



**ESKİŞEHİR TECHNICAL
UNIVERSITY**

**DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING**

EEM 334 - Digital Systems II

LAB 5 – SEQUENTIAL DESIGN I
DIGITAL CLOCK

1. PURPOSE

In this lab, you will learn how to design a synchronous sequential circuit in VHDL. A special kind of sequential circuits are counters which are frequently used in digital systems. Thus, you will first design an up-down counter with synchronous reset and enable inputs. After that, you will design, simulate, and realize a digital clock based on this counter.

2. BACKGROUND

A sequential circuit is a circuit with memory, which forms the internal state of the circuit. Unlike a combinational circuit, in which the output is a function of input only, the output of a sequential circuit is a function of the input and the internal state. The synchronous design methodology is the most commonly used practice in designing a sequential circuit. In this methodology, all storage elements are controlled (i.e., synchronized) by a global clock signal and the data is sampled and stored at the rising or falling edge of the clock signal.

The sequential domain is represented by a process that contains sequential statements. These statements are executed in the order in which they appear within the process, as in programming languages.

```
process_label : process(sensitivity_list)
-- declarative part
begin
-- sequential statement
end process process_label;
```

Fig. 1 Process structure

3. PROCEDURE

1. Design your 32-bit up_down counter with a synchronous reset and an enable input (it is already given).
2. Simulate your 32-bit up_down counter on ISE simulator.
3. Using a counter, obtain a new clock signal based on clock input by modifying the example code given as follows:
 - a. 1-bit “clock_out” output port will be needed for new clock signal output. 1-bit “clock_out” output port will be 0 for the half of the new clock period and 1 for the other half.
 - b. “result” output port will not be necessary.
 - b. New clock (clock_out) period will be 10x of the input clock period .

4. Using a counter, design a “Digital clock component” as follows.
 - a. The new clock signal obtained in Step 3 will be used as the clock input for the Sec and Min components.
 - b. You need a 1-bit “enable_out” output port and 4-bit “upperlimit” input port.
 - c. In up mode, your counter will count up to until this “upperlimit” and then reset to zero.
 - d. In down mode, your counter will count down to zero and then reset to this “upperlimit”.
 - e. “enable_out” output signal will be an asynchronous one. It will be 1 for a clock cycle if the current counter value is “upperlimit” and 0 for all other values while counting up.
5. Build your top module and instantiate all components according to Figure 2.

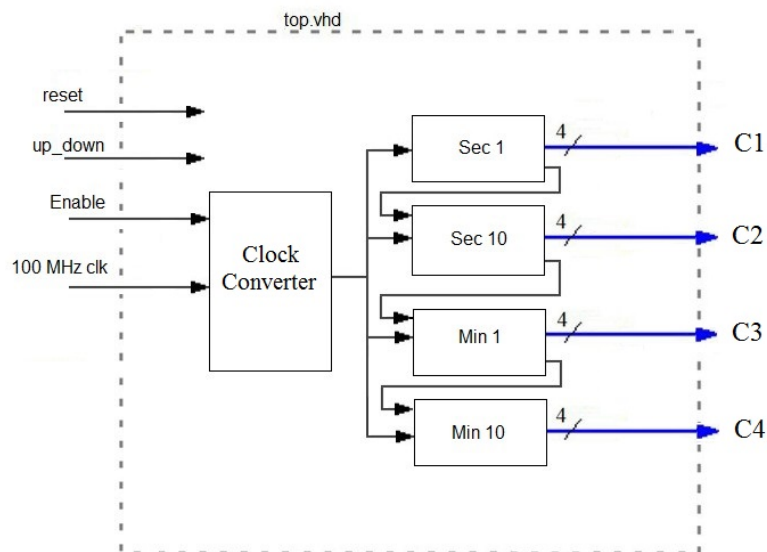


Fig. 2. Block diagram of top module

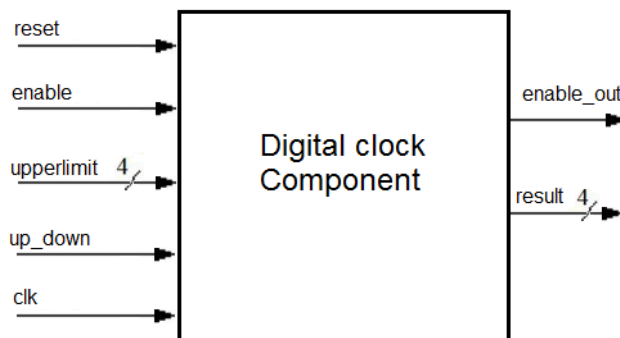


Fig. 3 Digital clock component

Note: Sec 1, Sec 10, Min 1 and Min 10 are the instantiations of the Digital clock component. Sec 1 enable input is connected to top enable port.

Example Up-Down Counter

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

entity universal_counter is
  Port ( clk : in  STD_LOGIC;
        rst  : in  STD_LOGIC;
        up_down : in  STD_LOGIC;
        enable : in  STD_LOGIC;
        result : out STD_LOGIC_VECTOR (31 downto 0));
end universal_counter;

architecture Behavioral of universal_counter is
  signal count : unsigned (31 downto 0) := x"00000000";
begin

  process (clk, rst, enable, up_down)
  begin
    if (clk = '1' and clk'event) then
      if (rst = '1') then
        count <= (others => '0');
      elsif (enable = '1') then
        if (up_down = '1') then
          count <= count + 1;
        else
          count <= count - 1;
        end if ;
      end if ;
    end if;
  end process;

  result <= count;

end Behavioral;
```