IC-UNICAMP

MC 613

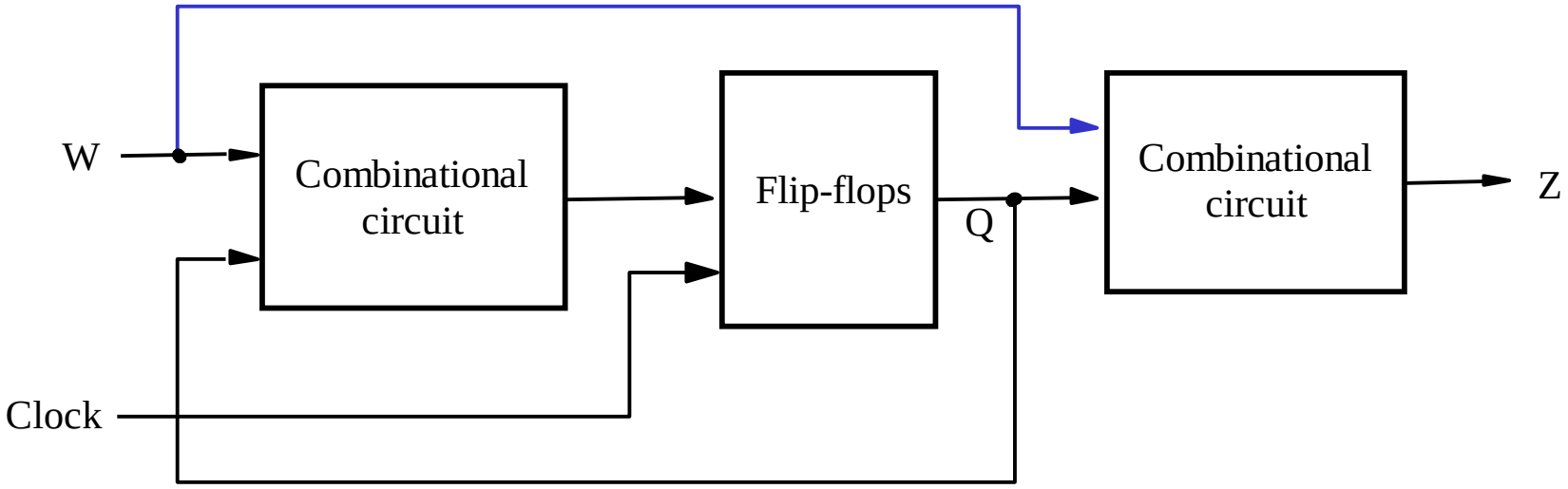
IC/Unicamp

Prof Guido Araújo
Prof Mario Côrtes
Prof Sandro Rigo

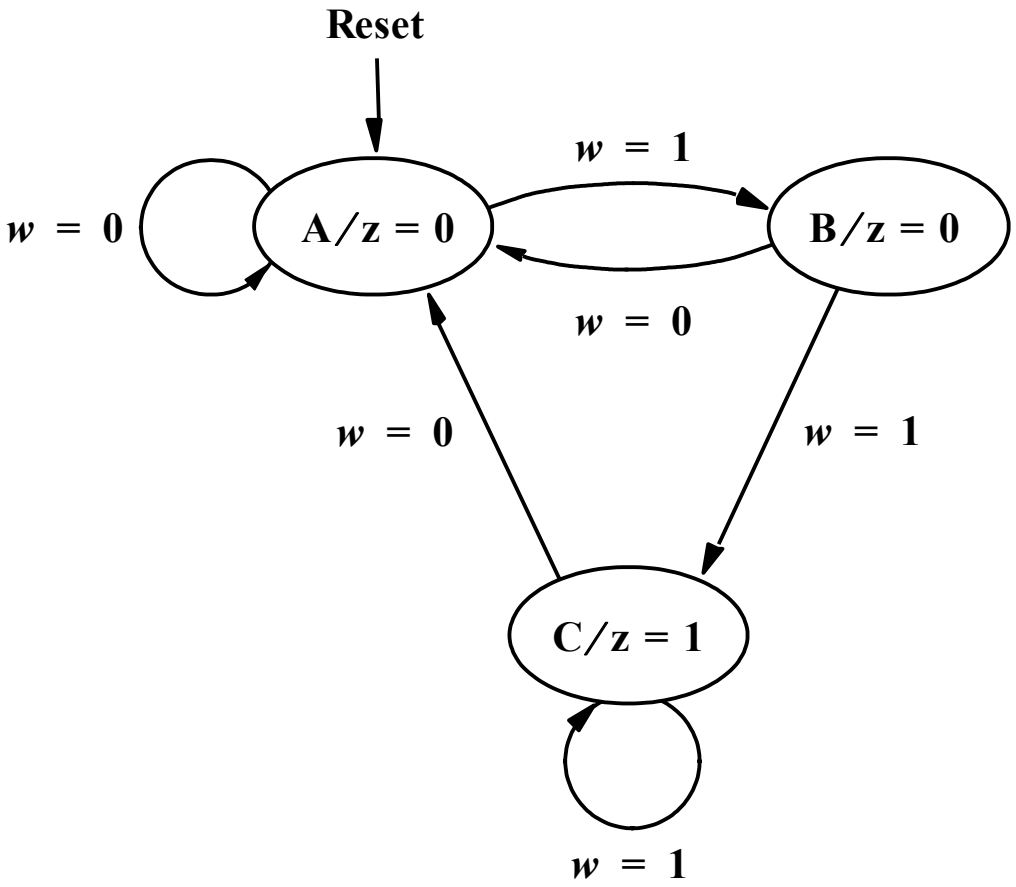
Máquinas de Estado

IC-UNICAMP

Forma geral de um circuito síncrono

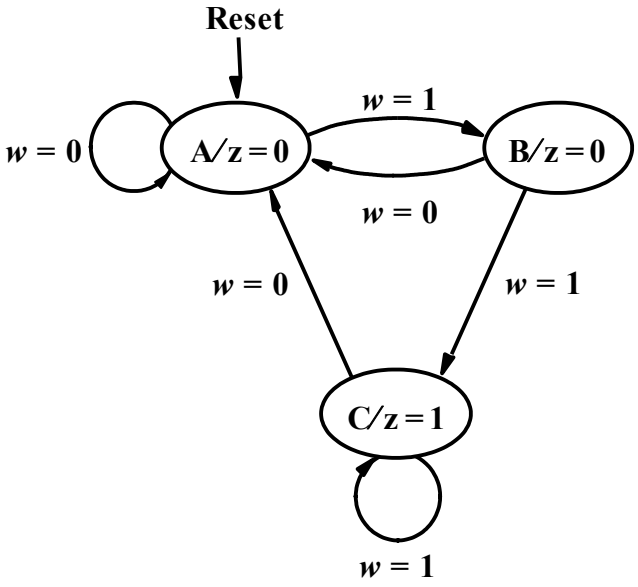


Máquina de Moore



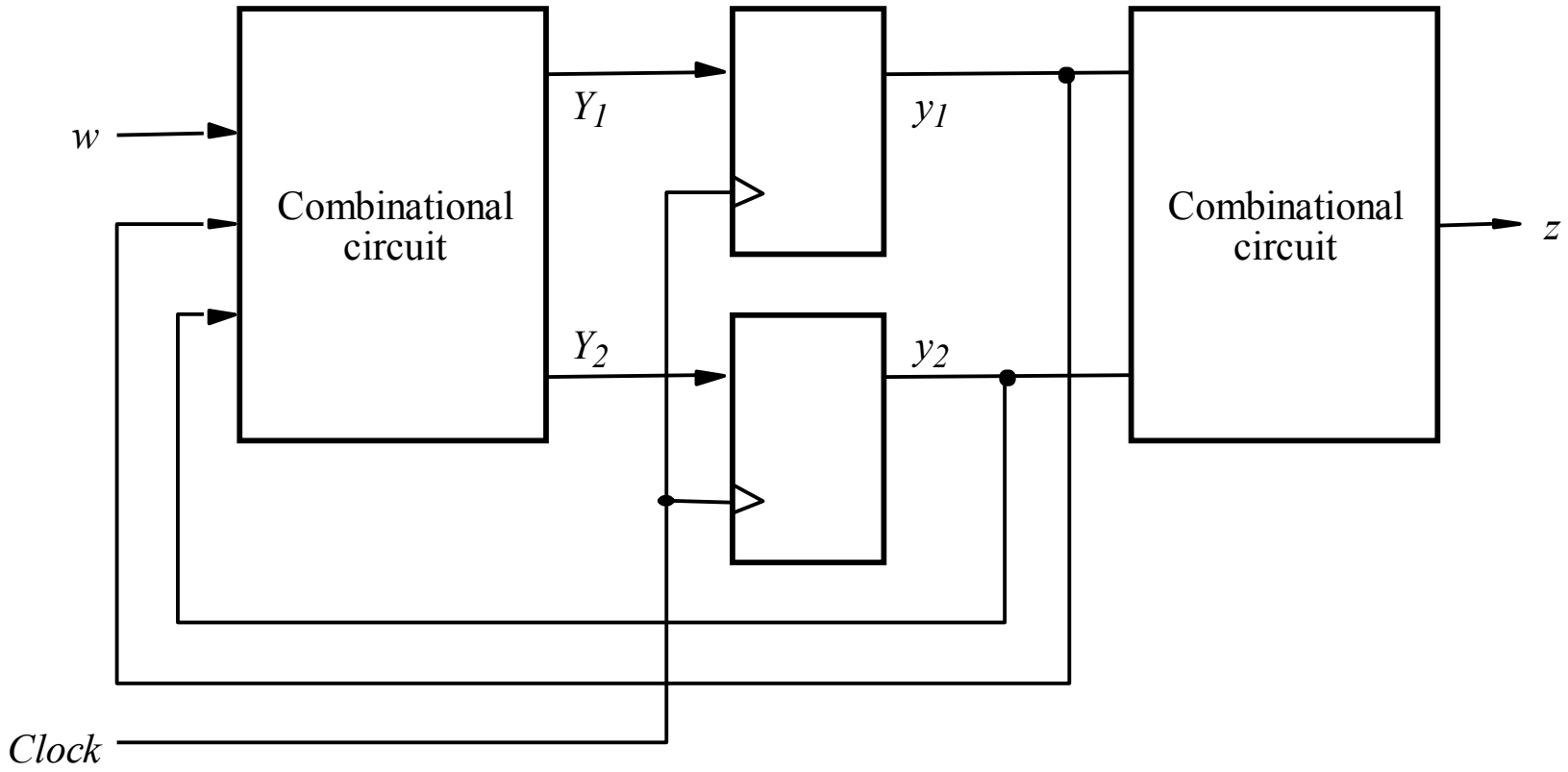
Clockcycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

Diagrama de Estados

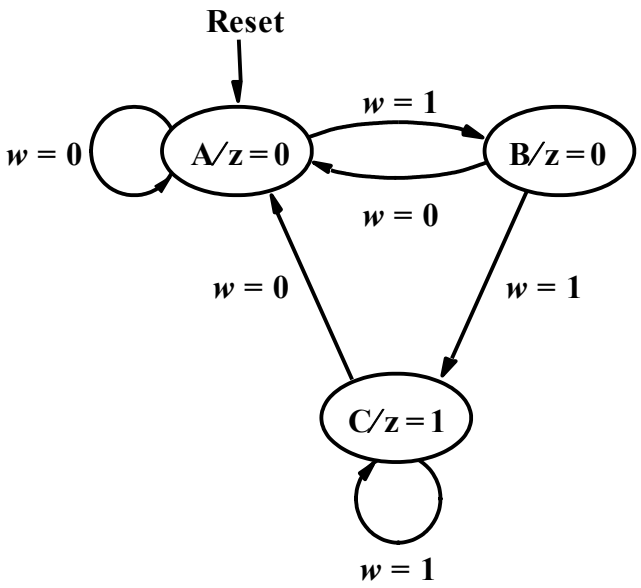


Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	C	1

Implementação

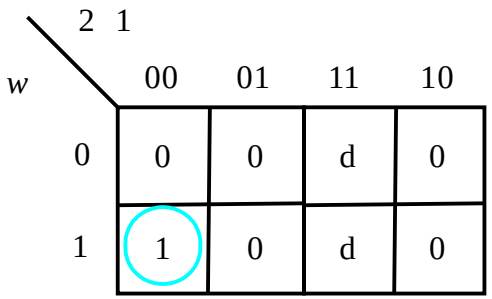


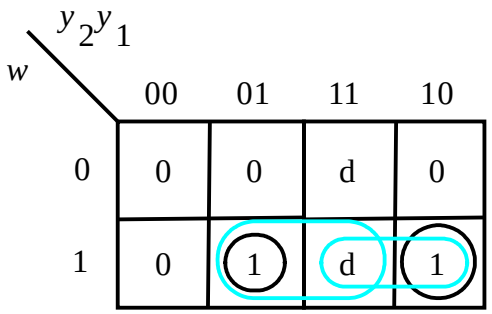
Atribuição de Estado

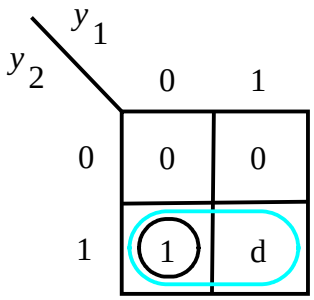


	Present state y_2y_1	Next state $w = 0$ Y_2Y_1	Next state $w = 1$ Y_2Y_1	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Derivação das expressões lógicas







Ignoring don't cares

$Y_1 = w\bar{y}_1\bar{y}_2$

$Y_2 = wy_1\bar{y}_2 + w\bar{y}_1y_2$

$z = \bar{y}_1y_2$

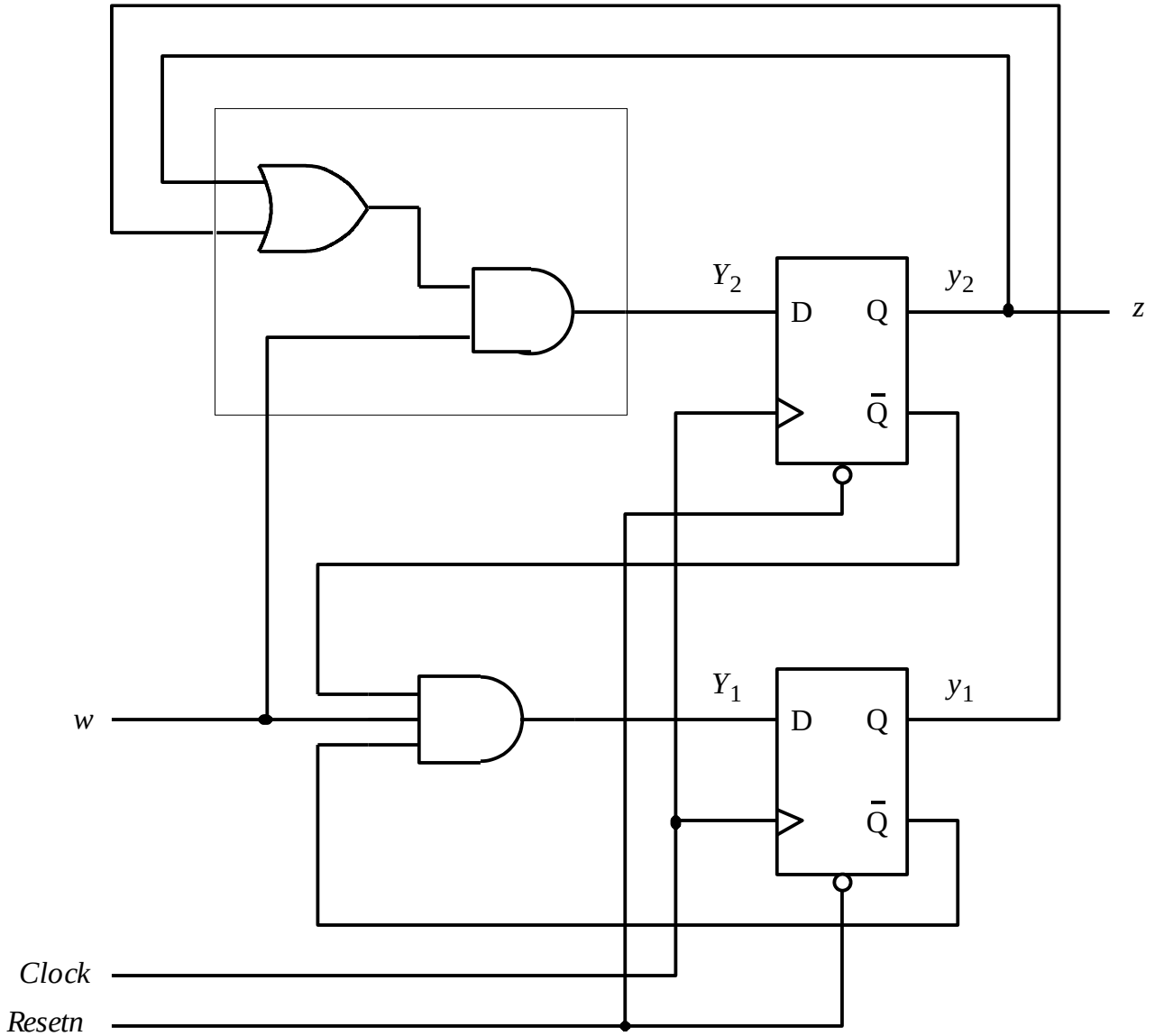
Using don't cares

$Y_1 = w\bar{y}_1\bar{y}_2$

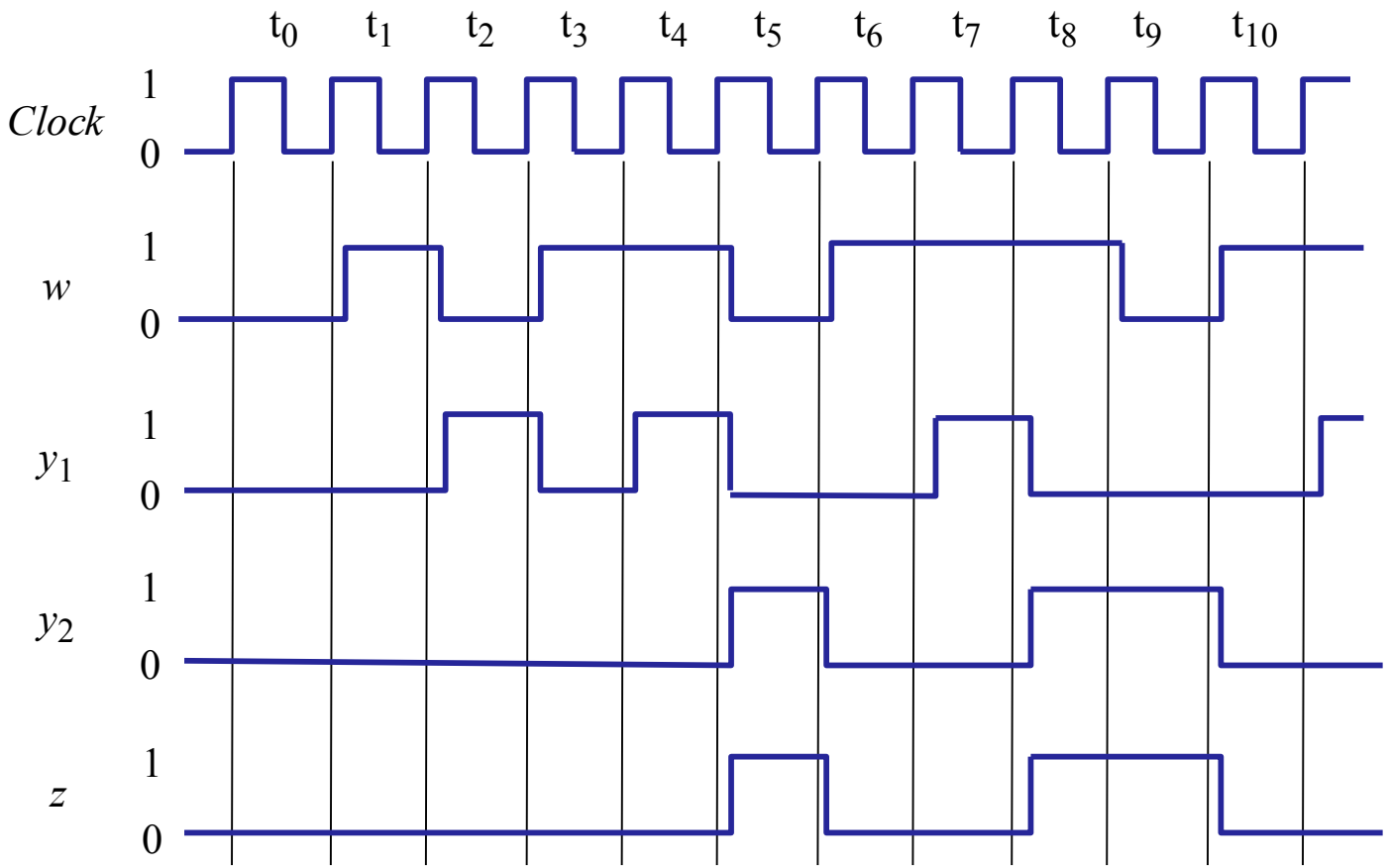
$Y_2 = wy_1 + wy_2$
$= w(y_1 + y_2)$

$z = y_2$

Circuito sequencial



Timing diagram



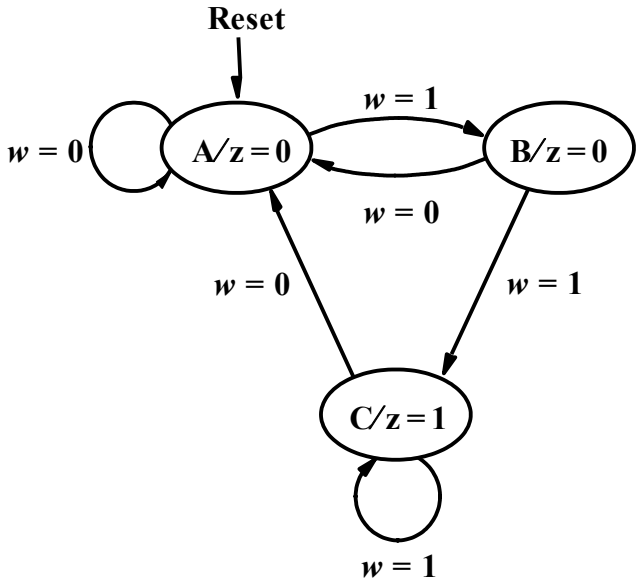
FSM de Moore

```
USE ieee.std_logic_1164.all;

ENTITY simple IS
   PORT (Clock, Resetn, w : IN     STD_LOGIC;
         z                : OUT    STD_LOGIC );
END simple;

ARCHITECTURE Behavior OF simple IS
   TYPE State_type IS (A, B, C); -- Tipo Enumerado para
                                 -- definir os  Estados
   SIGNAL y : State_type;
BEGIN
   PROCESS ( Resetn, Clock )
   BEGIN
       IF Resetn = '0' THEN  -- A é o estado inicial
             y <= A;
       ELSIF (Clock'EVENT AND Clock = '1') THEN

con't ...
```

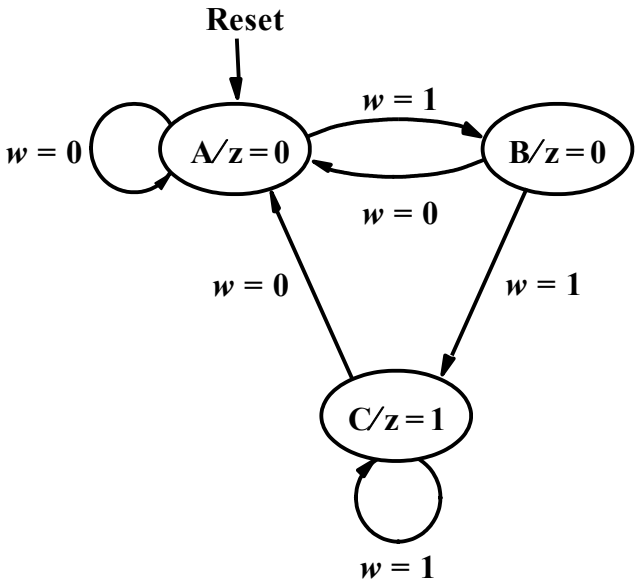
FSM de Moore

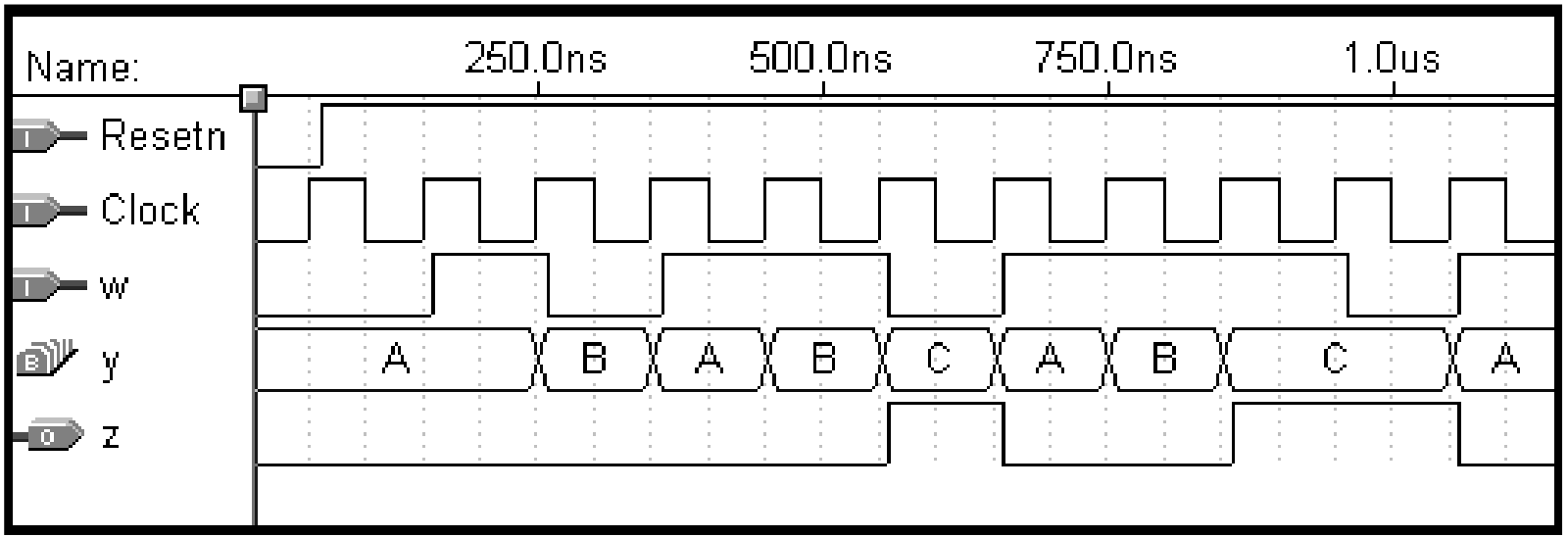
IC-UNICAMP

```
        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 <= C;
                END IF;
            WHEN C =>
                IF w = '0'
                    THEN   y <= A;
                    ELSE   y <= C;
                END IF;
        END CASE;
    END IF;
END PROCESS;
    z <= '1' WHEN y = C ELSE '0';
END Behavior;
```

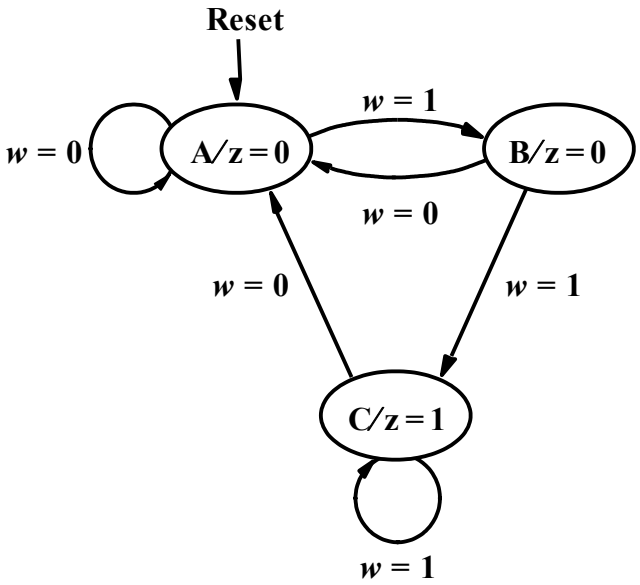


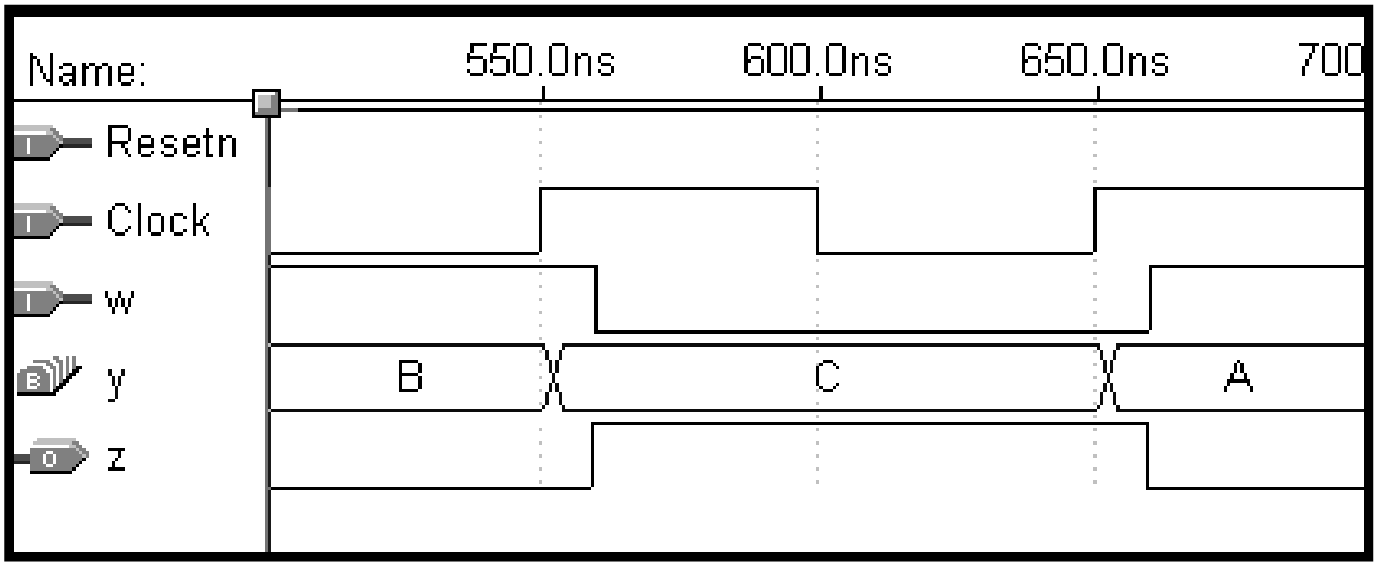
FSM de Moore - Simulação





FSM de Moore - Simulação





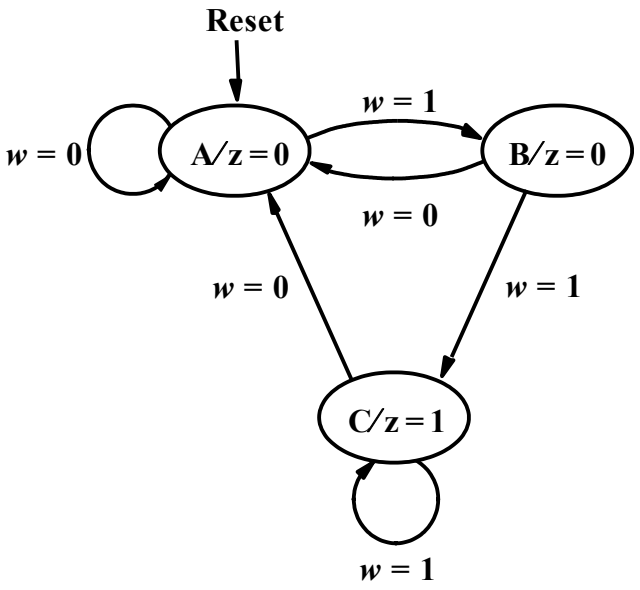
FSM de Moore
Codificação Alternativa (2 processos)

IC-UNICAMP

```
USE ieee.std_logic_1164.all;

ENTITY simple IS
   PORT (Clock, Resetn, w : IN     STD_LOGIC;
                 z        : OUT    STD_LOGIC );
END simple;

ARCHITECTURE Behavior OF simple IS
   TYPE State_type IS (A, B, C);

   SIGNAL y_present, y_next : State_type;
```

FSM de Moore
IC-UNICAMP Codificação Alternativa (2 processos)

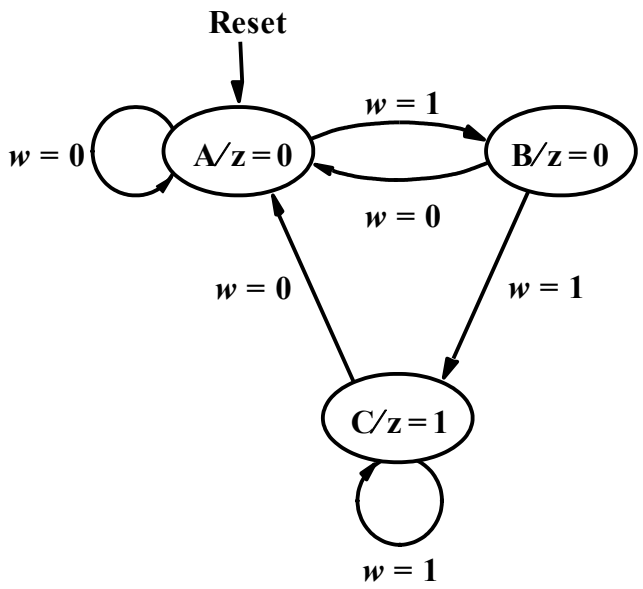
```
BEGIN
   PROCESS ( w, y_present )
   BEGIN
       CASE y_present IS
             WHEN A =>
                   IF w = '0' THEN
                       y_next <= A;
                   ELSE
                       y_next <= B;
                   END IF;
             WHEN B =>
                   IF w = '0' THEN
                       y_next <= A;
                   ELSE
                       y_next <= C;
                   END IF;
```



FSM de Moore - Codificação Alternativa

IC-UNICAMP

```
            WHEN C =>
                  IF w = '0' THEN
                      y_next <= A;
                  ELSE
                      y_next <= C;
                  END IF;
      END CASE;
   END PROCESS;

   PROCESS (Clock, Resetn)
   BEGIN
      IF Resetn = '0' THEN
            y_present <= A;
      ELSIF (Clock'EVENT AND Clock = '1') THEN
            y_present <= y_next;
      END IF;
   END PROCESS;

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



FSM - Especificando a Atribuição de Estados

IC-UNICAMP

```
ARCHITECTURE Behavior OF simple IS
   TYPE State_TYPE IS (A, B, C);
   ATTRIBUTE ENUM_ENCODING : STRING;
   ATTRIBUTE ENUM_ENCODING OF State_type: TYPE IS "00 01
   11";

   SIGNAL y_present, y_next : State_type;
  BEGIN

    con't ...
```

- Obs: Atributo Enum_Encoding é específico da ferramenta Quartus. Esta solução pode não funcionar em outras ferramentas CAD

FSM - Especificando a Atribuição de Estados

IC-UNICAMP

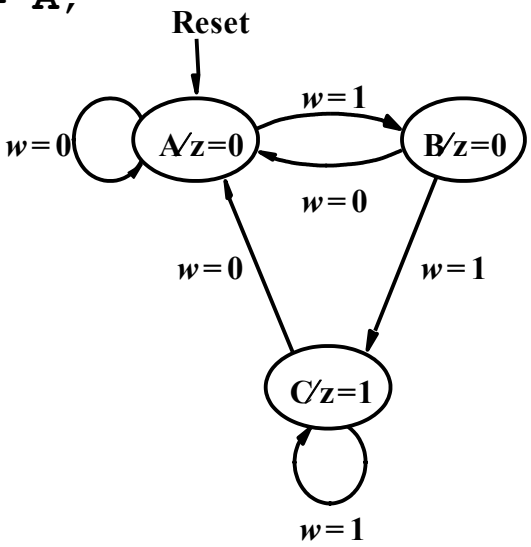
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY simple IS
        PORT ( Clock, Resetn, w : IN       STD_LOGIC;
               z                : OUT   STD_LOGIC );
END simple;
ARCHITECTURE Behavior OF simple IS
        SIGNAL y_present, y_next : STD_LOGIC_VECTOR(1 DOWNTO 0);
        CONSTANT A    : STD_LOGIC_VECTOR(1 DOWNTO 0) := "00";
        CONSTANT B    : STD_LOGIC_VECTOR(1 DOWNTO 0) := "01";
        CONSTANT C    : STD_LOGIC_VECTOR(1 DOWNTO 0) := "11";
BEGIN
        PROCESS ( w, y_present )
        BEGIN
                CASE y_present IS
                        WHEN A =>
                                IF w = '0' THEN y_next <= A;
                                ELSE y_next <= B;
                                END IF;
```

… con't

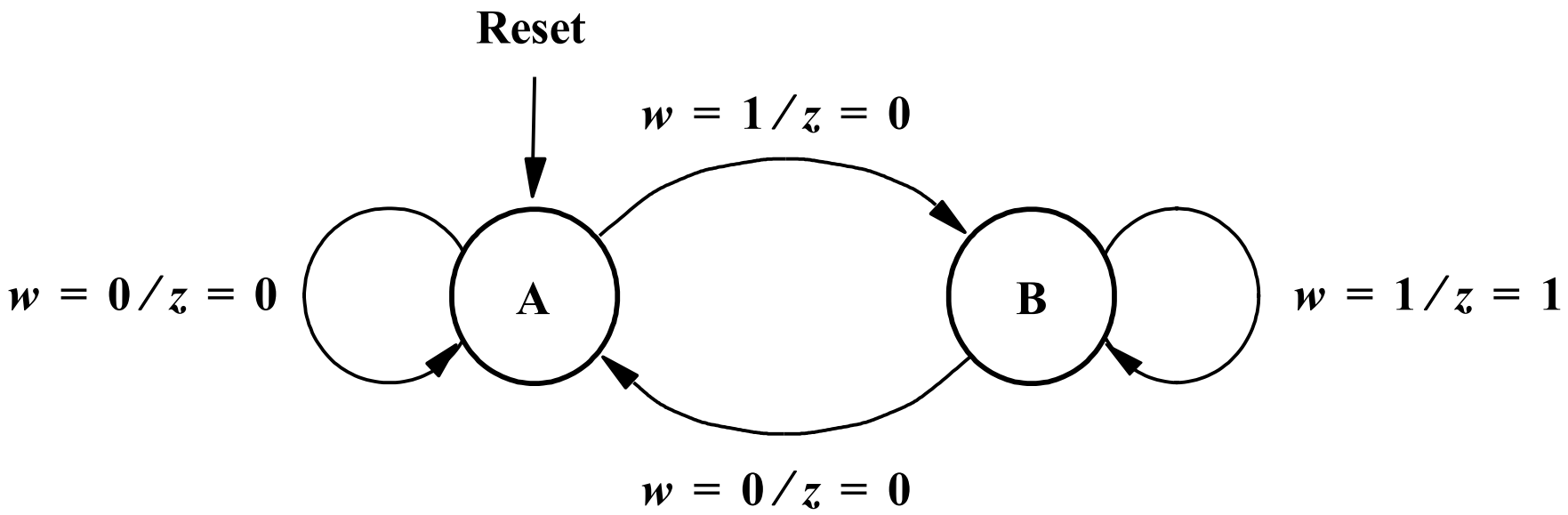
FSM - Especificando a Atribuição de Estados

IC-UNICAMP

```
                WHEN B =>
                     IF w = '0' THEN y_next <= A;
                     ELSE y_next <= C;
                     END IF;
                WHEN C =>
                     IF w = '0' THEN y_next <= A;
                     ELSE y_next <= C;
                     END IF;
                WHEN OTHERS =>
                     y_next <= A;
          END CASE;
     END PROCESS;

     PROCESS ( Clock, Resetn )
     BEGIN
          IF Resetn = '0' THEN
               y_present <= A;
          ELSIF (Clock'EVENT AND Clock = '1') THEN
               y_present <= y_next;
          END IF;
     END PROCESS;
     z <= '1' WHEN y_present = C ELSE '0';
END Behavior;
```



Máquina de Mealy

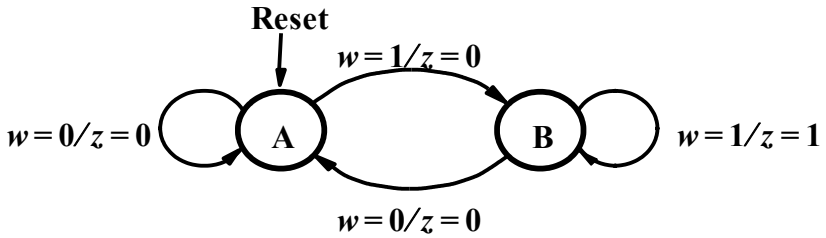


FSM de Mealy

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

ENTITY mealy IS
       PORT (Clock, Resetn, w: IN STD_LOGIC;
             z: OUT  STD_LOGIC );
END mealy;
```

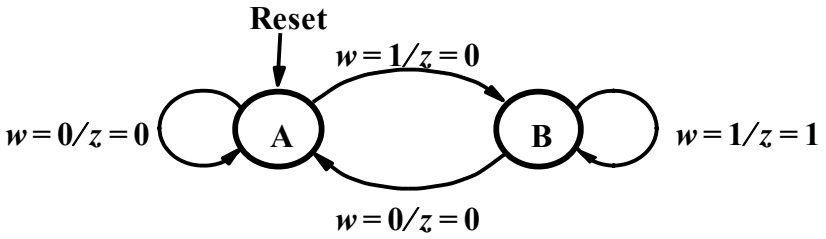
… con't

FSM de Mealy




```
ARCHITECTURE Behavior OF mealy IS
        TYPE State_type IS (A, B);
        SIGNAL y : State_type;
BEGIN
        PROCESS ( Resetn, Clock )
        BEGIN
                IF Resetn = '0' 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;
… con't
```

FSM de Mealy




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