



FEDERAL UNIVERSITY  
OF SANTA CATARINA

# Laboratório 6: FSMs

---

EEL5105 – Circuitos e Técnicas Digitais

## Objetivos

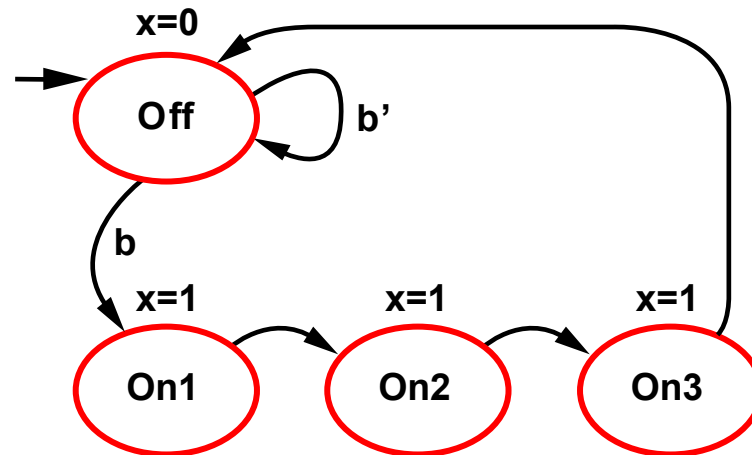
---

- Entender a implementação de **FSMs** usando **processos** em **VHDL**.
- Implementar **FSMs** usando **dois processos** (essa forma não é única).
- Fazer estudos de caso visando fixar os conceitos estudados.

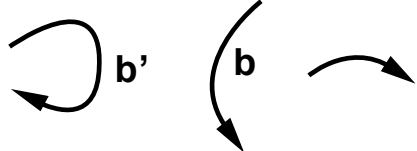
# Introdução

- **FSM** (Máquina de Estados Finitos) :

Entrada:  $b$ ; Saída:  $x$



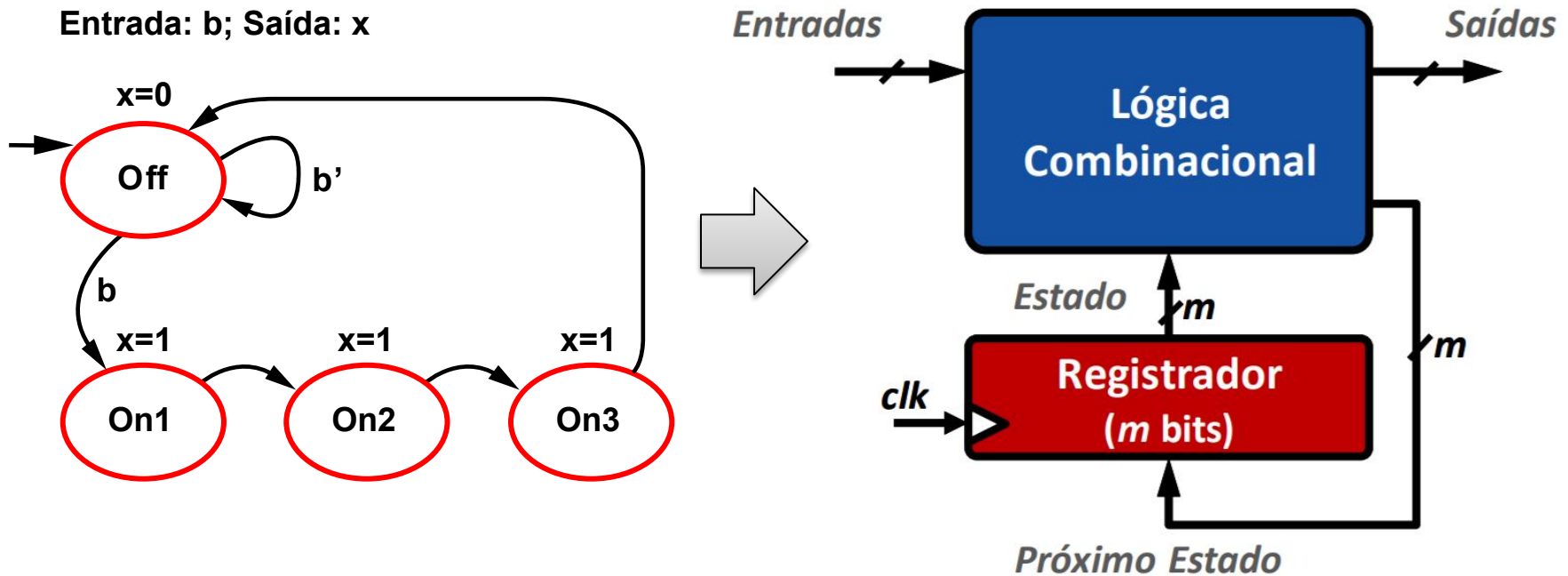
- **Estados:** Off On1 etc...

- **Transições:**  etc...

- **Ações (saídas):**  $x=1$   $x=0$  etc...

# Introdução

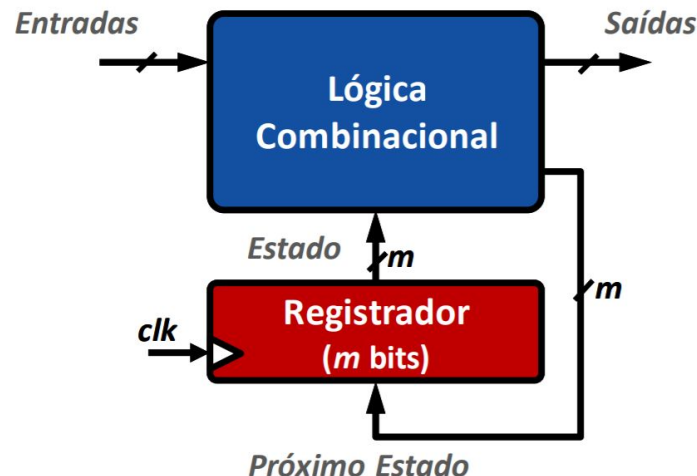
- Implementação de uma **FSM**:



# Introdução

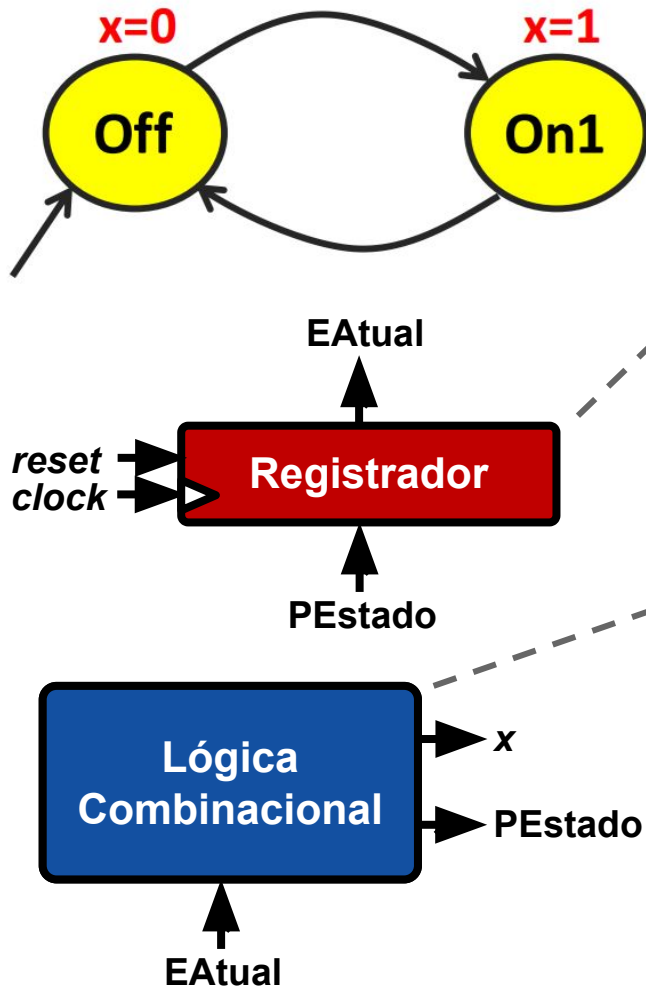
---

- Uma forma de implementação em **VHDL**:
  - Usar **dois** processos
    - **Processo 1**: implementa lógica combinacional
      - Cálculo do próximo estado
      - Cálculo da saída
    - **Processo 2**: implementa registrador
      - Atualiza estado na transição do *clock*



# Introdução

- Exemplo 1:



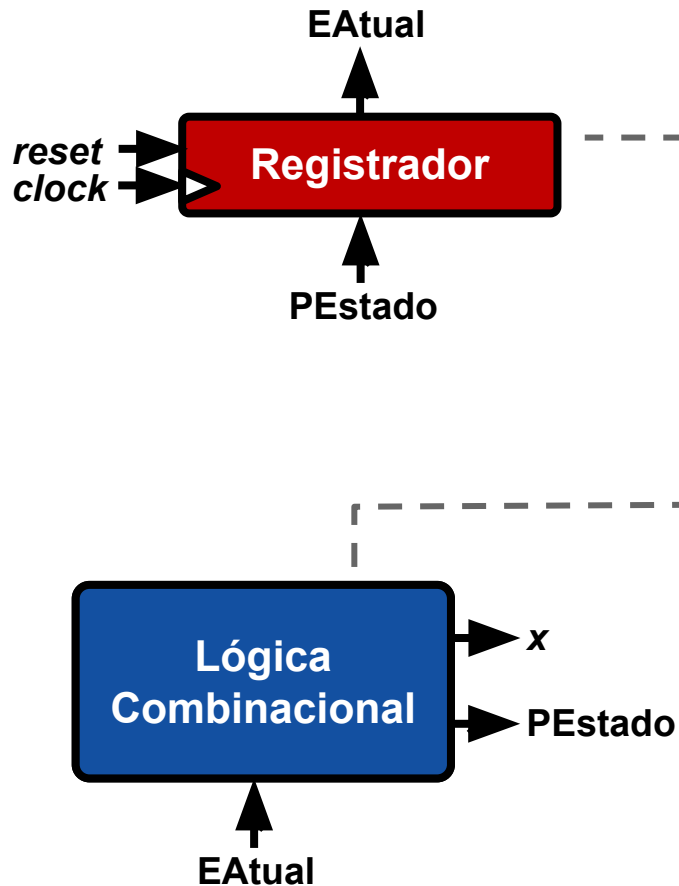
```
library ieee;
use ieee.std_logic_1164.all;
entity FSM1 is port (
    clock: in std_logic;
    reset: in std_logic;
    x: out std_logic );
end FSM1;

architecture fsm1arq of FSM1 is
    type STATES is (Off, On1);
    signal EAtual, PEstado: STATES;
begin
    process(clock, reset)
    begin
        if (reset = '1') then
            EAtual <= Off;
        elsif (clock'event AND clock = '1') then
            EAtual <= PEstado;
        end if;
    end process;

    process(EAtual)
    begin
        case EAtual is
            when Off => PEstado <= On1;
                        x <= '0';
            when On1 => PEstado <= Off;
                        x <= '1';
        end case;
    end process;
end fsm1arq;
```

# Introdução

- Exemplo 1:



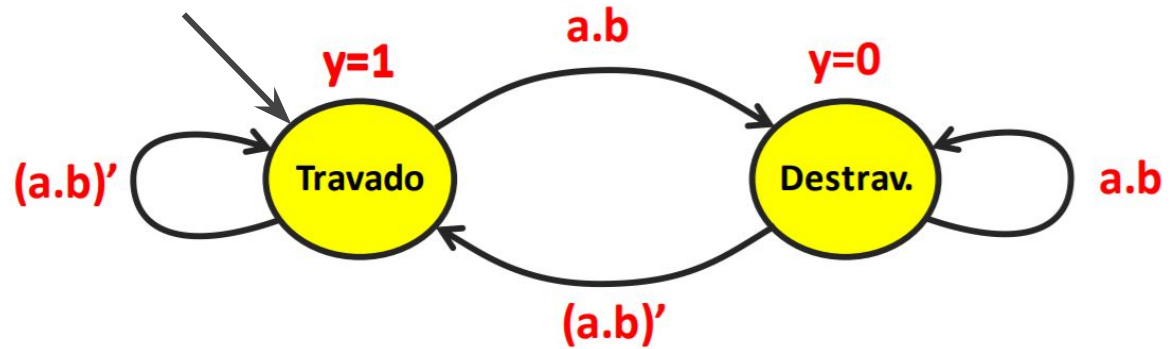
```
library ieee;
use ieee.std_logic_1164.all;
entity FSM1 is port (
    clock: in std_logic;
    reset: in std_logic;
    x: out std_logic );
end FSM1;

architecture fsm1arq of FSM1 is
    type STATES is (Off, On1);
    signal EAtual, PEstado: STATES;
begin
    REG: process(clock, reset)
    begin
        if (reset = '1') then
            EAtual <= Off;
        elsif (clock'event AND clock = '1') then
            EAtual <= PEstado;
        end if;
    end process;

    COMB: process(EAtual)
    begin
        case EAtual is
            when Off => PEstado <= On1;
                        x <= '0';
            when On1 => PEstado <= Off;
                       x <= '1';
        end case;
    end process;
end fsm1arq;
```

# Introdução

- Exemplo 2:



```
library ieee;
use ieee.std_logic_1164.all;
entity FSM2 is port (
    clock, reset, a, b: in std_logic;
    y: out std_logic );
end FSM2;

architecture fsm2arq of FSM2 is
    type STATES is (Travado, Destrav);
    signal EAtual, PEstado: STATES;
begin
    process(clock, reset)
    begin
        if (reset = '1') then
            EAtual <= Travado;
        elsif (clock'event AND clock = '1') then
            EAtual <= PEstado;
        end if;
    end process;
```

```
    process(EAtual, a, b)
    begin
        case EAtual is
            when Travado =>
                y <= '1';
                if ((a='1') AND (b='1')) then
                    PEstado <= Destrav;
                else
                    PEstado <= Travado;
                end if;
            when Destrav =>
                y <= '0';
                if ((a='1') AND (b='1')) then
                    PEstado <= Destrav;
                else
                    PEstado <= Travado;
                end if;
            end case;
        end process;
    end fsm2arq;
```



## Tarefa 1

---

- Projete um **circuito digital** que controla o acendimento de **3 leds** da seguinte forma:
  - Quando o circuito é resetado (a partir de uma entrada de **reset**), todos os leds devem ficar acesos.
  - Após 1 segundo, o primeiro **led** deve ser apagado;
  - Após mais 1 segundo, o segundo **led** também apaga;
  - Finalmente, 1 segundo depois, o terceiro **led** deve ser apagado;
  - Após 1 segundo com os **três leds apagados**, eles devem ser todos acesos e o processo iniciar novamente.
- Elabore o **diagrama de estados** e implemente a **máquina de estados** correspondente a esse circuito, usando a *port* **CLK\_1Hz** de **usertop** como *clock*.

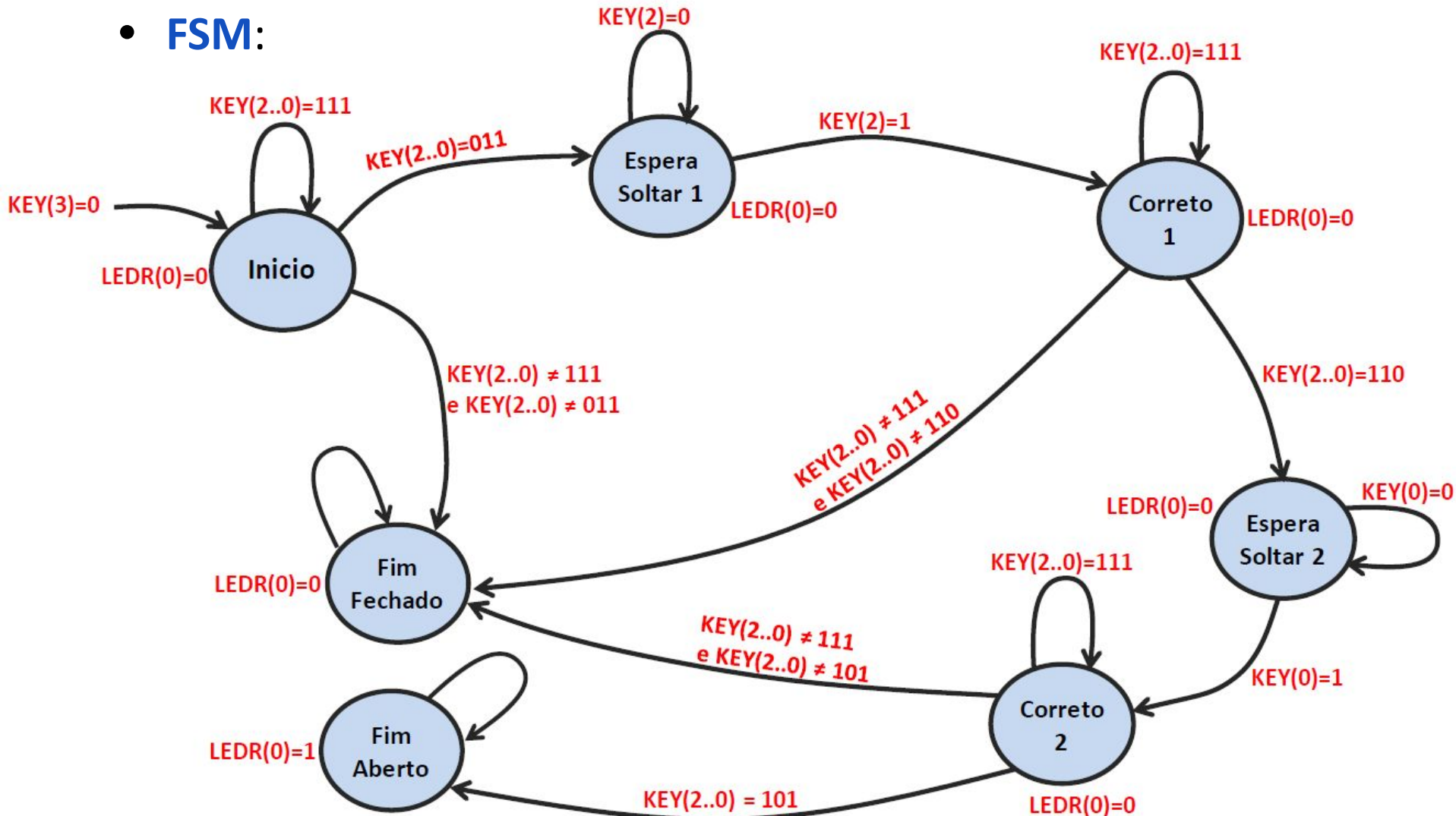
## Tarefa 2

---

- Implementar máquina de estados que controla **abertura de um cofre** a partir do acionamento dos botões **KEY(0)** a **KEY(3)** do **DE2**.
  - **Detalhe importante**: os botões **KEY(0)** a **KEY(3)** geram **0** quando são pressionados e **1** quando não estão pressionados.
  - O botão **KEY(3)** deve ser utilizado para dar **reset** no processo (ou seja, **KEY(3) = 0** leva a máquina para seu estado inicial).
  - Para abrir o cofre, os demais botões devem ser acionados na seguinte sequencia: **KEY(2)**, **KEY(0)**, **KEY(1)**.
  - **LEDR(0)** deve ser usado para indicar se cofre está travado ou destravado.
  - Utilize a port **CLK\_500Hz** do **usertop** como sinal de *clock* (trata-se de um *clock* de 500 Hz).

## Tarefa 2

- FSM:



## Tarefa Avançada

---

- Modifique sua FSM para que ela trabalhe com códigos de abertura de **4 cliques nos botões** ao invés de **3** (por exemplo, **KEY(2), KEY(2), KEY(0), KEY(1)**).