UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIAS
CURSO DE ENGENHARIA ELÉTRICA

LUCAS FILUS RAMOS - GRR20202598
VITORIA MARIANA DA ROCHA GASINO - GRR20202564

TE351 DA – LABORATÓRIO 3

CURITIBA
2024

UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

Lucas Filus Ramos
Vitoria Mariana da Rocha Gasino

TE 351 DA – LABORATÓRIO 3

Relatório técnico apresentado à disciplina Microeletrônica do Curso de Engenharia Elétrica do Setor de Tecnologia da Universidade Federal do Paraná, como parcial para completamento da disciplina.

Orientador: Sibilla Batista da Luz Franca.

CURITIBA

2024

Sumário

1. INTRODUÇÃO

Este relatório tem como objetivo descrever a atividade de laboratório número 3, desenvolvida para a disciplina de Microeletrônica I. O primeiro projeto trata-se de um contador de 0 a 9 a cada meio segundo. Nesta primeira atividade, foram contabilizados quantos ciclos de clock eram necessários para atingir o tempo de 0,5 s, sendo que foi utilizado o clock interno da placa com oscilação de 50MHz. O valor do contador é exibido no display BCD de sete segmentos. Neste projeto também foram implementados os sinais de entrada '*rst*' e '*enable*', no qual, os dois em nível alto, o primeiro zera a contagem de tempo e do valor exibido, e o segundo autoriza a realização das operações.

O segundo projeto também implementa um contador, com os mesmos sinais de entrada, agora utilizando o gatinho do aperto do botão para realizar uma soma ou subtração. Neste segundo código, foi necessário realizar o debounce do botão, sendo necessário que o estado lido no botão permaneça pelo menos 10 ms em valor alto para ser registrado como leitura válidfa. Uma nova entrada foi adicionada a este novo projeto, em que '*up_down*' em valor alto representa um incremento de 1 e em valor baixo representa um decréscimo de 1.

2. PROJETO 1 – Contador de 0,5 s

2.1. Desenvolvimento do código

Para este primeiro projeto, foi utilizado um processo. Com este tipo de estrutura em VHDL é possível construir partes de código sequenciais. Dentro dos parênteses foi passado a lista de sensibilidades, onde o sinal de clock(‘*clk*’) será utilizado para gatilhar nossas operações.

FIGURA 1 – Código da parte 1.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity lab3_proj1 is
       Port ( clk, rst, enable : in std_logic;
         y : out std_logic_vector(6 downto 0);
           a : out STD_LOGIC_vector (3 downto 0));
end entity;

architecture Behavioral of lab3_proj1 is
    signal d : std_logic_vector(3 downto 0);
begin

  process (clk,rst)
   variable contador_y: integer range 0 to 10:=0;
   variable contador_tempo: natural:=0;-- 50 Mhz B8 da placa
  -- a cada 30s -> +1
  -- t=1/p -> tempo=4*10^-8
   begin
      if (rst='1') then
         contador_y:=0;
         contador_tempo:=0;
      elsif ((clk'event and clk='1') AND enable = '1') then
            if (contador_tempo<=25000000) then --no codigo 25000000 no tb 5
               contador_tempo := contador_tempo + 1;
            else
            contador_y:=contador_y+1;
            contador_tempo := 0;
               if(contador_y=10) then
                    contador_y:=0;
               end if;
            end if;
        end if;
   d <=std_logic_vector(to_unsigned(contador_y, 4));
end process;
```

FONTE: Os autores (2024)

Quando o valor de ‘rst’ estiver em alto, são zerados os dois contadores: *contador_y* e *contador_tempo*. Caso o valor de rst esteja baixo, o enable esteja alto e ocorra uma borda positiva do clock, o código encontra duas possibilidades para executar: 1. caso o valor em *contador_tempo* seja menor que 25M(50MHz/2), o *contador_tempo* é incrementado em 1. 2. Caso

o valor de *contador_tempo* atinja o valor de 25M, zera-se o *contador_tempo* e soma +1 ao *contador_y*. Caso *contador_y* atinja o valor 10, ele retorna para 0.

Com o resultado de *contador_y*, é realizada a operação de codificação para o display de sete segmentos, assim como realizado no laboratório 2.

FIGURA 2 – Código da parte 2.

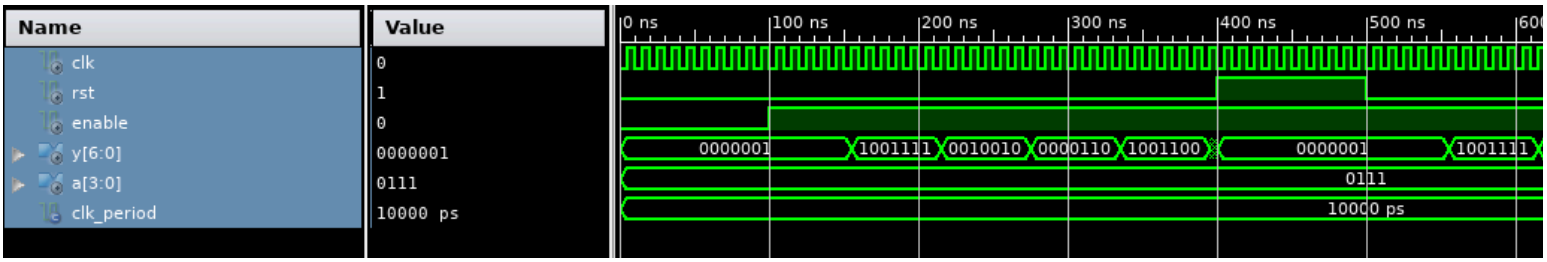
```
    a<= "0111";
    y <= "0000001" when d="0000" else
         "1001111" when d="0001" else
         "0010010" when d="0010" else
         "0000110" when d="0011" else
         "1001100" when d="0100" else
         "0100100" when d="0101" else
         "0100000" when d="0110" else
         "0001111" when d="0111" else
         "0000000" when d="1000" else
         "0000100" when d="1001" else
         "1111111";


end Behavioral;
```

FONTE: Os autores (2024)

2.2. Implementação do Test Bench e resultados

Para avaliar o funcionamento do circuito implementado acima,foi preparado um teste bench. Para fins de teste, foi utilizado ao invés de 25M para gatilhar a operação, 5, e um ciclo de clock igual a 10 ns.

FIGURA 3 – Simulação Behaviour do Projeto 1.



FONTE: Os autores (2024)

3. PROJETO 2 – Contador gatilhado pelo botão

3.1. Desenvolvimento do código

Neste projeto, a intenção era realizar uma soma ou subtração, dependendo do valor de entrada '*up_down*', sendo essa operação gatilhada pela mudança de estado na leitura do push button. Assim como no projeto anterior, a operação só é realizada caso '*enable*' esteja alto. A entrada '*rst*' zera os contadores de tempo e do valor.

Para evitar múltiplas leituras na hora do acionamento do botão, foi preciso realizar o debounce do botão. O tempo definido foi de 10 ms, ou seja, é preciso uma leitura contínua de 10 ms em valor alto para que seja identificado com sinal válido para realizar a operação.

Neste código, foram elaborados dois processos: o primeiro realiza a contagem na hora da leitura do botão. Foram declarados dois sinais internos: *tempo_d* e *temp*. Quando é detectado uma borda positiva do clock, ele verifica primeiramente se o sinal *tempo_d* é igual ao botão. Caso seja diferente, (*bt*='0' e *tempo_d*='0'), ele realiza a contagem até atingir o tempo definido. Quando é atingido, o código realiza a atribuição a *tempo_d* o valor de '*bt*'. Como agora '*bt*' e '*tempo_d*' são iguais, o código não entra na primeira condição, e zera *cont_d*. A variável *tempo_d* está operando como uma flag no nosso código, sendo necessário para o próximo processo.

FIGURA 4 – Código .vhd Projeto 2

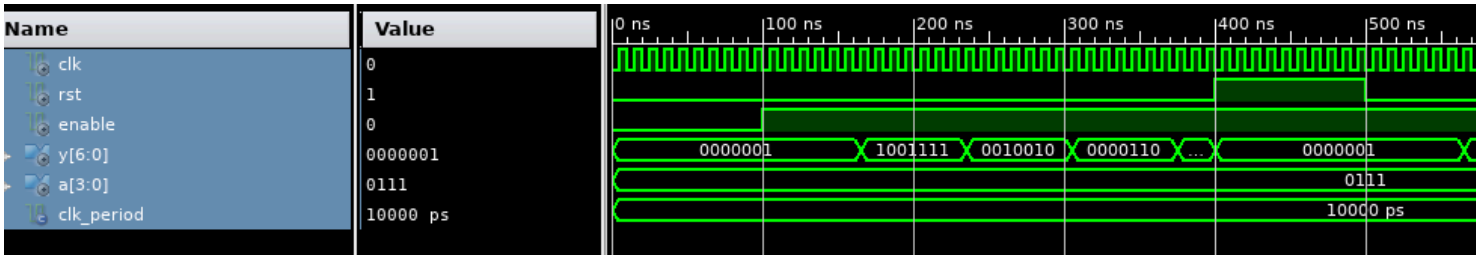
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lab3_proj2 is
    Port ( ena : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           bt : in  STD_LOGIC;
           up_down : in  STD_LOGIC;
           y : out  STD_LOGIC_VECTOR (6 downto 0);
           a : out std_logic_vector(3 downto 0));
end lab3_proj2;

architecture Behavioral of lab3_proj2 is

   signal temp_d : std_logic := '0';
   signal temp: integer;
begin

   --Processo para Debounce
      process(clk)
         variable cont_d: integer range 0 to 50000; --Contador debounce
      begin

         if (clk'event and clk = '1') then
               if(temp_d /= bt) then
                  cont_d := cont_d + 1;
                  if (cont_d = 2) then --NO codigo 50000
                     temp_d <= bt;
                     cont_d := 0;
                  end if;
               else
                  cont_d := 0;
               end if;
         end if;
      end process;
```

FIGURA 4 – Código .vhd Projeto 2- parte 2

```
    --Processo Contador
    process(temp_d,rst,ena,up_down)
       variable cont : integer := 0;
    begin
       if (rst = '0') then
          if (temp_d'event and temp_d = '1') then
             if (ena = '1') then
                if (up_down = '1') then
                   cont := cont + 1;
                   if(cont = 10) then
                      cont := 0;
                   end if;
                else
                   cont := cont - 1;
                   if(cont = -1) then
                      cont := 9;
                   end if;
                end if;
             end if;
          end if;
       else
          cont := 0;
       end if;
   temp <= cont;
end process;
   a <= "0111";
   y <= "0000001" when temp = 0 else  --0
        "1001111" when temp = 1 else  --1
        "0010010" when temp = 2 else  --2
        "0000110" when temp = 3 else  --3
        "1001100" when temp = 4 else  --4
        "0100100" when temp = 5 else  --5
        "1100000" when temp = 6 else  --6
        "0001111" when temp = 7 else  --7
        "0000000" when temp = 8 else  --8
        "0001100";                    --9

end Behavioral;
```

Com a flag *tempo_d*, este segundo processo irá realizar as operações de soma ou subtração. Caso o valor de 'rst' não esteja alto, o processo verifica se houve uma borda positiva em *tempo_d*. Se sim, ele entra nos condicionais para verificar qual operação deve realizar: se '*up_down*' estiver baixo, será realizado cont = cont -1, se estiver alto cont = cont -1. Além disso, o código verifica se os números ultrapassam os limites de 0 até 9.

Com o resultado de cont é realizada a operação de codificação para o display de sete segmentos, assim como realizado no laboratório 2



FONTE: Os autores (2024)

É possível observar que após 6 ciclos de clock e enable = ‘1’, o valor de y vai de ‘0000001’(exibição de 0 no display) para ‘1001111’(exibição de 1 no display). Quando a simulação atinge 400 ns , foi acionado o ‘rst’, é possível observar que o valor de y retorna para ‘0000001’(exibição de 0 no display).

4. CONCLUSÃO

Este relatório apresentou quatro projetos desenvolvidos e efetivos em suas respectivas funcionalidades. Após implementação e teste nos kits NEXYS 2, os resultados foram consistentes e satisfatórios, alinhando-se com os objetivos planejados para cada projeto.

Portanto, foi comprovado que os quatro projetos funcionaram conforme o esperado e atenderam aos requisitos de desempenho quando implementados na placa kit NEXYS 2. Esses resultados validam a eficácia do uso de processos concorrentes e de temporizadores na elaboração de circuitos digitais para aplicações diversas.