Relatório - Atividade 5: Luzes Piscando

Maria Eduarda Soares Romana & Tiago Fernandes Soucek

5 de junho de 2025

1 Resumo

Esta atividade teve como objetivo o desenvolvimento de acender LEDs de forma sequencial em FPGA, gerenciando o acendimento de 10 LEDs, um por vez, em um padrão progressivo de LED0 a LED9 e, em seguida, regressivo de LED9 a LED0.

O desenvolvimento envolveu duas etapas principais:

1. Controle de Estado e Sequência: Foi implementada a lógica para:
   - Resetar o circuito: Utilizando um botão (ativo em nível lógico '0'), esta função reinicia a sequência para o LED0, apaga todos os LEDs e define a direção da varredura como progressiva (de LED0 para LED9).
   - Habilitar/Pausar o circuito: Através de uma chave, permite-se iniciar ou interromper a progressão do LED aceso.
   - Gerenciamento da sequência: A lógica controla a posição do LED atualmente aceso e inverte automaticamente a direção da varredura quando os LEDs das extremidades (LED0 ou LED9) são alcançados.
2. Controle de Velocidade: Foi implementado um seletor de velocidade, controlado por uma chave, que oferece duas opções para a taxa de transição entre os LEDs:
   - Velocidade Lenta: A transição entre LEDs ocorre aproximadamente a cada 0,5 segundos. Isto é obtido configurando um contador divisor de clock para 25.000.000 ciclos (considerando o clock de entrada de 50 MHz).
   - Velocidade Rápida: A transição entre LEDs ocorre aproximadamente a cada 0,1 segundos, com o contador divisor de clock configurado para 5.000.000 ciclos.

2 Requisitos do Projeto

A atividade teve como objetivo o desenvolvimento de um circuito digital em FPGA para o controle sequencial de LEDs. Os seguintes itens foram requisitos do projeto:

1. O circuito de controle de LEDs deve ter 10 LEDs distintos, com apenas um LED esteja aceso por vez.
2. A sequência de acendimento dos LEDs deve ser começar pelo LED0 e progredindo até o LED9, para então inverter a direção e retornar do LED9 até o LED.
3. Deve ser implementada um **reset** para o circuito, controlado por um botão configurado como ativo em nível lógico baixo ('0'). A ativação do reset deve:
   - Reiniciar a contagem da posição do LED para o estado inicial (LED0).

- Apagar todos os 10 LEDs.
- Configurar a direção da sequência de acendimento como progressiva (LED0 a LED9).

4. O circuito deve ter uma entrada de **habilitação**, controlada por uma chave. Esta chave deve:
   - Iniciar ou continuar a sequência de acendimento dos LEDs, quando habilitada.
   - Pausar a sequência de acendimento dos LEDs, mantendo o estado atual, quando desabilitada.
5. O sistema deve ter duas opções de **velocidade** para a transição do LED aceso (por exemplo, lenta e rápida), através de uma chave.

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 05 - LUZES PISCANDO
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.STD_LOGIC_UNSIGNED.ALL;
10
11 entity atv5 is
12     Port (
13         clk: in  STD_LOGIC; -- sinal de clock
14         btn_reset: in  STD_LOGIC; -- botao de reset
15         habilita: in  STD_LOGIC; -- chave para habilitar/pausar
16         velocidade: in  STD_LOGIC; -- chave para escolher velocidade
17         leds: out STD_LOGIC_VECTOR (9 downto 0) -- saida para os 10 LEDs
18     );
19 end atv5;
20
21 architecture atv5 of atv5 is
22     signal clk_divisor: integer := 0; -- contador do divisor de clock
23     signal posicao: integer range 0 to 9 := 0; -- posicao atual do LED aceso
24     signal sentido: STD_LOGIC := '1'; -- '1': vai para frente (0 a 9), '0':
          volta (9 a 0)
25     signal pulso: STD_LOGIC := '0'; -- sinal de pulso para mudar o LED
26 begin
27     process(clk)
28         constant lento : integer := 25_000_000; -- ~0,5s para 50 MHz
29         constant rapido : integer := 5_000_000; -- ~0,1s
30         variable maximo : integer;
31     begin
32         if rising_edge(clk) then
33             if velocidade = '0' then
34                 maximo := lento;
35             else
36                 maximo := rapido;
```

```
37              end if;
38
39              if clk_divisor < maximo then
40                  clk_divisor <= clk_divisor + 1;
41                  pulso <= '0';
42              else
43                  clk_divisor <= 0;
44                  pulso <= '1';
45              end if;
46          end if;
47      end process;
48
49      -- Logica de controle dos LEDs
50      process(clk)
51      begin
52          if rising_edge(clk) then
53              if btn_reset = '0' then -- botao pressionado (reset ativo)
54                  posicao <= 0;
55                  sentido <= '1';
56                  leds <= (others => '0');
57              elsif habilita = '1' and pulso = '1' then
58                  leds <= (others => '0');
59                  leds(posicao) <= '1';
60
61                  if sentido = '1' then
62                      if posicao = 9 then
63                          sentido <= '0';
64                          posicao <= posicao - 1;
65                      else
66                          posicao <= posicao + 1;
67                      end if;
68                  else
69                      if posicao = 0 then
70                          sentido <= '1';
71                          posicao <= posicao + 1;
72                      else
73                          posicao <= posicao - 1;
74                      end if;
75                  end if;
76              end if;
77          end if;
78      end process;
79  end atv5;
```

Listing 1: Código VHDL: ATV O5 - Luzes Piscando

A entidade, denominada `atv5`, foi definida para controlar o sequenciamento de LEDs. O número de LEDs (10) e as constantes de temporização para as velocidades, são fixadas diretamente na descrição do hardware.

As portas da entidade foram declaradas da seguinte forma:

1. `clk : in STD_LOGIC`: Corresponde ao sinal de clock de entrada (especificado como 50 MHz), que sincroniza todas as operações internas.
2. `btn_reset : in STD_LOGIC`: Entrada para o botão de reset (ativo em nível lógico '0').
3. `habilita : in STD_LOGIC`: Chave de entrada para habilitar ('1') ou pausar ('0') a sequência de acendimento dos LEDs.

4. `velocidade : in STD_LOGIC`: Chave de entrada para selecionar a velocidade da sequência. '0' para lento e '1' para rápido.

5. `leds : out STD_LOGIC_VECTOR(9 downto 0)`: Saída de 10 bits que controla diretamente o estado dos LEDs (um bit para cada LED).

A arquitetura, também denominada `atv5`, implementa a lógica de controle sequencial dos LEDs utilizando dois processos principais, ambos síncronos com a borda de subida do sinal `clk`:

1. **Processo Divisor de Clock:** Este processo é responsável por gerar um sinal de pulso periódico (`pulso`), que dita o momento da transição para o próximo LED.
   - A frequência do `pulso` é determinada pela entrada `velocidade`.
   - Se `velocidade = '0'`, o contador `clk_divisor` atinge o valor da constante `lento`, gerando um pulso aproximadamente a cada 0,5 segundos com um clock de 50 MHz.
   - Se `velocidade = '1'`, o contador `clk_divisor` atinge o valor da constante `rapido`, gerando um pulso aproximadamente a cada 0,1 segundos.

2. **Processo de Controle dos LEDs:** Este processo gerencia o estado atual dos LEDs, a posição do LED aceso e a direção da sequência, respondendo ao sinal de `pulso` gerado pelo divisor de clock, bem como aos sinais de controle `btn_reset` e `habilita`.
   - **Lógica de Reset:** Se `btn_reset = '0'`, o circuito é inicializado.
   - **Lógica de Sequenciamento:** Se `habilita = '1'` e o `pulso = '1'`, indicando que é hora de mudar o LED.

Desta forma, a arquitetura controla um LED por vez, com direção e velocidade ajustáveis, e com a possibilidade de reset e pausa.

4 Resultados e Considerações

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

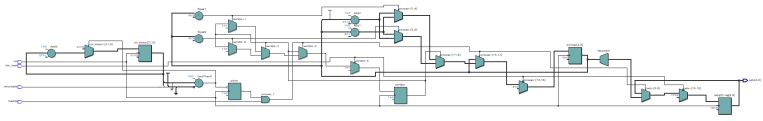

Figura 1: Esquema da distribuição das portas.

Fonte: Autoria própria.

Neste link há o vídeo demonstrando o funcionamento do projeto na placa.