Masauso Lungu 209533

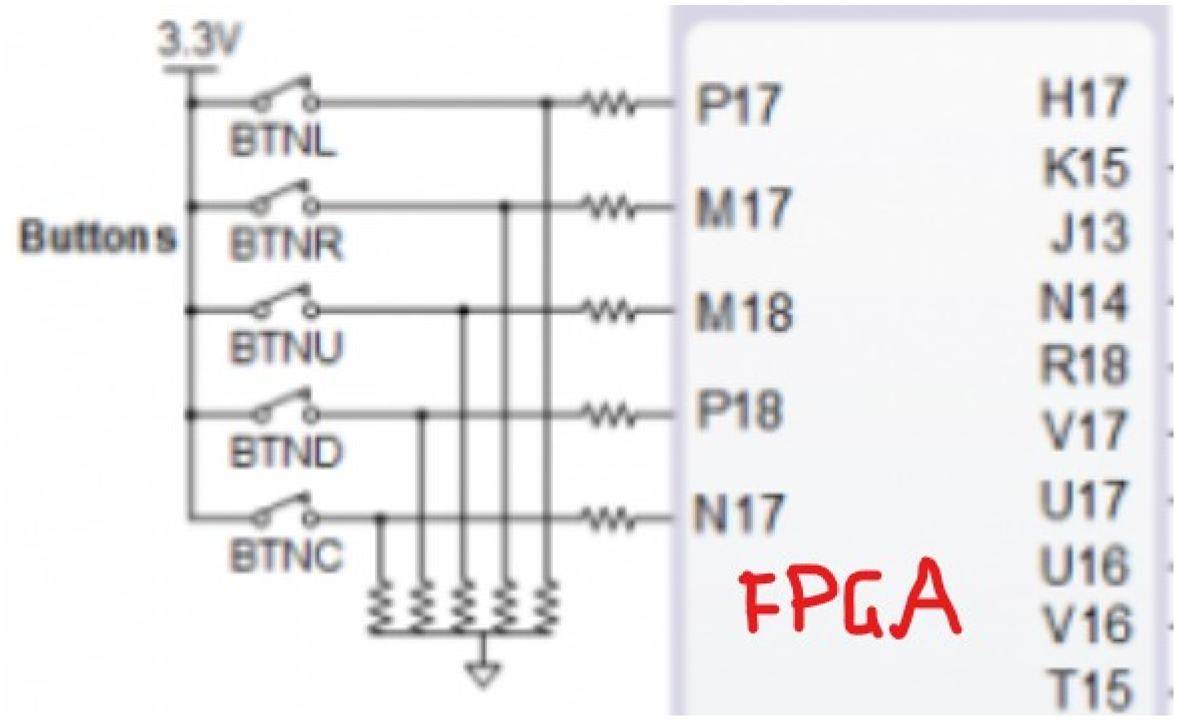
Lab 5: Binary counter

Links:

My github repository

Preparation tasks (done before the lab at home)

The Nexys A7 board provides six push buttons.The `five pushbuttons` arranged in a plus-sign configuration are "momentary" switches that normally generate a low output when they are at rest, and a high output only when they are pressed. The red pushbutton labeled `"CPU RESET,"` on the other hand, generates a high output when at rest and a low output when pressed. The connection of the five pushbuttons is shown on the picture below. CPU RESET button `(BTNRES)` is connected to pin `C12` of the FPGA.



Calculate how many periods of clock signal with frequency of 100 MHz contain time intervals 2 ms, 4 ms, 10 ms, 250 ms, 500 ms, and 1 s. Write values in decimal, binary, and hexadecimal forms.

`T_clk = 1/f_clk`

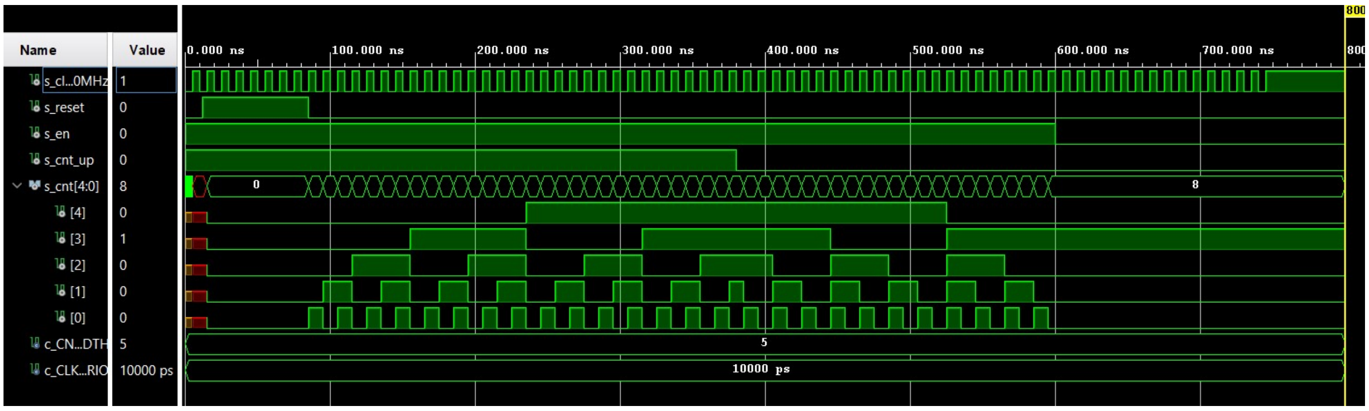
Time interval	Number of clk periods	Number of clk periods in hex	Number of clk periods in binary
2 ms	200 000	`x"3_0d40"`	`b"0011_0000_1101_0100_0000"`
4 ms	400 000	`x"61a80"`	`b"0110_0001_1010_1000_0000"`
10 ms	1000 000	`x"f4240"`	`b"1111_0100_0010_0100_0000"`

Time interval	Number of clk periods	Number of clk periods in hex	Number of clk periods in binary
250 ms	25 000 000	`x"17d7840"`	`b"0001_0111_1101_0111_1000_0100_0000"`
500 ms	50 000 000	`x"2faf080"`	`b"0010_1111_1010_1111_0000_1000_0000"`
1 sec	100 000 000	`x"5F5_E100"`	`b"0101_1111_0101_1110_0001_0000_0000"`

Listing of VHDL code of the process p_cnt_up_down

```vhdl
begin
    --------------------------------------------------------------------
    -- p_cnt_up_down:
    -- Clocked process with synchronous reset which implements n-bit
    -- up/down counter.
    --------------------------------------------------------------------
    p_cnt_up_down : process(clk)
    begin
        if rising_edge(clk) then

            if (reset = '1') then               -- Synchronous reset
                s_cnt_local <= (others => '0'); -- Clear all bits

            elsif (en_i = '1') then       -- Test if counter is enabled

                -- TEST COUNTER DIRECTION HERE
                if (cnt_up_i = '1') then
                     s_cnt_local <= s_cnt_local + 1;
                else
                     s_cnt_local <= s_cnt_local - 1;
                end if;

            end if;
        end if;
    end process p_cnt_up_down;

    -- Output must be retyped from "unsigned" to "std_logic_vector"
    cnt_o <= std_logic_vector(s_cnt_local);

end architecture behavioral;
```

updaown counter



Listing of VHDL reset and stimulus processes from testbench file tb_cnt_up_down.vhd

```
   --------------------------------------------------------------------
    -- Reset generation process
    --------------------------------------------------------------------
    p_reset_gen : process
    begin
        s_reset <= '0';
        wait for 12 ns;
        s_reset <= '1';                 -- Reset activated
        wait for 73 ns;
        s_reset <= '0';
        wait;
    end process p_reset_gen;

    --------------------------------------------------------------------
    -- Data generation process
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

        s_en     <= '1';                -- Enable counting
        s_cnt_up <= '1';
        wait for 380 ns;                -- Change counter direction
        s_cnt_up <= '0';
        wait for 220 ns;
        s_en     <= '0';                -- Disable counting

        report "Stimulus process finished" severity note;
        wait;
    end process p_stimulus;

end architecture testbench;
```

Listing of VHDL code from source file top.vhd with all instantiations for the 4-bit bidirectional counter

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;


entity top is
    Port ( CLK100MHZ : in STD_LOGIC;
           BTNC : in STD_LOGIC;
           SW   : in STD_LOGIC_VECTOR (0 downto 0);
           LED : out STD_LOGIC_VECTOR (3 downto 0);
           CA   : out STD_LOGIC;
           CB   : out STD_LOGIC;
           CC   : out STD_LOGIC;
           CD   : out STD_LOGIC;
           CE   : out STD_LOGIC;
           CF   : out STD_LOGIC;
           CG   : out STD_LOGIC;
           AN   : out STD_LOGIC_VECTOR(7 downto 0)
           );

end top;

------------------------------------------------------------------------
-- Architecture body for top level
------------------------------------------------------------------------
architecture Behavioral of top is

    -- Internal clock enable
    signal s_en  : std_logic;
    -- Internal counter
    signal s_cnt : std_logic_vector(4 - 1 downto 0);

begin

    --------------------------------------------------------------------
    -- Instance (copy) of clock_enable entity
    clk_en0 : entity work.clock_enable
        generic map(
            --- WRITE YOUR CODE HERE
            g_MAX => 25000000
        )
        port map(
            --- WRITE YOUR CODE HERE
            clk   => CLK100MHZ,
            reset => BTNC,
            ce_o  => s_en
        );

    --------------------------------------------------------------------
    -- Instance (copy) of cnt_up_down entity
    bin_cnt0 : entity work.cnt_up_down
        generic map(
            --- WRITE YOUR CODE HERE
            g_CNT_WIDTH => 4 -- 4bits
        )
        port map(
            --- WRITE YOUR CODE HERE
            clk      => CLK100MHZ,
            reset    => BTNC,
            en_i     => s_en,
            cnt_up_i => SW(0),
            cnt_o    => s_cnt

        );

    -- Display input value on LEDs
    LED(3 downto 0) <= s_cnt;

    --------------------------------------------------------------------
    -- Instance (copy) of hex_7seg entity
    hex2seg : entity work.hex_7seg
        port map(
            hex_i    => s_cnt,
            seg_o(6) => CA,
            seg_o(5) => CB,
            seg_o(4) => CC,
            seg_o(3) => CD,
            seg_o(2) => CE,
            seg_o(1) => CF,
            seg_o(0) => CG
        );

    -- Connect one common anode to 3.3V
    AN <= b"1111_1110";

end architecture Behavioral;
```