

UNIVERSIDADE FEDERAL DO PARANÁ

ELOIZA ARANDIA BRUSCHI GRR20212721

MARIA PAULA BASTOS GRR20212762

RELATÓRIO LABORATÓRIO 3: PROJETOS COM CÓDIGO SEQUENCIAL COM
PROCESSOS

CURITIBA

2024

ELOIZA ARANDA BRUSCHI GRR20212721
MARIA PAULA BASTOS GRR20212762

RELATÓRIO LABORATÓRIO 3: PROJETOS COM CÓDIGO SEQUENCIAL COM PROCESSOS

Relatório apresentada à disciplina de
Microeletrônica, Setor de Tecnologia da
Universidade Federal do Paraná, como atividade
avaliativa.
Professora: Dra. Sibilla Batista da Luz França.

CURITIBA
2024

1	INTRODUÇÃO	16
2	PROJETO 1- TEMPORIZADOR	17
2.1	ESQUEMÁTICOS RTL.....	17
2.2	SIMULAÇÕES	19
2.3	CÓDIGO COMENTADO	20
2.3.1	VHDL Module	20
2.3.2	VHDL Test Bench.....	20
2.4	RECURSOS LÓGICOS.....	21
3	PROJETO 2 - CONTADOR 0-9	22
3.1	ESQUEMÁTICOS RTL.....	23
3.2	SIMULAÇÕES	23
3.3	CÓDIGO	24
3.3.1	VHDL Module	24
3.3.2	VHDL Test Bench.....	25
3.4	RECURSOS LÓGICOS.....	25
	CONCLUSÃO.....	26
	REFERÊNCIAS.....	27
	ANEXO 1 – VHDL MODULE – PROJETO 1	16
	ANEXO 3 – VHDL MODULE – PROJETO 2	20
	ANEXO 4 – VHDL TEST BENCH – PROJETO 2.....	22

1 INTRODUÇÃO

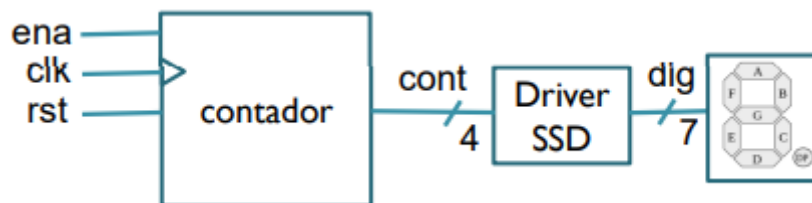
Este relatório apresenta a implementação de dois projetos desenvolvidos na disciplina de Microeletrônica I, focados na criação de temporizadores e contadores digitais, amplamente utilizados em diversos sistemas eletrônicos. Foram realizados dois projetos principais com a utilização da FPGA Nexys 2, ambos projetados para controlar e exibir contagens numéricas em displays de 7 segmentos.

O primeiro projeto consiste em um temporizador que realiza contagens de 0 a 9, com incrementos a cada meio segundo. O segundo projeto é um contador ajustável, que pode realizar contagens crescentes ou decrescentes, dependendo da configuração das entradas. O relatório detalha as simulações realizadas com diferentes combinações de entradas, o desenvolvimento dos circuitos e os resultados obtidos. Também são abordados aspectos técnicos importantes, como a utilização da técnica de debouncing no controle do botão e a implementação dos circuitos no kit de desenvolvimento.

2 PROJETO 1- TEMPORIZADOR

Foi utilizada a ferramenta ISE para implementar o temporizador solicitado no projeto 1, que faz uso do display de sete segmentos, similar ao que já havia sido utilizado na atividade anterior, do laboratório 2. O temporizador realiza a contagem de 0 a 9, com incrementos a cada meio segundo. Além disso, foi especificado o uso de uma chave para representar o 'enable', de modo que, ao ser desativada, a contagem para. A chave de reset foi configurada para reiniciar a contagem a partir de 0. A Figura 1 abaixo apresenta o esquemático conceitual, em que é possível abstrair que o sinal de 'clock' será utilizado como entrada do sistema.

FIGURA 1 – ESQUEMÁTICO CONCEITUAL DO PROJETO 1



FONTE: 'LAB 3', Sibilla França (20??).

Para garantir o cálculo correto do tempo, foi necessário considerar que o sinal de clock possui frequência de 50 MHz. A partir dessa referência, o cálculo dos períodos necessários para somar meio segundo foi realizado conforme a fórmula apresentada a seguir, resultando em 25 milhões de ciclos. Na lógica do programa, isso foi implementado utilizando um código sequencial, onde uma variável de controle foi incrementada até atingir esse valor. Quando o contador atingia 25 milhões, o valor no display era alterado, garantindo a precisão do temporizador.

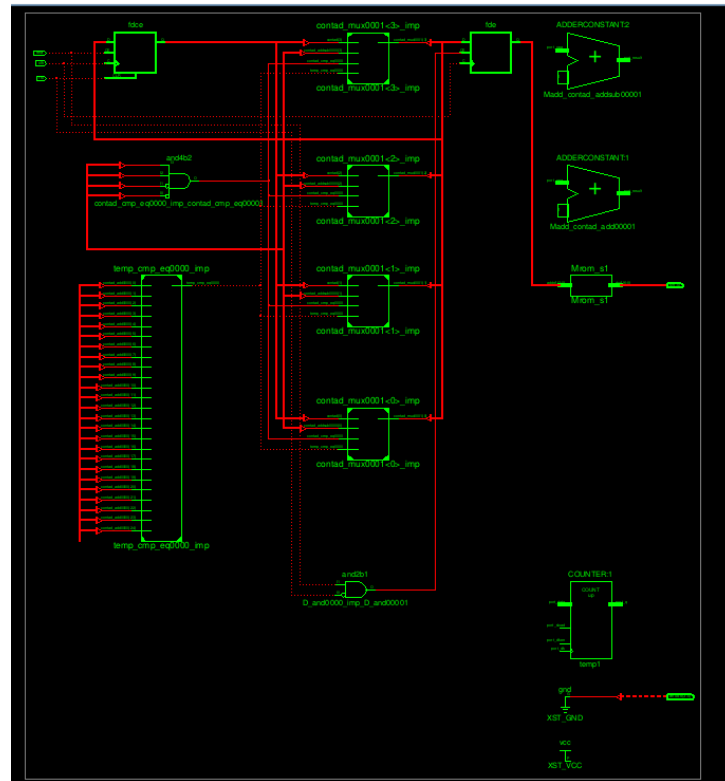
$$Quantidade\ de\ períodos = \frac{0,5 segundos}{período\ de\ clock} = \frac{0,5 segundos}{\frac{1}{50\ MHz}} = 25\ milhões$$

2.1 ESQUEMÁTICOS RTL

Os esquemáticos RTL são uma forma ilustrada de visualizar as entradas e saídas do projeto, proporcionando uma visão clara da estrutura interna do circuito. O esquemático RTL apresentado na figura abaixo, obtido após a etapa de síntese, demonstra a arquitetura lógica do circuito, confirmando a correta implementação da

lógica de decodificação BCD, o uso do clock e a inclusão das funcionalidades de enable e reset.

FIGURA 2 – ESQUEMÁTICO RTL DO PROJETO 1



FONTE: As autoras (2024).

A tabela verdade para o display de sete segmentos com ânodo comum é apresentada abaixo. Essa tabela será fundamental para a análise dos resultados obtidos durante a simulação, permitindo verificar o comportamento correto de cada segmento em relação aos dígitos a serem exibidos.

FIGURA 3 – ESQUEMÁTICO RTL DO PROJETO 1

Tabela Verdade para Display de 7 Segmentos (Anodo Comum)

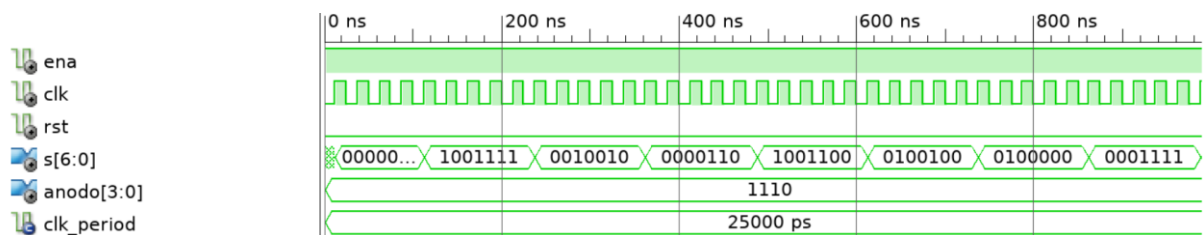
Dígito	a	b	c	d	e	f	g	Representação binária
0	0	0	0	0	0	0	1	1111110
1	1	0	0	1	1	1	1	0110000
2	0	0	1	0	0	1	0	1101101
3	0	0	0	0	1	1	0	1110001
4	1	0	0	1	1	0	0	0110011
5	0	1	0	0	1	0	0	1011011
6	0	1	0	0	0	0	0	1011111
7	0	0	0	1	1	1	1	1110000
8	0	0	0	0	0	0	0	1111111
9	0	0	0	0	1	0	0	1111011

FONTE: Desconhecida.

2.2 SIMULAÇÕES

Para simplificar a simulação, o número de ciclos de *clock* necessários para a troca de valor foi ajustado para 5 períodos. Uma simulação comportamental (Anexo 2) foi realizada para verificar o funcionamento do decodificador. O *testbench* foi configurado de modo que, a cada 5 ciclos de *clock*, o valor da saída mudasse, desde que o *enable* estivesse ativado. Se o reset estivesse em 1, os valores retornariam a 0. Os resultados da simulação, apresentados na figura abaixo, confirmaram que o temporizador funcionou corretamente para cada intervalo de tempo definido.

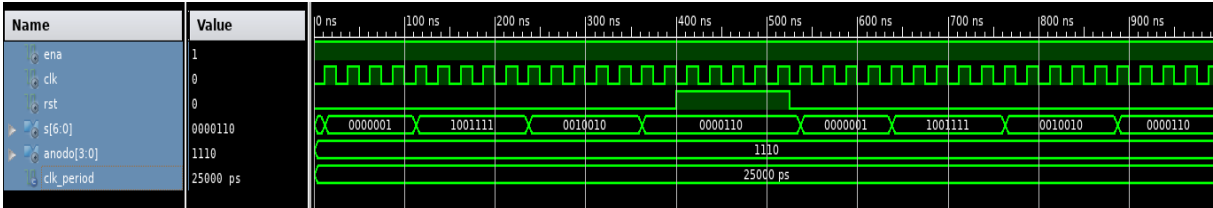
FIGURA 4 – SIMULAÇÃO DO PROJETO 1



FONTE: As autoras (2024).

Exemplo do funcionamento do reset durante simulação, em que é possível identificar que após o acionamento do reset, o contador é reiniciado, já que o valor da variável *S* se torna '0000001' após reset for acionado e desativado que pela tabela verdade do display é são os pontos necessários para formar o caractere 0.

FIGURA 5 – SIMULAÇÃO 2 DO PROJETO 1



FONTE: As autoras (2024).

2.3 CÓDIGO COMENTADO

O código comentado é dividido em duas partes, no código de fato da placa física e no código para simular, com as mudanças que seriam feitas se fosse possível testar na placa.

2.3.1 VHDL Module


O código VHDL Module, do projeto 1, está apresentado sem alterações no Anexo 1 deste presente trabalho, implementa um temporizador que conta de 0 a 9, exibindo os valores em um display de sete segmentos. Ele possui três entradas: ‘ena’ (enable), ‘clk’ (clock) e ‘rst’ (reset), e gera duas saídas: ‘S’, que controla os segmentos do display, assim é uma array de tamanho 7, e ‘anodo’, que habilita qual dígito está ativo sendo uma array de tamanho 4. Dentro do processo, o temporizador é reiniciado com o sinal de reset, e a cada ciclo de clock, se ‘ena’ estiver ativado, um contador interno (‘temp’) é incrementado até 25 milhões, representando meio segundo. Quando esse valor é alcançado, um contador (‘contad’) é incrementado, e, ao atingir 10, ele é resetado. A contagem em formato BCD é então convertida e utilizada para determinar quais segmentos do display devem ser ativados, permitindo a exibição correta dos números.

2.3.2 VHDL Test Bench

O código VHDL Test Bench, do projeto 1, está apresentado sem alterações no Anexo 2 deste presente trabalho.

2.4 RECURSOS LÓGICOS

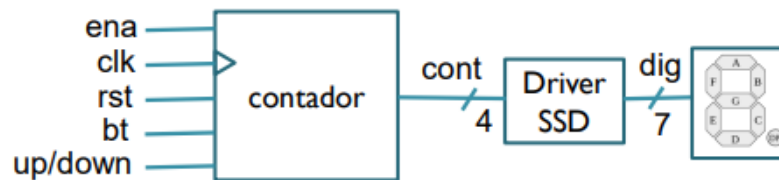
Os recursos lógicos resumem a utilização do dispositivo e os recursos do projeto 1 estão apresentados abaixo.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	33	9,312	1%	
Number of 4 input LUTs	48	9,312	1%	
Number of occupied Slices	53	4,656	1%	
Number of Slices containing only related logic	53	53	100%	
Number of Slices containing unrelated logic	0	53	0%	
Total Number of 4 input LUTs	96	9,312	1%	
Number used as logic	48			
Number used as a route-thru	48			
Number of bonded IOBs	14	232	6%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.16			

3 PROJETO 2 - CONTADOR 0-9

O circuito descrito na figura abaixo foi projetado como um contador que opera na faixa de 0 a 9, incrementando ou decrementando seu valor a cada vez que o botão 'bt' é pressionado.

FIGURA 6 – ESQUEMÁTICO CONCEITUAL

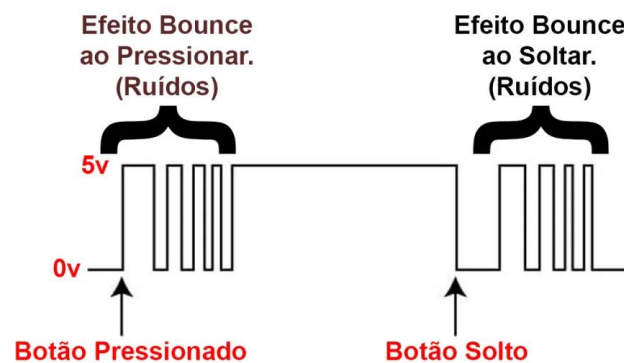


FONTE: 'LAB 3', Sibilla França (20??).

Para garantir um funcionamento correto, foi implementado um sistema de 'debouncing' para o botão, conforme ilustrado na figura abaixo. A lógica do contador foi configurada de forma que, quando a entrada 'up/down' está igual a '1', a contagem ocorre de maneira progressiva; por outro lado, quando essa entrada é igual a '0', a contagem é regressiva. Além disso, o circuito reinicia a contagem quando atinge o valor máximo (caso de contagem progressiva) ou o valor mínimo (caso de contagem regressiva). O sistema também inclui entradas de clock, 'clk', reset, 'rst', e habilitação, 'ena', sendo que a contagem só ocorre quando 'ena' está ativado.

FIGURA 7 – ESQUEMÁTICO CONCEITUAL

Efeito que ocorre ao pressionar o Botão 1 vez

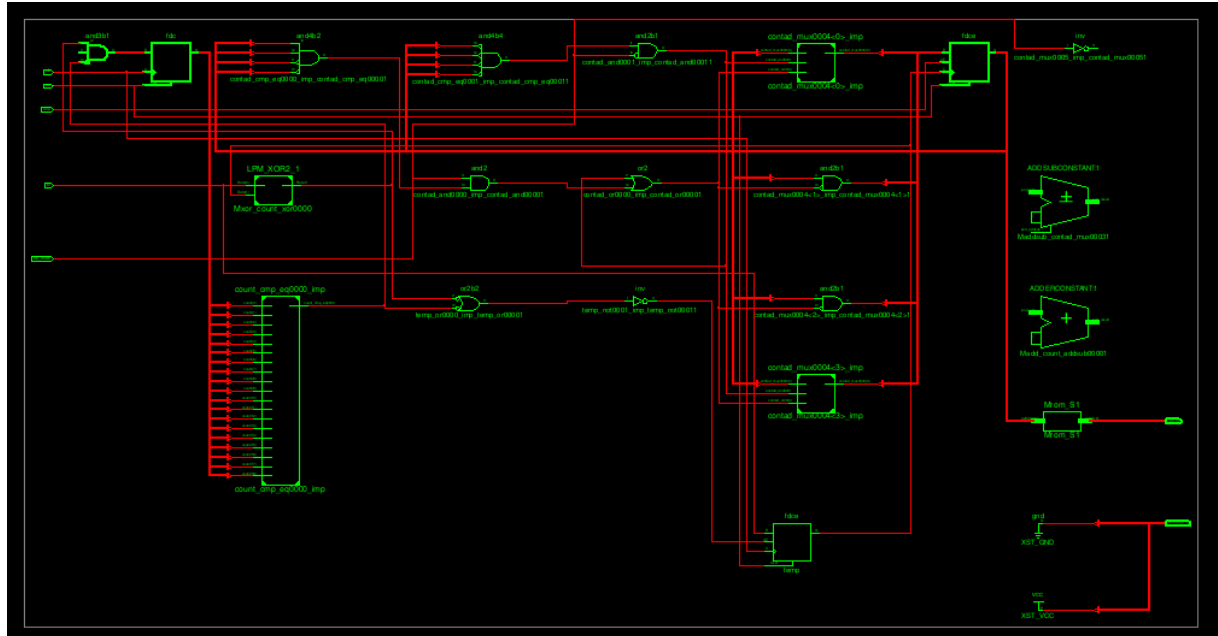


FONTE: Blog da Robótica (2021).

3.1 ESQUEMÁTICOS RTL

O esquemático RTL do projeto 2 está apresentado na figura abaixo, e mostra que as entradas e saídas do projeto.

FIGURA 8 – ESQUEMÁTICO RTL DO PROJETO 2

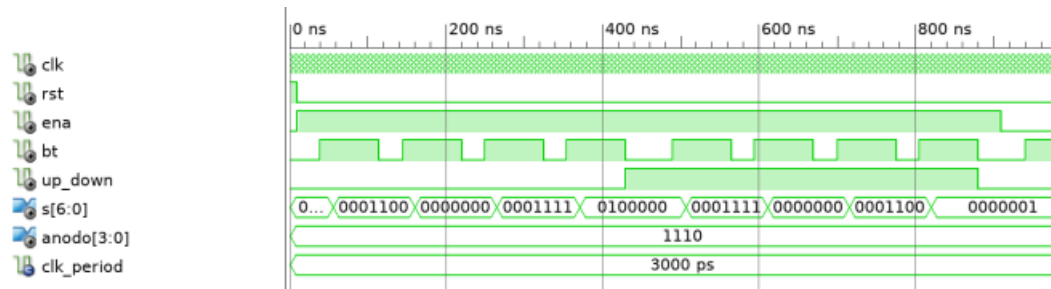


FONTE: As autoras (2024).

3.2 SIMULAÇÕES

Para simplificar a simulação, o número de períodos foi reduzido para apenas 5 ciclos fosse considerado o suficiente para ativar o processo de debounce do botão. A simulação mostra várias interações com o botão de incremento, o que resulta na alteração dos valores exibidos na saída 'S' a cada pressionamento. É importante voltar à tabela verdade apresentada anteriormente para entender como esses valores são representados. Além disso, a simulação revela que a mudança nos valores da entrada 'up/down' permite observar o comportamento reverso dos valores de 'S', ou seja, se antes estavam subindo, agora começam a descer. Por fim, a figura abaixo também ilustra os efeitos das entradas 'enable' e reset no funcionamento do circuito.

FIGURA 9 – SIMULAÇÃO DO PROJETO 2



FONTE: As autoras (2024).

3.3 CÓDIGO

Nessa seção será apresentado não apenas o código do VHDL Module, mas também o código realizado para que a simulação seja possível.

3.3.1 VHDL Module

O código VHDL implementado, disponível no Anexo 3 deste trabalho, foi estruturado em dois processos principais. O primeiro é responsável pelo debouncing do botão, e o sinal gerado por esse processo é essencial para o funcionamento do segundo, que realiza a contagem e inclui as funcionalidades de habilitação, reset e controle de direção (up/down).

O primeiro processo é sensível ao clock e ao reset. Quando o reset é ativado, um contador interno é zerado. Se o botão é pressionado e ocorre uma mudança de estado, o contador interno é incrementado até atingir um limite máximo, momento em que o estado do botão é atualizado. A janela de tempo considerada para reconhecer um clique no botão é de 10 ms. Os parâmetros 'generic' 'fclk' e 'windown' são definidos como 50.000 (em kHz, pois o sinal de clock da placa é de 50 MHz) e 10 (em milissegundos), respectivamente. Dessa forma, a multiplicação desses 'generics' resulta na quantidade de ciclos de clock necessários para formar uma janela de 10 ms, permitindo a aceitação do clique do botão.


No segundo processo, o contador é ajustado com base na habilitação e no estado do botão, incrementando ou decrementando conforme a direção escolhida. Um vetor de 4 bits representa o valor atual do contador, que é convertido para exibição em um display de sete segmentos. A saída do display é determinada por condições que ativam os segmentos correspondentes aos dígitos de 0 a 9, permitindo um controle intuitivo da contagem.

3.3.2 VHDL Test Bench

O código VHDL do Test Bench, referente ao projeto 2, é apresentado sem modificações no Anexo 4 deste trabalho.

3.4 RECURSOS LÓGICOS

Os recursos lógicos resumem a utilização do dispositivo e os recursos do projeto 2 estão apresentados abaixo.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	24	9,312	1%	
Number of 4 input LUTs	39	9,312	1%	
Number of occupied Slices	31	4,656	1%	
Number of Slices containing only related logic	31	31	100%	
Number of Slices containing unrelated logic	0	31	0%	
Total Number of 4 input LUTs	57	9,312	1%	
Number used as logic	39			
Number used as a route-thru	18			
Number of bonded IOBs	16	232	6%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	3.00			

CONCLUSÃO

Este relatório detalhou a implementação de dois projetos fundamentais na disciplina de Microeletrônica I, focando em temporizadores e contadores digitais. O primeiro projeto consistiu na criação de um temporizador, que contava de 0 a 9 com incrementos a cada meio segundo, enquanto o segundo abordou um contador ajustável, capaz de realizar contagens progressivas ou regressivas. Ambas as implementações foram realizadas utilizando a FPGA Nexys 2 e foram acompanhadas por simulações que confirmaram o funcionamento correto dos circuitos. A utilização de técnicas como *debouncing* e a implementação de circuitos lógicos demonstraram a eficácia das soluções propostas. Os resultados obtidos mostram a importância desses dispositivos em sistemas eletrônicos, evidenciando sua aplicabilidade em diversas áreas.

REFERÊNCIAS

PEDRONI, V. A. Circuit Design with VHDL. 2 ed. MIT Press, 2010.

ANEXO 1 – VHDL MODULE – PROJETO 1

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.NUMERIC_STD.ALL;entity Temporizador is
    Port ( ena : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           s : out STD_LOGIC_VECTOR (6 downto 0);
           anodo: out STD_logic_vector (3 downto 0))
    ;
end Temporizador;

architecture Behavioral of Temporizador is
    signal D : STD_logic_vector (3 downto 0);

begin

    process(rst, clk)
        variable temp : natural range 0 to 25_000_000;
        variable contad : natural range 0 to 9;

    begin

        if(rst='1') then
            temp := 0;
            contad := 0;

            elsif(clk'event and clk='1' and ena='1') then

                if(ena='1') then
                    temp := temp +1;
                    if(temp = 5) then
                        temp := 0;
                        contad := contad +1;
                        if(contad = 10) then
                            contad := 0;
                        end if;
                    end if;
                end if;
            end if;
        end if;
    end process;
end Behavioral;

```



```

    end if;

    D <= conv_STD_LOGIC_VECTOR(contad, 4);
end if;
end process;

anodo <= "1110";

S <= "0000001" WHEN D="0000" ELSE
    "1001111" WHEN D="0001" ELSE
    "0010010" WHEN D="0010" ELSE
    "0000110" WHEN D="0011" ELSE
    "1001100" WHEN D="0100" ELSE
    "0100100" WHEN D="0101" ELSE
    "0100000" WHEN D="0110" ELSE
    "0001111" WHEN D="0111" ELSE
    "0000000" WHEN D="1000" ELSE
    "0001100" WHEN D="1001" ELSE
    "1111111" ;

end Behavioral;

```

ANEXO 2 – VHDL TEST BENCH – PROJETO 1

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Temporizador_tb IS
END Temporizador_tb;

ARCHITECTURE behavior OF Temporizador_tb IS

    COMPONENT Temporizador
    PORT(
        ena : IN std_logic;
        clk : IN std_logic;
        rst : IN std_logic;
        s : OUT std_logic_vector(6 downto 0);
        anodo : OUT std_logic_vector(3 downto 0)
    );
    END COMPONENT;

    signal ena : std_logic := '0';
    signal clk : std_logic := '0';
    signal rst : std_logic := '0';

    signal s : std_logic_vector(6 downto 0);
    signal anodo : std_logic_vector(3 downto 0);

    constant clk_period : time := 25 ns;

BEGIN

    uut: Temporizador PORT MAP (
        ena => ena,
        clk => clk,
        rst => rst,
        s => s,

```

```
        anodo => anodo
    );
ena<='1';

clk_process :process
begin
    clk <= '0';
    wait for clk_period/2;
    clk <= '1';
    wait for clk_period/2;
end process;

stim_proc: process
begin

    wait for clk_period*10;

    wait;
end process;

END;
```

ANEXO 3 – VHDL MODULE – PROJETO 2

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_arith.ALL;
entity botao is
    generic (fclk: integer := 50_000;
            twindow: integer := 10);
    Port ( ena : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          rst : in  STD_LOGIC;
          bt  : in  STD_LOGIC;
          up_down : in  STD_LOGIC;
          S : out  STD_LOGIC_VECTOR (6 downto 0);
          anodo : out  STD_LOGIC_VECTOR (3 downto 0));
end botao;

architecture Behavioral of botao is
    signal temp: std_logic := '0';
    signal count: integer range 0 to (fclk * twindow) := 0;
    constant max: integer := 5;
    signal D : STD_LOGIC_VECTOR (3 downto 0);

begin

process (clk, rst)
    begin
        if rst = '1' then
            count <= 0;
            temp <= '0';
        elsif (clk'event and clk='1') then
            if(temp /= bt) then
                count <= count + 1;
                if (count = max) then
                    temp <= bt;
                    count <= 0;
                end if;
            else
                count <= 0;
            end if;
        end if;
    end process;
end Behavioral;

```

```

        end if;
    end if;
end process;

process(rst, temp)
    variable contad: natural range 0 to 9 := 0;
begin
    if rst = '1' then
        contad := 0;
    elsif (ena='1' and temp'event and temp = '1') then
        if contad = 9 and up_down = '1' then
            contad := 0;
        elsif contad = 0 and up_down = '0' then
            contad := 9;
        else
            if up_down = '1' then
                contad := contad + 1;
            else
                contad := contad - 1;
            end if;
        end if;
    end if;
end if;

    D <= std_logic_vector(to_unsigned(contad, 4));
end process;

anodo <= "1110";

S <= "0000001" when D = "0000" else
    "1001111" when D = "0001" else
    "0010010" when D = "0010" else
    "0000110" when D = "0011" else
    "1001100" when D = "0100" else
    "0100100" when D = "0101" else
    "0100000" when D = "0110" else
    "0001111" when D = "0111" else
    "0000000" when D = "1000" else
    "0001100" when D = "1001" else
    "1111111";

```

```
end Behavioral;
```

ANEXO 4 – VHDL TEST BENCH – PROJETO 2

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

entity botao_tb is
end botao_tb;
architecture behavior of botao_tb is
    signal clk      : STD_LOGIC := '0';
    signal rst      : STD_LOGIC := '1';
    signal ena      : STD_LOGIC := '0';
    signal bt       : STD_LOGIC := '0';
    signal up_down  : STD_LOGIC := '0';
    signal S        : STD_LOGIC_VECTOR (6 downto 0);
    signal anodo    : STD_LOGIC_VECTOR (3 downto 0);

    constant clk_period : time := 3 ns;

begin

    uut: entity work.botao
        generic map (
            fclk => 50_000,
            twindown => 10
        )
        port map (
            clk => clk,
            rst => rst,
            ena => ena,
            bt => bt,
            up_down => up_down,
            S => S,
            anodo => anodo
        );
```

```

clk_process : process
begin
    while true loop
        clk <= '0';
        wait for clk_period / 2;
        clk <= '1';
        wait for clk_period / 2;
    end loop;
end process;

stimulus_process : process
begin
    wait for 10 ns;
    rst <= '0';
    ena <= '1';

    wait for 10 * clk_period;
    bt <= '1';
    wait for 25 * clk_period;
    bt <= '0';

    wait for 10 * clk_period;
    bt <= '1';
    wait for 25 * clk_period;
    bt <= '0';

    wait for 10 * clk_period;
    bt <= '1';
    wait for 25 * clk_period;
    bt <= '0';

    wait for 10 * clk_period;
    bt <= '1';
    wait for 25 * clk_period;
    bt <= '0';

    up_down <= '1'; -- Testando o modo de contagem crescente
    wait for 10 * clk_period;

```

```

        wait for 10 * clk_period;
bt <= '1';
wait for 25 * clk_period;
bt <= '0';

        wait for 10 * clk_period;
bt <= '1';
wait for 25 * clk_period;
bt <= '0';

        wait for 10 * clk_period;
bt <= '1';
wait for 25 * clk_period;
bt <= '0';

        wait for 10 * clk_period;
bt <= '1';
wait for 25 * clk_period;
bt <= '0';

up_down <= '0'; -- Testando o modo de contagem decrescente
wait for 10 * clk_period;

ena <= '0';
        wait for 10 * clk_period;
bt <= '1';
wait for 25 * clk_period;
bt <= '0';

        wait for 10 * clk_period;
bt <= '1';
wait for 25 * clk_period;
bt <= '0';

        wait for 10 * clk_period;
bt <= '1';
wait for 25 * clk_period;
bt <= '0';

```



```
        wait for 10 * clk_period;
    bt <= '1';
    wait for 25 * clk_period;
    bt <= '0';
    wait for 10 * clk_period;
    ena <= '1';

    wait;
end process;

end behavior;
```