



**Universidade Federal de Santa Catarina**  
**Centro Tecnológico – CTC**  
**Departamento de Engenharia Elétrica**



# **“Circuitos e Técnicas Digitais”**

**Prof. Héctor Pettenghi Roldán\***

[Hector@eel.ufsc.br](mailto:Hector@eel.ufsc.br)

**Florianópolis, março de 2016.**

**\*Baseados nos slides do Professor Eduardo Bezerra e Eduardo Batista EEL5105 2015.2**

# Objetivos do laboratório

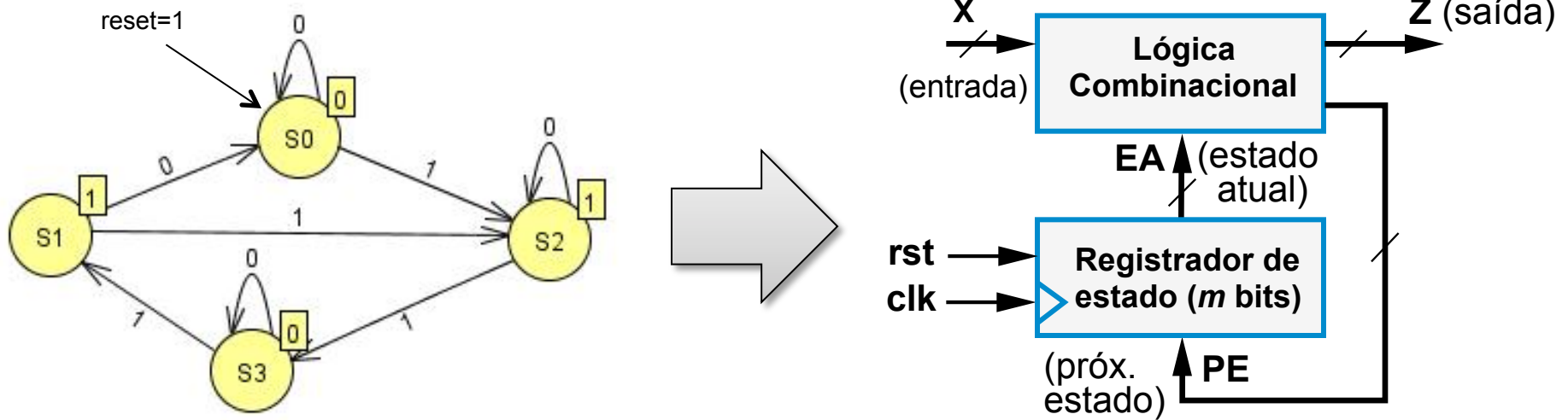
---

1. Entender o conceito de máquinas de estados (FSM).
2. Entender o conceito de circuito sequencial controlando o fluxo de atividades de circuito combinacional.
3. Entender o processo de síntese de FSMs em VHDL.
4. Entender o funcionamento de contadores.
5. Estudo de caso: projeto e implementação em VHDL de um contador baseado em máquinas de estados.

## **Síntese de FSMs**

# Estrutura de uma FSM

- Dois módulos:
  - Armazenamento do “estado atual”; e
  - Cálculo da “saída” e do “próximo estado”



# Uso de VHDL para descrever uma FSM

## VHDL típico de uma máquina de estados – 2 processos

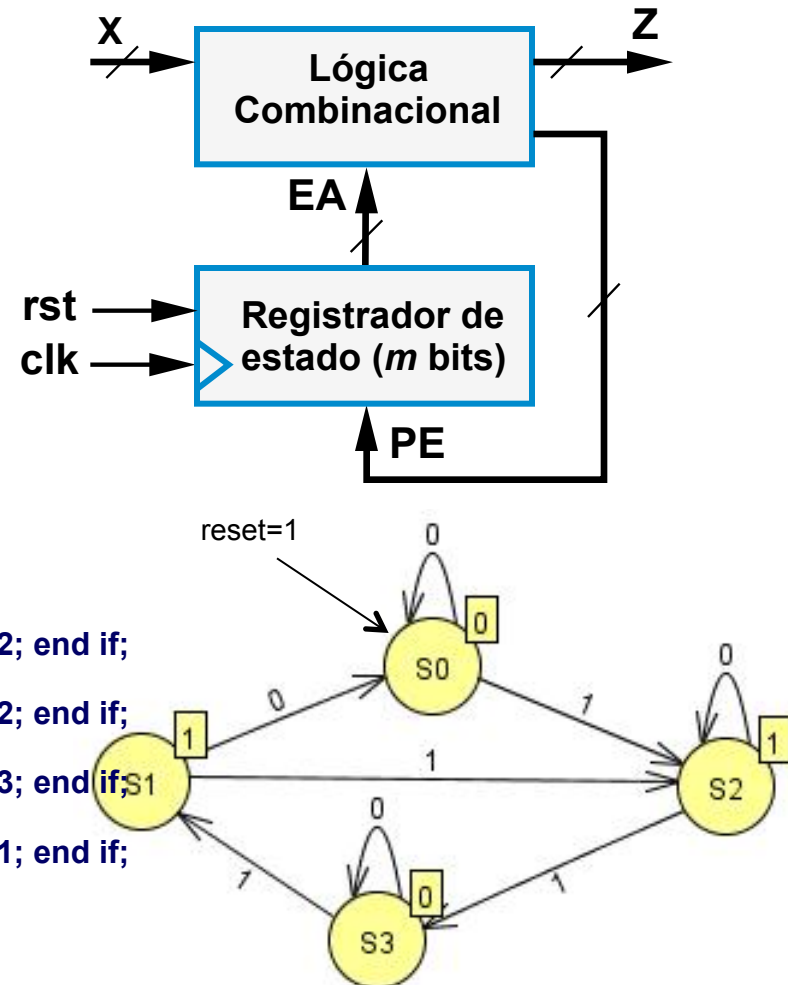
```
entity MOORE is port(X, clock, reset : in std_logic;  Z: out std_logic);  end;
```

```
architecture A of MOORE is
  type STATES is (S0, S1, S2, S3);
  signal EA, PE : STATES;
begin
```

```
  process (clock, reset)
  begin
    if reset= '1' then
      EA <= S0;
    elsif clock'event and clock='1' then
      EA <= PE ;
    end if;
  end process;
```

```
  process(EA, X)
  begin
    case EA is
      when S0 =>
        Z <= '0';
        if X='0' then PE <=S0; else PE <= S2; end if;
      when S1 =>
        Z <= '1';
        if X='0' then PE <=S0; else PE <= S2; end if;
      when S2 =>
        Z <= '1';
        if X='0' then PE <=S2; else PE <= S3; end if;
      when S3 =>
        Z <= '0';
        if X='0' then PE <=S3; else PE <= S1; end if;
    end case;
  end process;
```

```
end A;
```



# Uso de VHDL para descrever uma FSM

## VHDL típico de uma máquina de estados – 2 processos

```
entity MOORE is port(X, clock, reset : in std_logic;  Z: out std_logic);  end;
```

```
architecture A of MOORE is
```

```
  type STATES is (S0, S1, S2, S3);  
  signal EA, PE : STATES;
```

```
begin
```

```
  process (clock, reset)  
  begin
```

```
    if reset= '1' then  
      EA <= S0;
```

```
    elsif clock'event and clock='1' then  
      EA <= PE ;
```

```
    end if;
```

```
  end process;
```

```
  process(EA, X)  
  begin
```

```
    case EA is
```

```
      when S0 =>
```

```
        Z <= '0';
```

```
        if X='0' then PE <=S0; else PE <= S2; end if;
```

```
      when S1 =>
```

```
        Z <= '1';
```

```
        if X='0' then PE <=S0; else PE <= S2; end if;
```

```
      when S2 =>
```

```
        Z <= '1';
```

```
        if X='0' then PE <=S2; else PE <= S3; end if;
```

```
      when S3 =>
```

```
        Z <= '0';
```

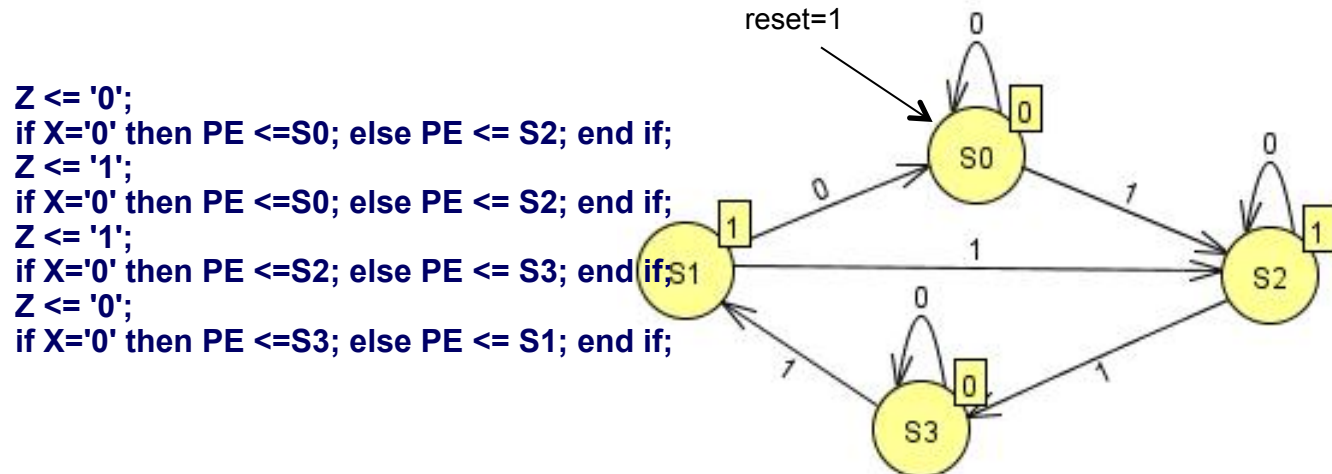
```
        if X='0' then PE <=S3; else PE <= S1; end if;
```

```
    end case;
```

```
  end process;
```

```
end A;
```

Definição dos estados possíveis da máquina e de variáveis que podem assumir esses estados.



# Uso de VHDL para descrever uma FSM

## VHDL típico de uma máquina de estados – 2 processos

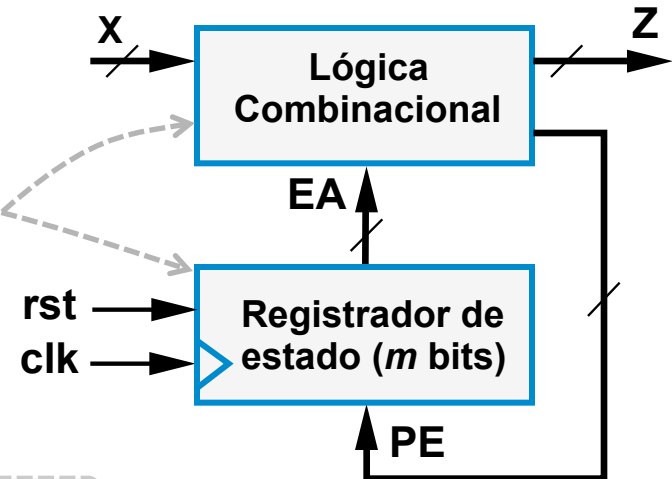
```
entity MOORE is port(X, clock, reset : in std_logic;  Z: out std_logic);  end;
```

```
architecture A of MOORE is
  type STATES is (S0, S1, S2, S3);
  signal EA, PE : STATES;
begin
```

```
process (clock, reset)
begin
  if reset= '1' then
    EA <= S0;
  elsif clock'event and clock='1' then
    EA <= PE ;
  end if;
end process;
```

```
process(EA, X)
begin
  case EA is
    when S0 =>    Z <= '0';
                  if X='0' then PE <=S0; else PE <= S2; end if;
    when S1 =>    Z <= '1';
                  if X='0' then PE <=S0; else PE <= S2; end if;
    when S2 =>    Z <= '1';
                  if X='0' then PE <=S2; else PE <= S3; end if;
    when S3 =>    Z <= '0';
                  if X='0' then PE <=S3; else PE <= S1; end if;
  end case;
end process;
```

```
end A;
```



# Uso de VHDL para descrever uma FSM

## VHDL típico de uma máquina de estados – 2 processos

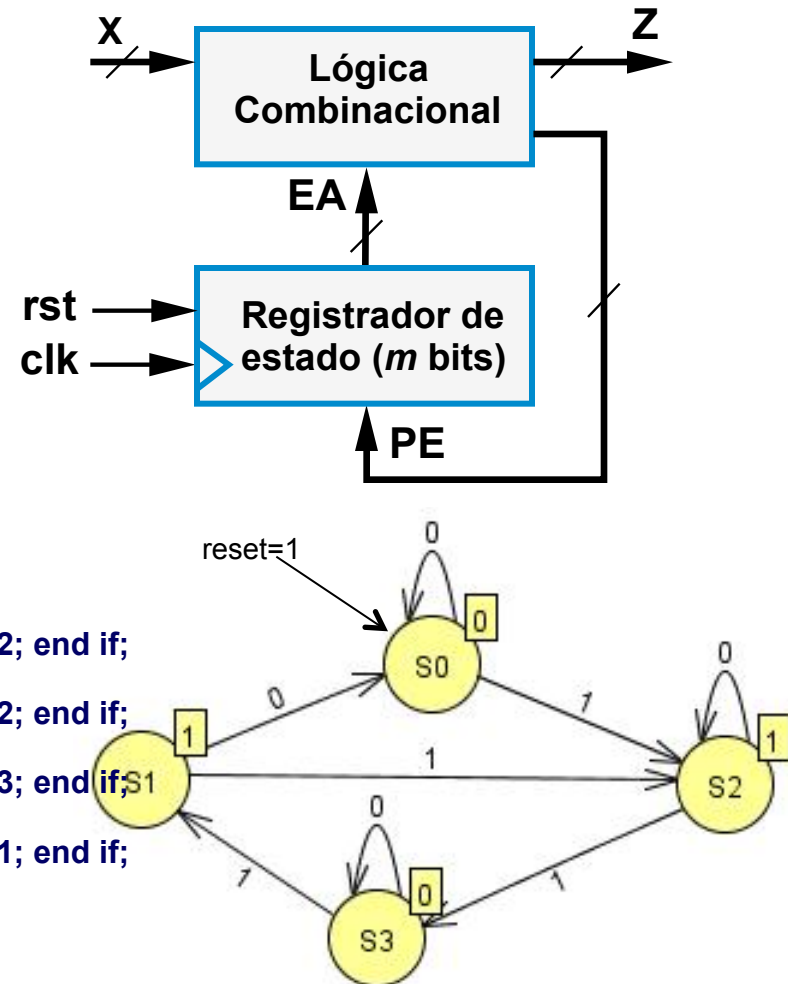
```
entity MOORE is port(X, clock, reset : in std_logic;  Z: out std_logic);  end;
```

```
architecture A of MOORE is
  type STATES is (S0, S1, S2, S3);
  signal EA, PE : STATES;
begin
```

```
  process (clock, reset)
  begin
    if reset= '1' then
      EA <= S0;
    elsif clock'event and clock='1' then
      EA <= PE ;
    end if;
  end process;
```

```
  process(EA, X)
  begin
    case EA is
      when S0 =>
        Z <= '0';
        if X='0' then PE <=S0; else PE <= S2; end if;
      when S1 =>
        Z <= '1';
        if X='0' then PE <=S0; else PE <= S2; end if;
      when S2 =>
        Z <= '1';
        if X='0' then PE <=S2; else PE <= S3; end if;
      when S3 =>
        Z <= '0';
        if X='0' then PE <=S3; else PE <= S1; end if;
    end case;
  end process;
```

```
end A;
```





# Uso de VHDL para descrever uma FSM

## VHDL típico de uma máquina de estados – 2 processos

```
entity MOORE is port(X, clock, reset : in std_logic;    Z: out std_logic);    end;
```

```
architecture A of MOORE is
```

```
    type STATES is (S0, S1, S2, S3);
```

```
    signal EA, PE : STATES;
```

```
begin
```

```
    process (clock, reset)
```

```
    begin
```

```
        if reset= '1' then
```

```
            EA <= S0;
```

```
        elsif clock'event and clock='1' then
```

```
            EA <= PE ;
```

```
        end if;
```

```
    end process;
```

```
    process(EA, X)
```

```
    begin
```

```
        case EA is
```

```
            when S0 =>
```

```
                Z <= '0';
```

```
                if X='0' then PE <=S0; else PE <= S2; end if;
```

```
            when S1 =>
```

```
                Z <= '1';
```

```
                if X='0' then PE <=S0; else PE <= S2; end if;
```

```
            when S2 =>
```

```
                Z <= '1';
```

```
                if X='0' then PE <=S2; else PE <= S3; end if;
```

```
            when S3 =>
```

```
                Z <= '0';
```

```
                if X='0' then PE <=S3; else PE <= S1; end if;
```

```
        end case;
```

```
    end process;
```

```
end A;
```

**Importante:** essa é uma máquina do tipo **Moore** já que as saídas dependem apenas do estado atual!

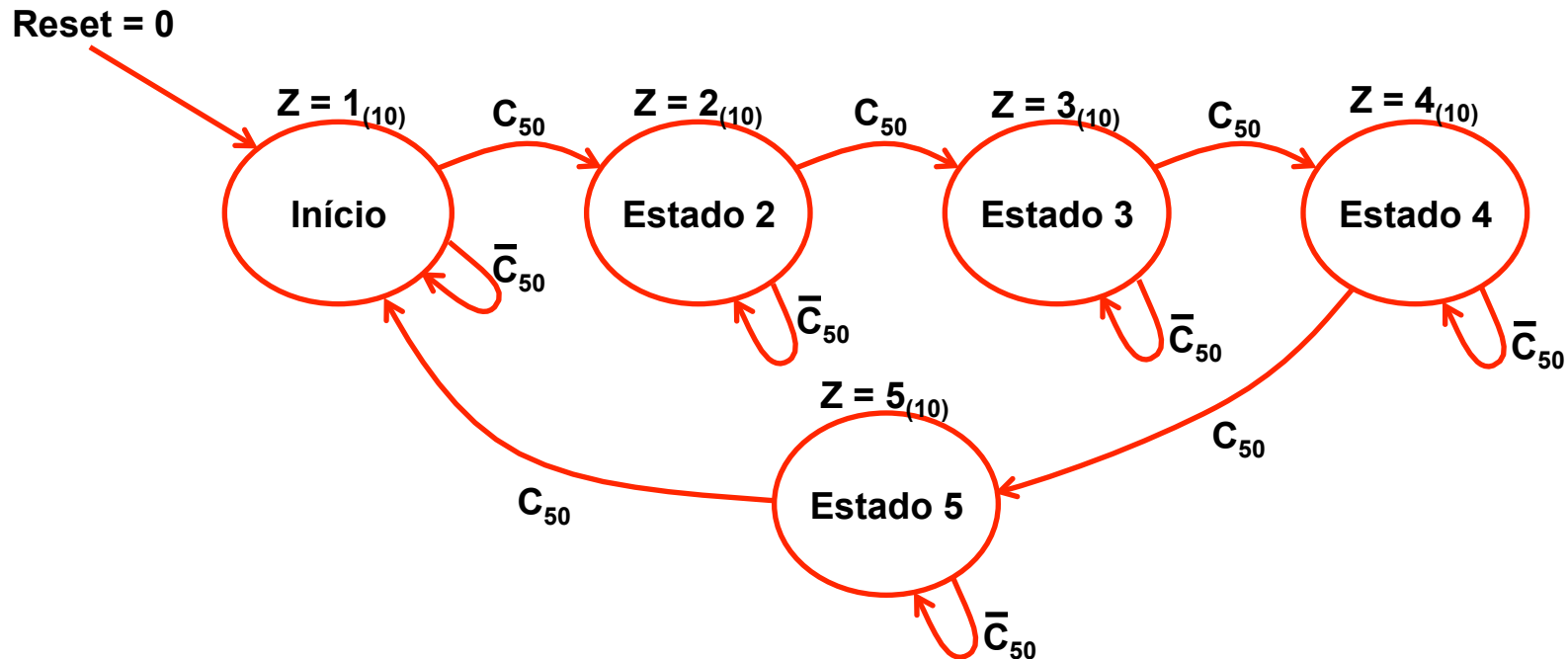
**Tarefa a ser realizada:**  
**Contador de  $1_{(10)}$  a  $5_{(10)}$  com Período de 1 Segundo**

# Descrição Funcional: Contador com Período de 1 Segundo

---

- **Ideia:** trocar o *clock* manual da FSM anterior por um *clock* de **1 Hz**.
- **Problema:** o *clock* disponível no kit é de **50 MHz** (período = **20 ns**).
- **Solução:** usar um contador que conta de **0** a **49.999.999** e só mudar de estado quando a contagem for igual a **49.999.999**.
  - $49.999.999 = 2FAF07F_H$
  - Em VHDL:  $2FAF07F_H = \text{0x"2FAF07F"}$  (com **28 bits**)

# Diagrama de estados: Contador com Período de 1 Segundo



# Descrição VHDL: Contador com Período de 1 Segundo

- “Embutindo” um contador na sua FSM

- **Passo 1:** Crie os seguintes signals na FSM\_Conta.

```
signal C50: std_logic;    -- vai para 1 quando o valor da contagem for 49.999.999  
signal contador: std_logic_vector(27 downto 0); -- registra valor da contagem
```

- **Passo 2:** Inclua um trecho de código na FSM\_Conta que faça com que o signal contador receba 0000000<sub>H</sub> quando a FSM for resetada.

```
contador <= 0x"0000000";
```

- **Passo 3:** Inclua um trecho de código na FSM\_Conta que faça com que o signal contador seja incrementado quando ocorrer uma transição positiva de clock.

```
contador <= contador + 1;
```

**Observação:** para usar o código acima, será preciso incluir a biblioteca que define a operação de soma para std\_logic\_vector.

```
use ieee.std_logic_unsigned.all;
```

# Descrição VHDL: Contador com Período de 1 Segundo

- “Embutindo” um contador na sua FSM
  - **Passo 4:** Inclua o trecho de código que, quando a contagem chegar a 49.999.999 ( $2\text{FAF}07\text{F}_{\text{H}}$ ), faça  $C_{50} = 1$  e contador = 0. Senão, esse trecho deverá fazer  $C_{50} = 0$ .  
**Atenção:** esse teste deverá ser feito somente na transição positiva do clock.

```
if contador = x"2FAF07F" then
    contador <= x"0000000";
    C50 <= '1';
else
    C50 <= '0';
end if;
```

- **Passo 5:** modifique a sua FSM\_Conta para que a atribuição de próximo estado só ocorra quando  $C_{50} = 1$  (ou seja, **EA <= PE** somente para **C50 = '1'**).
- **Passo 6:** incluir entrada CLOCK\_50 no topo e conectá-la na entrada de clock de FSM\_Conta.

```
entity topo is
    port ( LEDR: out std_logic_vector(3 downto 0);
          KEY: in std_logic_vector(1 downto 0);
          CLOCK_50: in std_logic );
end topo;
```