulses

- You have to write the architecture of the file *break_measure.vhd*.
- This component has to count the number of clock cycles between the falling edge of the first pulse and the rising edge of the second pulse.
- It has two outputs:
  1. *count_out* : it is an unsigned type. It represents the numbers of clock cycles between the two pulses (It is equals to *PULSE_DIST*). In order to visualize this value an ILA core has to be instantiated.
  2. *done_out* : it is connected to a led. When the count is done the led has to be on for one second.

Hints

- You have to implement a Moore FSM.
- Four states are enough.
- A reverse engineering of the FSM used to implement the two pulses generator can be a source of inspiration.
- The ILA core can be instantiated or in the top module or in the *break_measure.vhd* file.
- In order to monitor the state of the FSM through the ILA core you can use this example code (copied from the *pulses_generator.vhd* file):



```
ila : ila_0
  PORT MAP (

      probe2(0) => y,
      probe3    => std_logic_vector(to_unsigned(state'pos(state_fsm),3))
  );
```

Outline

Homework

- **This homework will be graded.**
- Write a report where there is:
  1. the code of the *break_measure.vhd* architecture;
  2. the screenshot of the ILA triggered in the rising edge of the *done_out* signal;
  3. NOTHING ELSE.
- **This homework is for the 15th January.**