Práctica 3 SED

Por simón Mateo de Pedraza

Contenido

Tarea 1: completar el process

Se le da prioridad al reset. Se asigna la señal con el flanco de sbida del reloj.

```
begin state_register:
    process (RESET, CLK)
    begin
        --COMPLETAR
        if reset='0' then
            current_state <= S0;
        elsif CLK'event and CLK='1' then
            current_state <= next_state;
        end if;
    end process;
```

Tarea 2: crear testbench

Se crea un testbench con la ayuda del generador de testbench online lapinoo

VHDL Testbench

```
-- Testbench automatically generated online
-- at https://vhdl.lapinoo.net
-- Generation date : 15.11.2021 09:07:16 UTC

library ieee;
use ieee.std_logic_1164.all;

entity tb_fsm is
end tb_fsm;

architecture tb of tb_fsm is

    component fsm
        port (RESET      : in std_logic;
              CLK        : in std_logic;
              PUSHBUTTON : in std_logic;
              LIGHT      : out std_logic_vector (0 to 3));
    end component;

    signal RESET      : std_logic;
    signal CLK        : std_logic;
    signal PUSHBUTTON : std_logic;
    signal LIGHT      : std_logic_vector (0 to 3);

    constant TbPeriod : time := 100 ns; -- EDIT Put right period here
    signal TbClock : std_logic := '0';
    signal TbSimEnded : std_logic := '0';

begin

    dut : fsm
    port map (RESET      => RESET,
              CLK        => CLK,
              PUSHBUTTON => PUSHBUTTON,
              LIGHT      => LIGHT);

    -- Clock generation
    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

    -- EDIT: Check that CLK is really your main clock signal
    CLK <= TbClock;

    stimuli : process
    begin
        -- EDIT Adapt initialization as needed
        PUSHBUTTON <= '0';

        -- Reset generation
        -- EDIT: Check that RESET is really your reset signal
```

```
        RESET <= '1';
        wait for 100ns;
        RESET <= '0';
        wait for 100ns;

        -- EDIT Add stimuli here

        --wait for 100 * TbPeriod;
        --Estado 1
        PUSHBUTTON <= '1';
        wait for 100ns;
        PUSHBUTTON <= '0';
        wait for 100ns;

        --Estado 2
        PUSHBUTTON <= '1';
        wait for 100ns;
        PUSHBUTTON <= '0';
        wait for 100ns;

        --Estado 3
        PUSHBUTTON <= '1';
        wait for 100ns;
        PUSHBUTTON <= '0';
        wait for 100ns;

        --Estado 0
        PUSHBUTTON <= '1';
        wait for 100ns;
        PUSHBUTTON <= '0';
        wait for 100ns;

        --Estado 1
        PUSHBUTTON <= '1';
        wait for 100ns;
        PUSHBUTTON <= '0';
        wait for 100ns;

        --reseteo estado 0
        RESET <= '1';
        wait for 100 ns;
        RESET <= '0';
        wait for 100 ns;

        -- Stop the clock and hence terminate the simulation
        TbSimEnded <= '1';
        wait;
    end process;

end tb;

-- Configuration block below is required by some simulators. Usually no need to
edit.

configuration cfg_tb_fsm of tb_fsm is
    for tb
    end for;
end cfg_tb_fsm;
```

Tarea 3: implementación en la placa

Fichero de restricciones

Utilizamos el reloj de la placa, 4 leds, 1 botón para cambiar y el reset de la placa.

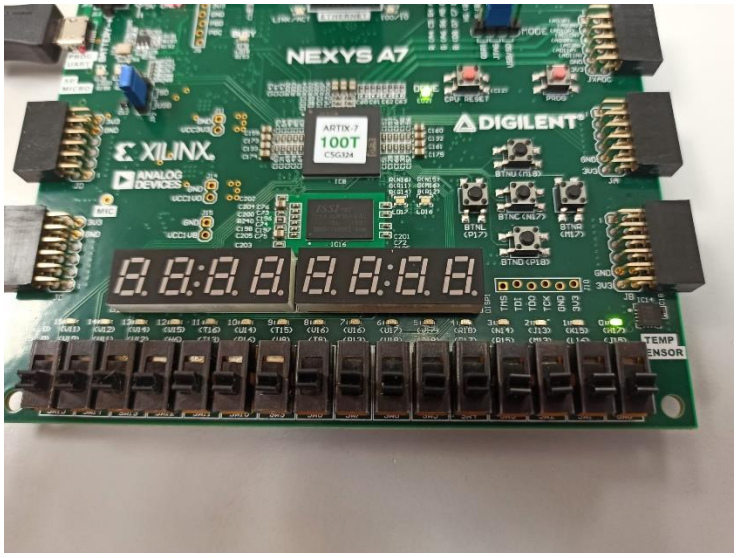
```
6  ## Clock signal
7  set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
8  #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];
9
```

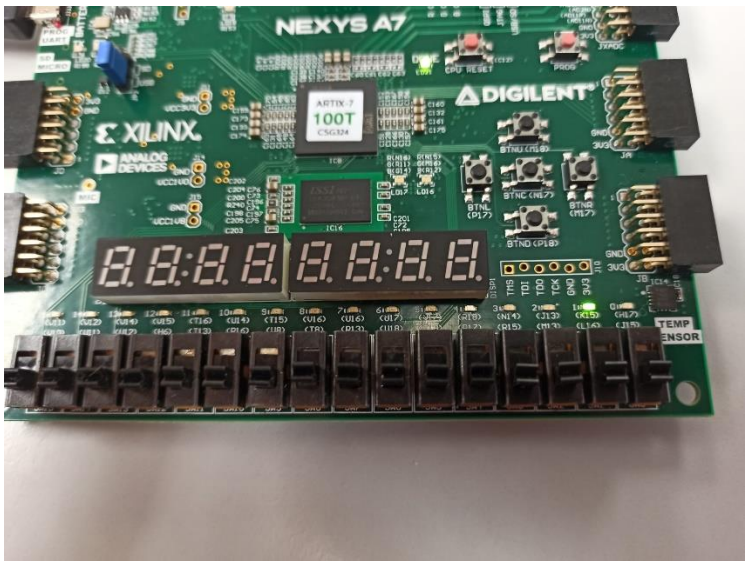
```
33 set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {LIGHT[0]}]
34 set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {LIGHT[1]}]
35 set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {LIGHT[2]}]
36 set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports {LIGHT[3]}]
37 #set_property -dict {PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports {LED[4]}]
```

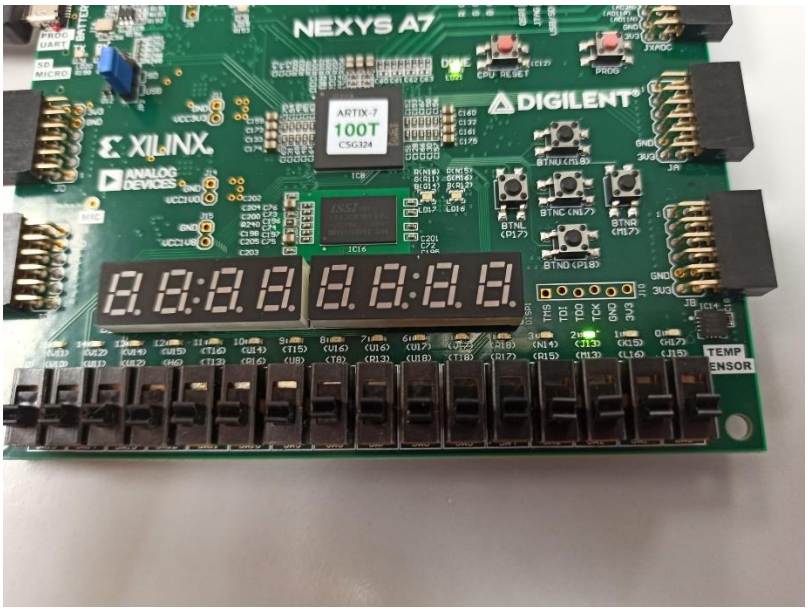
```
82 set_property -dict { PACKAGE_PIN C12   IOSTANDARD LVCMOS33 } [get_ports { RESET }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn
83
84 set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { PUSHBUTTON }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

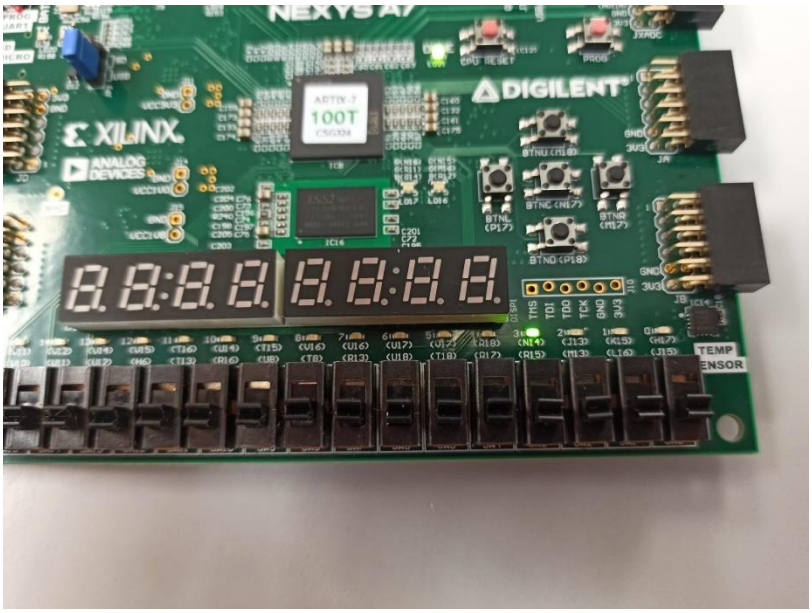
Pulsando el botón

Observamos como pasa del estado cero a los siguietes estados mediante la luz que se va “moviendo” hacia la izquierda cada vez que se pulsa el botón.

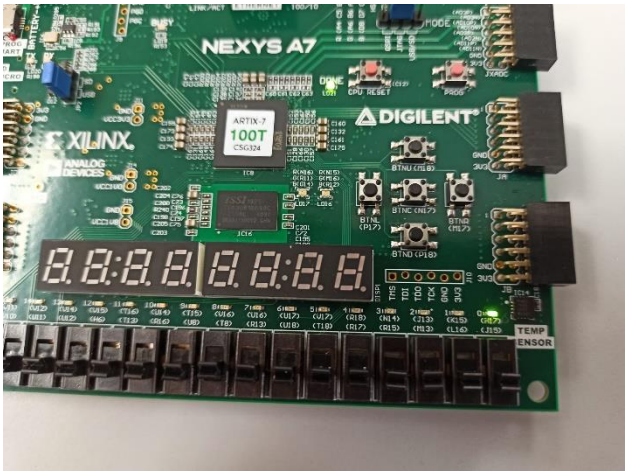








Pulsando reset



Conclusiones

Resulta sorprendentemente sencillo implementar una máquina de estados en una FPGA de la forma vista en la práctica. Estoy seguro de que será de gran ayuda para los trabajos de la asignatura.

También se ve que si no se añade algo para contrarrestar el rebote mecánico de los botones, esto puede ser un problema, a si que se debe tener en cuenta a la hora de diseñar productos con una FPGA.