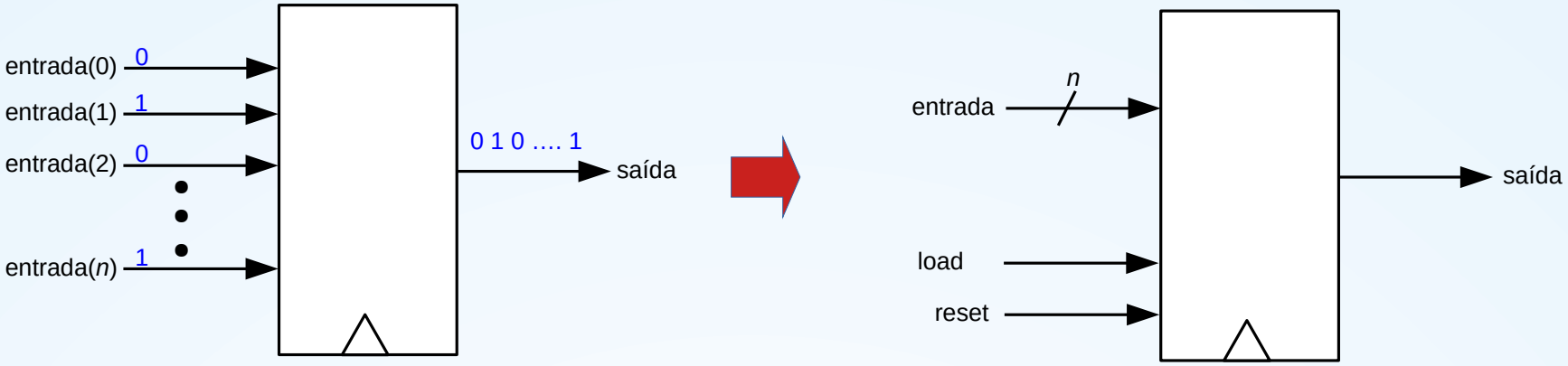
Aula 14 – Circuitos especiais

Tópicos da aula

- **Conversor Paralelo para Serial**
- **Conversor Serial para Paralelo**
- **Detector de Borda de sinais**
- **Lógica de rejeição (*Debouce Logic*)**

Conversor Paralelo → Serial

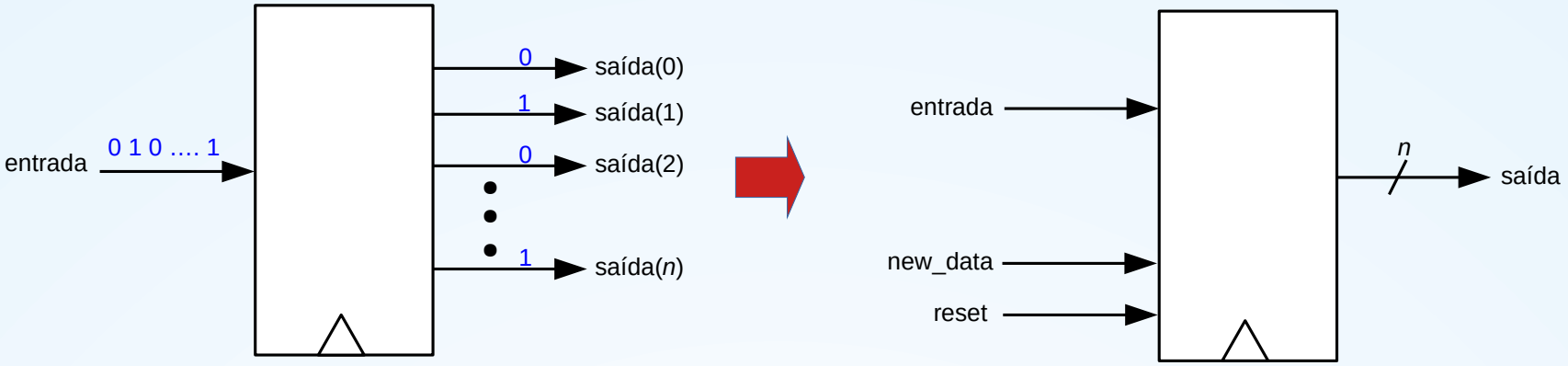


- Possíveis usos: seriais síncronas (SPI, I2C, etc) ou assíncronas (UART, por exemplo)

Conversor Paralelo → Serial

```
24  entity PAR2SER is port
25     (
26        i_RST        : in std_logic;
27        i_CLK        : in std_logic;
28        --
29        i_LOAD       : in std_logic;
30        i_ND         : in std_logic;
31        i_DATA       : in std_logic_vector(7 downto 0);
32        o_TX         : out std_logic
33     );
34  end PAR2SER;
35
36  architecture Behavioral of PAR2SER is
37  ----------------------------------------------------------------------------------------------------
38  -- Internal signals.
39  ----------------------------------------------------------------------------------------------------
40      signal w_DATA : std_logic_vector (i_DATA'range);
41      signal w_ND   : std_logic;
42
43  ----------------------------------------------------------------------------------------------------
44  begin
45  ----------------------------------------------------------------------------------------------------
46  -- Serializer (MSB first).
47  ----------------------------------------------------------------------------------------------------
48     U1 : process (i_CLK, i_RST)
49     begin
50        if(i_RST = '1') then
51           w_ND <= '0';
52
53        elsif falling_edge(i_CLK) then
54           if(i_ND = '1') then
55              o_TX <= w_DATA(7);
56              w_ND <= '1';
57           else
58              w_ND <= '0';
59           end if;
60        end if;
61
62     end process U1;
63
64     --
65     -- Loading (or Shifting) the data into the serializer.
66     --
67     U2 : process (i_CLK)
68     begin
69
70        if rising_edge(i_CLK) then
71           if(i_LOAD = '1') then
72               w_DATA <= i_DATA;
73
74           elsif(w_ND = '1') then
75               w_DATA <= w_DATA(6 downto 0) & '0';
76           end if;
77        end if;
78
79     end process U2;
80
81  ----------------------------------------------------------------------------------------------------
82  end Behavioral;
```

Conversor Serial → Paralelo



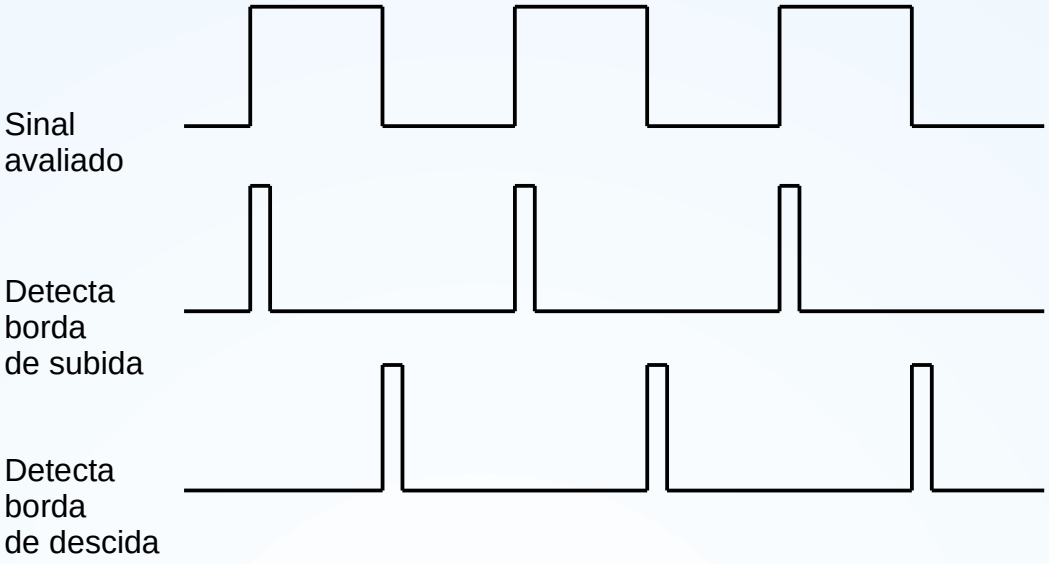
- Caminho inverso em comunicações seriais síncronas ou assíncronas

Conversor Serial → Paralelo

```
23  entity SER2PAR is port
24     (
25        i_RST       : in std_logic;
26        i_CLK       : in std_logic;
27        --
28        i_ND        : in std_logic;
29        o_DATA      : out std_logic_vector(7 downto 0);
30        i_RX        : in std_logic
31     );
32  end SER2PAR;
33
34  architecture Behavioral of SER2PAR is
35  ----------------------------------------------------------------------------------------------
36  -- Internal signals.
37  ----------------------------------------------------------------------------------------------
38      signal w_DATA : std_logic_vector (o_DATA'range);
39
40
41  ----------------------------------------------------------------------------------------------
42  begin
43  ----------------------------------------------------------------------------------------------
44
45     U1 : process (i_RST, i_CLK)
46     begin
47        if (i_RST = '1') then
48           w_DATA <= (others => '1');
49
50        else
51           if rising_edge (i_CLK) then
52              if(i_ND = '1') then
53                 w_DATA <= w_DATA(6 downto 0) & i_RX;
54              end if;
55           end if;
56        end if;
57
58     end process U1;
59
60     o_DATA <= w_DATA;
61
62  ----------------------------------------------------------------------------------------------
63  end Behavioral;
```

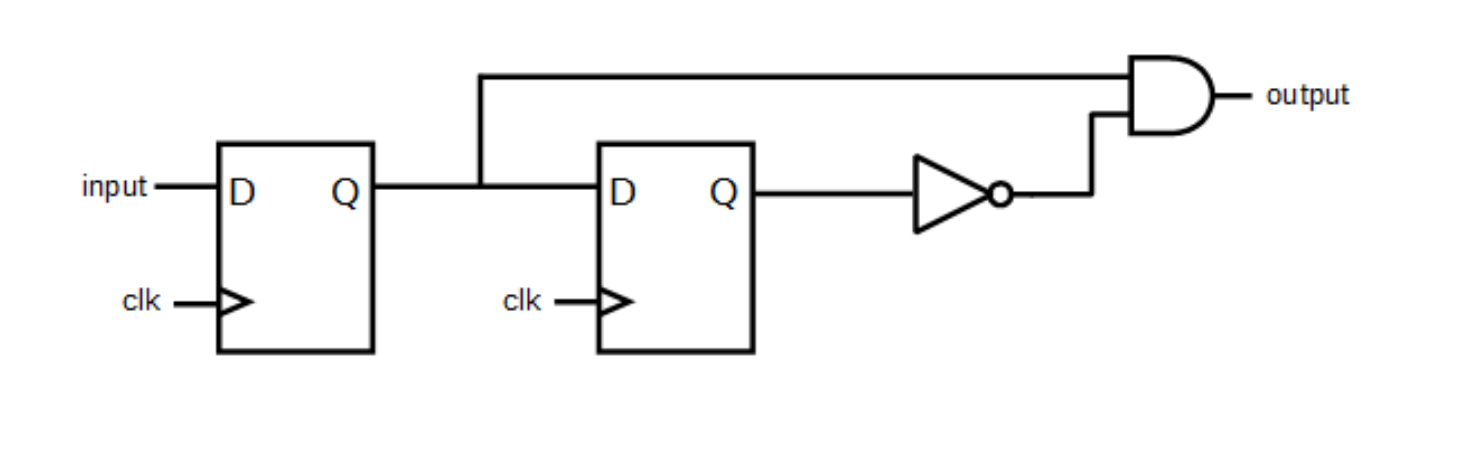
Detector de Borda

- Só se deve usar diretivas de análise de borda (rising_edge ou falling_edge) para o sinal de **Clock**
- Outros sinais precisam ter as bordas detectadas por um circuito apropriado



Circuito Detector de Borda

- **Sinal** a ser avaliado entra na entrada do primeiro Flip-flop (input) e detecção da borda é indicada em output

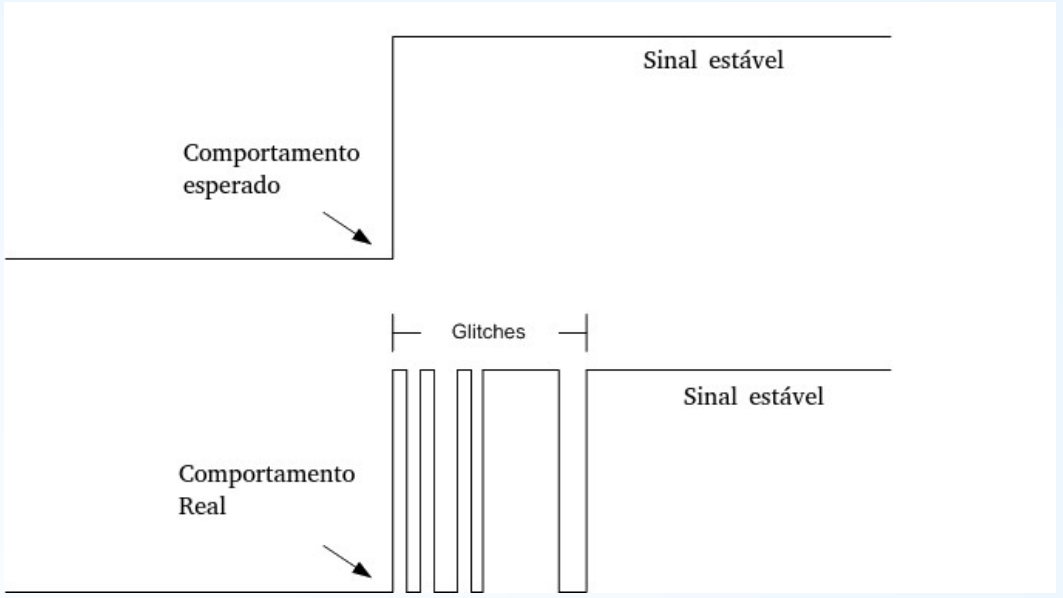


Código de um Detector de Borda

```
1.  library ieee;
2.  use ieee.std_logic_1164.all;
3.
4.  entity EdgeDetector is
5.     port (
6.        clk      :in std_logic;
7.        d        :in std_logic;
8.        edge     :out std_logic
9.     );
10. end EdgeDetector;
11.
12. architecture EdgeDetector_rtl of EdgeDetector is
13.
14.    signal reg1 :std_logic;
15.    signal reg2 :std_logic;
16.
17. begin
18. reg: process(clk)
19. begin
20.    if rising_edge(clk) then
21.       reg1  <= d;
22.       reg2  <= reg1;
23.   end if;
24. end process;
25.
26. edge <= reg1 and (not reg2);
27.
28. end EdgeDetector_rtl;
```

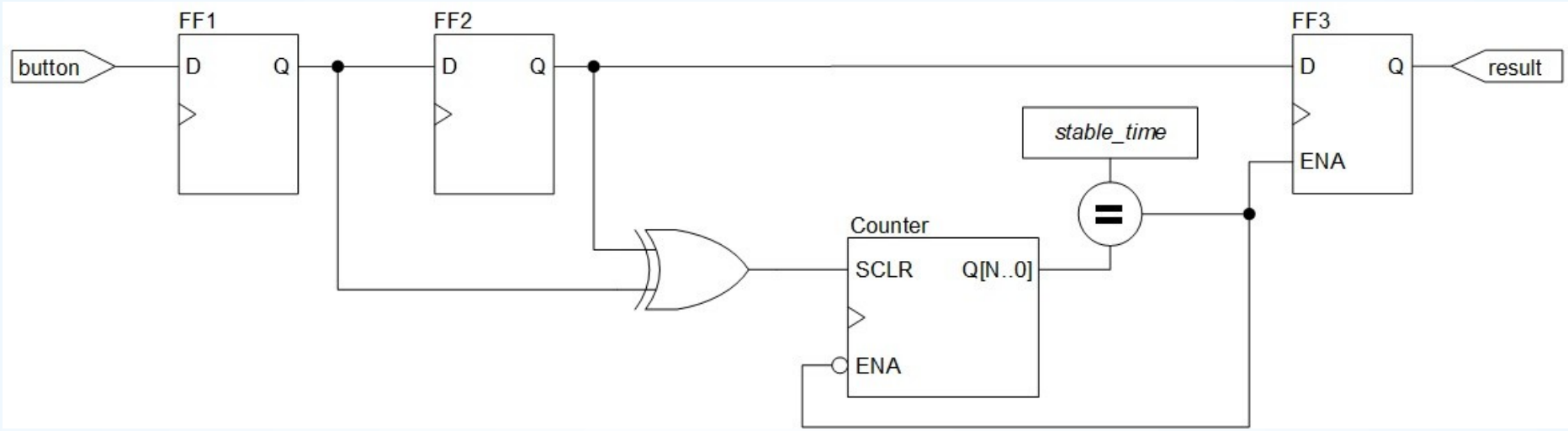
Lógica de Rejeição

- Interruptores físicos, como **botões** de pressão e **interruptores** de alternância, estão todos sujeitos a ruído (ressaltos)
- O ruído ocorre quando o interruptor é pressionado ou invertido. Isso acontece entre os contatos de metal ao se unirem (ou separarem) rapidamente antes que eles tenham tempo de se estabelecer de modo estável



Circuito de *Debounce*

- A ideia é utilizar um contador para verificar se o sinal fica estável por um determinado tempo
- Para chave e botões, é usual usar contadores de milisegundos (20 ms ou mais)



FIM AULA 14