



Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Graduação em Ciências da Computação



Sistemas Digitais

INE 5406

Aula 8-P

Descrição em VHDL, síntese e simulação de máquinas de estados finitos (FSMs).

Prof. José Luís Güntzel
guntzel@inf.ufsc.br

Est. Vinícius Livramento
vini@inf.ufsc.br

www.inf.ufsc.br/~guntzel/ine5406/ine5406.html

Descrição, Síntese e Simulação de FSMs

Máquinas de Estados

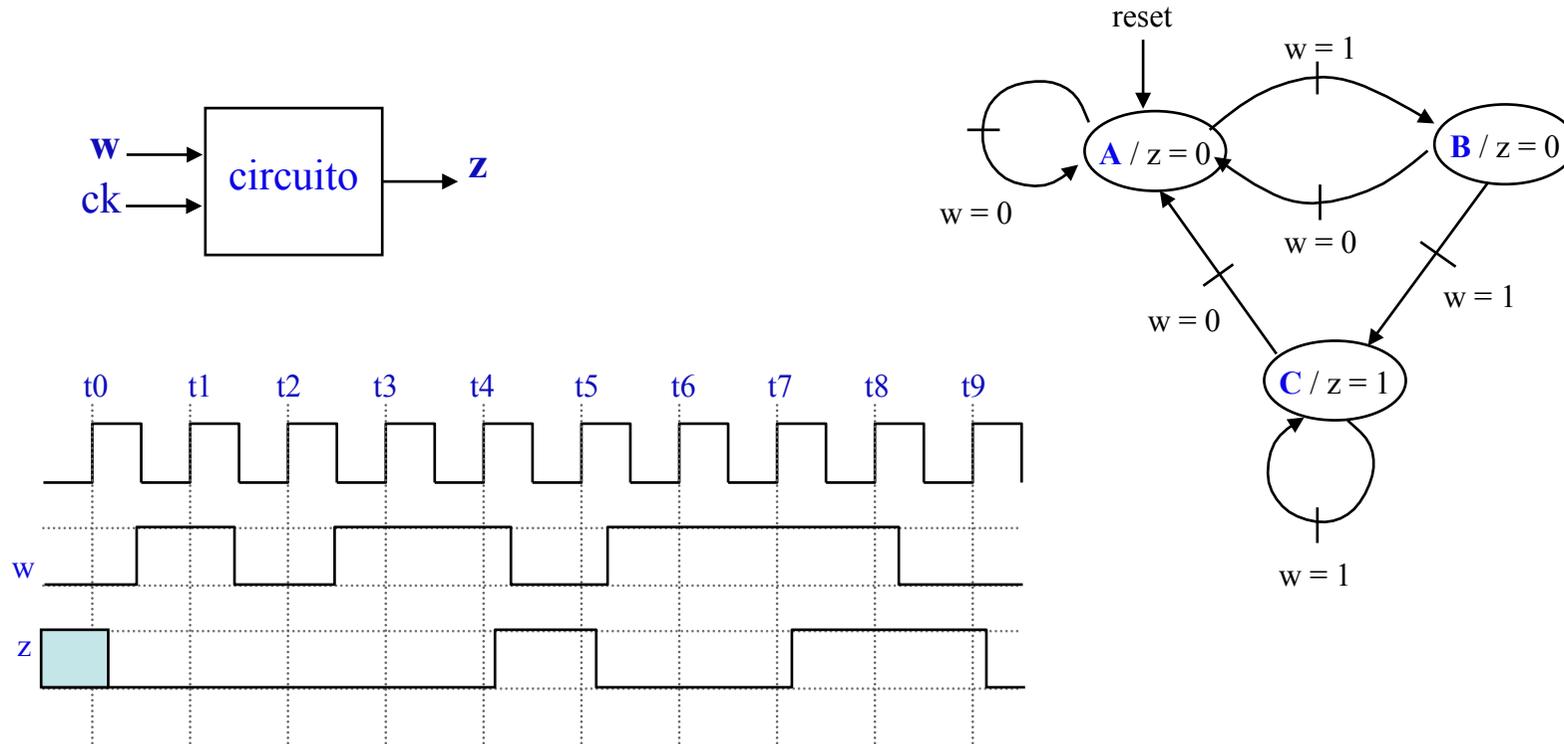
- **VHDL não define padrão para a descrição de máquinas de estados finitos**
- **Existe mais de uma maneira de se descrever uma dada FSM**

Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

Considere a FSM com o seguinte diagrama de estados

- Esta FSM segue o modelo de Moore



Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

```
1 LIBRARY ieee ;
2 USE ieee.std_logic_1164.all ;

3 ENTITY contabits1 IS
4     PORT ( Clock, Reset, w      : IN  STD_LOGIC ;
           z                        : OUT
           STD_LOGIC ) ;
5 END contabits1 ;
```

```
7 ARCHITECTURE Behavior OF contabits1 IS
8     TYPE Tipo_estado IS (A, B, C) ;
9     SIGNAL y : Tipo_estado ;
10 BEGIN
11     PROCESS ( Reset, Clock )
12     BEGIN
13         IF Reset = '1' THEN
14             y <= A ;
15         ELSIF (Clock'EVENT AND Clock = '1') THEN
16             CASE y IS
17                 WHEN A =>
18                     IF w = '0' THEN
19                         y <= A ;
20                     ELSE
21                         y <= B ;
22                     END IF ;
```

FSM descrita segundo o Modelo de Moore, Versão 1 (somente 1 processo)

“TYPE” permite criar um tipo de sinal definido pelo usuário.

Neste caso, se está definindo um dado chamado State_type que pode assumir um entre 3 valores simbólicos: A, B, C

Algumas ferramentas de EDA (p.ex., o Quartus II) assumem que o primeiro estado da lista corresponde ao estado de “reset”

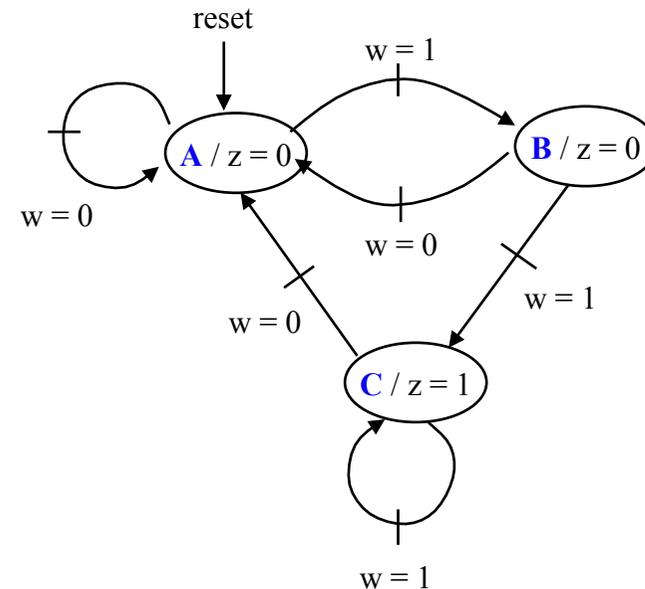
O sinal y representa as saídas dos flip-flops que armazenam os estados desta FSM.

Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

```
15     ELSIF (Clock'EVENT AND Clock = '1') THEN
16         CASE y IS
17             WHEN A =>
18                 IF w = '0' THEN
19                     y <= A ;
20                 ELSE
21                     y <= B ;
22                 END IF ;
23             WHEN B =>
24                 IF w = '0' THEN
25                     y <= A ;
26                 ELSE
27                     y <= C ;
28                 END IF ;
29             WHEN C =>
30                 IF w = '0' THEN
31                     y <= A ;
32                 ELSE
33                     y <= C ;
34                 END IF ;
35         END CASE ;
36     END IF ;
37 END PROCESS ;
38     z <= '1' WHEN y = C ELSE '0' ;
39 END Behavior ;
```

FSM descrita segundo o Modelo de Moore, Versão 1 (continuação)

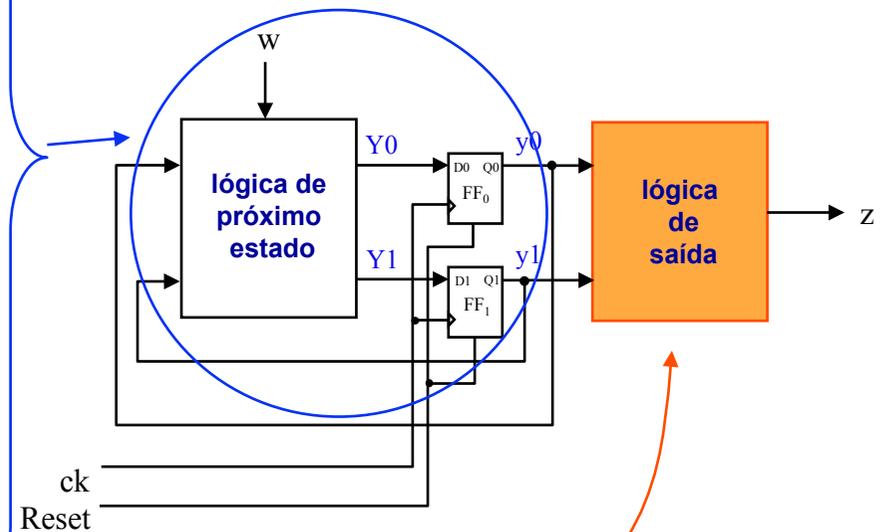


Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

```
15     ELSIF (Clock'EVENT AND Clock = '1') THEN
16         CASE y IS
17             WHEN A =>
18                 IF w = '0' THEN
19                     y <= A ;
20                 ELSE
21                     y <= B ;
22                 END IF ;
23             WHEN B =>
24                 IF w = '0' THEN
25                     y <= A ;
26                 ELSE
27                     y <= C ;
28                 END IF ;
29             WHEN C =>
30                 IF w = '0' THEN
31                     y <= A ;
32                 ELSE
33                     y <= C ;
34                 END IF ;
35         END CASE ;
36     END IF ;
37 END PROCESS ;
38     z <= '1' WHEN y = C ELSE '0' ;
39 END Behavior ;
```

FSM descrita segundo o Modelo de Moore, Versão 1 (continuação)



Descrição, Síntese e Simulação de FSMs

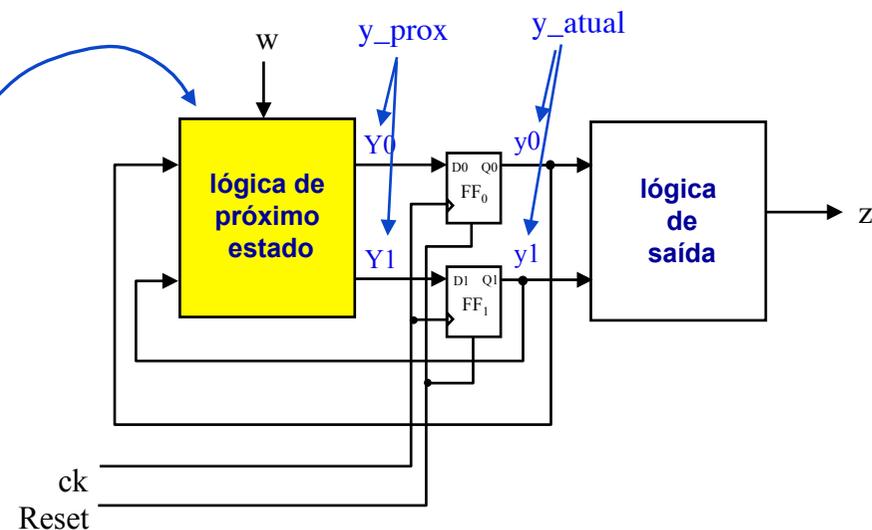
▶ Máquinas de Estados FSM descrita segundo o Modelo de Moore, Versão 2 (2 processos)

```
ARCHITECTURE Behavior OF contabits1 IS
  TYPE Tipo_estado IS (A, B, C);
  SIGNAL y_atual, y_prox : Tipo_estado;
BEGIN
  PROCESS ( w, y_atual )
  BEGIN
    CASE y_atual IS
      WHEN A =>
        IF w = '0' THEN
          y_prox <= A;
        ELSE
          y_prox <= B;
        END IF;
      WHEN B =>
        IF w = '0' THEN
          y_prox <= A;
        ELSE
          y_prox <= C;
        END IF;
      WHEN C =>
        IF w = '0' THEN
          y_prox <= A;
        ELSE
          y_prox <= C;
        END IF;
    END CASE;
  END PROCESS;
```

Em termos da notação que usamos:

$y_atual \rightarrow y$ (estado atual)

$y_prox \rightarrow Y$ (próximo estado)



Slide 8P.7

Prof. José Luís Güntzel
Estagiário Vinícius Livramento

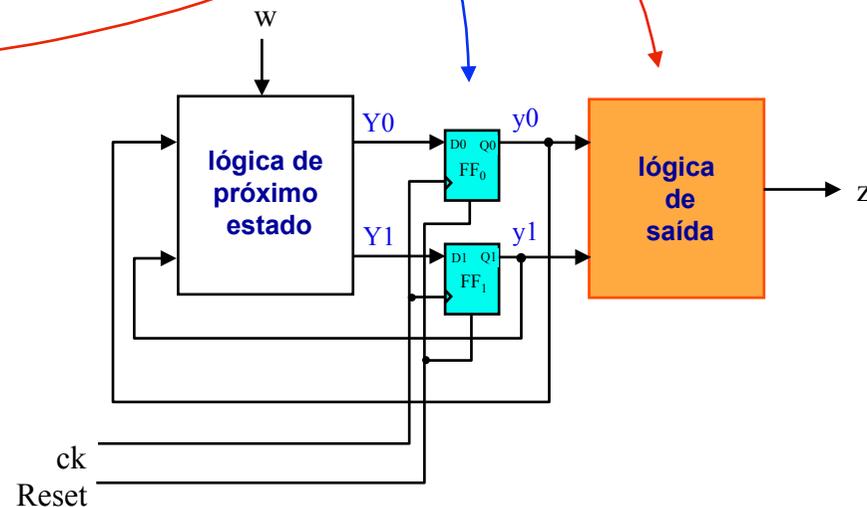
Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

FSM descrita segundo o Modelo de Moore, Versão 2 (2 processos)

```
PROCESS (Clock, Reset)
BEGIN
  IF Reset = '1' THEN
    y_atual <= A ;
  ELSIF (Clock'EVENT AND Clock = '1') THEN
    y_atual <= y_prox ;
  END IF ;
END PROCESS ;

z <= '1' WHEN y_atual = C ELSE '0' ;
END Behavior ;
```



Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

Codificação Manual de Estados

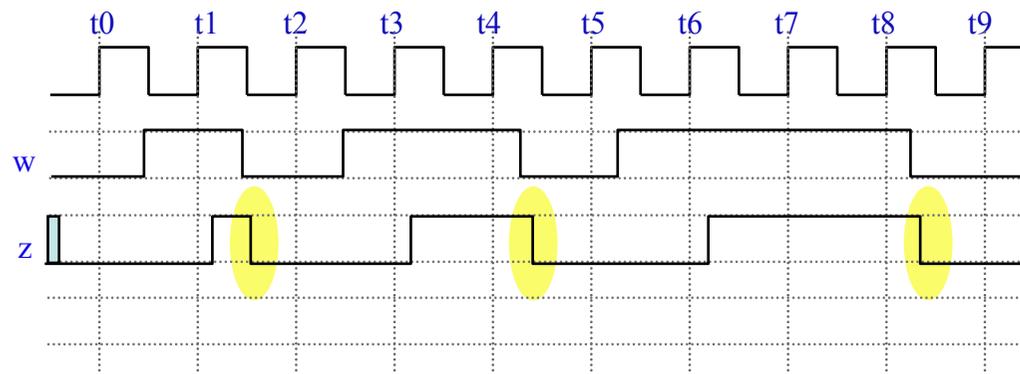
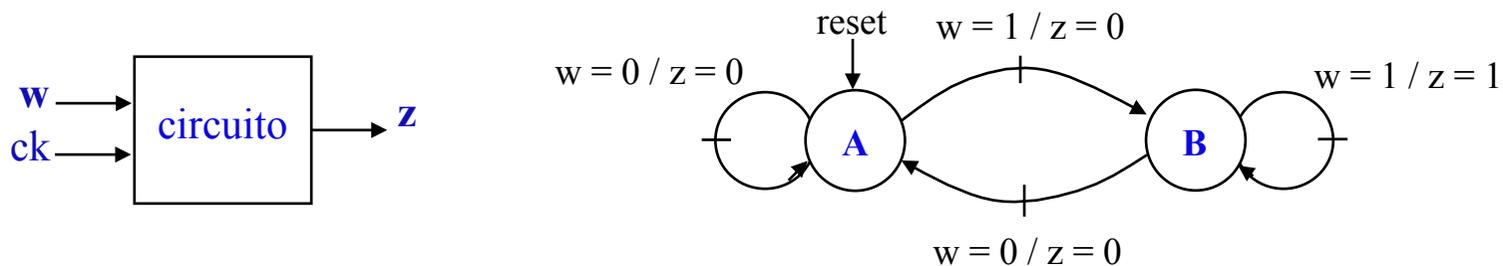
```
ARCHITECTURE Behavior OF contabits1 IS
  TYPE Tipo_estado IS (A, B, C) ;
  ATTRIBUTE ENUM_ENCODING:      STRING;
  ATTRIBUTE ENUM_ENCODING OF Tipo_estado: TYPE IS "00 01 10";
  SIGNAL y_atual, y_prox :      Tipo_estado ;
```

...

Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

- Considere a FSM descrita pelo diagrama de estados que segue, a qual segue o **modelo de Mealy**



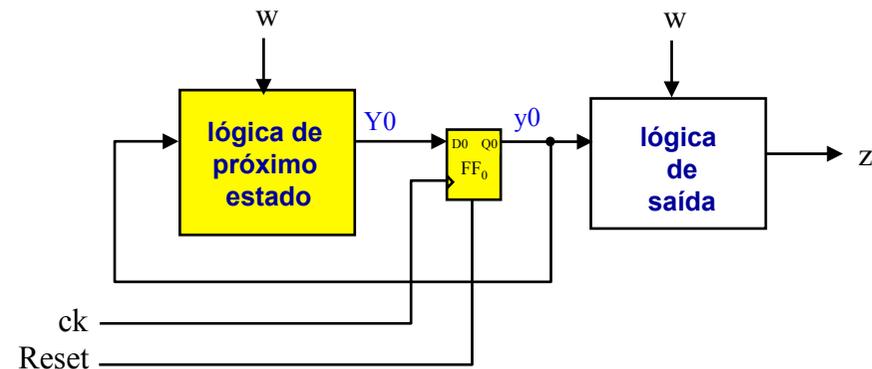
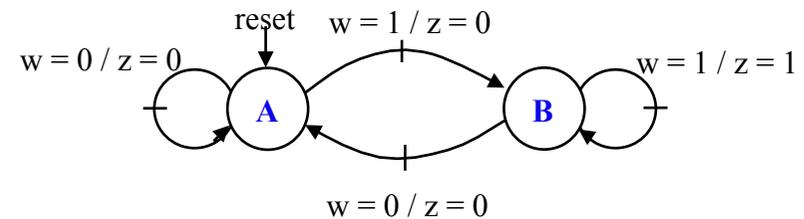
Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY contabits2 IS
    PORT ( Clock, Reset, w : IN  STD_LOGIC ;
          z : OUT  STD_LOGIC ) ;
END contabits2 ;
```

```
ARCHITECTURE Behavior OF contabits2 IS
    TYPE Tipo_estado IS (A, B) ;
    SIGNAL y : Tipo_estado ;
BEGIN
    PROCESS ( Reset, Clock )
    BEGIN
        IF Reset = '1' THEN
            y <= A ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            CASE y IS
                WHEN A =>
                    IF w = '0' THEN y <= A ;
                    ELSE y <= B ;
                    END IF ;
                WHEN B =>
                    IF w = '0' THEN y <= A ;
                    ELSE y <= B ;
                    END IF ;
            END CASE ;
        END IF ;
    END PROCESS ;
END contabits2 ;
```

FSM descrita segundo o Modelo de Mealy (2 processos)



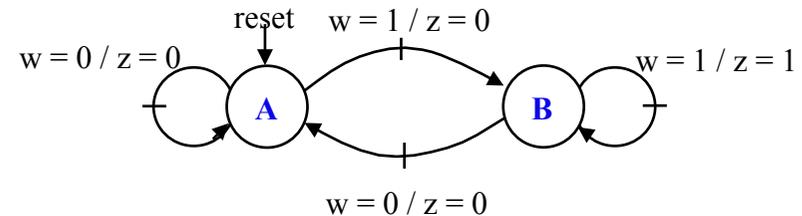
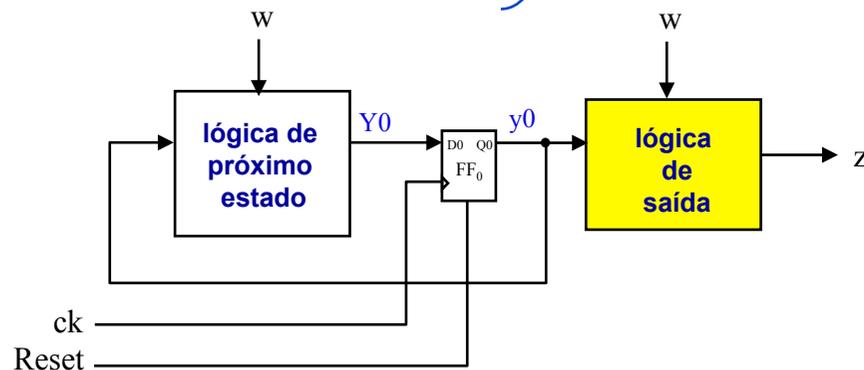
Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

FSM descrita segundo o
Modelo de Mealy
(2 processos) Continuação

```
PROCESS ( y, w )  
BEGIN  
  CASE y IS  
    WHEN A =>  
      z <= '0' ;  
    WHEN B =>  
      z <= w ;  
  END CASE ;  
END PROCESS ;  
END Behavior ;
```

A saída “z” precisa estar em um processo diferente porque suas mudanças de valor não estão sujeitas ao relógio (pois trata-se de uma FSM de Mealy)



Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados

Outro Exemplo de FSM de Mealy

```
Use ieee.std_logic_1164.all;
```

```
ENTITY Cont IS
```

```
  PORT(clk, reset, flag1 : IN STD_LOGIC;  
        S1, S2, S3 : OUT STD_LOGIC);
```

```
END Cont;
```

```
ARCHITECTURE comportamento OF Cont IS
```

```
SIGNAL estado: STD_LOGIC_VECTOR(1 DOWNTO 0);
```

```
BEGIN
```

```
  PROCESS (clk, reset)
```

```
  BEGIN
```

```
    IF (reset='0') THEN  
      estado <= "00";
```

← Definição das
variáveis de
estado

Descrição, Síntese e Simulação de FSMs

▶ Máquinas de Estados (continuação)

Outro Exemplo de FSM de Mealy

```
ELSIF (clk'EVENT AND clk = '1') THEN
  CASE estado IS
    WHEN "00" => estado <= "01"; S1 <= '0'; S2 <= '1'; S3 <= '1';
    WHEN "01" => IF flag1 = '1' THEN estado <= "10";
                  ELSE estado <= "11"; END IF;
                  S1 <= '1'; S2 <= '1'; S3 <= '0';
    WHEN "10" => estado <= "11"; S1 <= '1'; S2 <= '1'; S3 <= '1';
    WHEN "11" => estado <= "00"; S1 <= '1'; S2 <= '0'; S3 <= '1';
    WHEN OTHERS => estado <= "00";
  END CASE;
END IF;
END PROCESS;
END comportamento;
```

Descrição, Síntese e Simulação de FSMs

Experimento

Especificação do árbitro de barramento (FSM de Moore)

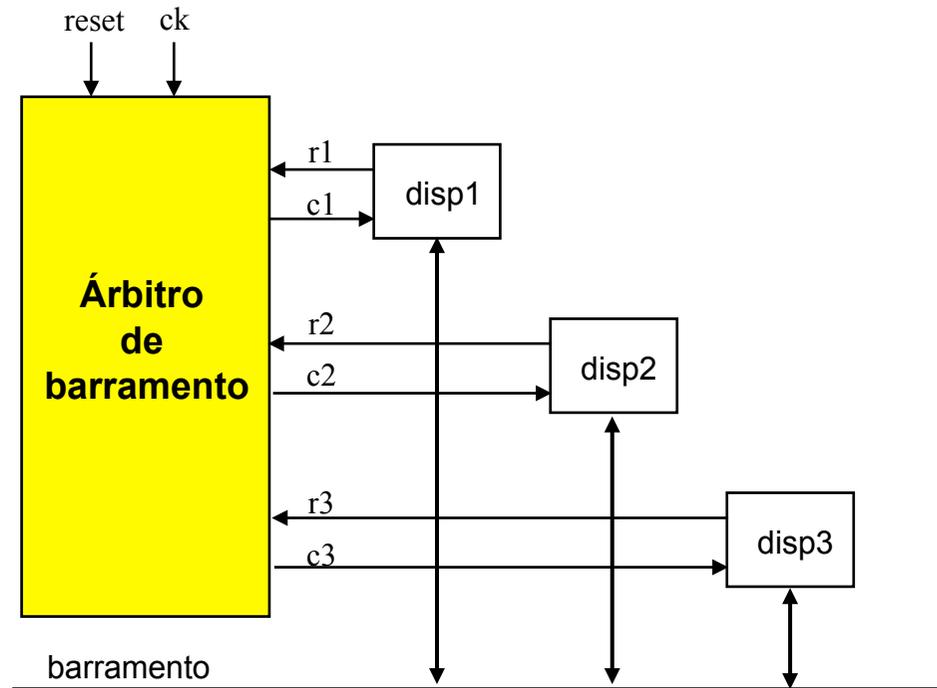
Este árbitro recebe requisições para uso do barramento (r_1, r_2, r_3) provenientes de três dispositivos de entrada/saída ($disp_1, disp_2, disp_3$, respectivamente), sendo que $disp_1$ possui a maior prioridade no uso do barramento (e $disp_3$ possui a menor prioridade). Somente um dispositivo por vez pode receber a concessão do barramento. Para sinalizar qual dispositivo pode usar o barramento, o árbitro faz o respectivo sinal de concessão valer “1” (por exemplo, $c_1=1$ sinaliza que $disp_1$ pode usar o barramento). Uma vez que um dispositivo recebe a concessão para usar o barramento, ele permanece com esta concessão durante todo o tempo que ele necessitar usar o barramento. Para tanto, o dispositivo mantém seu sinal de requisição no valor “1”. O estado “BD” significa barramento disponível. D1 significa barramento concedido ao dispositivo 1 (e assim por diante).

Descrição, Síntese e Simulação de FSMs

► Experimento

Especificação do árbitro de barramento

Diagrama de Blocos do Sistema (contextualização)



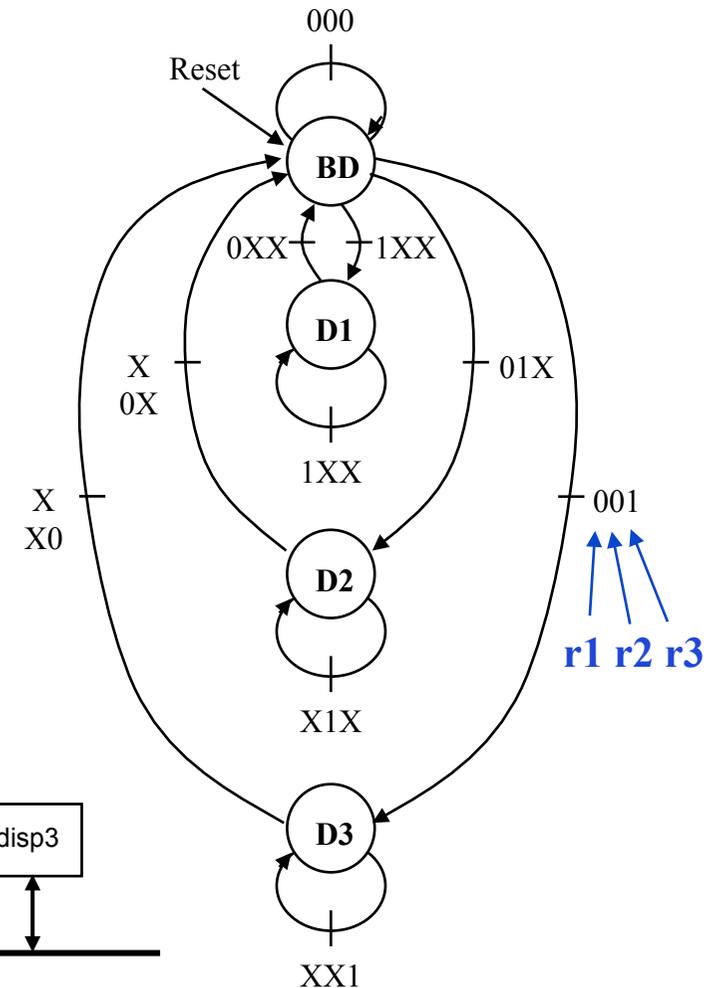
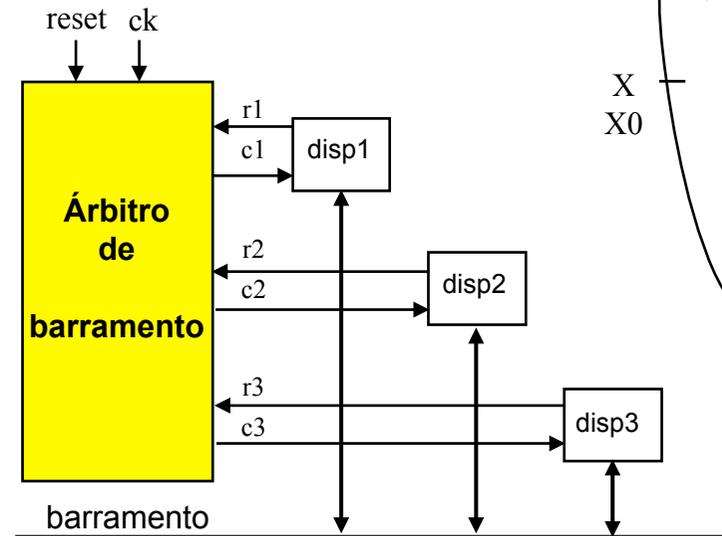
Descrição, Síntese e Simulação de FSMs

Experimento

Especificação do árbitro de barramento
Tabela de Saídas e Diagrama de Estados
(FSM de Moore)

saídas

estado	c1	c2	c3
BD	0	0	0
D1	1	0	0
D2	0	1	0
D3	0	0	1



Descrição, Síntese e Simulação de FSMs

Experimento

1. Na pasta Meus_documentos, criar uma pasta com o seu nome (p. ex., “Paulo”). Na pasta “Paulo”, criar uma pasta com nome de “arbitro”.
2. Acessar o sítio “www.inf.ufsc.br/~guntzel/ine5406/aula8P” e baixar para a pasta o arquivo “arbitro.vhd”
3. Abrir o Quartus II e criar um projeto na pasta “[arbitro](#)”, selecionando “[arbitro.vhd](#)” como toplevel. Escolher o dispositivo FPGA EP2C35F672C6 e selecionar o ModelSim-Altera como EDA Simulation Tool.
4. Completar o arquivo “[arbitro.vhd](#)”. Observe que esta versão utiliza um tipo especial, definido pelo usuário (“State_type”) para as variáveis de estado.
5. Compilar o projeto criado.

Descrição, Síntese e Simulação de FSMs

Experimento

6. A partir do Quartus II, chame o ModelSim-Altera e inicie uma simulação com atrasos (“Gate-level Simulation”).
7. Crie um arquivo de estímulos, nomeando-o “estimulos.do”. Preparar os estímulos de modo a simular a seguinte situação:
 - O sinal de relógio deve ter período de 20ns, sendo que a primeira borda de subida ocorre em 10ns.
 - O sinal Reset deve valer “1” entre 0ns e 20ns.
 - Após os 20ns iniciais, cada dispositivo usa o barramento durante 2 ciclos de relógio seguidos (i.e., 2 bordas de subida seguidas), na seguinte ordem: dips1, disp2, disp3.
 - A simulação termina quando o disp3 tiver liberado o barramento, e conseqüentemente, a máquina de estados tiver retornado ao estado BD.

Descrição, Síntese e Simulação de FSMs

Experimento para Casa

Refaça o experimento anterior adotando a seguinte codificação de estados:
BD=00, D1=01, D2=10, D3=11 (conforme slide 7P.9)