Laboratório 3

Contadores, Timers e Relógios de Tempo Real

O objetivo deste laboratório é a construção e o uso de contadores e 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 a seguir.

```
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
begin
    PROCESS(clock, reset_n)
    variable value : unsigned(n-1 downto 0);
    begin
        if (reset_n = '0') then
            value := 0;
        elsif (rising_edge(clock)) then
            value := value + 1;
        end if;
        Q <= std_logic_vector(value);
    end process;
end rtl;
```

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:

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

Ao se utilizar os componentes genéricos, podemos instancializar contadores de diferentes tamanhos em um circuito lógico sem a necessidade de criar um novo módulo para cada contador.

Parte I

Considere o circuito da Figura 1. Este é um contador síncrono de 4 bits que utiliza quatro flip-flops do tipo T. O contador incrementa seu valor em cada subida do clock se o sinal *Enable* estiver ativo. O contador reseta para 0 quando o sinal *Clear* se tornar baixo. Você deve implementar um contador de 8 bits deste tipo.

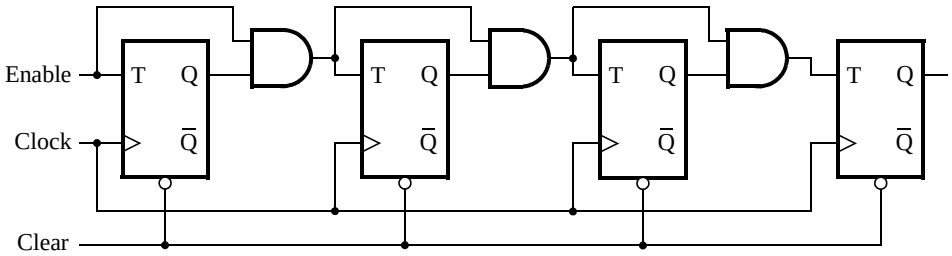

Figura 1: Um contador de 4 bits.

1. Escreva um arquivo VHDL que defina um contador de 4 bits conforme a estrutura da Figura 1. Você deve incluir o módulo que instancia o flip-flop T quatro vezes.
2. Crie um novo projeto Quartus II que defina este contador e conecte o botão KEY_0 como *Clock*, as chaves SW_1 e SW_0 como as entradas *Enable* e *Clear*, e o display de 7 segmentos *HEX0* mostra a contagem hexadecimal do circuito
3. Compile e teste seu circuito. **Apresente este projeto para o professor**.

Parte II

Outra forma de especificar um contador é através de código comportamental. Em um process, crie uma variável que aumente em 1 seu valor na subida do clock. A biblioteca `ieee.numeric_std.all` pode ser útil.
Compile uma versão de 16 bits deste contador. **Apresente o código para o professor**.

Parte III

Crie um contador módulo-k modificando o projeto de um contador de 8-bits para conter um parâmetro 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 IV

Projete e implemente um circuito que funcione como um relógio. Ele deve mostrar a hora (de 0 a 23) nos displays de 7-segmentos *HEX7−6*, os minutos (de 0 a 59) em *HEX5−4* e os segundos (de 0 a 59) em *HEX3−2*. Use o botão KEY_0 como um reset assíncrono e o botão KEY_1 junto com as chaves SW_{15-11} para definir a hora e SW_{10-5} para definir os minutos mostrados no relógio. O contador deve ser incrementado pelo clock de 50 MHz (CLOCK_50) existente na placa. Você pode modificar o contador da parte anterior para que este inclua uma segunda saída que indique que o contador completou um ciclo até seu valor máximo. **Envie este código ao Ensino Aberto**.