Módulo 6

Sistemas sequenciais

P1A. Controlador de portão automático: enunciado

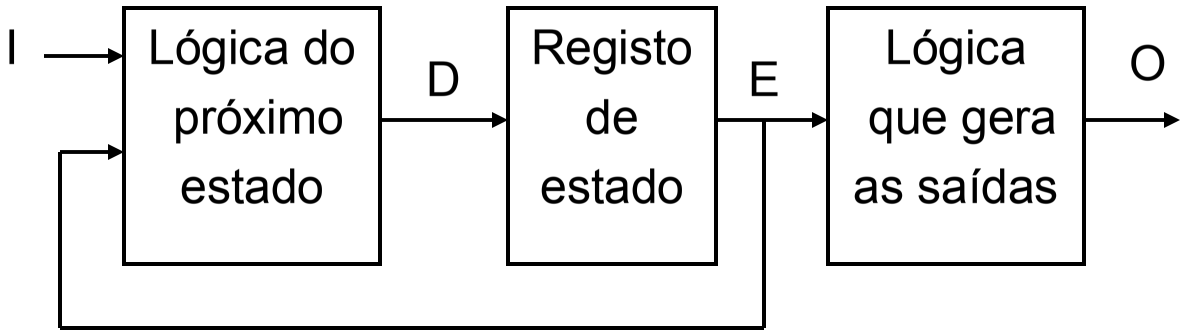
- O portão é deslocado por um motor, controlado pelos sinais ***Ma*** e ***Mf***. Estes sinais nunca poderão estar activos simultaneamente
- Para manobrar o portão existe 1 controlo remoto com um botão ***Ac***.
- Existem 2 sensores de fim de curso (***Ia*** e ***If***) para indicar quando o portão está completamente aberto ou fechado.
- O motor não deverá tentar abrir o portão quando este se encontra aberto, nem fechá-lo quando este se encontra fechado.
- O motor está equipado com uma sirene, controlada pelo sinal ***Av***, que deve estar ligada enquanto o portão se desloca.
- Se o portão estiver fechado e *Ac* for premido o portão começa a abrir. Caso esteja aberto e o *Ac* for activado então começa a fechar.
- Assumir que não é possível o portão parar a meio do movimento. Mas é possível que o utilizador active o sinal *Ac* durante o movimento, o que faz inverter o sentido desse movimento.

P1B. Síntese manual: objectivos

- Pretende-se:
  - desenhar o diagrama de blocos
  - desenhar o diagrama de estados (máquina Moore)
  - construir a tabela de estados
  - atribuir uma combinação binária a cada estado e construir a tabela de verdade
  - construir a tabela de excitação para *flip-flops* D
  - obter as expressões minimizadas para as saídas dos 2 blocos de lógica combinatória, utilizando mapas de *Karnaugh*

P1B. Síntese manual: Moore vs Mealy

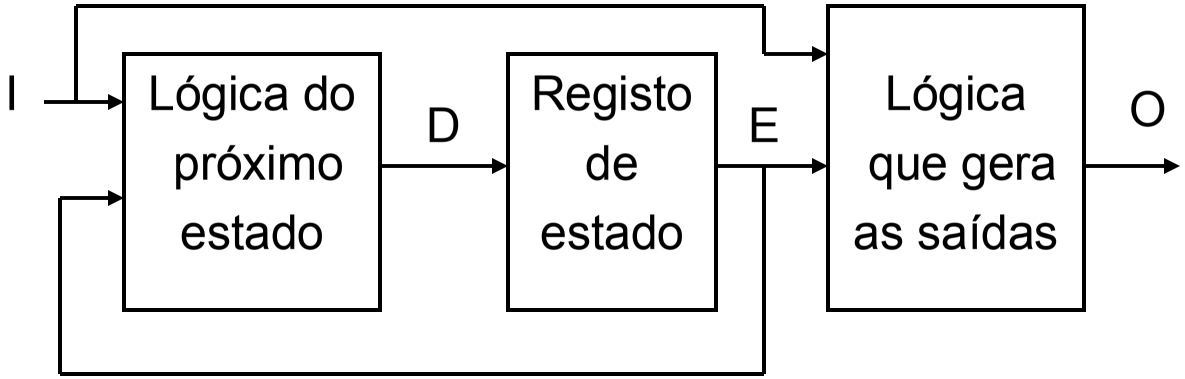
- Máquina de Moore



D = *f1* (I,E)
O = *f2* (E)

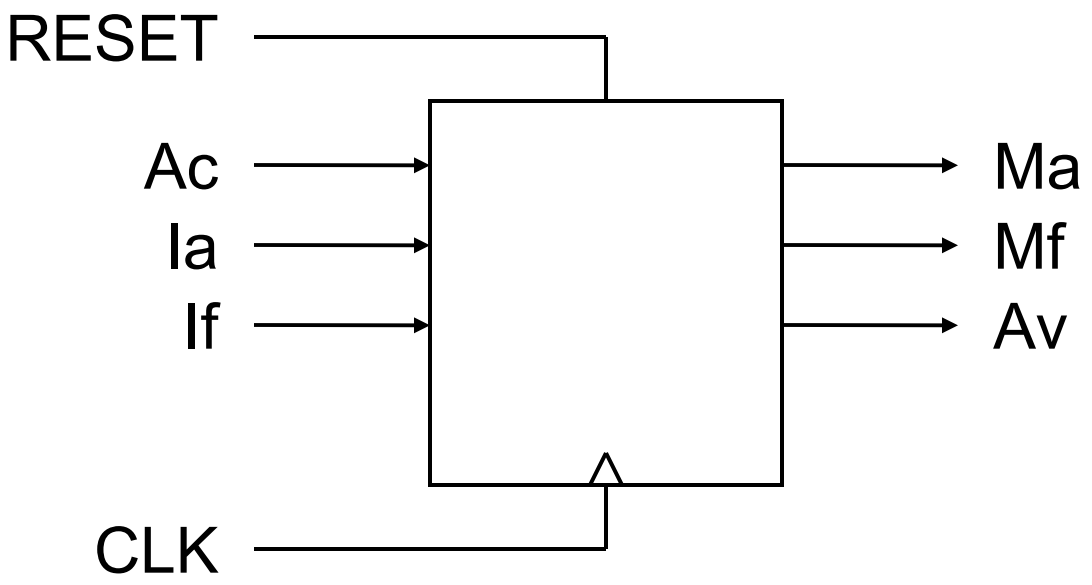
I - entradas
O - saídas
E - estado
D - entrada do registo de estado

- Máquina de Mealy



D = *f3* (I,E)
O = *f4* (I,E)

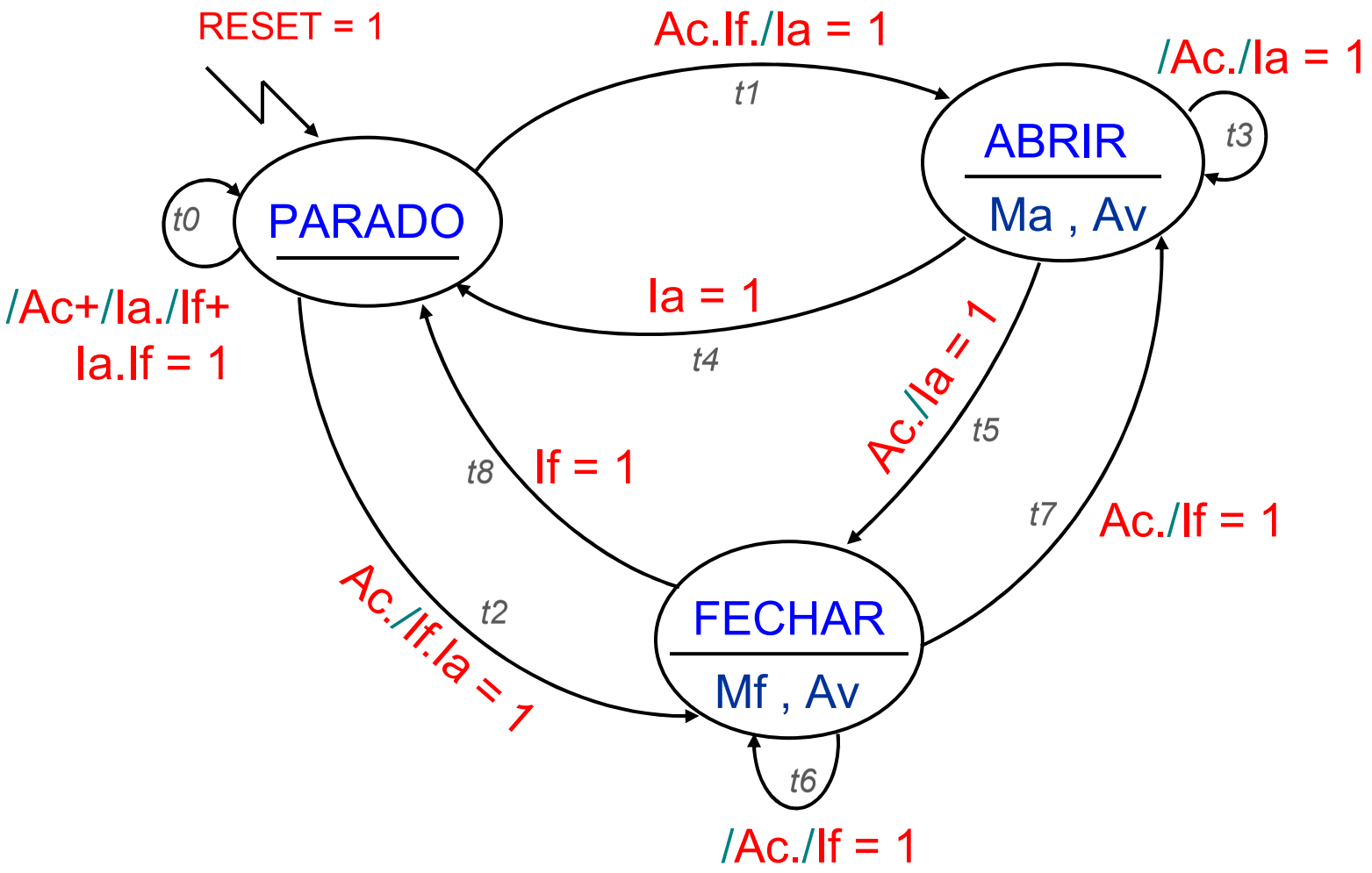
P1B. Síntese manual: diagrama de blocos



ENTRADAS	
Sinal	**Descrição**
Ac	Inicia ou inverte movimento
Ia	Portão completamente aberto
If	Portão completamente fechado

SAÍDAS	
Sinal	**Descrição**
Ma	Motor abre portão
Mf	Motor fecha portão
Av	Liga a sirene

P1B. Síntese manual: diagrama de estados



P1B. Síntese manual: diagrama de estados

Transições do estado **PARADO**

Ac If Ia	Transição
0 0 0	*t0*
0 0 1	*t0*
0 1 0	*t0*
0 1 1	*t0*
1 0 0	*t0*
1 0 1	*t2*
1 1 0	*t1*
1 1 1	*t0*

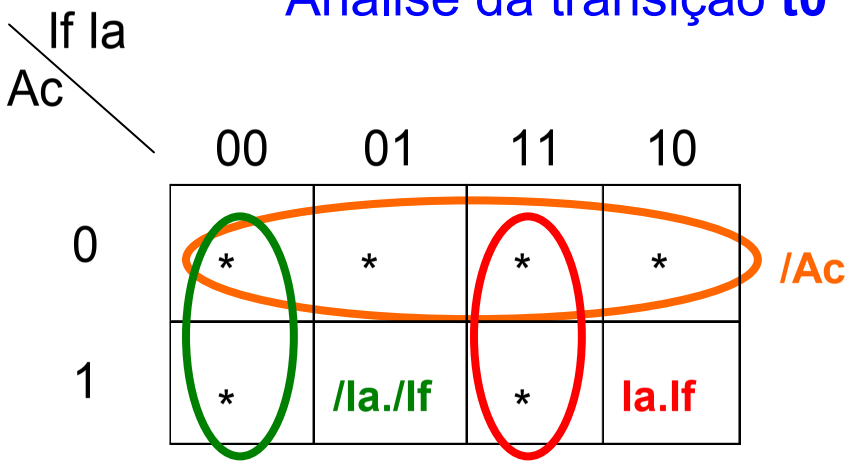
Transições do estado **ABRIR**

Ac Ia	Transição
0 0	*t3*
0 1	*t4*
1 0	*t5*
1 1	*t4*

Transições do estado **FECHAR**

Ac If	Transição
0 0	*t6*
0 1	*t8*
1 0	*t7*
1 1	*t8*

Análise da transição **t0**



P1B. Síntese manual: tabela de estados

Estado actual	Entradas Ac	If	Ia	Próximo estado	Saídas Av	Ma	Mf
PARADO	0	X	X	PARADO	0	0	0
	X	0	0	PARADO			
	X	1	1	PARADO			
	1	0	1	FECHAR			
	1	1	0	ABRIR			
ABRIR	0	X	0	ABRIR	1	1	0
	1	X	0	FECHAR			
	X	X	1	PARADO			
FECHAR	0	0	X	FECHAR	1	0	1
	1	0	X	ABRIR			
	X	1	X	PARADO			

P1B. Síntese manual: tabela de de verdade

- Atribuição duma combinação binária a cada estado

Estado	Código
PARADO	00
ABRIR	01
FECHAR	10

- Considerando *flip-flops D, a entrada a aplicar no registo de estado coincide com o valor do próximo estado porque a equação característica dos flip-flops D é* $Q^* = D$

- Aplicando a informação dos dois itens anteriores à tabela de estados constroi-se a tabela de verdade dos 2 blocos de lógica combinacional da máquina de estados

P1B. Síntese manual: tabela de de verdade

Estado actual E1 E0	Entradas Ac If Ia	Próximo estado E1* E0*	Saídas Av Ma Mf
0 0	0 x x	0 0	0 0 0
	x 0 0	0 0	
	x 1 1	0 0	
	1 0 1	1 0	
	1 1 0	0 1	
0 1	0 x 0	0 1	1 1 0
	1 x 0	1 0	
	x x 1	0 0	
1 0	0 0 x	1 0	1 0 1
	1 0 x	0 1	
	x 1 x	0 0	

P1B. Síntese manual: tabela de excitação para FFs D

Estado actual		Entradas			Saídas da lógica do próximo estado	
E1	E0	Ac	If	Ia	D1	D0
0	0	0	x	x	0	0
		x	0	0	0	0
		x	1	1	0	0
		1	0	1	1	0
		1	1	0	0	1
0	1	0	x	0	0	1
		1	x	0	1	0
		x	x	1	0	0
1	0	0	0	x	1	0
		1	0	x	0	1
		x	1	x	0	0
1	1	x	x	x	x	x

P1B. Síntese manual: tabela de verdade das saídas da FSM

Estado actual E1 E0	Saídas Av Ma Mf
0 0	0 0 0
0 1	1 1 0
1 0	1 0 1
1 1	x x x

P1B. Síntese manual: expressão minimizada para D1

E1 = 0

E0	Ac	Ia=0, If=0	Ia=1, If=0	Ia=1, If=1	Ia=0, If=1
0	0	0	0 (i2)	0	0
0	1	0	1	0	0
1	1	1 (i1)	0	0	1 (i1)
1	0	0	0	0	0

E1 = 1

E0	Ac	Ia=0, If=0	Ia=1, If=0	Ia=1, If=1	Ia=0, If=1
0	0	1 (i3)	1 (i3)	0	0
0	1	0	0	0	0
1	1	X (i1)	X	X	X (i1)
1	0	X (i3)	X (i3)	X	X

D1 = i1 + i2 + i3

D1 = E0.Ac./Ia + /E1./E0.Ac./If.Ia + E1./Ac./If

P1B. Síntese manual: expressão minimizada para D0

E1 = 0

E0 Ac \ Ia If	0 0	0 1	1 1	1 0	
0 0	0	0	0	0	
0 1	0	0	0	1	i2
1 1	0	0	0	0	
1 0	1	0	0	1	i1
	i1				

E1 = 1

E0 Ac \ Ia If	0 0	0 1	1 1	1 0	
0 0	0	0	0	0	
0 1	1 (i3)	1 (i3)	0	0	
1 1	X	X	X	X	
1 0	X	X	X	X	i1
	i1				

D0 = i1 + i2 + i3

D0 = E0./Ac./Ia + /E1./E0.Ac.If./Ia + E1.Ac./If

P1B. Síntese manual: expressão minimizada para Av, Ma e Mf

Av

E1 \ E0	0	1
		i2
0	0	1
1	1	X

i1

Ma

E1 \ E0	0	1
0	0	1
1	0	X

i3

Mf

E1 \ E0	0	1
0	0	0
1	1	X

i4

```
Av =  i1+i2 = E1 + E0

Ma =  i3    = E0

Mf =  i4    = E1
```

P1C. Simulação em VHDL: controlador

```
library IEEE;
use     IEEE.std_logic_1164.all;
use     IEEE.std_logic_arith.all;
use     IEEE.std_logic_unsigned.all;

entity ctl_portao is
  port (
    clk        : in  STD_LOGIC;
    reset      : in  STD_LOGIC;
    Ac         : in  STD_LOGIC;
    I_f        : in  STD_LOGIC;
    Ia         : in  STD_LOGIC;
    Av         : out STD_LOGIC;
    Ma         : out STD_LOGIC;
    Mf         : out STD_LOGIC
    );
end;

architecture funcional of ctl_portao is

  type tipoEstado is
       (parado, abrir, fechar);

  signal estado: tipoEstado;

begin
```

```
-- transições de estado

process (clk, reset)
begin

  if reset='1' then
    estado  <= parado;

  elsif clk'event and clk='1' then

    case estado is

      when parado =>

        if ( Ac='1' and I_f='1' and
             Ia='0' ) then       -- t1
          estado <= abrir ;

        elsif ( Ac='1' and I_f='0'
             and Ia='1' ) then   -- t2
          estado <= fechar ;

        else                     -- t0
          -- ( (Ac='0') or
          -- (I_f='0' and Ia='0')
          -- or (I_f='1' and Ia='1'))
          estado <= parado ;
        end if;
```

(*)

P1C. Simulação em VHDL: controlador

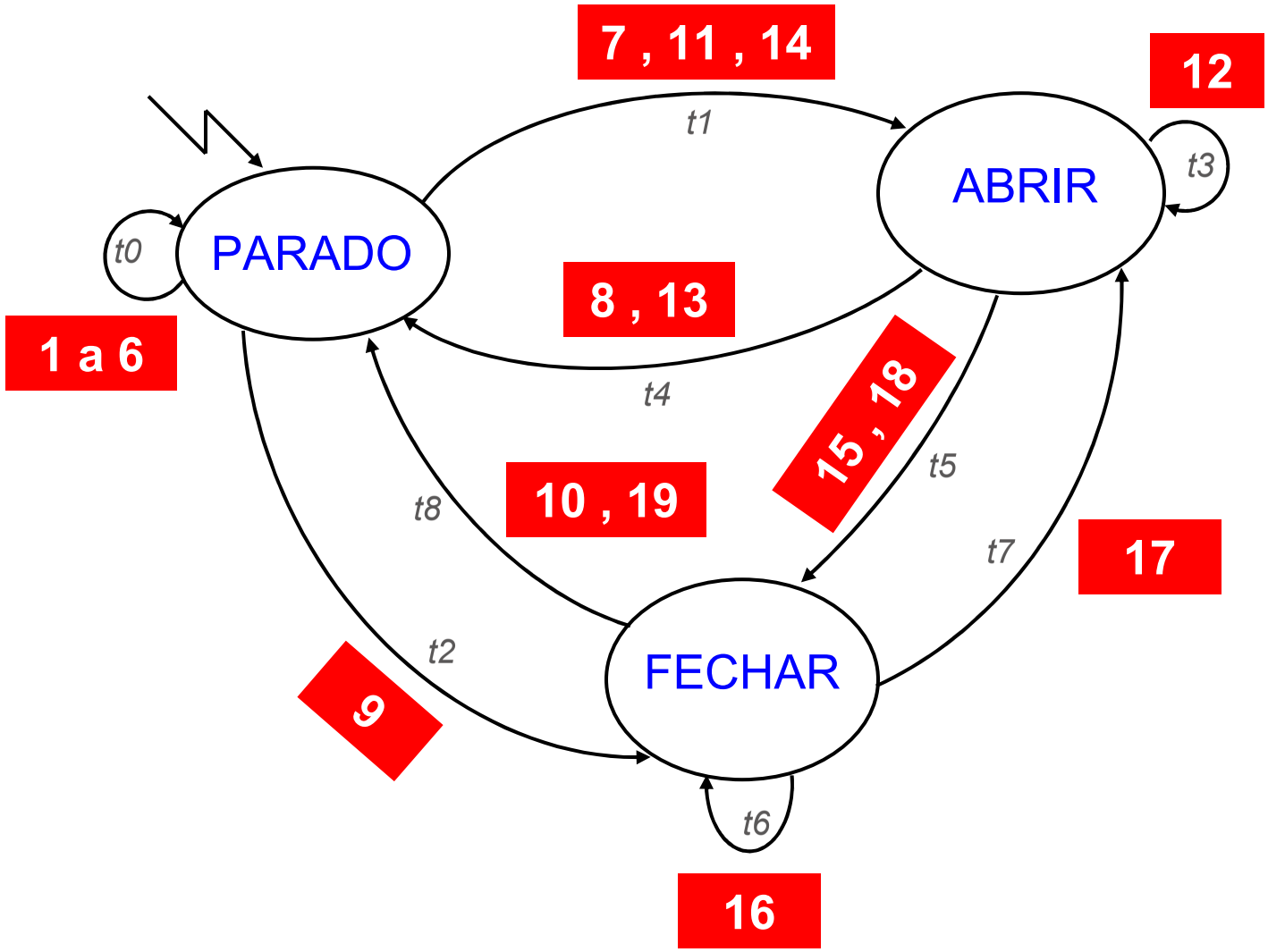
```
when abrir =>

  if ( Ac='0' and Ia='0' ) then          -- t3
    estado <= abrir ;

  elsif ( Ac='1' and Ia='0' ) then       -- t5
    estado <= fechar ;

  else -- ( Ia='1' )                     -- t4
    estado <= parado ;
  end if;

when fechar  =>

  if ( Ac='0' and I_f='0' ) then         -- t6
    estado <= fechar ;

  elsif ( Ac='1' and I_f='0' ) then      -- t7
    estado <= abrir ;

  else -- ( I_f='1' )                    -- t8
    estado <= parado ;
  end if;

 when others =>
        null;
 end case;
end if;
end process;
```

```
  process (estado) -- saídas
  begin

    case estado is
      when parado =>

        Av <= '0' ;
        Ma <= '0' ;
        Mf <= '0' ;

      when abrir =>

        Av   <= '1' ;
        Ma   <= '1' ;
        Mf   <= '0' ;

      when fechar  =>

        Av   <= '1' ;
        Ma   <= '0' ;
        Mf   <= '1' ;

      when others =>
        null;

      end case;
  end process;

end funcional;
```

P1C. Simulação em VHDL: cenários da simulação

Cenário	Estado actual	Ac	If	Ia	Transição que dispara	Próximo estado
1	PARADO	0	0	0	t0	PARADO
2		0	0	1		
3		0	1	0		
4		0	1	1		
5		1	0	0		
6		1	1	1		
7	PARADO	1	1	0	t1	ABRIR
8	ABRIR	1	X	1	t4	PARADO
9	PARADO	1	0	1	t2	FECHAR
10	FECHAR	1	1	X	t8	PARADO
11	PARADO	1	1	0	t1	ABRIR
12	ABRIR	0	X	0	t3	ABRIR
13		0	X	1	t4	PARADO
14	PARADO	1	1	0	t1	ABRIR
15	ABRIR	1	X	0	t5	FECHAR
16	FECHAR	0	0	X	t6	FECHAR
17		1	0	X	t7	ABRIR
18	ABRIR	1	X	0	t5	FECHAR
19	FECHAR	0	1	X	t8	PARADO

P1C. Simulação em VHDL: cenários da simulação



P1C. Simulação em VHDL: *testbench*

```
library IEEE;
use     IEEE.std_logic_1164.all;
use     IEEE.std_logic_arith.all;
use     IEEE.std_logic_unsigned.all;

entity teste_ctl_portao is
end;

architecture arch_ctl_portao of
             teste_ctl_portao is
  component ctl_portao is
    port (
      clk             : in  STD_LOGIC;
      reset           : in  STD_LOGIC;

      Ac              : in  STD_LOGIC;
      I_f             : in  STD_LOGIC;
      Ia              : in  STD_LOGIC;
      Av              : out STD_LOGIC;
      Ma              : out STD_LOGIC;
      Mf              : out STD_LOGIC
      );
  end component;

  signal clk_i       : STD_LOGIC;
  signal reset_i     : STD_LOGIC;
```

```
  signal Ac_i        : STD_LOGIC;
  signal If_i        : STD_LOGIC;
  signal Ia_i        : STD_LOGIC;

  signal Av_o        : STD_LOGIC;
  signal Ma_o        : STD_LOGIC;
  signal Mf_o        : STD_LOGIC;

begin

  -- Activar o reset durante 95ns

  process
  begin
    reset_i <= '1';
    wait for 95 ns;
    reset_i <= '0';
    wait;
  end process;

  -- Gerar sinal relogio com T=20ns
  process
  begin
    clk_i <= '0';
    wait for 10 ns;
    clk_i <= '1';
    wait for 10 ns;
  end process;
```

P1C. Simulação em VHDL: *testbench*

```
-- Gerar padroes nas entradas
-- Ac, Ia e If de modo a testar
-- todas as transicoes

process
begin
  -- ESTADO = parado

  Ac_i <= '0';
  If_i <= '0';
  Ia_i <= '0';
  wait for 120 ns; -- (disparou T0)

  Ac_i <= '0';
  If_i <= '0';
  Ia_i <= '1';
  wait for 20 ns;  -- (disparou T0)

  Ac_i <= '0';
  If_i <= '1';
  Ia_i <= '0';
  wait for 20 ns;  -- (disparou T0)

  Ac_i <= '0';
  If_i <= '1';
  Ia_i <= '1';
  wait for 20 ns;  -- (disparou T0)
```

```
  Ac_i <= '1';
  If_i <= '0';
  Ia_i <= '0';
  wait for 20 ns;  -- (disparou T0)

  Ac_i <= '1';
  If_i <= '1';
  Ia_i <= '1';
  wait for 20 ns;  -- (disparou T0)

  Ac_i <= '1';
  If_i <= '1';
  Ia_i <= '0';
  wait for 20 ns;
  -- ESTADO = abrir (disparou T1)

  Ac_i <= '1';
  If_i <= '-';
  Ia_i <= '1';
  wait for 20 ns;
  -- ESTADO = parado (disparou T4)

  Ac_i <= '1';
  If_i <= '0';
  Ia_i <= '1';
  wait for 20 ns;
  -- ESTADO = fechar (disparou T2)
```

P1C. Simulação em VHDL: *testbench*

```
Ac_i <= '1';
If_i <= '1';
Ia_i <= '-';
wait for 20 ns;
-- ESTADO = parado (disparou T8)

Ac_i <= '1';
If_i <= '1';
Ia_i <= '0';
wait for 20 ns;
-- ESTADO = abrir (disparou T1)

Ac_i <= '0';
If_i <= '-';
Ia_i <= '0';
wait for 20 ns; -- (disparou T3)

Ac_i <= '0';
If_i <= '-';
Ia_i <= '1';
wait for 20 ns;
-- ESTADO = parado (disparou T4)

Ac_i <= '1';
If_i <= '1';
Ia_i <= '0';
wait for 20 ns;
-- ESTADO = abrir (disparou T1)
```

```
Ac_i <= '1';
If_i <= '-';
Ia_i <= '0';
wait for 20 ns;
-- ESTADO = fechar (disparou T5)

Ac_i <= '0';
If_i <= '0';
Ia_i <= '-';
wait for 20 ns; -- (disparou T6)

Ac_i <= '1';
If_i <= '0';
Ia_i <= '-';
wait for 20 ns;
-- ESTADO = abrir (disparou T7)

Ac_i <= '1';
If_i <= '-';
Ia_i <= '0';
wait for 20 ns;
-- ESTADO = fechar (disparou T5)

Ac_i <= '0';
If_i <= '1';
Ia_i <= '-';
wait for 20 ns;
-- ESTADO = parado (disparou T8)
```

P1C. Simulação em VHDL: *testbench*

```
    Ac_i <= '0';
    If_i <= '1';
    Ia_i <= '0'; -- (disparou T0)
    wait;

  end process;

-- Instanciar a entidade
-- controlador de portao

fsm:
  ctl_portao
    port map (
      clk          => clk_i       ,
      reset        => reset_i     ,
      Ac           => Ac_i        ,
      I_f          => If_i        ,
      Ia           => Ia_i        ,

      Av           => Av_o        ,
      Ma           => Ma_o        ,
      Mf           => Mf_o
    );

end arch_ctl_portao ;
```