FPGA & VHDL

Tutorial – Aula 2

Computação Digital

VHDL: CKTs sequenciais

Flip-Flop D

```
library ieee;
use ieee.std_logic_1164.all;

entity d_ff_reset is
  port(
    clk, reset: in std_logic;
    d: in std_logic;
    q: out std_logic
  );
end d_ff_reset;

architecture arch of d_ff_reset is
begin
  process(clk,reset)
  begin
    if (reset='1') then
      q <= '0';
    elsif (clk'event and clk='1') then
      q <= d;
    end if;
  end process;
end arch;
```

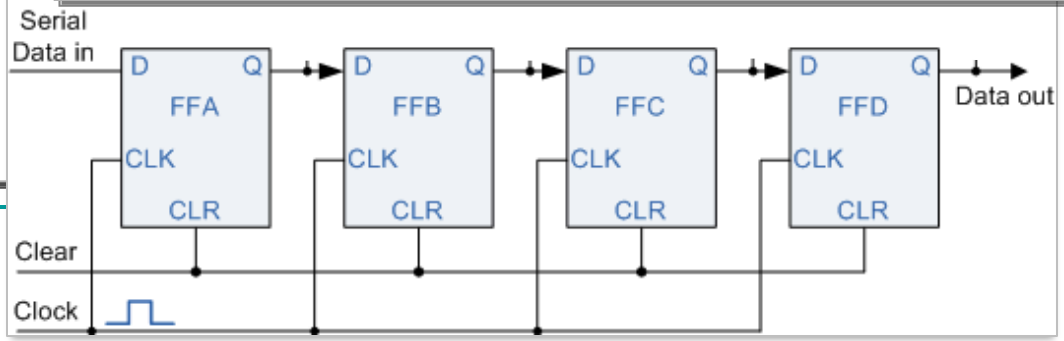
VHDL: CKTs sequenciais

Registrador de deslocamento

```
library ieee;
use ieee.std_logic_1164.all;

entity shift_reg is
  generic(N: integer := 8);
  port(
    clk, reset: in std_logic;
    s_in: in std_logic;
    s_out: out std_logic
    );
end shift_reg;

architecture arch of shift_reg is
  signal r_reg: std_logic_vector(N-1 downto 0);
  signal r_next: std_logic_vector(N-1 downto 0);
begin
  process(clk,reset)
  begin
    if (reset='1') then
      r_reg <= (others=>'0');
    elsif (clk'event and clk='1') then
      r_reg <= r_next;
    end if;
  end process;
  r_next <= s_in & r_reg(N-1 downto 1);
  s_out <= r_reg(0);
end arch;
```

Testbench

```
library ieee;
use ieee.std_logic_1164.all;

entity shift_reg_tb is
end shift_reg_tb;

architecture arch of shift_reg_tb is
  constant N: integer := 4; -- no. of bits
  constant T: time := 20 ns;
  signal clk, reset, s_in, s_out: std_logic;
begin
  shift_reg_unit: entity work.shift_reg(arch)
    generic map(N => N)
    port map(clk => clk, reset => reset, s_in => s_in, s_out => s_out);
    process
    begin
      clk <= '0';
      wait for T/2;
      clk <= '1';
      wait for T/2;
    end process;
    s_in <= '1',
      '0' after T,
      '1' after 2*T,
      '0' after 3*T;
      reset <= '1', '0' after T/4;
end arch;
```



VHDL: CKTs sequenciais

Contador binário

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter is
  generic(N: integer := 8);
  port(
    clk, reset: in std_logic;
    max_tick: out std_logic;
    q: out std_logic_vector(N-1 downto 0)
  );
end counter;

architecture arch of counter is
  signal r_reg: unsigned(N-1 downto 0);
  signal r_next: unsigned(N-1 downto 0);
begin
  process(clk,reset)
  begin
    if (reset='1') then
      r_reg <= (others=>'0');
    elsif (clk'event and clk='1') then
      r_reg <= r_next;
    end if;
  end process;
  r_next <= r_reg + 1;
  q <= std_logic_vector(r_reg);
  max_tick <= '1' when r_reg=(2**N-1) else '0';
end arch;
```

Testbench

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_tb is
end counter_tb;

architecture arch of counter_tb is
  constant N: integer := 4;
  constant T: time := 20 ns;
  signal clk, reset, max_tick: std_logic;
  signal q: std_logic_vector(N-1 downto 0);
begin
  counter_unit: entity work.counter(arch)
    generic map(N => N)
    port map(clk => clk, reset => reset, max_tick => max_tick, q => q);
    process
    begin
      clk <= '0';
      wait for T/2;
      clk <= '1';
      wait for T/2;
    end process;
    reset <= '1', '0' after T/4;
end arch;
```

VHDL: CKTs sequenciais

Contador binário *up/down*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_up_down is
  generic(N: integer := 8);
  port(
    clk, reset, up: in std_logic;
    max_tick: out std_logic;
    q: out std_logic_vector(N-1 downto 0)
  );
end counter_up_down;

architecture arch of counter_up_down is
  signal r_reg: unsigned(N-1 downto 0);
begin
  process(clk,reset)
  begin
    if (reset='1') then
      r_reg <= (others=>'0');
    elsif (clk'event and clk='1') then
      if (up='1') then
        r_reg <= r_reg + 1;
      else
        r_reg <= r_reg -1;
      end if;
    end if;
  end process;
  q <= std_logic_vector(r_reg);
  max_tick <= '1' when r_reg=(2**N-1) else '0';
end arch;
```

Simplificação: apenas 1 *signal*

Testbench

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_up_down_tb is
end counter_up_down_tb;

architecture arch of counter_up_down_tb is
  constant N: integer := 4;
  constant T: time := 20 ns;
  signal clk, reset, up, max_tick: std_logic;
  signal q: std_logic_vector(N-1 downto 0);
begin
  counter_unit: entity work.counter_up_down(arch)
    generic map(N => N)
    port map(clk => clk, reset => reset, up => up,
             max_tick => max_tick, q => q);
    process
    begin
      clk <= '0';
      wait for T/2;
      clk <= '1';
      wait for T/2;
    end process;
    reset <= '1', '0' after T/4;
    up <= '1', '0' after T*(2**N);
end arch;
```

VHDL: CKTs sequenciais

Contador módulo-m

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mod_m_counter is
  generic(
    N: integer := 4; -- no. of bits
    M: integer := 10 -- mod-M
  );
  port(
    clk, reset: in std_logic;
    max_tick: out std_logic;
    q: out std_logic_vector(N-1 downto 0)
  );
end mod_m_counter;

architecture arch of mod_m_counter is
  signal r_reg: unsigned(N-1 downto 0);
  signal r_next: unsigned(N-1 downto 0);
begin
  process(clk,reset)
  begin
    if (reset='1') then
      r_reg <= (others=>'0');
    elsif (clk'event and clk='1') then
      r_reg <= r_next;
    end if;
  end process;
  r_next <= (others=>'0') when r_reg=M-1 else r_reg + 1;
  q <= std_logic_vector(r_reg);
  max_tick <= '1' when r_reg=M-1 else '0';
end arch;
```

Testbench

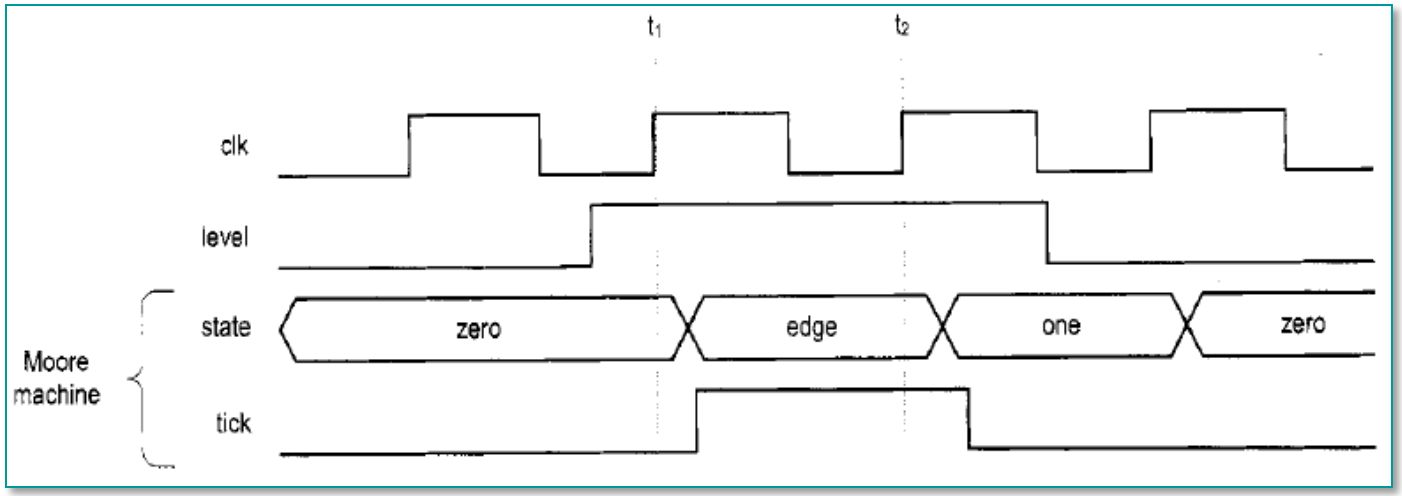
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mod_m_counter_tb is
end mod_m_counter_tb;

architecture arch of mod_m_counter_tb is
  constant N: integer := 6;
  constant M: integer := 4;
  constant T: time := 20 ns;
  signal clk, reset, max_tick: std_logic;
  signal q: std_logic_vector(N-1 downto 0);
begin
  mod_m_counter_unit: entity work.mod_m_counter(arch)
    generic map(N => N, M => M)
    port map(clk => clk, reset => reset, max_tick => max_tick, q => q);
    process
    begin
      clk <= '0';
      wait for T/2;
      clk <= '1';
      wait for T/2;
    end process;
    reset <= '1', '0' after T/2;
end arch;
```

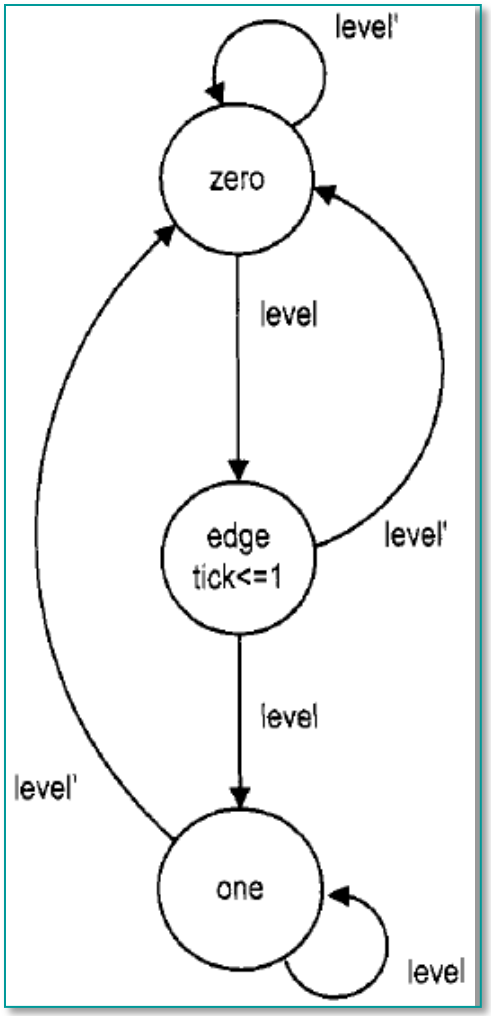
Clock da placa:
$F = 50$ MHz; $T = 20 \times 10^{-9}$ s

Para $T = 1{,}0$ s:
$M = 50.000.000$
$N = 26$, pois $2^{26} = 67.108.864$

Máquinas de Estados Finitos (FSM)

- Detector de borda ascendente

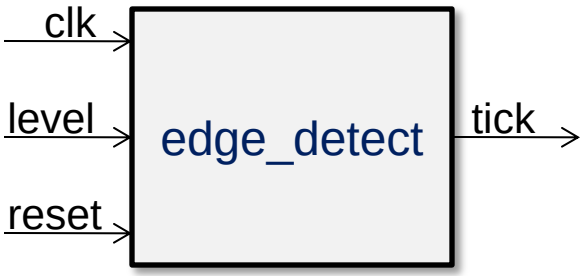


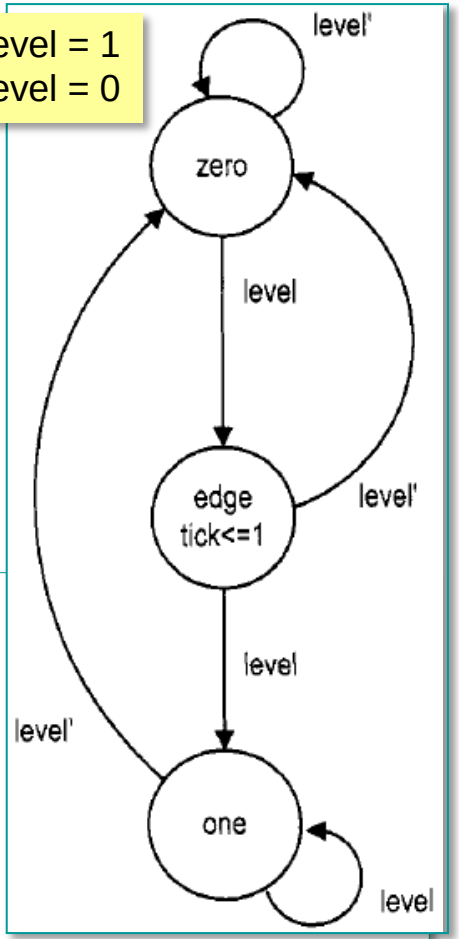


Detector de borda

```
library ieee;
use ieee.std_logic_1164.all;
entity edge_detect is
  port(
    clk, reset: in std_logic;
    level: in std_logic;
    tick: out std_logic
    );
end edge_detect;

architecture moore_arch of edge_detect is
  type state_type is (zero, edge, one);
  signal state_reg, state_next: state_type;
begin
  -- state register
  process(clk,reset)
  begin
    if (reset='1') then
      state_reg <= zero;
    elsif (clk'event and clk='1') then
      state_reg <= state_next;
    end if;
  end process;
  -- next-state / output logic
  process(state_reg,level)
  begin
    state_next <= state_reg;
    tick <= '0';
    case state_reg is
      when zero =>
        if level= '1' then
          state_next <= edge;
        end if;
      when edge =>
        tick <= '1';
        if level= '1' then
          state_next <= one;
        else
          state_next <= zero;
        end if;
      when one =>
        if level= '0' then
          state_next <= zero;
        end if;
    end case;
  end process;
end moore_arch;
```

`level` : level = 1
`level'`: level = 0





Testbench

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity edge_moore_tb is
end edge_moore_tb;

architecture arch of edge_moore_tb is
  constant T: time := 20 ns;
  signal clk, reset, level, tick: std_logic;
begin
  edge_unit: entity work.edge_detect(moore_arch)
    port map(clk => clk, reset => reset, level => level, tick => tick);
    process
    begin
      clk <= '0';
      wait for T/2;
      clk <= '1';
      wait for T/2;
    end process;
    reset <= '1', '0' after T/4;
    level <= '0',
      '1' after 2.25*T,
      '0' after 5.25*T;
end arch;
```

Circuito de "Debounce"

- Necessário para chaves de *clock* manual
  - Código fonte fornecido