VHDL

Circuitos Síncronos

4 de outubro de 2015

Rafael Corsi Ferrão - IMT

rafael.corsi@maua.br
http://www.maua.br

CONTEÚDO

CONTEÚDO

CONTEÚDO

Definição

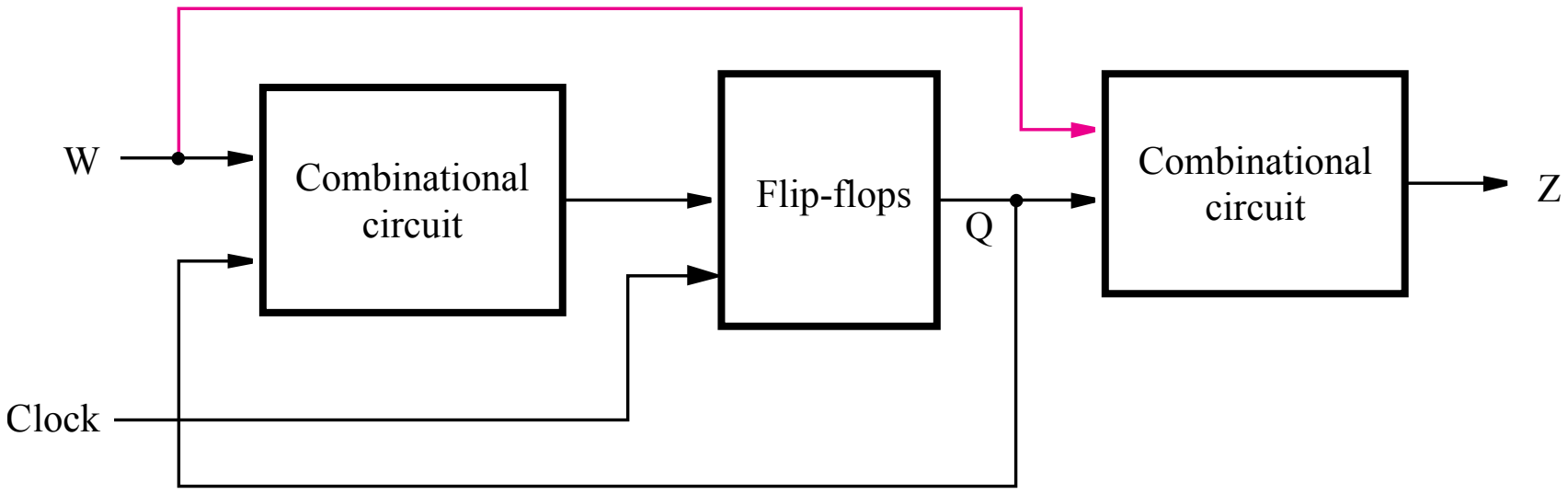
Circuitos síncronos são aqueles que utilizam um sinal de sincronismos para trigar sua execução, esse sinais são conhecidos como *clock*

Em quase toda sua totalidade, projetos digitais são do tipo síncronos. Alguns exemplos:

- Relógio
- filtros digitais
- televisão
- computador
- microprocessador
- ...

- Circuitos síncronos possuem necessariamente em sua formação elementos de memória (flip-flop)
- o *clock* é o sinal responsável pela sincronização do circuito
- as mudanças dos estados (variáveis e saídas) ocorrem na transição do *clock* - borda de subida **ou** borda de descida.

- Para a sintetização de circuitos síncronos, a ferramenta irá utilizar de elementos de memória como Flip-Flop, memórias RAM e ROM.
- Porém temos que mostrar para o sintetizador que a região descrita deve ser entendida como síncrona.

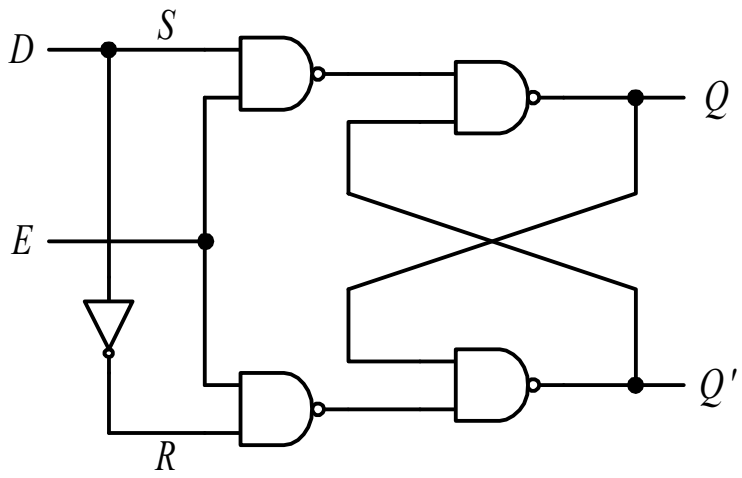


LATCH VS FLIP-FLOP

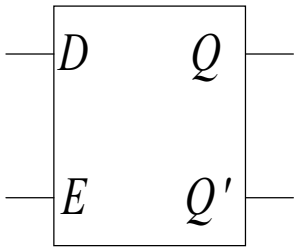
Principais diferenças entre Flip-Flop e Latchs são:

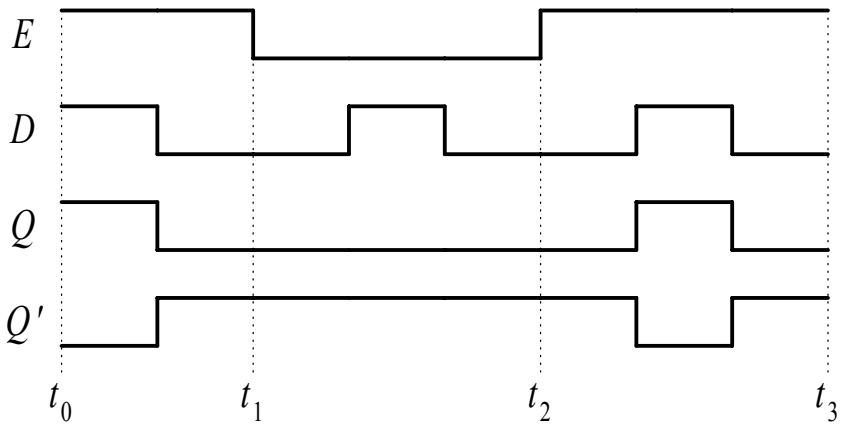
- Flip-flops são sensíveis a transições, latchs são sensíveis a níveis
- Use preferencialmente em projetos com FPGA os flip-flops;
- Latchs apresentam problema com temporização (veremos mais tarde)

LATCH

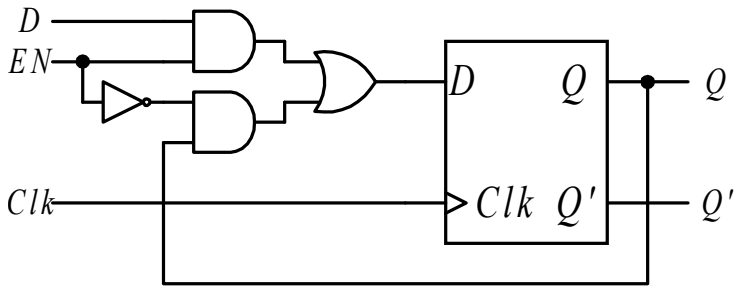


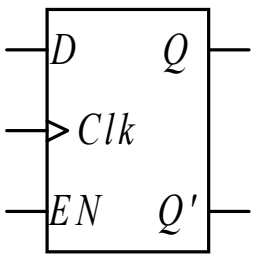
E	D	Q	Q_{next}	Q_{next}'
0		0	0	1
0		1	1	0
1	0		0	1
1	1		1	0





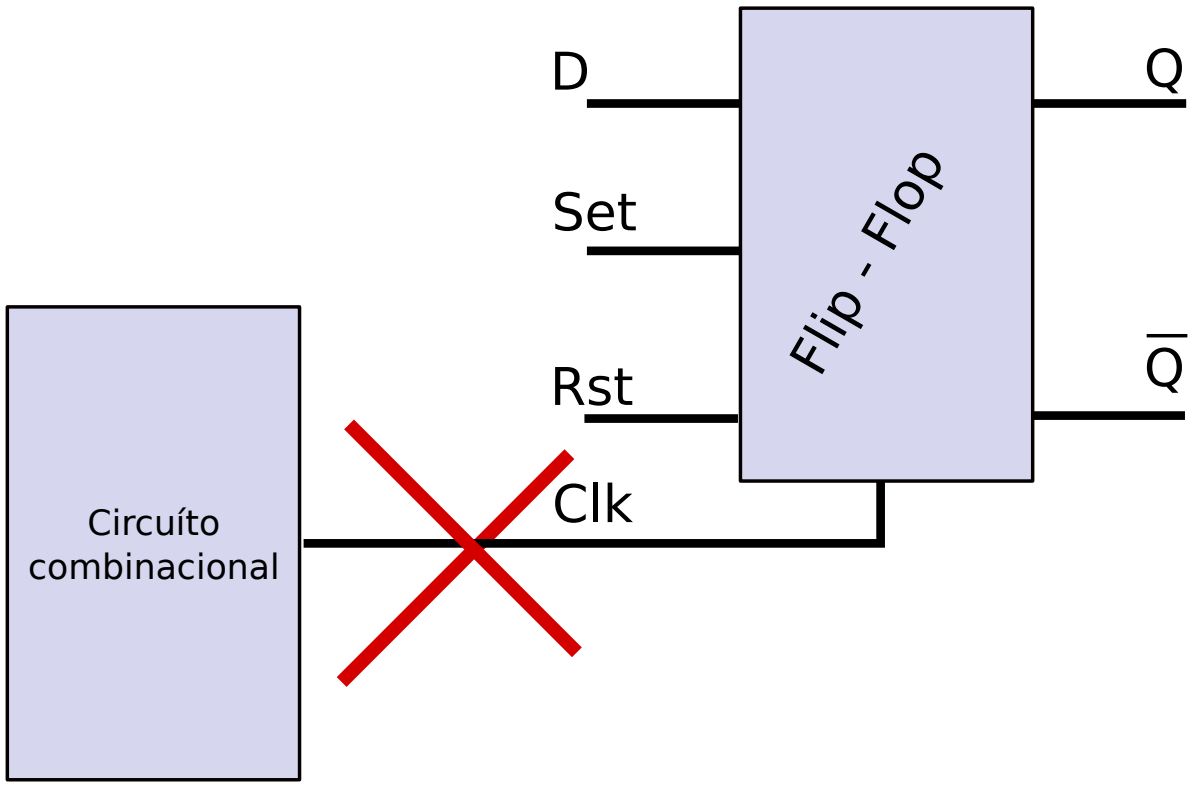
FLIP-FLOP



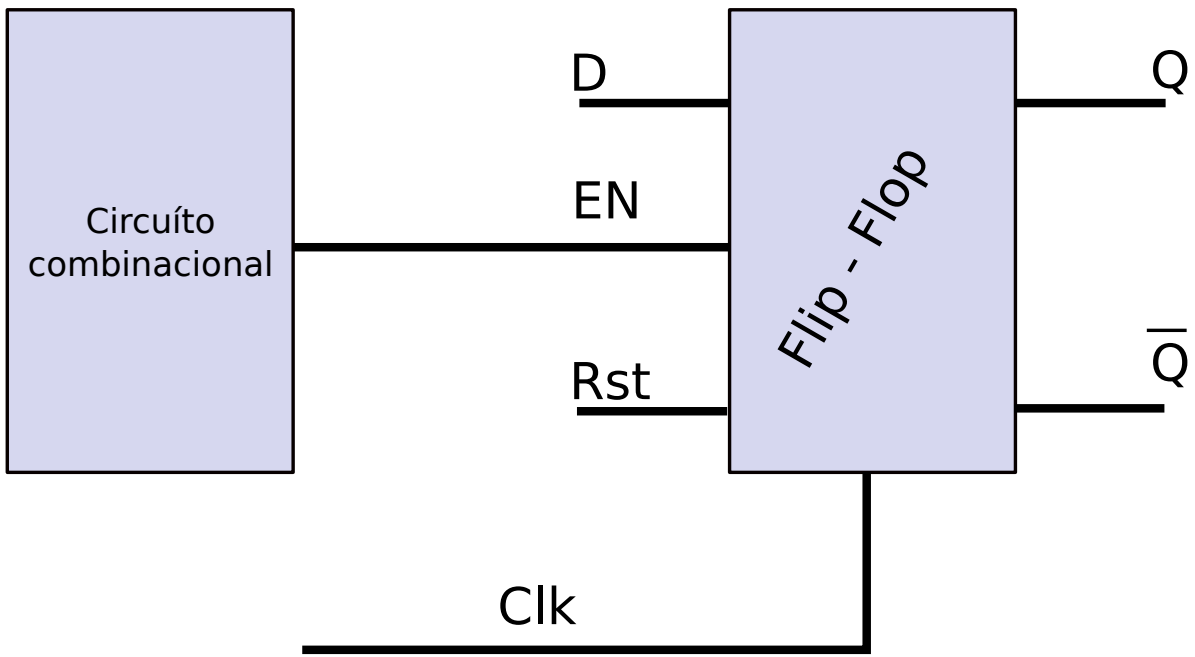


Clk	EN	D	Q	Q_{next}	Q_{next}'
0			0	0	1
0			1	1	0
1			0	0	1
1			1	1	0
↑	0		0	0	1
↑	0		1	1	0
↑	1	0		0	1
↑	1	1		1	0

Nunca deve-se usar uma lógica para gerar o sinal para trigar um clock:



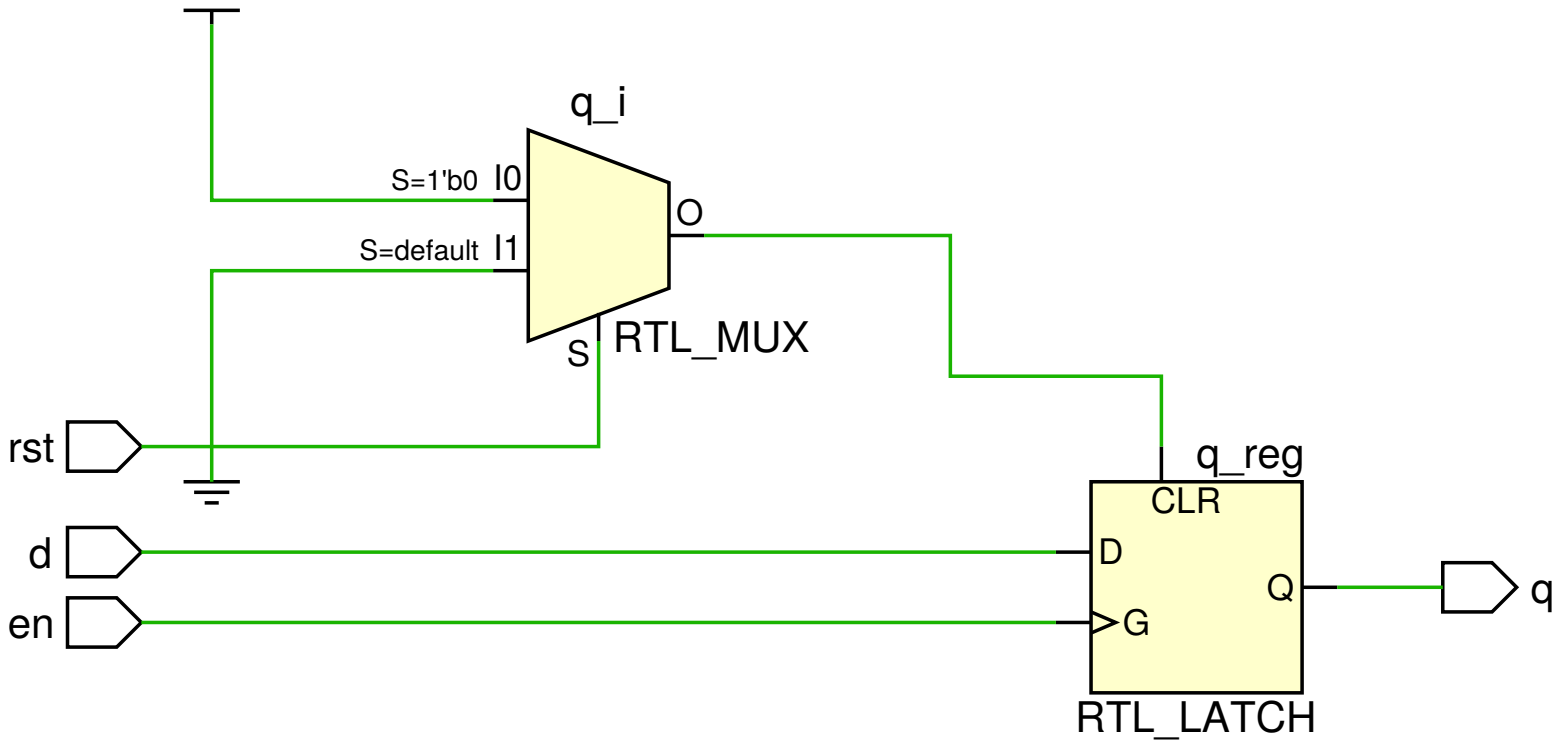
Porém, deve-se utilizar o circuíto combinacional como enable ou rst para o flip-flop

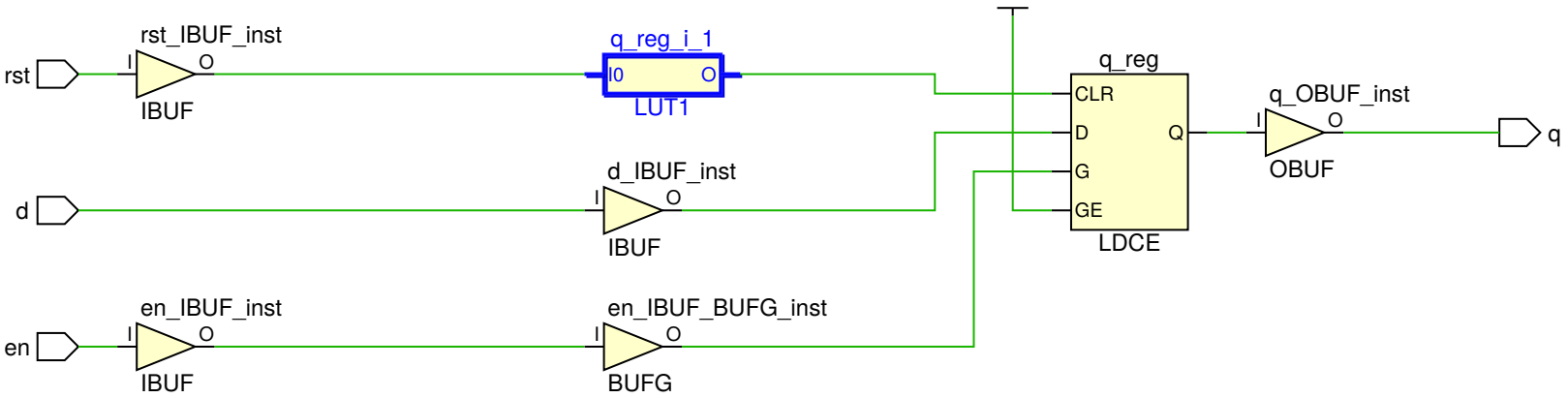


CONTEÚDO

LATCH

```
ENTITY latch IS
   PORT(
      rst : IN  STD_LOGIC; -- entradas
      set  : IN  STD_LOGIC;
      d   : IN  STD_LOGIC;
      q   : OUT STD_LOGIC
   );
END latch;

ARCHITECTURE bhv OF latch IS
BEGIN

   PROCESS(rst, en, d)
   BEGIN
       IF rst = '0' THEN
        q <= '0';
       ELSIF set = '1' THEN
        q <= d;
       END IF;
   END PROCESS;
END bhv;
```


q_i
S=1'b0
I0
S=default
I1
O
S
RTL_MUX
rst
d
en
q_reg
CLR
D
G
Q
q
RTL_LATCH



LDCE : Transparent Data Latch with Asynchronous Clear and Gate Enable

CONTEÚDO

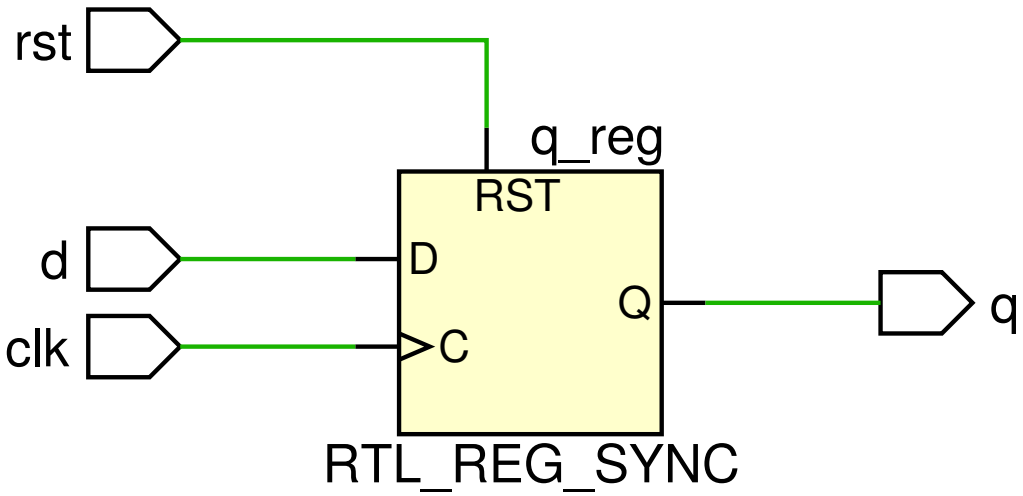
Circuitos síncronos em VHDL são implementados em regiões de códigos sequenciais (**process**).
Uma transição em VHDL pode ser detectada de duas maneiras diferentes :

```
(ck'EVENT and CK = '1') -- Borda de subida
(ck'EVENT and CK = '0') -- Borda de descida
(rising_edge(ck))       -- Borda de subida
(falling_edge(ck))      -- Borda de descida
```

Onde **ck** é o um sinal/entrada que propaga o **clock** .

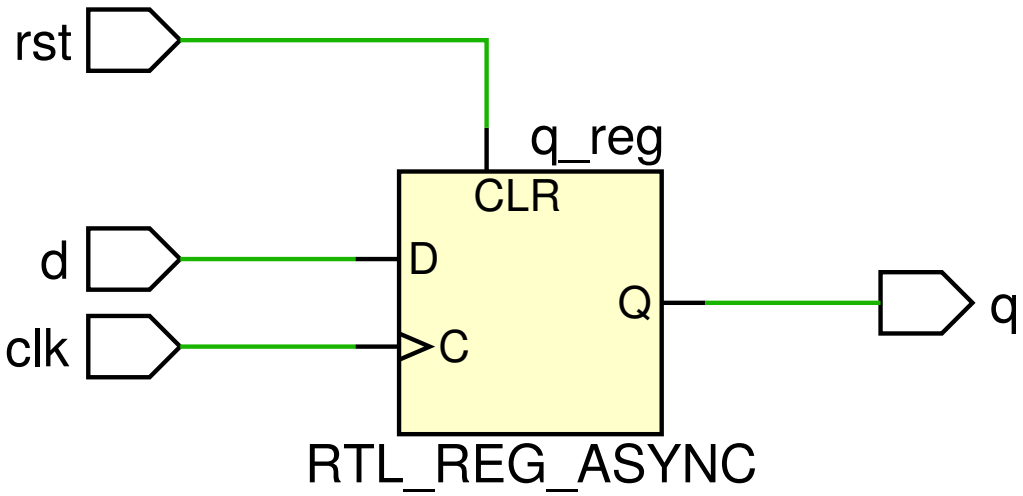
```
entity sincrono is
    Port ( clk : in  std_logic;
           rst : in  std_logic;
           d   : in  std_logic;
           q   : out std_logic
         );
end sincrono;

architecture Behavioral of sincrono is

begin
    process(clk)
    begin
        if(rising_edge(clk)) then
            if(rst = '1') then
                q <= '0';
            else
                q <= d;
            end if;
        end if;
    end process;
end Behavioral;
```

REGISTRADOR SENSÍVEL A BORDA DE SUBIDA - RST SÍNCRONO



```
entity sincrono is
    Port ( clk : in  std_logic;
           rst : in  std_logic;
           d   : in  std_logic;
           q   : out std_logic
          );
end sincrono;

architecture Behavioral of sincrono is

begin
    process(clk, rst)
    begin
            if(rst = '1') then
                q <= '0';
     elsif(clk'EDGE = '1' and clk = '1') then
         q <= d;
     end if;
   end process;
end Behavioral;
```

REGISTRADOR SENSÍVEL A BORDA DE SUBIDA - RST ASSÍNCRONO



```
entity sincrono is
    Port ( clk : in  std_logic;
           rst : in  std_logic;
           set : in  std_logic;
           d   : in  std_logic;
           q   : out std_logic
         );
end sincrono;

architecture Behavioral of sincrono is

begin
    process(clk, rst)
    begin
            if(rst = '1') then
                q <= '0';
            elsif(set = '1') then
                q <= '1';
     elsif(clk'EDGE = '1' and clk = '1') then
         q <= d;
     end if;
   end process;
end Behavioral;
```

CONTEÚDO

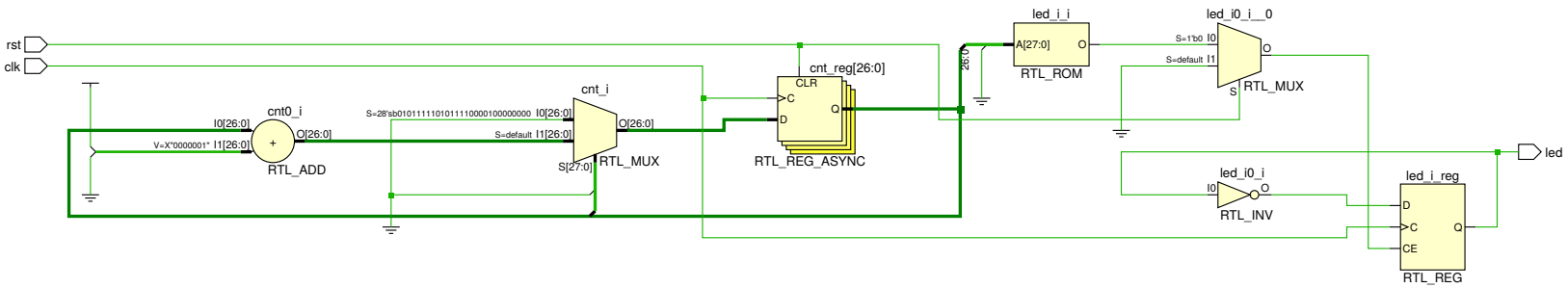
CONTADOR DE 0 A 100

```
entity contador is
    Port ( clk : in  std_logic;
           rst : in  std_logic;
           d   : in  std_logic;
           q   : out std_logic
         );
end contador;

architecture Behavioral of contador is

    signal cnt : integer range 0 to 100;

begin
    process(clk, rst)
    begin
        if(rst = '1') then
            cnt <= 0;
        elsif(rising_edge(clk)) then
            cnt <= cnt + 1;
        end if;
    end process;
end Behavioral;
```

PISCA LED A CADA 1 SEGUNDO COM CLK DE 100MHZ

```
entity contador_led is
    Port ( clk : in  std_logic; -- 100Mhz
           rst : in  std_logic;
           led : out  std_logic
          );
end contador_led;

architecture Behavioral of contador_led is

    signal cnt   : integer range 0 to 100_000_000 := 0;
    signal led_i : std_logic := '0';

begin
    process(clk, rst)
    begin
        if(rst = '1') then
            cnt <= 0;
        elsif(rising_edge(clk)) then
            if (cnt = 100_000_000) then
                cnt   <= 0;
                led_i <= not led_i;
            else
                cnt <= cnt + 1;
            end if;
        end if;
    end process;

   led <= led_i;

end Behavioral;
```

RTL


rst
clk
cnt0_i
I0[26:0]
V=X"0000001"
I1[26:0]
O[26:0]
RTL_ADD
cnt_i
S=28'sb01011111010111100001000000000
I0[26:0]
S=default
I1[26:0]
O[26:0]
S[27:0]
RTL_MUX
cnt_reg[26:0]
CLR
C
D
Q
RTL_REG_ASYNC
26:0
led_i_i
A[27:0]
O
RTL_ROM
led_i0_i__0
S=1'b0
I0
S=default
I1
O
S
RTL_MUX
led_i0_i
I0
O
RTL_INV
led_i_reg
D
C
CE
Q
RTL_REG
led

ESQUEMÁTICO SÍNTESE