

My DE1 repisitory link

[DE1 - Jiří Navrátil 222721](#)

Binary counter

Table with button connections for Nexys A7 board

Output pins	FPGA pin	FPGA pin name
BTNL	IO_L12P_T1_MRCC_14	P17
BTNR	IO_L10N_T1_D15_14	M17
BTNU	IO_L4N_T0_D05_14	M18
BTND	IO_L9N_T1_DQS_D13_14	P18
BTNC	IO_L9P_T1_DQS_14	N17

Intervals of clock signal

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"6_1a80"	b"0110_0001_1010_1000_0000"
10 ms	1 000 000	x"f_4240"	b"1111_0100_0010_0100_0000"
250 ms	25 000 000	x"17d_7840"	b"0001_0111_1101_0111_1000_0100_0000"
500 ms	50 000 000	x"2fa_f080"	b"0010_1111_1010_1111_0000_1000_0000"
1 sec	100 000 000	x"5f5_e100"	b"0101_1111_0101_1110_0001_0000_0000"

VHDL code for binary counter

Process code

```
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
```

```
        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;
```

Reset and stimulus processes from testbench

```
p_reset_gen : process
begin
    s_reset <= '0';
    wait for 32 ns;

    -- Reset activated
    s_reset <= '1';
    wait for 94 ns;

    s_reset <= '0';
    wait;
end process p_reset_gen;

p_stimulus : process
begin
    report "Stimulus process started" severity note;

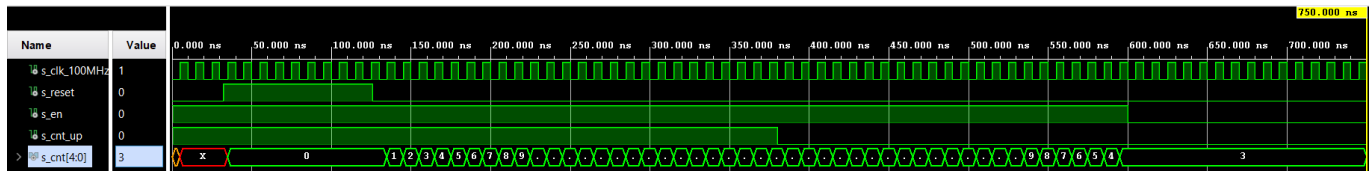
    -- Enable counting
    s_en      <= '1';

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

    -- Disable counting
    s_en      <= '0';

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

Result



VHDL code of *top.vhd* file

```
-- Instance (copy) of clock_enable entity
clk_en0 : entity work.clock_enable
    generic map(
        g_MAX => 100000000
    )
    port map(
        clk     => CLK100MHZ,
        reset   => BTNC,
        ce_o    => s_en
    );
```

```
-- Instance (copy) of cnt_up_down entity
bin_cnt0 : entity work.cnt_up_down
    generic map(
        g_CNT_WIDTH => 4
    )
    port map(
        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";
```

The block diagram illustrates the system architecture. It features four main functional blocks: **cnt_up_down**, **cnt_up_down_16**, **clock_enable**, and **clock_enable_16**. The **cnt_up_down** and **cnt_up_down_16** blocks are counters with inputs *clk*, *en_i*, *cnt_up_i*, and *reset*, and outputs *cnt_o* and *s_en*. The **clock_enable** and **clock_enable_16** blocks are clock enable blocks with inputs *clk*, *reset*, and *cnt_o*, and outputs *ce_o* and *s_en_16*. The outputs *s_en* and *s_en_16* are connected to an **OR** gate, which produces the **AN** signal. The outputs *cnt_o* and *cnt_o_16* are connected to **hex_7seg** and **hex_7seg_16**, which produce the **LED** signal. The system is controlled by **SW**, **CLK100MHZ**, and **BTNC** inputs.