Aula 6 - Timers and Real-time Clock

Parte 4: desenvolvemos um circuito que recebe uma das 8 primeiras letras do alfabeto, selecionada por 3 chaves na FPGA, e retorna um sinal em código morse da letra com os LEDs da FPGA.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY partel IS
    PORT (
        clk     : in  STD_LOGIC;    -- Clock de 50 MHz
        reset   : in  STD_LOGIC;    -- Reset para inicializar o contador
        enable  : in  STD_LOGIC;    -- Sinal que controla a exibição do codigo morse
        SW0 : IN STD_LOGIC;
        SW1 : IN STD_LOGIC;
        SW2 : IN STD_LOGIC;
        led_output : OUT STD_LOGIC
    );
END ENTITY;

ARCHITECTURE Behavior OF partel IS
    CONSTANT LINHA : INTEGER := 50000000;
    CONSTANT PONTO : INTEGER := 20000000;

    SIGNAL counter : INTEGER := 0;
    SIGNAL intervalo : INTEGER := 50000000;
    SIGNAL comprimento : INTEGER := 0;
    SIGNAL led_state : STD_LOGIC := '0'; -- Inicializando o estado do LED
    SIGNAL cod_morse : STD_LOGIC_VECTOR(4 DOWNTO 0); -- Vetor de 5 bits

    FUNCTION bin_to_morse (d : STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR IS
    BEGIN
        CASE d IS
            WHEN "000" =>
                RETURN "XXX01";  -- A
            WHEN "001" =>
                RETURN "x1110";  -- B
            WHEN "010" =>
                RETURN "x1010";  -- C
            WHEN "011" =>
                RETURN "xx110";  -- D
            WHEN "100" =>
                RETURN "XXXX1";  -- E
            WHEN "101" =>
                RETURN "x1011";  -- F
            WHEN "110" =>
                RETURN "xx100";  -- G
            WHEN "111" =>
                RETURN "x1111";  -- H
            WHEN OTHERS =>
                RETURN "XXXXX";   -- erro
        END CASE;
    END FUNCTION;

    FUNCTION bin_to_comprimento(d : STD_LOGIC_VECTOR) RETURN INTEGER IS
    BEGIN
      CASE d IS
            WHEN "000" =>
                RETURN 2;  -- A
            WHEN "001" =>
                RETURN 4;  -- B
            WHEN "010" =>
                RETURN 4;  -- C
            WHEN "011" =>
                RETURN 3;  -- D
            WHEN "100" =>
                RETURN 1;  -- E
            WHEN "101" =>
                RETURN 4;  -- F
            WHEN "110" =>
                RETURN 3;  -- G
            WHEN "111" =>
                RETURN 4;  -- H
            WHEN OTHERS =>
                RETURN 0;   -- erro
        END CASE;
    END FUNCTION;
```

```
BEGIN
    PROCESS (clk, reset)
        VARIABLE bin_letra : STD_LOGIC_VECTOR (2 DOWNTO 0);
        VARIABLE i : INTEGER := 0;
        VARIABLE flag : INTEGER := 0;
    BEGIN
        IF reset = '0' THEN
            -- reset
            counter <= 0;
            intervalo <= 50000000; -- Resetando intervalo
            led_state <= '1'; -- Desligar LED no reset
            i := 0;
            flag := 0;

        ELSIF enable = '0' THEN
            flag := 1;

        ELSIF rising_edge(clk) THEN

            IF flag = 1 THEN
               IF counter = 0 AND intervalo = 0 THEN

                   bin_letra := SW2 & SW1 & SW0;

                   cod_morse <= bin_to_morse(bin_letra);
                   comprimento <= bin_to_comprimento(bin_letra);

                   if i < comprimento THEN
                      CASE cod_morse(i) IS
                          WHEN '1' =>
                              counter <= PONTO;
                              intervalo <= 20000000;
                              i := i + 1;
                          WHEN '0' =>
                              counter <= LINHA;
                              intervalo <= 20000000;
                              i := i + 1;
                      END CASE;
                   else
                     i := 0;
                     intervalo <= 100000000;
                   end if;

               ELSIF counter /= 0 THEN
                   counter <= counter - 1;

                   IF led_state = '1' THEN
                       led_state <= '0';  -- Acentuar LED
                   END IF;

               ELSIF intervalo /= 0 THEN
                   intervalo <= intervalo - 1;

                   IF led_state = '0' THEN
                       led_state <= '1';  -- Apagar LED
                   END IF;

               END IF;

           END IF;
      END IF;
    END PROCESS;

    led_output <= led_state; -- Saída para o LED

END Behavior;
```

As funções que acompanham este programa são “bin_to_comprimento” e “bin_to_morse”, sendo que a primeira é responsável por retornar a quantidade de sinais de morse, usados para representar uma determinada letra selecionada por uma sequência de chaves, para o sinal “comprimento”, enquanto a segunda retorna um vetor de binários referente ao código morse de alguma letra. Note que a segunda função usa o valor indefinido “x”, pois para manter o tamanho dos vetores retornados constante, precisou-se preencher as lacunas do vetor com esse valor. Definimos as constantes LINHA, PONTO e INTERVALO, com cada uma significando o intervalo de tempo que cada ação deve ocorrer. Como o período do clock é de 50 MHz, o sinal de linha e o intervalo entre os sinais de morse duram 1 segundo, enquanto o sinal de ponto dura 0.4 segundo. O vetor cod_morse armazena uma sequência em código

morse com a seguinte convenção: 1 simboliza ponto, 0 simboliza linha, x simboliza o fim do vetor e o vetor e o primeiro elemento do vetor é seu elemento mais à direita. A saída do código é direcionada para um único LED que pisca de acordo com a saída led_output, sendo seu valor alterado pelo sinal led_state.

O programa começa na seleção da letra do código morse pelo usuário através de chaves na FPGA. Os 3 valores são concatenados no vetor bin_letra, e a condição i < comprimento é executada. O sinal counter recebe o valor correspondente ao valor do primeiro elemento de cod_morse, intervalo recebe o valor 20*10^6 e i é incrementado. Nesse momento, o primeiro ciclo de clock se encerra, e o valor de counter é atualizado, fazendo com que o fluxo do programa seja redirecionado para a condição “counter \= 0”. As instruções abaixo da condição são executadas, a cada ciclo de clock, até que essa condição se torne falsa. Com isso, o valor de led_state e, consequentemente, de led_output permanecem em nível baixo, condição para acender o LED, por um tempo apropriado, 0.4 s para o sinal de ponto, e 1 s para o sinal de linha.

Quando counter chega a 0, o fluxo do programa é direcionado para a condição ELSIF intervalo /= 0, onde o intervalo é decrementado a cada ciclo de clock até chegar a 0, e led_state permance ligado (LED desligado) até que a condição se torne falsa. Quando isso acontece, o fluxo do programa volta para a primeira instrução do processo a condição “if i < comprimento”, em que o próximo sinal de código morse da letra é escolhido. Quando todos os sinais foram lidos, o bloco else abaixo desse if é executado, mudando o valor de intervalo fazendo com que o LED fique apagado por 2 segundos. Após esse tempo, o ciclo dentro do laço i é reiniciado e a última letra inserida pelo usuário é transmitida pelo LED em código morse até que ele acione reset para escolher outra letra ou encerrar o programa.