Relatório - Atividade 7: Timer

Maria Eduarda Soares Romana & Tiago Fernandes Soucek

6 de junho de 2025

1 Resumo

Esta atividade teve como objetivo o desenvolvimento de um temporizador digital regressivo em VHDL, projetado para ser implementado em uma placa FPGA. O sistema realiza uma contagem regressiva em segundos a partir de um valor pré-selecionado e exibe o tempo restante em quatro displays de sete segmentos.

O desenvolvimento do projeto abrangeu as seguintes funcionalidades e componentes principais:

1. **Contagem Regressiva e Controle:** A base é um contador que decrementa a cada segundo, com:
   - **Reset:** Um botão que reinicia a contagem para o valor e apaga o LED de finalização.
   - **Pause:** Um botão que pausa a contagem enquanto estiver pressionado.
   - **Seleção de Tempo:** Duas chaves permitem selecionar um de quatro valores para a contagem regressiva (10s, 30s, 60s, ou 90s).
2. **Geração de Base de Tempo:** Foi implementado um divisor de clock para converter o sinal de entrada de 50 MHz em um pulso de habilitação com frequência de 1 Hz para que a contagem regressiva ocorra em segundos.
3. **Conversão e Exibição de Dados:** Para exibir o tempo restante:
   - **Conversão para BCD:** O valor inteiro do contador é convertido para o formato BCD (*Binary-Coded Decimal*).
   - **Decodificação para 7 Segmentos:** Uma função converte cada dígito BCD no padrão de 7 bits necessário para acender os segmentos corretos nos displays.
4. **Sinalização de Conclusão:** Ao final da contagem regressiva, o sistema para de contar e acende um LED (`LED_DONE`) para mostrar que o tempo acabou.

2 Requisitos do Projeto

Os seguintes itens foram requisitos do projeto:

1. O circuito deve funcionar como um temporizador que realiza uma contagem regressiva em segundos até chegar a zero.
2. O tempo restante da contagem deve ser continuamente apresentado em quatro displays de sete segmentos (SSDs).
3. Deve ser implementada uma função para converter o valor numérico do contador para o formato BCD (*Binary-Coded Decimal*), para permitir a correta exibição nos SSDs.

4. O projeto deve incluir um botão de **reset** que, ao ser pressionado, reinicia a contagem para o valor inicial selecionado.

5. Deve haver um botão para **pausar/desabilitar** a contagem. O timer deve parar enquanto o botão estiver pressionado e continuar a contagem quando for liberado.

6. O sistema deve oferecer diferentes opções de tempo para a contagem regressiva (pelo menos duas), selecionáveis através de chaves.

7. Ao final da contagem , um **LED de sinalização** deve ser aceso para indicar que o tempo se esgotou.

3 Métodos e Técnicas

A implementação foi realizada no software Quartus Prime, e a Figura 1 apresenta o código VHDL desenvolvido para resolver o problema proposto.

```
1  -- ----------------------------------------
2  -- ATV 07 - TIMER
3  -- MARIA EDUARDA SOARES ROMANA SILVA
4  -- TIAGO FERNANDES SOUCEK
5  -- ----------------------------------------
6
7  library IEEE;
8  use IEEE.STD_LOGIC_1164.ALL;
9  use IEEE.NUMERIC_STD.ALL;
10
11 entity atividade7 is
12     Port (
13         CLK_50MHz  : in  STD_LOGIC;
14         BTN_RESET  : in  STD_LOGIC; --ativo-baixo (pressionado = '0')
15         BTN_PAUSE  : in  STD_LOGIC; --ativo-baixo (pressionado = '0')
16         SW_SEL     : in  STD_LOGIC_VECTOR(1 downto 0); -- selecao do tempo
17         LED_DONE   : out STD_LOGIC;
18         HEX0, HEX1, HEX2, HEX3 : out STD_LOGIC_VECTOR(6 downto 0)
19     );
20 end atividade7;
21
22 architecture atividade7 of atividade7 is
23
24     --clock divisor para gerar enable de 1 Hz
25     signal clk_div      : INTEGER range 0 to 49999999 := 0;
26     signal enable_1Hz   : STD_LOGIC := '0';
27
28     --contador
29     signal count        : INTEGER range 0 to 9999 := 0;
30     signal preset_time  : INTEGER range 0 to 9999 := 10;
31     signal running      : STD_LOGIC := '1';
32
33     --BCD
34     signal bcd_0, bcd_1, bcd_2, bcd_3 : STD_LOGIC_VECTOR(3 downto 0);
35
36     --conversor BCD para 7 segmentos
37     function bcd_to_7seg(bcd : STD_LOGIC_VECTOR(3 downto 0)) return
          STD_LOGIC_VECTOR is
38     begin
```

```
39          case bcd is
40              when "0000" => return "1000000"; -- 0
41              when "0001" => return "1111001"; -- 1
42              when "0010" => return "0100100"; -- 2
43              when "0011" => return "0110000"; -- 3
44              when "0100" => return "0011001"; -- 4
45              when "0101" => return "0010010"; -- 5
46              when "0110" => return "0000010"; -- 6
47              when "0111" => return "1111000"; -- 7
48              when "1000" => return "0000000"; -- 8
49              when "1001" => return "0010000"; -- 9
50              when others => return "1111111"; -- Off
51          end case;
52      end function;
53
54  begin
55
56      --clock 1 Hz (gera um pulso enable a cada segundo)
57      process(CLK_50MHz)
58      begin
59          if rising_edge(CLK_50MHz) then
60              if clk_div = 49999999 then
61                  clk_div <= 0;
62                  enable_1Hz <= '1';
63              else
64                  clk_div <= clk_div + 1;
65                  enable_1Hz <= '0';
66              end if;
67          end if;
68      end process;
69
70      --tempo inicial com base nos switches
71      process(SW_SEL)
72      begin
73          case SW_SEL is
74              when "00"   => preset_time <= 10;
75              when "01"   => preset_time <= 30;
76              when "10"   => preset_time <= 60;
77              when others => preset_time <= 90;
78          end case;
79      end process;
80
81      --contador principal
82      process(CLK_50MHz)
83      begin
84          if rising_edge(CLK_50MHz) then
85              -- Reset e ativo-baixo ('0')
86              if BTN_RESET = '0' then
87                  count <= preset_time;
88                  running <= '1';
89                  LED_DONE <= '0';
90
91              -- logica de pause agora responde ao botao pressionado
92              elsif BTN_PAUSE = '0' then
93                  running <= '0';
94
95              elsif enable_1Hz = '1' then
96                  -- contador so decrementa se estiver rodando (running = '1')
```

```
97                  if running = '1' then
98                      if count > 0 then
99                          count <= count - 1;
100                     else -- count chegou a 0
101                         running <= '0';  --para de contar
102                         LED_DONE <= '1'; --acende o LED
103                     end if;
104                 else --se nao estiver pausado nem resetado, pode voltar a
                        rodar
105                     running <= '1';
106                 end if;
107             end if;
108         end if;
109     end process;
110
111     -- este processo combinacional calcula os digitos BCD a cada mudanca no
           'count'
112     process(count)
113     begin
114         bcd_3 <= std_logic_vector(to_unsigned((count / 1000) rem 10, 4));
115         bcd_2 <= std_logic_vector(to_unsigned((count / 100) rem 10, 4));
116         bcd_1 <= std_logic_vector(to_unsigned((count / 10) rem 10, 4));
117         bcd_0 <= std_logic_vector(to_unsigned(count rem 10, 4));
118     end process;
119
120     --saida para os displays de 7 segmentos
121     HEX0 <= bcd_to_7seg(bcd_0);
122     HEX1 <= bcd_to_7seg(bcd_1);
123     HEX2 <= bcd_to_7seg(bcd_2);
124     HEX3 <= bcd_to_7seg(bcd_3);
125
126 end atividade7;
```

Listing 1: Código VHDL: ATV O7 - TIMER

A entidade, denominada `atividade7`, foi definida para implementar um temporizador digital regressivo. Suas características, como os tempos pré-selecionáveis e o número de displays, são fixadas diretamente na descrição do hardware.

As portas da entidade foram declaradas da seguinte forma:

1. `CLK_50MHz : in STD_LOGIC`: Sinal de clock de entrada de 50 MHz.
2. `BTN_RESET : in STD_LOGIC`: Entrada para o botão de reset.
3. `BTN_PAUSE : in STD_LOGIC`: Entrada para o botão de pausa.
4. `SW_SEL : in STD_LOGIC_VECTOR(1 downto 0)`: Entradas de duas chaves para a seleção de um dos quatro tempos pré-configurados para a contagem.
5. `LED_DONE : out STD_LOGIC`: Saída para um LED que é aceso para sinalizar o término da contagem.
6. `HEX0, HEX1, HEX2, HEX3 : out STD_LOGIC_VECTOR(6 downto 0)`: Saídas para controlar quatro displays de sete segmentos.

A arquitetura, denominada `atividade7`, implementa a lógica do temporizador:

1. **Processo Divisor de Clock:** Este processo é responsável por gerar um sinal de habilitação (`enable_1Hz`) com uma frequência de 1 Hz. Ele utiliza um contador inteiro (`clk_div`) que, ao atingir o valor correspondente a um segundo (49.999.999 ciclos de um clock de 50 MHz), ativa o sinal `enable_1Hz` por um ciclo de clock.

2. **Processo de Seleção de Tempo:** É um processo correspondente às chaves de seleção (`SW_SEL`). Utilizando um `case`, ele atribui um valor inicial ao sinal `preset_time` (10, 30, 60 ou 90 segundos) com base na combinação das chaves.

3. **Processo do Contador Principal:** Este é o processo sequencial central do temporizador.

   - **Lógica de Reset:** Se `BTN_RESET = '0'`, o contador (`count`) é carregado com o valor de `preset_time`, a máquina de estados é habilitada (`running <= '1'`) e o LED de finalização é apagado.
   - **Lógica de Pausa:** Se `BTN_PAUSE = '0'`, a contagem é pausada ao desabilitar a máquina de estados (`running <= '0'`). Ao soltar o botão, a contagem é retomada.
   - **Lógica de Contagem:** A cada pulso do sinal `enable_1Hz`, se a máquina de estados estiver habilitada (`running = '1'`) e a contagem for maior que zero, o valor de `count` é decrementado.
   - **Lógica de Finalização:** Quando `count` atinge zero, a máquina de estados é desabilitada (`running <= '0'`) e o `LED_DONE` é aceso.

4. **Processo de Conversão para BCD:** Este processo é sensível a qualquer alteração no valor do contador `count`. Ele calcula e atualiza continuamente os quatro dígitos BCD que representam o valor atual do contador.

5. **Atribuições para os Displays:** As saídas `HEX0` a `HEX3` são atribuídas de forma concorrente. Cada uma recebe o resultado da função `bcd_to_7seg`, que converte o respectivo dígito BCD para o código de 7 segmentos correspondente.

4 Resultados e Considerações

Na Figura 1, é possível visualizar o esquema lógico gerado pelo Quartus Prime após a compilação do código.

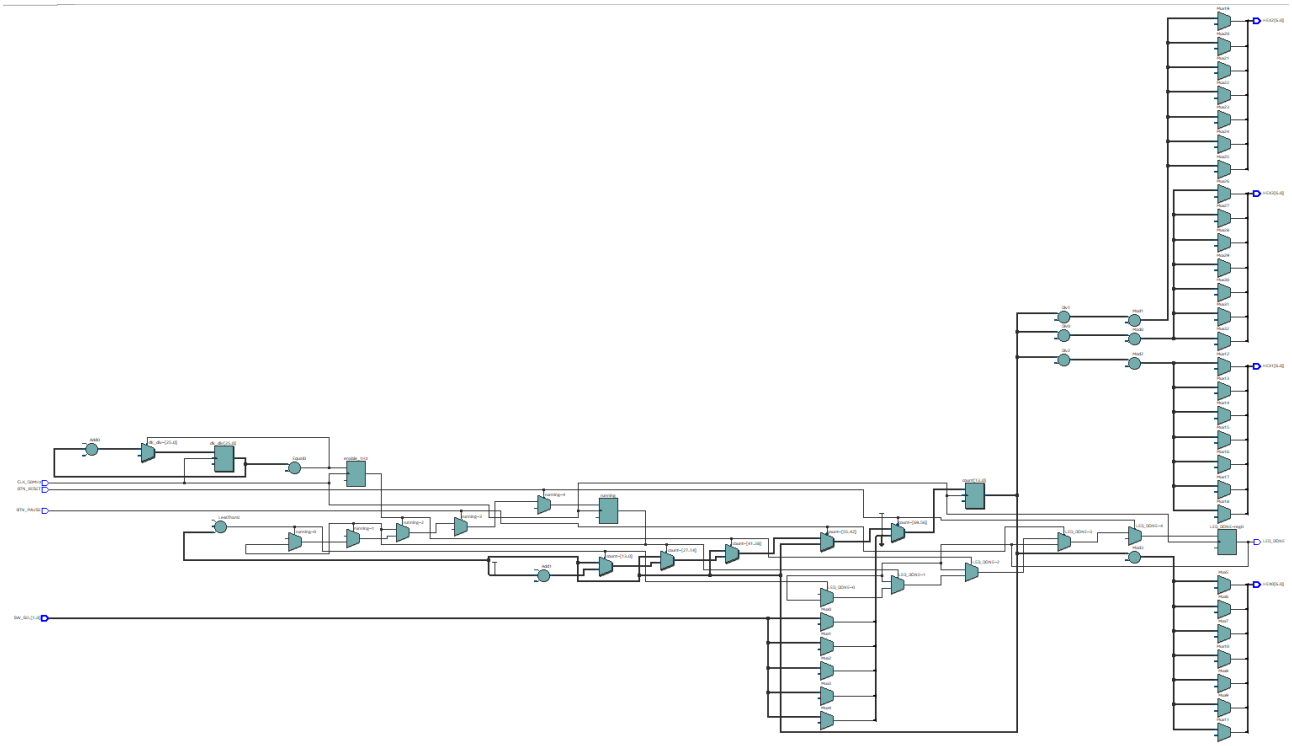

Figura 1: Esquema da distribuição das portas.

Fonte: Autoria própria.

Neste `LINK` (clique aqui) há o vídeo demonstrando o funcionamento do projeto na placa.