Laboratório 5

Timers e Relógios de Tempo Real

O objetivo deste laboratório é o estudo do uso de clocks em circuitos temporizados.

Background

Na linguagem VHDL, podemos descrever um contador de tamanho variável usando uma declaração *generic*. Um exemplo de um contador de *n*-bits é mostrado na Figura 1.

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

entity counter is
    generic (
        n : natural := 4
    );
    port (
        clock : in STD_LOGIC;
        reset_n : in STD_LOGIC;
        Q : out STD_LOGIC_VECTOR(n-1 downto 0)
    );
end entity;

architecture rtl of counter is
signal value : unsigned(n-1 downto 0);
begin
    PROCESS(clock, reset_n)
    begin
        if (reset_n = '0') then
            value <= (OTHERS => '0');
        elsif (rising_edge(clock)) then
            value <= value + 1;
        end if;
    end process;
    Q <= std_logic_vector(value);
end rtl;
```

Figura 1: Uma descrição em VHDL de um contador de *n*-bits.

O parâmetro *n* especifica o número de bits no contador. Um valor particular deste parâmetro é definido usando uma declaração **generic map**. Por exemplo, um contador de 8-bits pode ser especificado através de:

```
eight_bit: counter
    generic map( 8 )
    port map (clock, reset_n, Q);
```

Ao se utilizar os componentes genéricos, podemos criar contadores de diferentes tamanhos em um circuito lógico

sem a necessidade de criar um novo módulo para cada contador.

Parte I

Crie um contador módulo-k modificando o projeto de um contador de 8-bits para conter uma entrada adicional (*ValMax*). O contador deve contar de 0 até *ValMax* -1. Quando o contador alcança este valor, o valor que segue deve ser 0.

Seu circuito deve usar o botão KEY_0 como um reset assíncrono, KEY_1 como uma entrada de clock manual e *SW* para definir o valor máximo. O conteúdo do seu contador deve ser mostrado nos LEDs vermelhos.

Mostre ao professor a implementação deste código na placa de desenvolvimento considerando um contador de 8 bits.

Parte II

Nesta parte, faremos o projeto de um contador BCD de 000 a 999. Cada sinal BCD será apresentado em um display de 7 segmentos. O circuito obrigatoriamente deverá ser síncrono, ou seja, todos os contadores devem receber o mesmo sinal de relógio. Este sinal deverá ser criado a partir do sinal CLOCK_50.

Você deverá modificar o projeto do contador anterior para incluir uma entrada adicional enable e uma saída adicional rollout. A entrada enable é usada para liberar que o acréscimo síncrono na contagem seja efetuado e, portanto, deve ser implementada com um teste dentro da subida do relógio. A saída rollout deve ser estar em nível alto apenas quando o contador estiver em seu último número antes de retornar a zero, de forma que seja possível, a partir dele, criar a lógica dos enables dos demais contadores. Observe que o rollout é apenas um comparador/multiplexador, e pode ser facilmente implementado fora do process principal através da diretiva when:

```
rollover <= '1' when (counter = k-1) else '0';
```

onde k é o módulo do contador e counter é o sinal interno que mantém o valor da contagem.

Neste projeto precisamos de 4 contadores deste tipo. O primeiro é responsável por, a partir do sinal CLOCK_50, gerar um clock de 1s. Os outros três contadores irão fazer a contagem em BCD de cada um dos dígitos.