Relatório

Tópico: Aulas 4 & 5 - Contadores
Grupo: Adhemar Molon Neto 14687681
Alessandro Rodrigues Pereira da Silva 15436838
Link do Github: <>

Parte 1

Descrição
Foi desenvolvido em VHDL um contador de 8 bits, usando oito instâncias de flip-flops tipo T, de modos que cada um muda seu valor quando a anterior muda 2 vezes. Foram usados aproximadamente 102 elementos lógicos e 33 pins.

Códigos VHDL
d_latch.vhd
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity d_latch is
	port(
		Data, Clock	: in std_logic;
		Qu, notQu	: out std_logic
	);
END d_latch;
ARCHITECTURE main of d_latch is
	signal R_g, S_g, Qa, Qb : std_logic;
	attribute keep : boolean;
	attribute keep of R_g, S_g, Qa, Qb : signal is true;
	begin
		S_g <= not (Data and Clock);
		R_g <= not ((not Data) and Clock);
		Qa <= not (S_g and Qb);
		Qb <= not (R_g and Qa);
		Qu <= Qa;
		notQu <= Qb;
END main;
```
d_flip_flop.vhd
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

ENTITY d_flip_flop is
	port(
		D, Clk	: in std_logic;
		Q, nQ	: out std_logic
```

```
     );
END d_flip_flop;
ARCHITECTURE Behavioral of d_flip_flop is
     COMPONENT d_latch is
          port(
               Data, Clock     : in std_logic;
               Qu, notQu  : out std_logic
          );
     END COMPONENT;
     signal Qm, Qs : std_logic;
     signal nClk : std_logic;
     attribute keep : boolean;
     attribute keep of Qm, Qs : signal is true;
     begin
     nClk <= not Clk;
     Master: entity work.d_latch
          port map (
               Data => D,
               Clock => nClk,
               Qu => Qm,
               notQu => open
          );
     Slave: entity work.d_latch
          port map (
               Data => Qm,
               Clock => Clk,
               Qu => Qs,
               notQu => open
          );
     Q <= Qs;
     nQ <= not Qs;
END Behavioral;
```

t_flip_flop_clear.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
ENTITY t_flip_flop_clear IS
     port(
          T, Clock, Clr        : in std_logic;
          Qu, notQu            : out std_logic
     );
END t_flip_flop_clear;
ARCHITECTURE Behavioral OF t_flip_flop_clear IS
     signal Data : std_logic;
     signal Qf: std_logic;
     component d_flip_flop is
          port(
               D, Clk     : in std_logic;
               Q, nQ      : out std_logic
```

```
          );
     end component;
     begin
     Data <= (not Clr) and (T xor Qf);
     d1 : entity work.d_flip_flop
          port map(
               D => Data,
               Clk => Clock,
               Q => Qf,
               nQ => notQu
          );
     Qu <= Qf;

END Behavioral;
```

ex1.vhd

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

ENTITY ex1 is
     port(
          Enable, Clk, Clear         : in std_logic;
          b0,b1,b2,b3,b4,b5,b6,b7    : out std_logic;
          num                                : out
std_logic_vector(7 downto 0);
          hex0                               : out
std_logic_vector(6 downto 0);
          hex1                               : out
std_logic_vector(6 downto 0)
     );
END ex1;
ARCHITECTURE main of ex1 is
COMPONENT t_flip_flop_clear IS
     port(
          T, Clock, Clr         : in std_logic;
          Qu, notQu             : out std_logic
     );
END COMPONENT;
COMPONENT hex_display IS
     port(
          num4  : in std_logic_vector(3 downto 0);
          hex   : out std_logic_vector(6 downto 0)
     );
END COMPONENT;
signal E1, E2, E3, E4, E5, E6, E7 : std_logic;
signal Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7 : std_logic;
signal nClear : std_logic;
signal num8 : std_logic_vector(7 downto 0);
begin
```

```
nClear <= not Clear;
E1 <= Enable and Q0;
E2 <= E1 and Q1;
E3 <= E2 and Q2;
E4 <= E3 and Q3;
E5 <= E4 and Q4;
E6 <= E5 and Q5;
E7 <= E6 and Q6;
tff1 : entity work.t_flip_flop_clear
     port map(
          T => Enable,
          Clock => Clk,
          Clr => nClear,
          Qu => Q0,
          notQu => open
     );
tff2 : entity work.t_flip_flop_clear
     port map(
          T => E1,
          Clock => Clk,
          Clr => nClear,
          Qu => Q1,
          notQu => open
     );
tff3 : entity work.t_flip_flop_clear
     port map(
          T => E2,
          Clock => Clk,
          Clr => nClear,
          Qu => Q2,
          notQu => open
     );
tff4 : entity work.t_flip_flop_clear
     port map(
          T => E3,
          Clock => Clk,
          Clr => nClear,
          Qu => Q3,
          notQu => open
     );
tff5 : entity work.t_flip_flop_clear
     port map(
          T => E4,
          Clock => Clk,
          Clr => nClear,
          Qu => Q4,
          notQu => open
     );
```

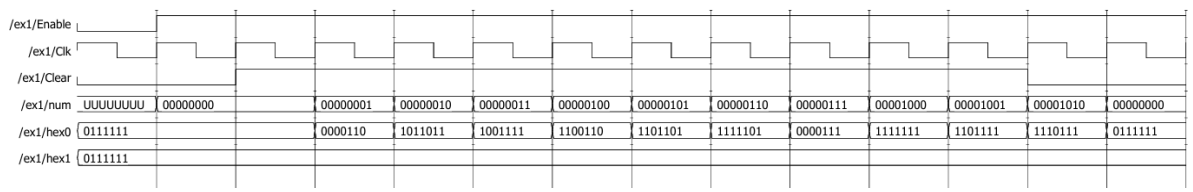
```
tff6 : entity work.t_flip_flop_clear
     port map(
          T => E5,
          Clock => Clk,
          Clr => nClear,
          Qu => Q5,
          notQu => open
     );
tff7 : entity work.t_flip_flop_clear
     port map(
          T => E6,
          Clock => Clk,
          Clr => nClear,
          Qu => Q6,
          notQu => open
     );
tff8 : entity work.t_flip_flop_clear
     port map(
          T => E7,
          Clock => Clk,
          Clr => nClear,
          Qu => Q7,
          notQu => open
     );

b0 <= Q0;
b1 <= Q1;
b2 <= Q2;
b3 <= Q3;
b4 <= Q4;
b5 <= Q5;
b6 <= Q6;
b7 <= Q7;
num8(0) <= Q0;
num8(1) <= Q1;
num8(2) <= Q2;
num8(3) <= Q3;
num8(4) <= Q4;
num8(5) <= Q5;
num8(6) <= Q6;
num8(7) <= Q7;
num <= num8;
hd0 : entity work.hex_display
     port map(
          num4 => num8(3 downto 0),
          hex => hex0
     );
hd1 : entity work.hex_display
```

```
            port map(
                  num4 => num8(7 downto 4),
                  hex => hex1
            );

end main;
```

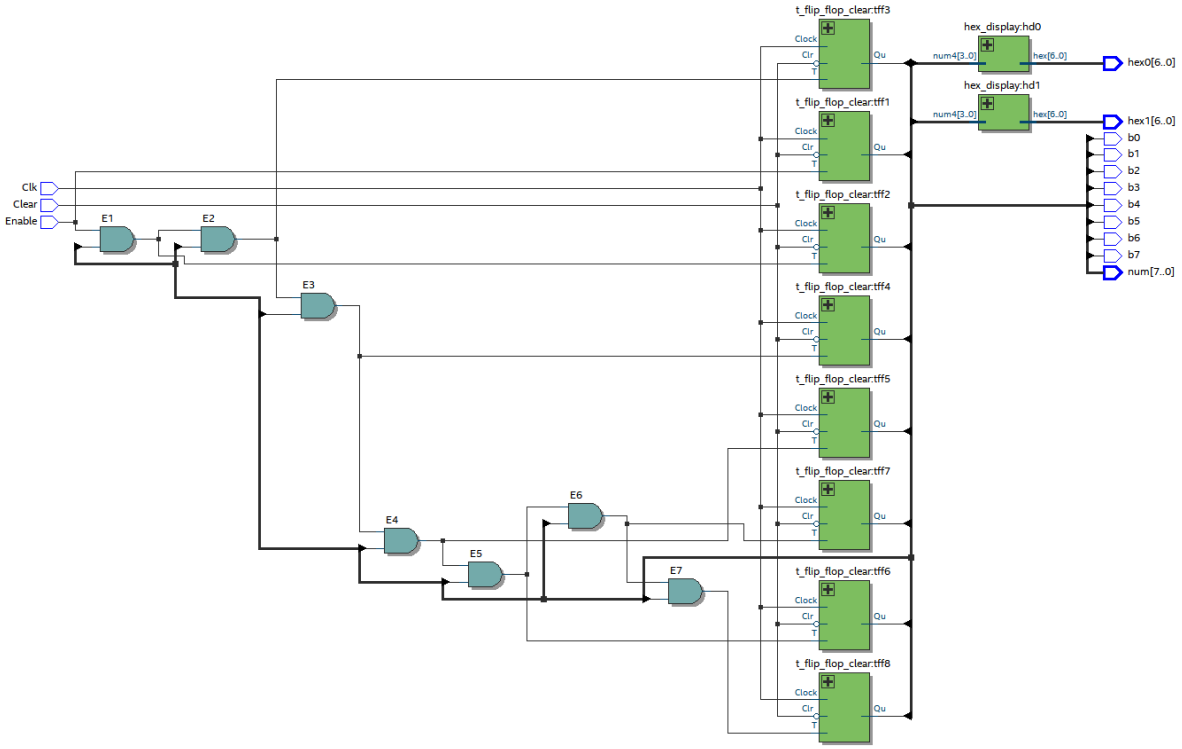
hex_display.vhd

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

ENTITY hex_display IS
      port(
            num4  : in std_logic_vector(3 downto 0);
            hex   : out std_logic_vector(6 downto 0)
      );
END hex_display;
ARCHITECTURE Behavioral OF hex_display IS
begin
with num4 select
      hex <=     "1000000" when "0000", -- 0
                        "1111001" when "0001", -- 1
                        "0100100" when "0010", -- 2
                        "0110000" when "0011", -- 3
                        "0011001" when "0100", -- 4
                        "0010010" when "0101", -- 5
                        "0000010" when "0110", -- 6
                        "1111000" when "0111", -- 7
                        "0000000" when "1000", -- 8
                        "0010000" when "1001", -- 9
                        "0001000" when "1010", -- A
                        "0000011" when "1011", -- B
                        "1000110" when "1100", -- C
                        "0100001" when "1101", -- D
                        "0000110" when "1110", -- E
                        "0001110" when "1111", -- F
                        "1111111" when others;
END Behavioral;
```

Simulação no ModelSim



RTL Viewer



Parte 2

Descrição

Foi desenvolvido um contador de 16 bits em VHDL utilizando flip-flops tipo T. Cada flip-flop foi configurado para mudar seu estado com base no sinal do anterior. Este design empregou aproximadamente 204 elementos lógicos e 47 pinos.

Código VHDL (com flip-flops)

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

ENTITY aulas4e5part2 is
	port(
		Enable, Clk, Clear		: in std_logic;
		num					: out
std_logic_vector(15 downto 0);
		hex0					: out
std_logic_vector(6 downto 0);
		hex1					: out
std_logic_vector(6 downto 0);
		hex2					: out
std_logic_vector(6 downto 0);
		hex3					: out
std_logic_vector(6 downto 0)
	);
END aulas4e5part2;
ARCHITECTURE main of aulas4e5part2 is
```

```
COMPONENT t_flip_flop_clear IS
     port(
          T, Clock, Clr         : in std_logic;
          Qu, notQu             : out std_logic
     );
END COMPONENT;
COMPONENT hex_display IS
     port(
          num4  : in std_logic_vector(3 downto 0);
          hex   : out std_logic_vector(6 downto 0)
     );
END COMPONENT;
signal E : std_logic_vector(15 downto 0);
signal Q : std_logic_vector(15 downto 0);
signal nClear : std_logic;
signal num8 : std_logic_vector(15 downto 0);
begin
     nClear <= not Clear;
     E(1) <= Enable and Q(0);
     E(2) <= E(1) and Q(1);
     E(3) <= E(2) and Q(2);
     E(4) <= E(3) and Q(3);
     E(5) <= E(4) and Q(4);
     E(6) <= E(5) and Q(5);
     E(7) <= E(6) and Q(6);
     E(8) <= E(7) and Q(7);
     E(9) <= E(8) and Q(8);
     E(10) <= E(9) and Q(9);
     E(11) <= E(10) and Q(10);
     E(12) <= E(11) and Q(11);
     E(13) <= E(12) and Q(12);
     E(14) <= E(13) and Q(13);
     E(15) <= E(14) and Q(14);
     tff1 : entity work.t_flip_flop_clear
          port map(
               T => Enable,
               Clock => Clk,
               Clr => nClear,
               Qu => Q(0),
               notQu => open
          );
     tff2 : entity work.t_flip_flop_clear
          port map(
               T => E(1),
               Clock => Clk,
               Clr => nClear,
               Qu => Q(1),
               notQu => open
```

```
      );
tff3 : entity work.t_flip_flop_clear
      port map(
            T => E(2),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(2),
            notQu => open
      );
tff4 : entity work.t_flip_flop_clear
      port map(
            T => E(3),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(3),
            notQu => open
      );
tff5 : entity work.t_flip_flop_clear
      port map(
            T => E(4),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(4),
            notQu => open
      );
tff6 : entity work.t_flip_flop_clear
      port map(
            T => E(5),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(5),
            notQu => open
      );
tff7 : entity work.t_flip_flop_clear
      port map(
            T => E(6),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(6),
            notQu => open
      );
tff8 : entity work.t_flip_flop_clear
      port map(
            T => E(7),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(7),
            notQu => open
```

```
      );
tff9 : entity work.t_flip_flop_clear
      port map(
            T => E(8),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(8),
            notQu => open
      );
tff10 : entity work.t_flip_flop_clear
      port map(
            T => E(9),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(9),
            notQu => open
      );
tff11 : entity work.t_flip_flop_clear
      port map(
            T => E(10),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(10),
            notQu => open
      );
tff12 : entity work.t_flip_flop_clear
      port map(
            T => E(11),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(11),
            notQu => open
      );
tff13 : entity work.t_flip_flop_clear
      port map(
            T => E(12),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(12),
            notQu => open
      );
tff14 : entity work.t_flip_flop_clear
      port map(
            T => E(13),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(13),
            notQu => open
```

```
        );
    tff15 : entity work.t_flip_flop_clear
        port map(
            T => E(14),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(14),
            notQu => open
        );
    tff16 : entity work.t_flip_flop_clear
        port map(
            T => E(15),
            Clock => Clk,
            Clr => nClear,
            Qu => Q(15),
            notQu => open
        );

    num8 <= Q;
    num <= num8;
    hd0 : entity work.hex_display
        port map(
            num4 => num8(3 downto 0),
            hex => hex0
        );
    hd1 : entity work.hex_display
        port map(
            num4 => num8(7 downto 4),
            hex => hex1
        );
    hd2 : entity work.hex_display
        port map(
            num4 => num8(11 downto 8),
            hex => hex2
        );
    hd3 : entity work.hex_display
        port map(
            num4 => num8(15 downto 12),
            hex => hex3
        );

end main;
```

Código VHDL (usando aritmética)

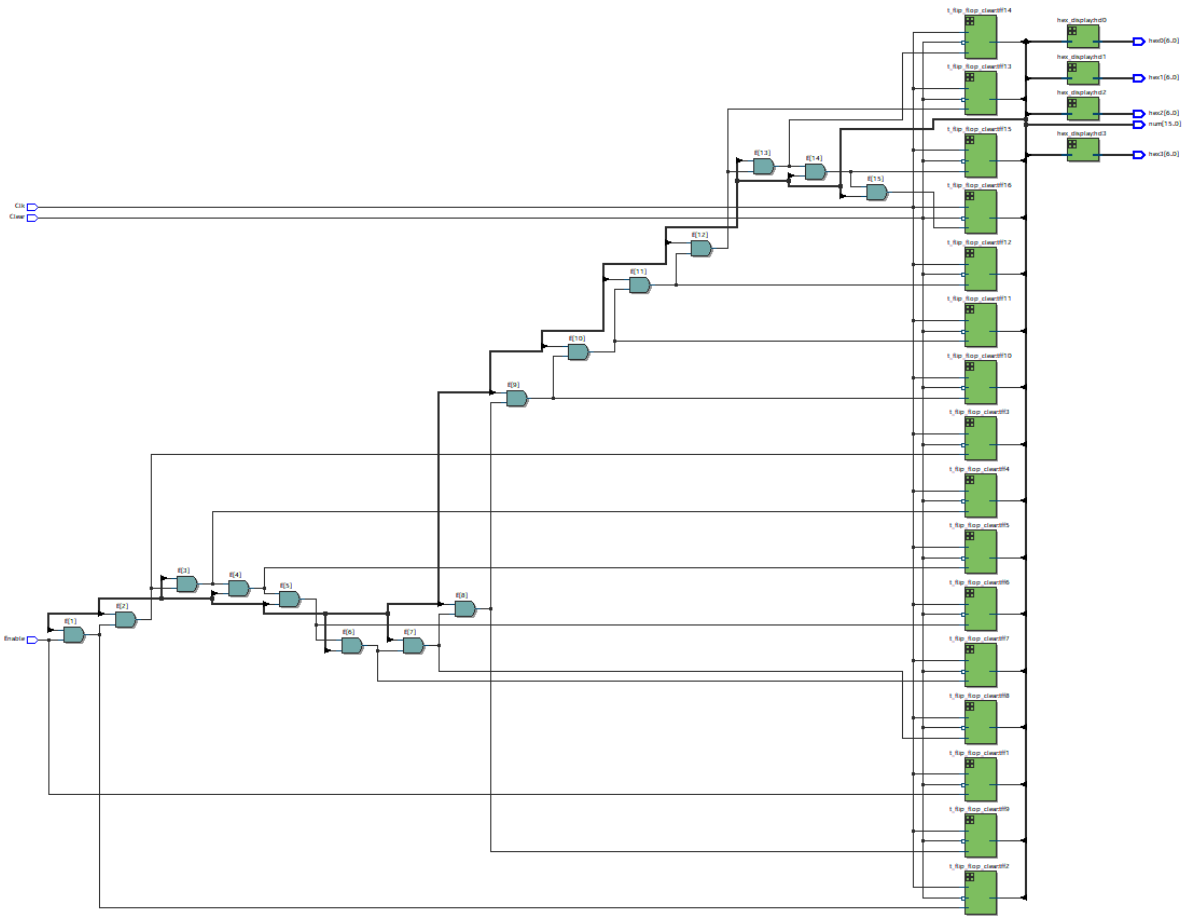
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;
```

```
ENTITY part2 is
	port(
		Enable, Clk, Clear    : in std_logic;
		hex0,hex1,hex2,hex3   : out std_logic_vector(6 downto 0)
	);
END part2;
ARCHITECTURE main of part2 is
COMPONENT hex_display IS
	port(
		num4  : in std_logic_vector(3 downto 0);
		hex   : out std_logic_vector(6 downto 0)
	);
END COMPONENT;
signal Qvec : std_logic_vector(15 downto 0);
signal num0,num1,num2,num3  : std_logic_vector(3 downto 0);
begin
	process (Clk) is
	begin
	if(rising_edge(Clk)) then
		if(Clear = '0') then
			Qvec <= "0000000000000000";
		elsif (Enable = '1') then
			if(unsigned(Qvec) < 65536) then
				Qvec <= std_logic_vector(unsigned(Qvec) + 1);
			else
				Qvec <= "0000000000000000";
			end if;
		end if;
	end if;
	end process;
	num0 <= Qvec(3 downto 0);
	num1 <= Qvec(7 downto 4);
	num2 <= Qvec(11 downto 8);
	num3 <= Qvec(15 downto 12);
	hd0 : entity work.hex_display
		port map(
			num4 => num0,
			hex => hex0
		);
	hd1 : entity work.hex_display
		port map(
			num4 => num1,
			hex => hex1
		);
	hd2 : entity work.hex_display
		port map(
			num4 => num2,
```

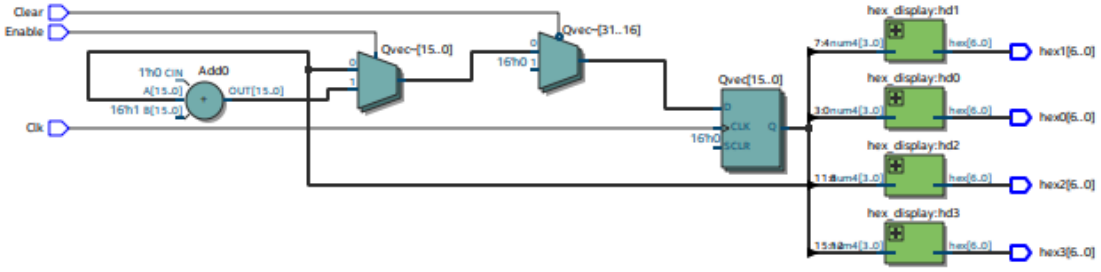
```
                hex => hex2
            );
    hd3 : entity work.hex_display
            port map(
                num4 => num3,
                hex => hex3
            );


end main;
```

RTL Viewer

com flip-flops:



com aritmética:



Parte 3

Descrição

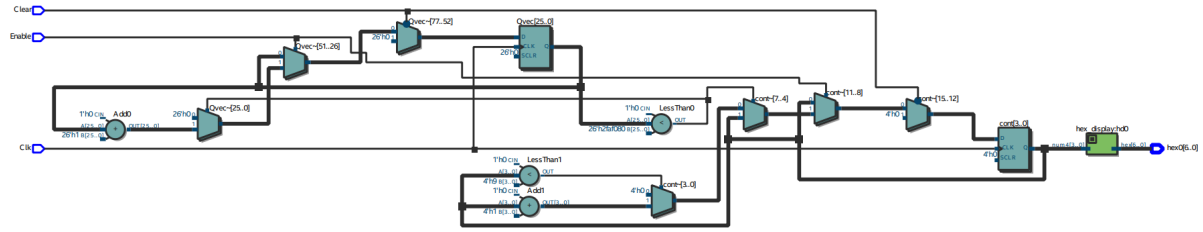
Foi implementado em VHDL um contador decimal com variável de 26 bits, utilizado para gerenciar ciclos de contagem. A lógica assegura a reinicialização após alcançar 9. O código resultante emprega técnicas para controle de tempo, incluindo exibição em um único display hexadecimal.

Código VHDL

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

use IEEE.NUMERIC_STD.ALL;

ENTITY part3comvar is
	port(
		Enable, Clk, Clear	: in std_logic;
		hex0                                : out std_logic_vector(6
downto 0)
	);
END part3comvar;
ARCHITECTURE main of part3comvar is
COMPONENT hex_display IS
	port(
		num4  : in std_logic_vector(3 downto 0);
		hex   : out std_logic_vector(6 downto 0)
	);
END COMPONENT;
signal Qvec : std_logic_vector(25 downto 0);
signal cont : std_logic_vector(3 downto 0);
signal num  : std_logic_vector(3 downto 0);
begin
	process (Clk) is
	begin
	if(rising_edge(Clk)) then
		if(Clear = '0') then
			Qvec <= "00000000000000000000000000";
```

```
                cont <= "0000";
            elsif (Enable = '1') then
                if(unsigned(Qvec) < 50000000) then
                    Qvec <= std_logic_vector(unsigned(Qvec) + 1);
                else
                    Qvec <= "00000000000000000000000000";
                    if(unsigned(cont) < 9) then
                        cont <= std_logic_vector(unsigned(cont)
+ 1);
                    else
                        cont <= "0000";
                    end if;
                end if;
            end if;
        end if;
        end process;
        num <= cont;
        hd0 : entity work.hex_display
            port map(
                num4 => num,
                hex => hex0
            );


end main;
```

RTL Viewer



Parte IV

Descrição

Foi desenvolvido em VHDL um contador com lógica para exibição de uma sequência rotativa de palavras em displays hexadecimais. Utilizando um vetor de 26 bits para controle de tempo e um contador de 2 bits, o design permite a rotação cíclica de quatro estados. A implementação utiliza aproximadamente 4 displays hexadecimais para exibição sequencial.

Código VHDL

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

use IEEE.NUMERIC_STD.ALL;

ENTITY part4 is
     port(
          Enable, Clk, Clear    : in std_logic;
          hex0,hex1,hex2,hex3   : out std_logic_vector(6 downto 0)
     );
END part4;
ARCHITECTURE main of part4 is
signal Tempo : std_logic_vector(25 downto 0);
signal cont : std_logic_vector(1 downto 0);

type ARRAY_PALAVRA is array (0 to 3) of std_logic_vector(6 downto
0);
constant word : ARRAY_PALAVRA := (0 => "1111111", 1 => "0100001",
2 => "0000110", 3 => "1000000");

begin
     process (Clk) is
     begin
     if(rising_edge(Clk)) then
          if(Clear = '0') then
               Tempo <= "00000000000000000000000000";
               cont <= "00";
          elsif (Enable = '1') then
               if(unsigned(Tempo) < 50000000) then
                    Tempo <= std_logic_vector(unsigned(Tempo) +
1);
               else
                    Tempo <= "00000000000000000000000000";
                    if(unsigned(cont) < 3) then
                         cont <= std_logic_vector(unsigned(cont)
+ 1);
                    else
                         cont <= "00";
                    end if;
               end if;
          end if;
     end if;
     end process;
     hex3 <= word(to_integer(unsigned(cont)));
     hex2 <= word(to_integer(unsigned(cont) + 1) mod 4);
     hex1 <= word(to_integer(unsigned(cont) + 2) mod 4);
```

```
      hex0 <= word(to_integer(unsigned(cont) + 3) mod 4);


end main;
```

Part V

Descrição

Foi implementado um contador rotativo mais avançado em VHDL, utilizando lógica semelhante à parte 4, mas com 6 estados diferentes exibidos em 6 displays hexadecimais. O design inclui um vetor de 26 bits para controle de tempo e um contador de 3 bits, permitindo uma rotação contínua entre os estados definidos.

Código VHDL

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

use IEEE.NUMERIC_STD.ALL;

ENTITY part5 is
      port(
            Enable, Clk, Clear         : in std_logic;
            hex0,hex1,hex2,hex3,hex4,hex5   : out std_logic_vector(6
downto 0)
      );
END part5;
ARCHITECTURE main of part5 is
signal Tempo : std_logic_vector(25 downto 0);
signal cont : std_logic_vector(2 downto 0);

type ARRAY_PALAVRA is array (0 to 5) of std_logic_vector(6 downto
0);
constant word : ARRAY_PALAVRA := (0 => "1111111", 1 => "1111111",
2 => "1111111", 3 => "0100001", 4 => "0000110", 5 => "1000000");

begin
      process (Clk) is
      begin
      if(rising_edge(Clk)) then
            if(Clear = '0') then
                  Tempo <= "00000000000000000000000000";
                  cont <= "000";
```

```
           elsif (Enable = '1') then
                if(unsigned(Tempo) < 50000000) then
                     Tempo <= std_logic_vector(unsigned(Tempo) +
1);
                else
                     Tempo <= "00000000000000000000000000";
                     if(unsigned(cont) < 5) then
                          cont <= std_logic_vector(unsigned(cont)
+ 1);
                     else
                          cont <= "000";
                     end if;
                end if;
           end if;
      end if;
      end process;
      hex5 <= word(to_integer(unsigned(cont)));
      hex4 <= word(to_integer(unsigned(cont) + 1) mod 6);
      hex3 <= word(to_integer(unsigned(cont) + 2) mod 6);
      hex2 <= word(to_integer(unsigned(cont) + 3) mod 6);
      hex1 <= word(to_integer(unsigned(cont) + 4) mod 6);
      hex0 <= word(to_integer(unsigned(cont) + 5) mod 6);


end main;
```