ENG1448: Computação Digital - PUC-Rio, 2024.1, 2024.03.11

Laboratório 3: Debounce:

Professor: Felipe Calliari | Monitor: Cristiano Nascimento

Aluno: Pedro Gabriel Serodio Sales e Thiago Levis *Matrícula: 2211911 e 1812899*

1 Introdução

O objetivo deste relatório é apresentar as soluções para o laboratório 3 da disciplina ENG1448 assim como apresentações do laboratório em vídeo.

2 Resolução

1 : **Implementação do debounce para o botão de rotação**

Considerando o funcionamento mecânico interno do Rotary Encoder e suas oscilações de sinal a cada rotação, utilizamos o código proposto nos materiais da disciplina para realizar o debouncing. Para tanto, no código VHDL temos o processo síncrono rotary-filter, que detecta apenas a primeira mudança do sinal e ignora a atividade seguinte do mesmo até que o outro sinal também mude de estado. Dessa forma, filtra-se as oscilações dos sinais A e B originados nos switches do Rotary Enconder, obtendo respectivamente os sinais rotary-q1 e rotary-q2 (estes serão usados para determinar o sentido de rotação).

```
rotary_filter: process(clk)
   begin
      if clk'event and clk='1' then
         rotary_in <= B & A;
         case rotary_in is
            when "00" => rotary_q1 <= '0';
                         rotary_q2 <= rotary_q2;

            when "01" => rotary_q1 <= rotary_q1;
                         rotary_q2 <= '0';

            when "10" => rotary_q1 <= rotary_q1;
                         rotary_q2 <= '1';

            when "11" => rotary_q1 <= '1';
                         rotary_q2 <= rotary_q2;

            when others => rotary_q1 <= rotary_q1;
                           rotary_q2 <= rotary_q2;
         end case;
      end if;
end process rotary_filter;
```

(1)

Figura 1: Rotary Filter

Foi definido outro processo síncrono para definir uma rotação e seu sentido. Assim, foi necessário guardar o valor anterior de rotary-q1 para detectar a sua transição de low para high. Nesse caso, um evento de rotação é sinalizado, e o sentido é definido pelo sinal rotary-q2, como planejado acima.

```
direction: process(clk)
   begin
      if clk'event and clk='1' then

         delay_rotary_q1 <= rotary_q1;
         if rotary_q1='1' and delay_rotary_q1='0' then
            rotary_event <= '1';
            rotary_left <= rotary_q2;
         else
            rotary_event <= '0';
            rotary_left <= rotary_left;
         end if;
      end if;
end process direction;
```

(2)

Figura 2: Direction

2 Implementação do tratamento de debounce do push-button

Num terceiro processo síncrono, além de atualizarmos os LEDs rotacionados para esquerda ou direita, também fazemos a inversão do padrão de LEDs acesos e apagados. Para implementar isso, conectamos o sinal emitido com oscilações do push-button do Rotary Enconder à entrada de um debounce; quando este emite um pulso na sua saída tick, o ROT-CENTER-TICK assume valor lógico 1 durante um ciclo de clock. Assim, dentro do processo síncrono shift-leds avaliamos ROT-CENTER-TICK = 1 e invertemos o padrão de LEDs.

Para a rotação dos LEDs usou-se a lógica de um shift-register circular, ou seja, o bit mais significativo é inserido no menos significativo ou vice-versa, dependendo do sentido de rotação do encoder.

```
shift_leds: process(clk)
begin
   if rising_edge(clk) then

      if (RST = '1') then
         LEDS_reg <= "00000001";
      end if;

      if (rotary_event = '1') then
         if (rotary_left = '1') then
            LEDS_reg <= LEDS_reg(6 downto 0) & LEDS_reg(7);
         else
            LEDS_reg <= LEDS_reg(0) & LEDS_reg(7 downto 1);
         end if;
      end if;

      if (ROT_CENTER_TICK = '1') then
         LEDS_reg <= not LEDS_reg;
      end if;

   end if;
end process shift_leds;
```

(3)

Figura 3: Implementação do Shift e da Inversão dos LED's

2 Funcionamento do Algoritmo

Antes de fazer o upload do código para a placa, foi necessário definir o arquivo de constraints, no qual definiu-se o Rotary Encoder nas entradas A e B para os sinais e ROT-CENTER para o push-button do mesmo. Além disso, o push-button acima do encoder foi associado ao RST, e por fim associaram-se os 8 LEDs e o clock, como visto a seguir:

```
NET "CLK" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "CLK" PERIOD = 20.0ns HIGH 50%;

NET "B" LOC = "K18" | IOSTANDARD = LVTTL | PULLUP ;
NET "A" LOC = "G18" | IOSTANDARD = LVTTL | PULLUP ;
NET "ROT_CENTER" LOC = "V16" | IOSTANDARD = LVTTL | PULLDOWN ;

NET "LEDS<7>" LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDS<6>" LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDS<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDS<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDS<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDS<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDS<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDS<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

NET "RST" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
```

(4)

Figura 4: Constraint File

A placa começa com o LED mais a direita aceso e ao girar a esquerda ou a direita, o código age como o esperado. O botao central é apertado e os LED's invertidos continuam a funcionar normalmente como visto no vídeo abaixo. E por fim, se efetuado um giro para além de uma das bordas, o LED selecionado passa para outro lado.

3 Vídeo de Apresentação

O vídeo de apresentação segue ao lado: **Lab 3: Debounce**
ou acessar pelo url:**`https://youtu.be/_fq4-pWdJOE`**