VLSI Digital Design

LAB ASSIGNMENT 1

Practical aspects of synchronous digital design

0. Objectives

This laboratory work is intended to bring to light some problems that turn up when unsuitable design techniques are used. After illustrating cases of defective designs several necessary modifications are proposed to be carried out to correct the problems that arise.

The final goal of the lab assignment is that the student adopts a completely synchronous design methodology that avoids an important number of situations as the previously shown ones while simplifying system debugging. In general, asynchronous designs will be avoided, except in very well characterized and delimited situations (for example, asynchronous communication interfaces).

1. Work specification

In general, when designing digital systems of certain complexity, all the sequential elements of the system are switched by means of a single active edge of a unique clock signal. The clock is distributed throughout the circuit with the minimal delays and trying to equalize the instant when it attains all the circuit flip-flops, in order to minimize skew.

In this lab assignment, simple circuits are used to show some of the problems and undesired effects that asynchronous designs can produce. These problems become worse with the increase of design complexity.

1.1. 1, 3, 5 detector

Three VHDL descriptions of a subsystem, that has to detect codes 1, 3 and 5 from the 4-bit counter output and send them to a 7-segment decoder, are provided. The decoder output is connected to a led display that shows the current code.

Each code version, prac1_1a.vhd, prac1_1b.vhd and prac1_1c.vhd is provided in the annex. Notice that all the three versions use a frequency divider of the mclk clock, which operates at 50 MHz.

Tasks to be carried out:

- Analyze carefully each code version, trying to find out eventual problems of operation.
- Synthesize the descriptions and observe the resulting elements. Indicate whether they are the expected ones or not.
- Simulate the post-synthesis descriptions with delays. Are the results correct? Why?
  Important note: To speed up the simulation time, reduce the number of bits of the frequency divider, changing *cntClk* to the range *28 downto 23.* Modify the necessary sentences and synthesize the new version.
- Make a new description *prac1_1d.vhd* that avoids the former designs' problems. Synthesize and simulate it

Optional task:

- File *main_prac1_1.vhd* allows incorporating any of the former entities as a component. To check real-time operation, the synthesized and compiled file can be uploaded to the lab Xilinx evaluation board. It is necessary to introduce, together with the EDIF file generated by the synthesizer, the pin configuration UCF constraints file that will be provided in the lab, substituting the one eventually obtained from the synthesis process.
  In order to be able to observe operation with a visible time scale, **it is necessary to restore the signal cntClk range to *28 downto 0***.

1.2. Modulo-16 counter

Second part of this lab consists in showing an operation fault that may arise in asynchronous designs due to glitches.

In the file *asinc_count16.vhd* design, a system which pretends to accumulate for 16 times a four-bit number defined by the user by means of four external switches. This modulo-16 accumulation is performed periodically every 3 seconds, approximately. Thus, if the number to accumulate is set to 1, and the accumulator is initially reset to 0, after 16 accumulations by 1, the final result remains to 0. This also happens for any other number to accumulate, regardless its value, the final accumulator value after 16 accumulations should remain to 0.

The file *asinc_count16.vhd* design, listed in the annex, uses as a component an enabled T flip-flop (file *etff.vhd*), that is instantiated 27 times to divide the 50 MHz mclk clock signal. The AND function of stages 22 and 27 attempts to generate a stream of 16 clock pulses to control the accumulation process.

Tasks to carry out:

- Carefully analyze the *asinc_count16.vhd* description, trying to find out potential problems.
- Synthesize the description. Simulate the description with delays. Are the results correct? Why? **Important note.** Like in section 1.1, speed up the time simulation cutting the number of frequency dividing stages, changing the cntClk range to 27 downto 21 and modifying the necessary sentences and synthesizing the new version.
- Create a new description file *sinc_count16.vhd* that avoids the problems of the former designs. Synthesize it and simulate it.

Optional task:

- The file *main_prac1_2.vhd* incorporates the former entity as a component. This file, after being synthesized and compiled can be uploaded to the Xilinx evaluation board to check real-time operation. It is necessary to introduce, together with the EDIF file generated by the synthesizer, the pin configuration UCF constraints file that will be provided in the lab, substituting the one eventually obtained from the synthesis process.

In order to be able to observe operation with a visible time scale, **it is necessary to restore the signal cntClk range to *27 downto 0***.

2. Annex

VHDL code for section 1.1

prac1_1a.vhd

```
----------------------------------------------------------------------
-- 1, 3, 5 detector
-- Version 1a
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity prac1_1a is
  Port ( mclk       : in std_logic;
         sb_btn     : in std_logic;
              cntClk     : out std_logic_vector(28 downto 0);
              Ssegcnt    : out std_logic_vector(3 downto 0));
end prac1_1a;

architecture behavioral of prac1_1a is

----------------------------------------------------------------------
-- Signal Declarations
----------------------------------------------------------------------

  signal cntClk_aux          : std_logic_vector(28 downto 0);
  signal enable: std_logic;

----------------------------------------------------------------------
-- Module Implementation - DIO5 tests
----------------------------------------------------------------------

begin

  -- Divide the clock to produce various timing signals
        process (mclk, sb_btn)
     begin
                if (sb_btn = '1') then cntClk_aux <= "00000000000000000000000000000";

          elsif mclk = '1' and mclk'Event then
             cntClk_aux <= cntClk_aux + 1;
         end if;
        end process;


        enable <= '1' when (cntClk_aux(28 downto 25)="0001" or cntClk_aux(28 downto
25)="0011" or cntClk_aux(28 downto 25)="0101")
                            else enable;
        process
        begin
                wait until(enable'event and enable='1');
                if (sb_btn = '1') then
                        Ssegcnt <= (others => '0');
                else
                        Ssegcnt <= cntClk_aux(28 downto 25);  -- DIO4 Sseg display digit
                end if;
        end process;

        cntClk <= cntClk_aux;

end behavioral;
```

prac1_1b.vhd

```
------------------------------------------------------------------------
-- 1, 3, 5 detector
-- Version 1b
------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity prac1_1b is
   Port ( mclk       : in std_logic;
          sb_btn     : in std_logic;
               cntClk     : out std_logic_vector(28 downto 0);
               Ssegcnt    : out std_logic_vector(3 downto 0));
end prac1_1b;

architecture behavioral of prac1_1b is

------------------------------------------------------------------------
-- Signal Declarations
------------------------------------------------------------------------

   signal cntClk_aux          : std_logic_vector(28 downto 0);
   signal enable: std_logic;

------------------------------------------------------------------------
-- Module Implementation - DIO5 tests
------------------------------------------------------------------------

begin

   -- Divide the clock to produce various timing signals
        process (mclk, sb_btn)
      begin
                if (sb_btn = '1') then cntClk_aux <= "00000000000000000000000000000";

          elsif mclk = '1' and mclk'Event then
             cntClk_aux <= cntClk_aux + 1;
        end if;
        end process;


process(cntClk_aux, sb_btn)
```

```
	begin
		if (sb_btn = '1') then
			enable <= '0';
		elsif (cntClk_aux(28 downto 25)="0001" or cntClk_aux(28 downto 25)="0011" or
cntClk_aux(28 downto 25)="0101") then
			enable <= '1';
		end if;
	end process;

process
	begin
		wait until(enable'event and enable='1');
		if (sb_btn = '1') then
			Ssegcnt <= (others => '0');
		else
			   Ssegcnt <= cntClk_aux(28 downto 25);  -- DIO4 Sseg display digit
		end if;
	end process;

	cntClk <= cntClk_aux;

end behavioral;
```

prac1_1c.vhd

```
-----------------------------------------------------------------------
-- 1, 3, 5 detector
-- Version 1c
-----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity prac1_1c is
   Port ( mclk       : in std_logic;
          sb_btn     : in std_logic;
               cntClk     : out std_logic_vector(28 downto 0);
               Ssegcnt    : out std_logic_vector(3 downto 0));
end prac1_1c;

architecture behavioral of prac1_1c is

-----------------------------------------------------------------------
-- Signal Declarations
-----------------------------------------------------------------------

   signal cntClk_aux           : std_logic_vector(28 downto 0);
   signal enable: std_logic;

-----------------------------------------------------------------------
-- Module Implementation - DIO5 tests
-----------------------------------------------------------------------

begin

   -- Divide the clock to produce various timing signals
        process (mclk, sb_btn)
     begin
```

```
                if (sb_btn = '1') then cntClk_aux <= "00000000000000000000000000000";

        elsif mclk = '1' and mclk'Event then
           cntClk_aux <= cntClk_aux + 1;
      end if;
        end process;

        process(cntClk_aux, sb_btn)
        begin
                if (sb_btn = '1') then
                        enable <= '0';
                elsif (cntClk_aux(28 downto 25)="0001" or cntClk_aux(28 downto 25)="0011" or
cntClk_aux(28 downto 25)="0101") then
                        enable <= '1';
                else
                        enable <= '0';
                end if;
        end process;

        process
        begin
                wait until(enable'event and enable='1');
                if (sb_btn = '1') then
                        Ssegcnt <= (others => '0');
                else
                           Ssegcnt <= cntClk_aux
                           (28 downto 25);  -- DIO4 Sseg display digit
                end if;
        end process;

        cntClk <= cntClk_aux;

end behavioral;
```

VHDL code for section 1.2

etff.vhd

```
library ieee;
  use ieee.std_logic_1164.all;

  entity ETFF is

  -- 1-bit enable rising-edge register
    port(
       T: in std_logic;-- Toggle enable
       A_RES: in std_logic; -- Asynchronous reset
       CK: in std_logic;  -- Clock input
       Q: out std_logic -- Output
       );

  end ETFF;

  architecture behavior of ETFF is

    signal Qaux: std_logic;

  begin

  ff:process(CK, A_RES)
    begin
      if (a_res = '1') then Qaux<='0' after 10 ps;-- asynchronous reset condition
      elsif (CK'event and CK = '1') then
        if (T='1') then
          Qaux <= not(Qaux) after 10 ps;
        end if;
      end if;
    end process;

    Q <= Qaux;

  end behavior;
```

asinc_count16.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity asinc_count16 is
   port ( mclk      : in std_logic;
          sb_btn     : in std_logic;
          IO4_btn    : in std_logic_vector(4 downto 0);
          IO4_swt    : in std_logic_vector(7 downto 0);
                  cntClk     : out std_logic_vector(27 downto 0);
                  Ssegcnt    : out std_logic_vector(3 downto 0));
end asinc_count16;

architecture behavioral of asinc_count16 is

--------------------------------------------------------------------
```

```
-- Component Declarations
----------------------------------------------------------------------

    component etff
      port(
         T: in std_logic;-- Toggle enable
         A_RES: in std_logic; -- Asynchronous reset
         CK: in std_logic;          -- Clock input
         Q: out std_logic -- Output
         );
    end component;

----------------------------------------------------------------------
-- Signal Declarations
----------------------------------------------------------------------

   signal cntClk_aux          : std_logic_vector(27 downto 0);
   signal Ssegcnt_aux       : std_logic_vector(3 downto 0);
   signal EN_CTL: std_logic_vector(9 downto 0);
   signal A_RES_aux1, EN_CKGEN, CKdiv_aux, conta_ck : std_logic; -- external and internal
reset;

begin

-- Divisor de mclk per flanc de pujada
    -- Combinational reset
    A_RES_aux1 <= sb_btn; -- Resets when sb_btn is pressed
    EN_CKGEN <= not IO4_BTN(4); -- Disables when button 4 is pressed

         cntClk_aux(0) <= mclk;
  -- Central clock generation
    GEN_FF: for i in 1 to 27 generate
    ETFF_i: ETFF
         port map(
           T => EN_CKGEN, A_RES => A_RES_aux1, CK => cntClk_aux(i-1), Q =>
cntClk_aux(i)
           );
    end generate;

conta_ck <= cntClk_aux(27) and cntClk_aux(22); -- Produeix un glitch

-- Acumulació
  process(conta_ck, sb_btn)
  begin
    if (sb_btn = '1') then
                        Ssegcnt_aux <= (others => '0');
                elsif conta_ck'event and conta_ck='1' then
                        Ssegcnt_aux <= Ssegcnt_aux + IO4_swt(3 downto 0);
                end if;
        end process;

Ssegcnt <= Ssegcnt_aux;
cntClk <= cntClk_aux;

end behavioral;
```