Universidade de São Paulo

Departamento de Engenharia de Computação e Sistemas Digitais

PCS

Projeto de Máquinas de Estado com VHDL

PCS3115 Sistemas Digitais I - Aula 19

Prof. Edson S. Gomi

PCS - Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Maio, 2020

Projeto de Máquinas de Estado com VHDL

O uso de uma linguagem de descrição de hardware (HDL - *Hardware Description Language*) permite representar a solução para um problema que será implementado por meio de um circuito digital de uma maneira mais abstrata. Linguagens HDL dão suporte à técnica do projeto estruturado, facilitando a descrição e o design de circuitos de alta complexidade. Também aceleram o tempo de desenvolvimento do circuito e facilitam o processo de depuração, na medida em que com as simulações podemos postergar ao máximo a realização física dos circuitos.

Projeto de Máquinas de Estado com VHDL

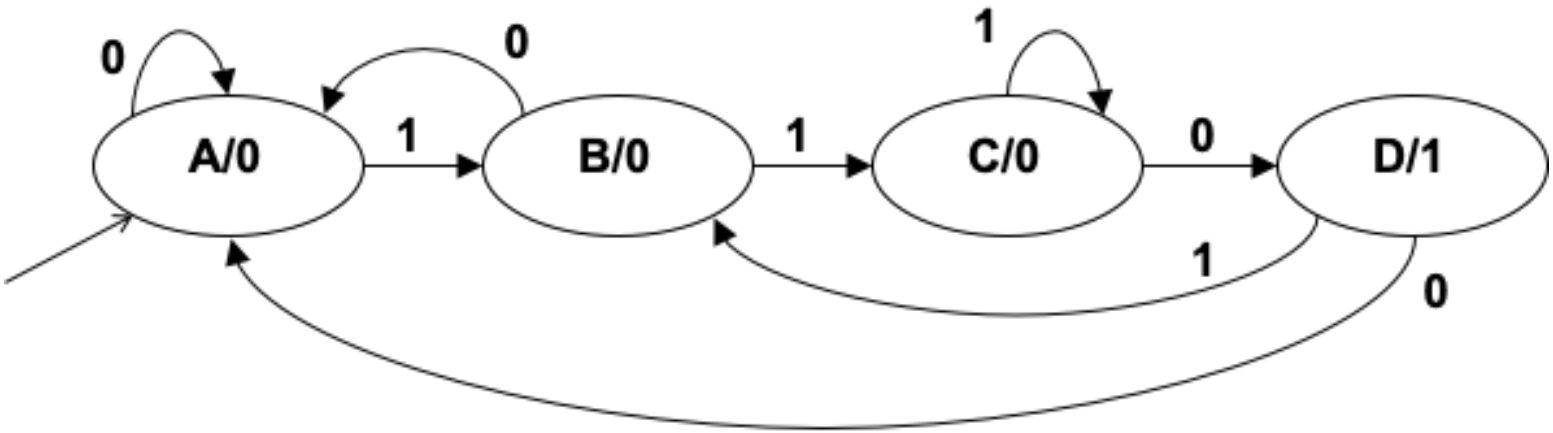
Nesta aula aprenderemos como descrever o diagrama de transições de estado de uma Máquina de Estados diretamente em VHDL. Tendo a descrição, um compilador VHDL assumirá a responsabilidade pela síntese do circuito digital ou a geração das estruturas para a realização de simulações.

Exemplo : detector de 0 após 2 ou mais 1s

Enunciado: Projetar um circuito sequencial síncrono que reconhece o primeiro ZERO apos a ocorrência de dois ou mais UNS consecutivos. Adotar uma solução do tipo Moore.

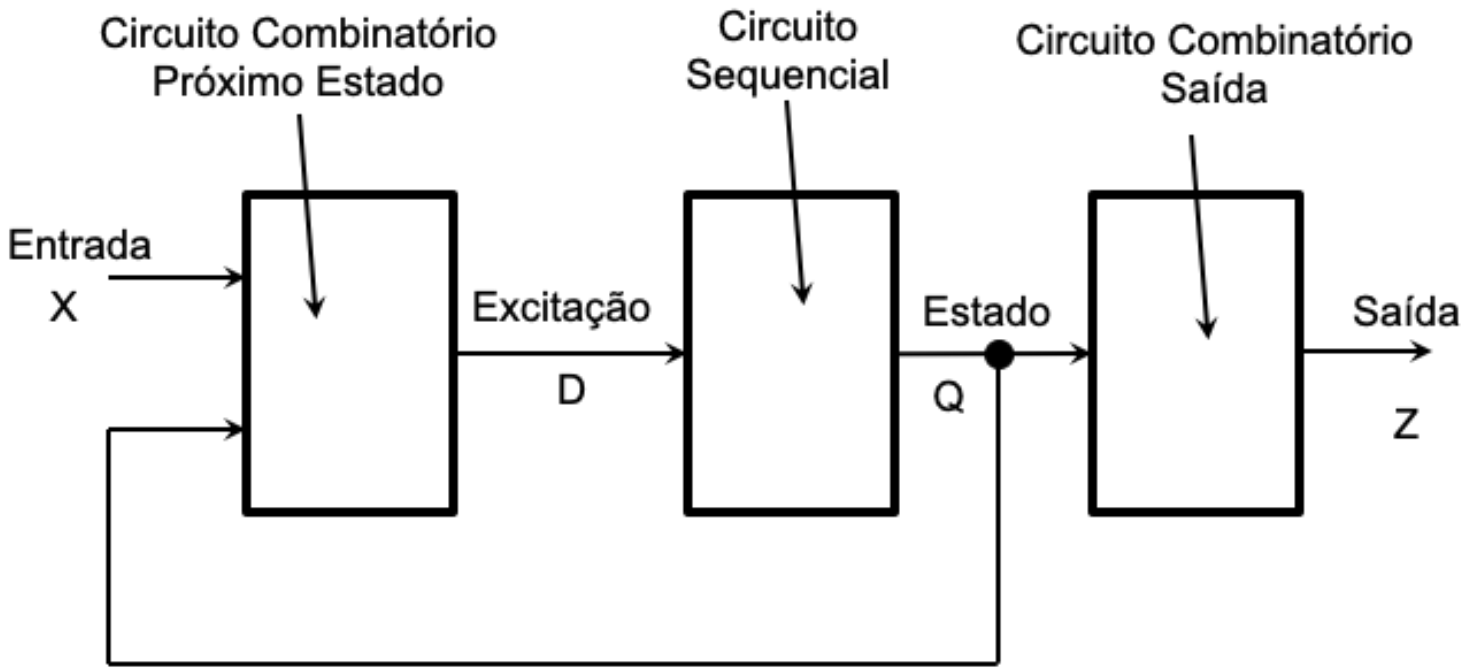
Entrada	0	1	0	1	1	0	0	1	1	1	0	1	0	0	0
Saída	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0

Máquina de Estado do Detector



A	Estado inicial: ZEROS
B	Ocorrência de 1 UM
C	Ocorrência de 2+ UNS
D	Ocorrência do 1o ZERO após 2+ UNS

Detector em VHDL



Entidade do Detector

```
1 entity detector is
2   port (
3     RESET : in bit;
4     X : in bit;
5     CLOCK : in bit;
6     Z : out bit
7     );
8 end entity detector;
```

Arquitetura do Detector

```
1  architecture behavioral of detector is
2    type state_type is (A,B,C,D);
3    signal present_state, next_state : state_type;
4  begin
5
6  -- Bloco sequencial : Estado Atual
7
8  -- Bloco combinatorio : Proximo Estado
9
10 -- Bloco combinatorio : Saida
11
12 end architecture behavioral;
```

Bloco Estado Atual

```
1  ESTADO_ATUAL: process (RESET,CLOCK) is
2  begin
3     if (RESET = '1') then
4        present_state <= A;
5     elsif (rising_edge(CLOCK)) then
6        present_state <= next_state;
7     end if;
8  end process ESTADO_ATUAL;
```

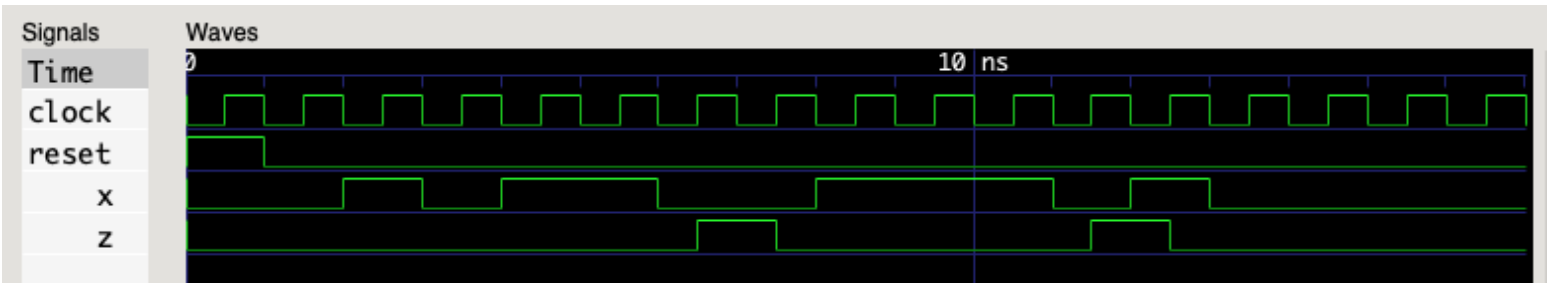
Bloco Próximo Estado

```
1    next_state <=
2      A when (present_state = A) and (X = '0') else
3      B when (present_state = A) and (X = '1') else
4      A when (present_state = B) and (X = '0') else
5      C when (present_state = B) and (X = '1') else
6      D when (present_state = C) and (X = '0') else
7      C when (present_state = C) and (X = '1') else
8      B when (present_state = D) and (X = '0') else
9      A when (present_state = D) and (X = '1') else
10     A;
```

Bloco Saída

```
1    Z <=
2        '0' when present_state = A else
3        '0' when present_state = B else
4        '0' when present_state = C else
5        '1' when present_state = D else
6        '0';
```

Simulação do Detector



Testbench para o Detector

```
1 -- A testbench has no ports.
2 entity detector_tb is
3 end entity detector_tb;
```

Arquitetura do Testbench

```
1  architecture testbench of detector_tb is
2  --  Declaration of the component to be tested.
3  -- Declaration of signals
4
5  begin
6  -- Component instantiation
7      DUT : Device Under Test
8  -- Clock generator
9  -- Stimulus process
10     Tests to apply
11     Test application loop
12     Check the outputs
13 end architecture testbench;
```

Declaração do DUT e dos sinais

```
1   component detector
2     port (
3       RESET : in bit;
4       X : in bit;
5       CLOCK : in bit;
6       Z : out bit
7       );
8   end component detector;
9
10  signal reset : bit;
11  signal x : bit;
12  signal clock : bit;
13  signal z : bit;
```

Instanciação do DUT

```
1   DUT: entity work.detector port map (
2      RESET => reset,
3      X => x,
4      Clock => clock,
5      Z => z
6      );
```

Gerador do clock

```
1   clk: process is
2   begin
3     clock <= '0';
4     wait for 0.5 ns;
5     clock <= '1';
6     wait for 0.5 ns;
7   end process clk;
```

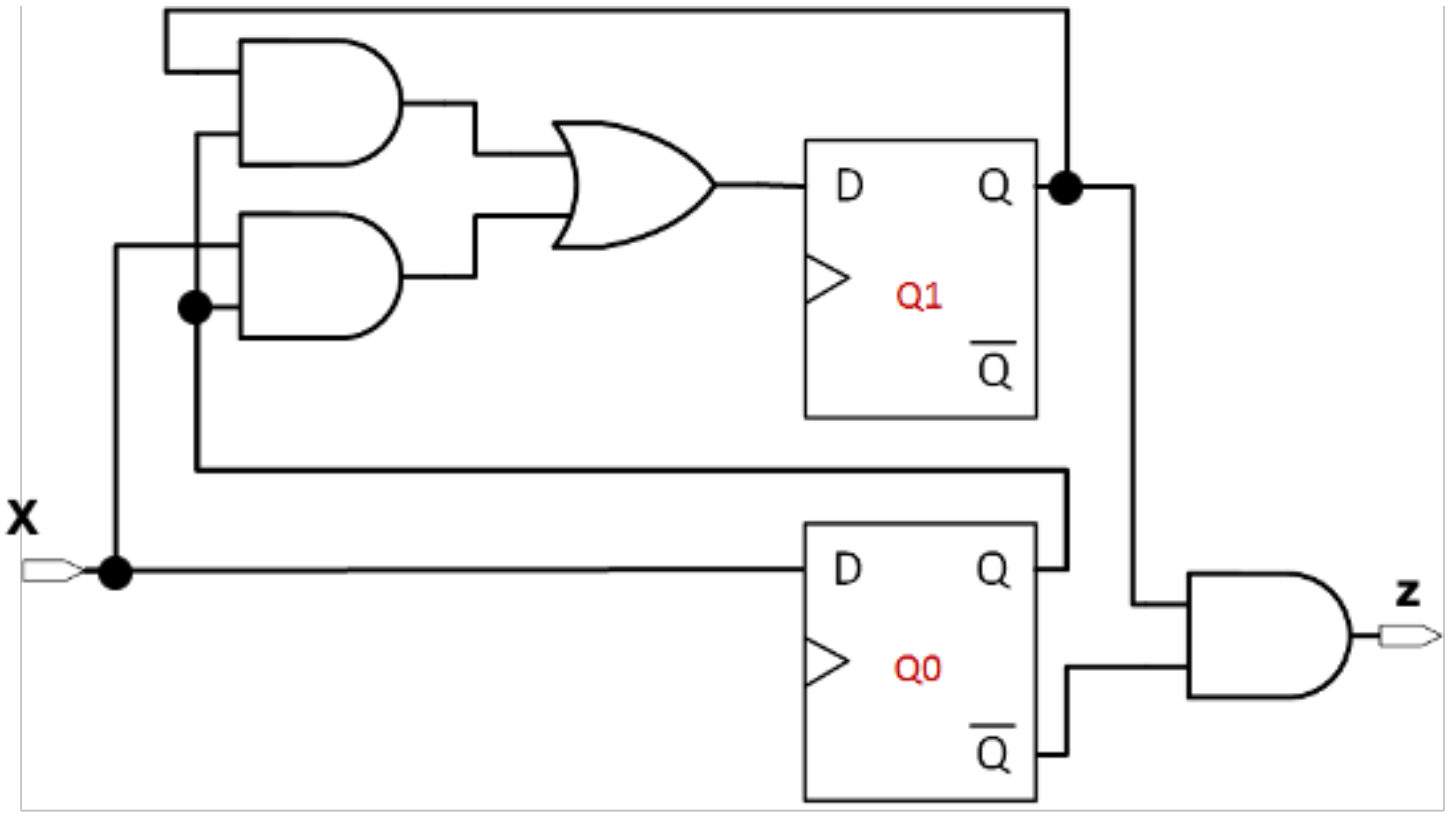
Casos de Teste

```
1      type pattern_type is record
2        -- The inputs of the circuit.
3        reset : bit;
4        x : bit;
5        -- The expected outputs of the circuit.
6        z : bit;
7      end record;
8
9      type pattern_array is array (natural range <>)
10     of pattern_type;
11     constant patterns : pattern_array :=
12       (
13         ('1','0','0'),
14         ('0','0','0'),
15         ('0','1','0'),
16         ..............
17         ('0','0','0'),
18         ('0','0','0'));
```

Aplicação dos Testes

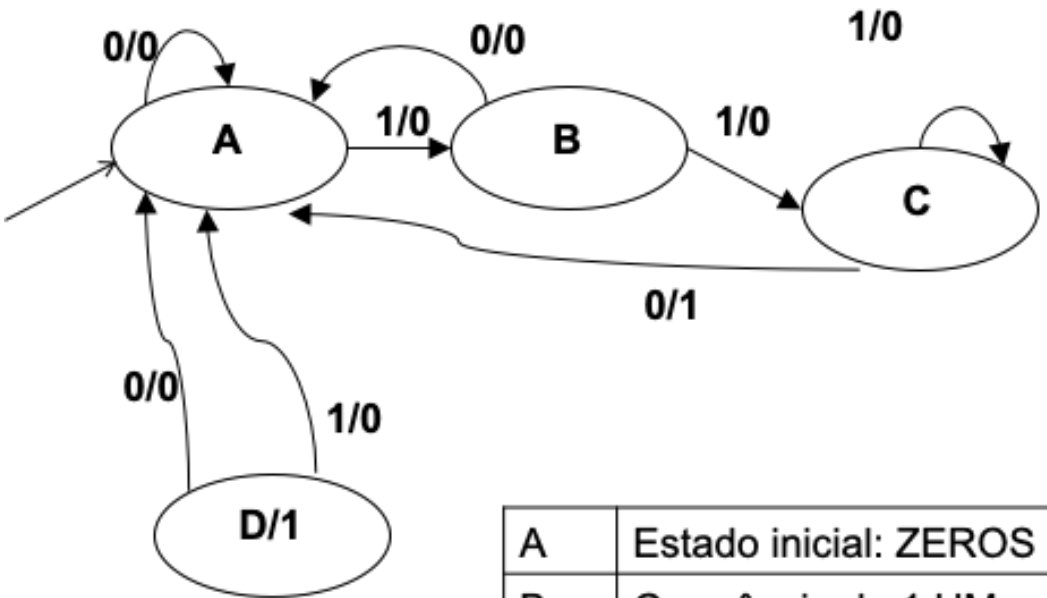
```
1  stimulus_process: process is
2  begin
3      --  Check each pattern.
4      for k in patterns'range loop
5        --  Set the inputs.
6        reset <= patterns(k).reset;
7        x <= patterns(k).x;
8         --  Wait for the results.
9        wait for 1 ns;
10       --  Check the outputs.
11       assert z = patterns(k).z
12         report "bad Z" severity error;
13     end loop;
14     assert false report "end of test" severity note;
15     --  Wait forever; this will finish
16     -- the simulation.
17     wait;
18 end process;
```

Detector - Circuito Modelo Moore



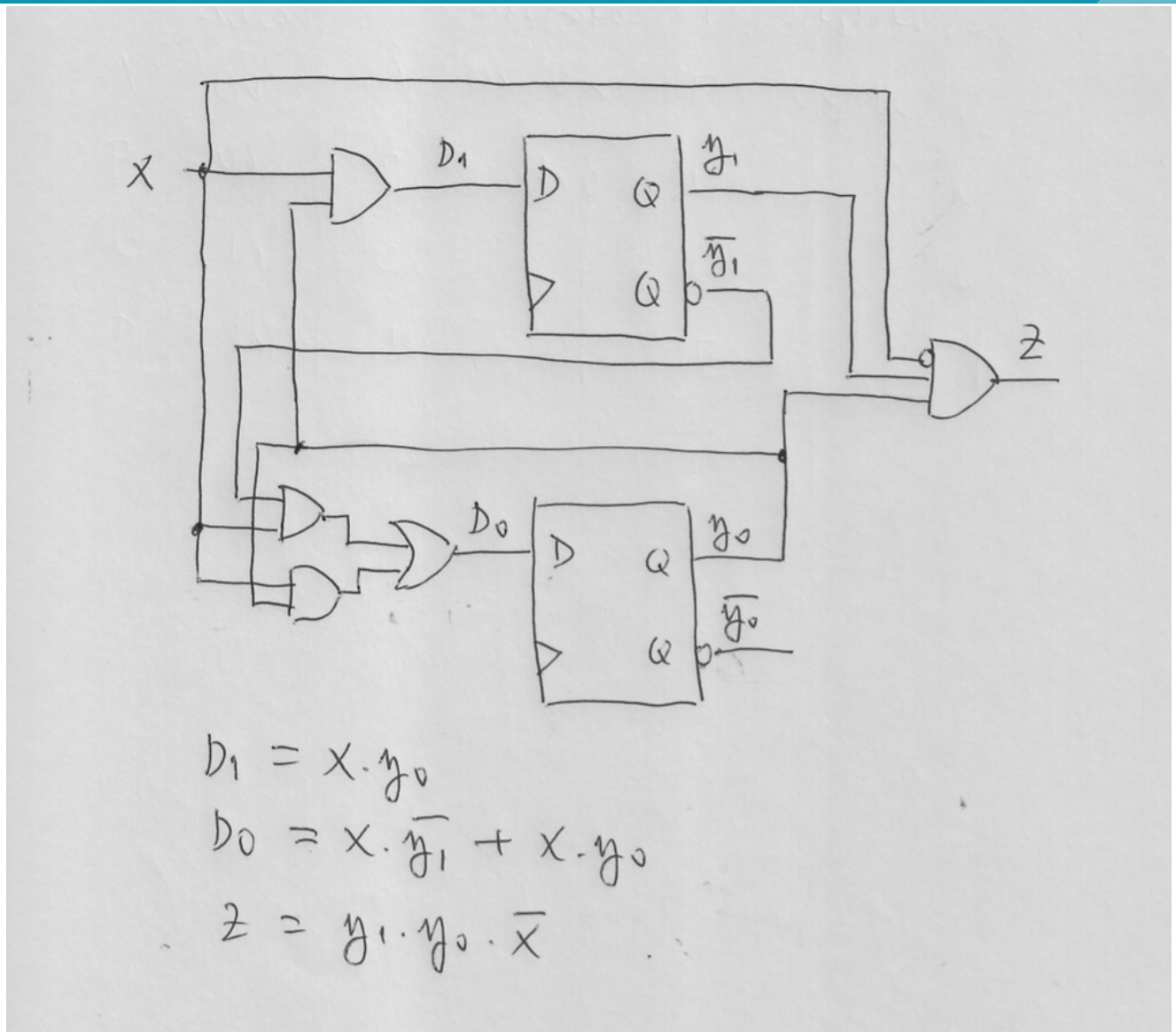
$D1^t = x \cdot y0 + y1 \cdot y0$ $\quad D0^t = x$ $\quad z^t = y1 \cdot y0'$

Detector - Modelo Mealy



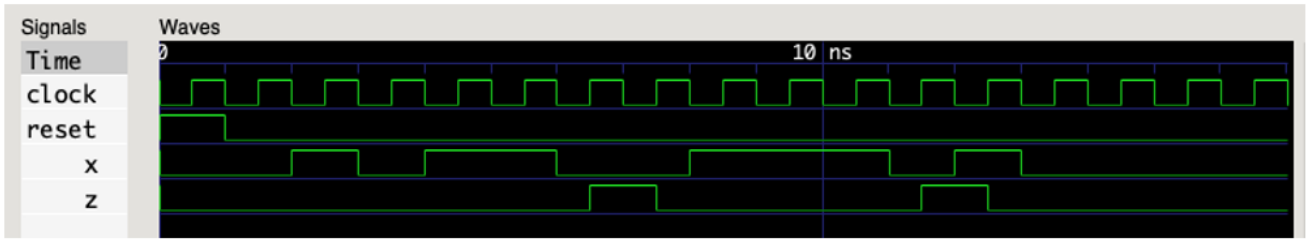
A	Estado inicial: ZEROS
B	Ocorrência de 1 UM
C	Ocorrência de 2+ UNS
D	Estado inatingível

Detector - Circuito Modelo Mealy

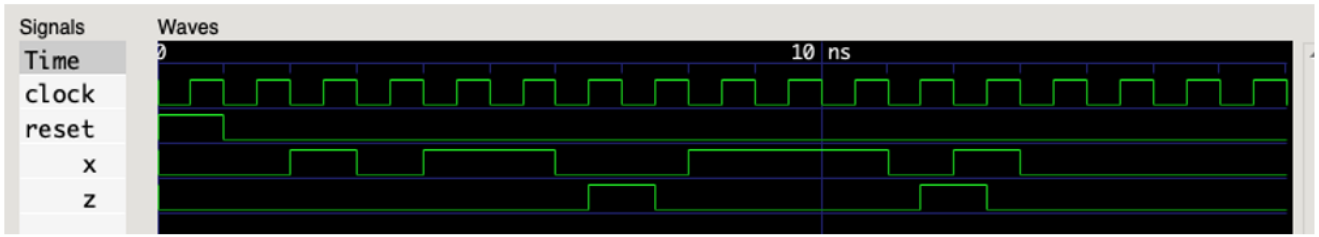


Detector - Comparação das Simulações

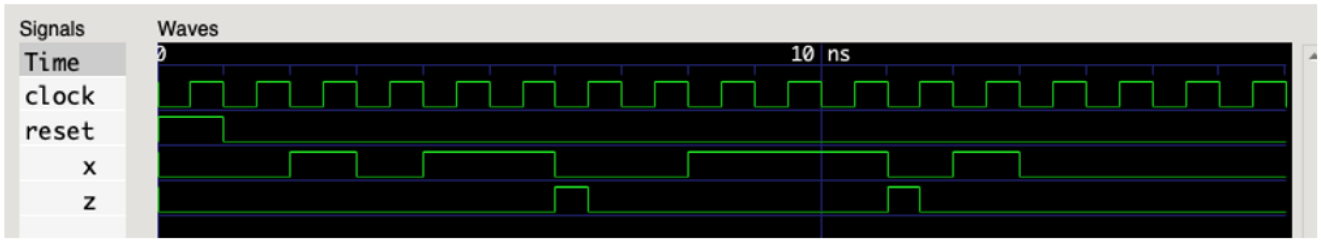
VHDL Behavioral



Circuito Moore



Circuito Mealy



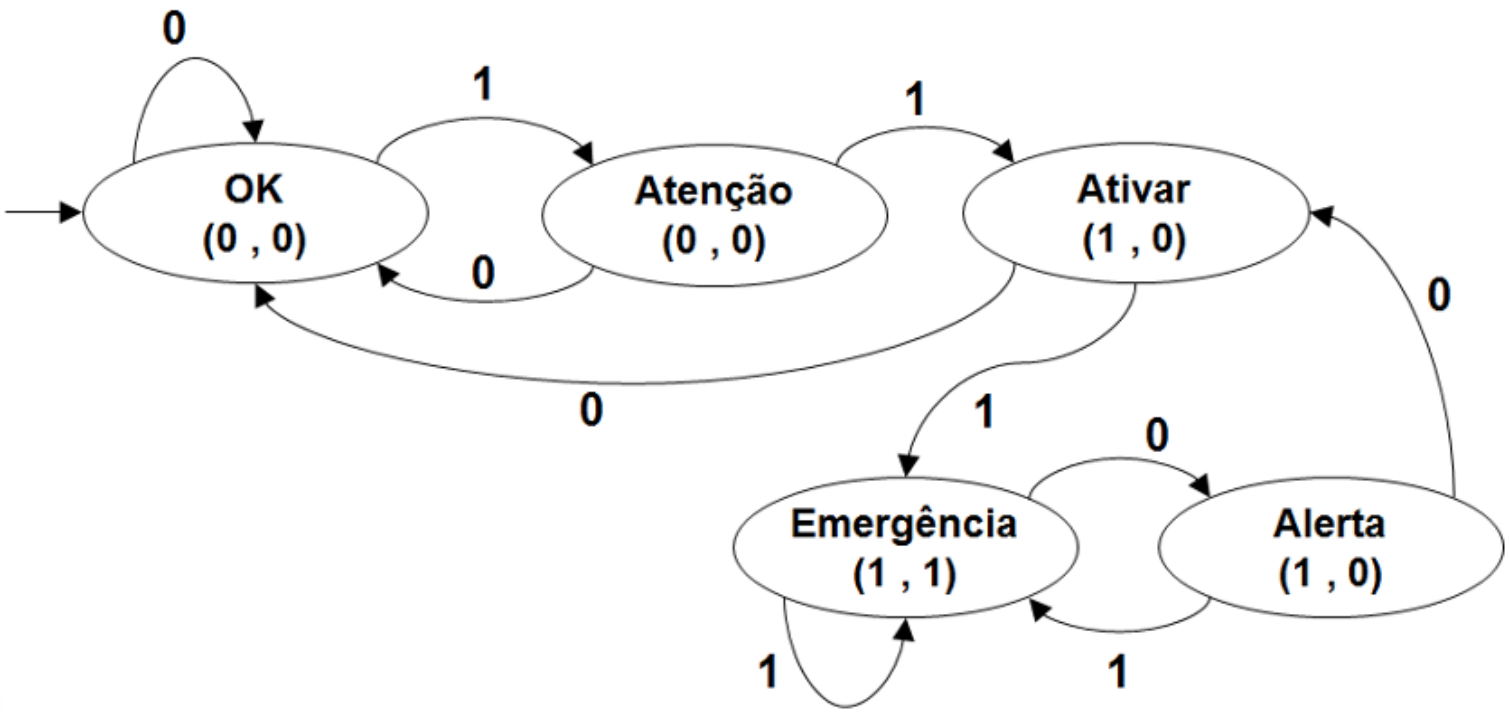
Exercício : controle de reação química

- Projetar módulo de controle de reação química, para evitar temperaturas excessivas, usando o método da descrição em VHDL.
- Especificação:
  - Entrada (bit): X. 1 se temperatura excessiva for detectada.
  - Saída (bits): T e R, comandos para ativar, respectivamente, redução de temperatura e adição de catalisador à reação. Operação: T deve ser ativado após dois 1s consecutivos na entrada X, voltando ao estado inicial sem ativar R caso isso seja suficiente para obter X = 0. Caso contrário, entra-se em um modo de "emergência"em que T e R mantêm-se ambos em 1 enquanto X também for 1. Caso o sistema chegue nesse estado, ele só voltará à normalidade após três leituras consecutivas de X = 0. Enquanto isso não ocorrer, qualquer leitura de X = 0 faz com que R volte a 0, mas não com que T volte a 0; além disso, qualquer leitura de X=1 nesse período também reativa R além de manter ativado T.

Exemplo de entradas e saídas

X: 00101101011<u>1101001010</u>0010110<u>11111011000</u>10
T: 00000100001<u>1111111111</u>0000100<u>1111111111</u>000
R: 00000000000<u>1101001010</u>0000000<u>1111011000</u>00

Máquina de Estado



Obrigado!


Universidade de São Paulo
Departamento de Engenharia de Computação e Sistemas Digitais
PCS

Referências