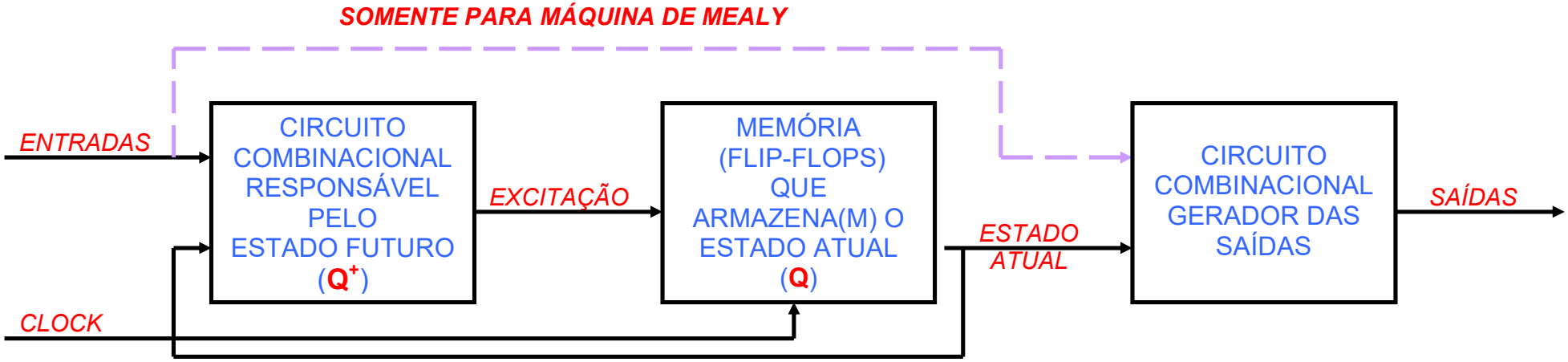
MÁQUINAS DE MOORE E DE MEALY
(FINITE STATE MACHINES – FSM)

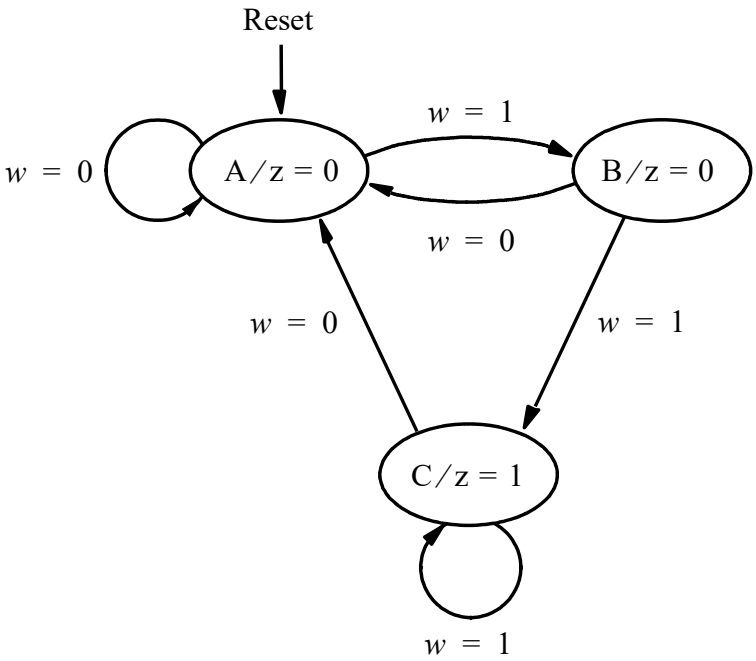
MEALY - Z(Q,w) / MOORE Z(Q)



- As FSM são implementadas através de circuitos combinacionais e flip-flops.
- Quando as ***SAÍDAS*** dependem **diretamente** do estado atual **Q** e das ***ENTRADAS***, dizemos que se trata de uma **Máquina de Mealy**. Caso esta dependência **direta** se restrinja ao estado **Q**, será chamada de **Máquina de Moore**.
- Apesar desta definição, é comum encontrarmos circuitos que têm algumas de suas ***SAÍDAS*** determinadas como uma **Máquina de Mealy** e outras, como **Moore**.

Exemplo: *Deseja-se construir uma FSM com as seguintes especificações: (a) O circuito possui uma entrada w e uma saída z. (b) Todas as mudanças ocorrem em uma borda positiva de clock. (c) A saída z* ***somente*** *será* ***1*** *se após dois ciclos consecutivos de clock a entrada w se mantiver em* ***1****.*

Solução: Escolhendo projetar o circuito como uma Máquina de Moore, podemos representar a operação através do seguinte **Diagrama de Estados**:



Simbologia do Diagrama:

- A, B e C representam os estados (de interesse) a serem armazenados nos ffs.
- Uma seta indica o efeito de uma transição de clock.

Uma vez que existem 3 estados no diagrama, serão necessárias 2 *variáveis de estado* Q (Q_1,Q_0) para referenciá-los. De forma geral, para uma máquina com **N** estados, serão necessárias **M** variáveis de estado, onde $2^{M} \geq N$.

Este diagrama pode ser transformado numa **Tabela de Estados**:

Estado Atual (**Q**)	Estado Futuro (Q^+)		Saída z
	w=0	*w*=1	
A	A	B	0
B	A	C	0
C	A	C	1

Fazendo, de forma arbitrária, A≡(Q_1=0,Q_0=0), B≡(Q_1=0,Q_0=1), C ≡ (Q_1=1,Q_0=0) e escolhendo flip-flops tipo D para a memória, reescrevemos a tabela de estados:

	Estado Atual (**Q**)		Estado Futuro (Q^+)								Saída z
			w=0				*w*=1				
	Q_1	Q_0	Q_1^+	Q_0^+	D_1	D_0	Q_1^+	Q_0^+	D_1	D_0	
A	0	0	0	0	0	0	0	1	0	1	0
B	0	1	0	0	0	0	1	0	1	0	0
C	1	0	0	0	0	0	1	0	1	0	1
	1	1	X	X	X	X	X	X	X	X	X

A partir deste diagrama devemos, agora, escrever as **Equações de Excitação**, que determinam o circuito combinacional responsável pela excitação dos flip-flops e a equação referente à saída z.

Estas equações expressam $D_1 = D_1(w,Q_1,Q_0)$, $D_0 = D_0(w,Q_1,Q_0)$, $z = (Q_1,Q_0)$ e podem ser obtidas, por exemplo, através do método do Mapa de Karnaugh:

Q_1Q_0 / w	00	01	11	10
0	0	0	X	0
1	0	1	X	1

$$D_1 = w(Q_1 + Q_0)$$

Q_1Q_0 / w	00	01	11	10
0	0	0	X	0
1	1	0	X	0

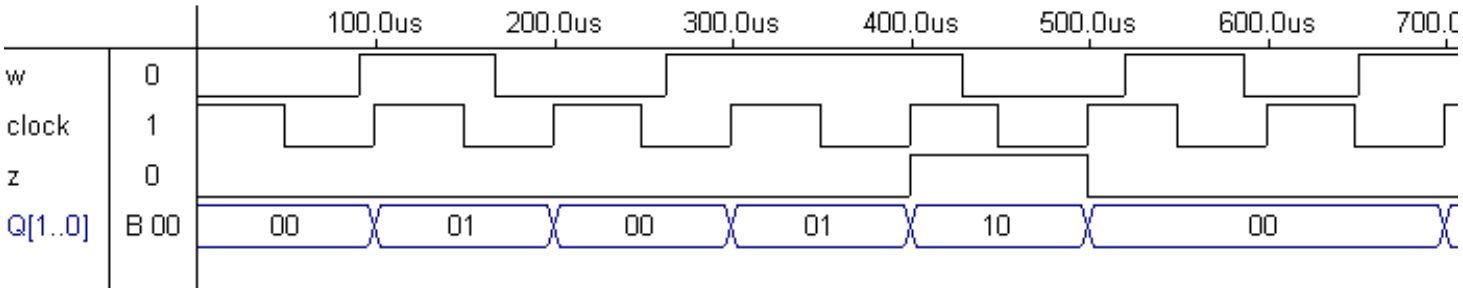
$$D_0 = w\,\overline{Q_1}\,\overline{Q_0}$$

Q_0 / Q_1	0	1
0	0	0
1	1	X

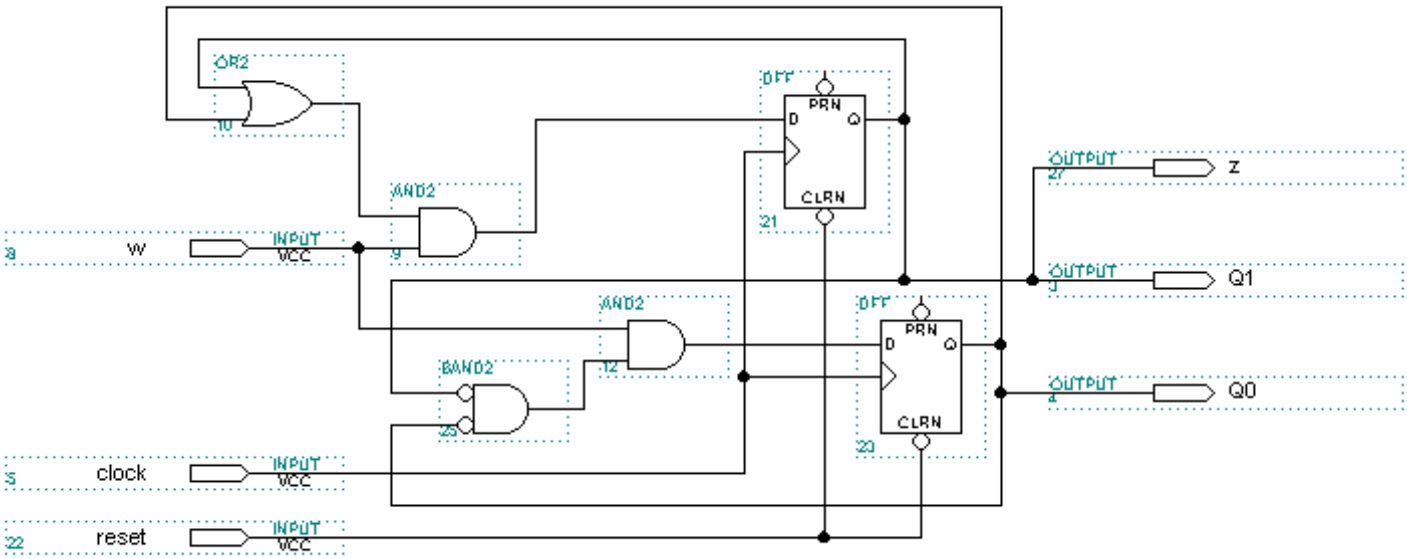
$$z = Q_1$$

$wQ_0 + w\,Q_1$

Exemplo de Temporização



Esquemático do Circuito



A.9.3 CASE Statement

The general form of a CASE statement is shown in Figure A.18. The *constant_value* can be a single value, such as 2, a list of values separated by the | pipe, such as 2|3, or a range, such as 2 to 4. An example of a CASE statement used to describe combinational logic is

```
CASE expression IS
    WHEN constant_value =>
        statement ;
        {statement ;}
    WHEN constant_value =>
        statement ;
        {statement ;}
    WHEN OTHERS =>
        statement ;
        {statement ;}
END CASE ;
```

Figure A.18 The general form of a CASE statement.

```
1  LIBRARY ieee ;
2  USE ieee.std_logic_1164.all ;
3
4  ENTITY simple IS
5      PORT (  Clock, Resetn, w    : IN    STD_LOGIC ;
6              z                   : OUT   STD_LOGIC ) ;
7  END simple ;
8
9  ARCHITECTURE Behavior OF simple IS
10     TYPE State_type IS (A, B, C) ;
11     SIGNAL Q : State_type ;
12 BEGIN
13     PROCESS ( Resetn, Clock )
14     BEGIN
15         IF Resetn = '0' THEN
16             Q <= A ;
17         ELSIF (Clock'EVENT AND Clock = '1') THEN
18             CASE Q IS
19                 WHEN A =>
20                     IF w = '0' THEN
21                         Q <= A ;
22                     ELSE
23                         Q <= B ;
24                     END IF ;
25                 WHEN B =>
26                     IF w = '0' THEN
27                         Q <= A ;
28                     ELSE
29                         Q <= C ;
30                     END IF ;
31                 WHEN C =>
32                     IF w = '0' THEN
33                         Q <= A ;
34                     ELSE
35                         Q <= C ;
36                     END IF ;
37             END CASE ;
38         END IF ;
39     END PROCESS ;
40
41     z <= '1' WHEN Q = C ELSE '0' ;
42 END Behavior ;
```

Exemplo: *Deseja-se construir uma FSM com as seguintes especificações: (a) O circuito possui uma entrada w e uma saída z. (b) A saída z* ***somente*** *será* ***1*** *se após dois ciclos consecutivos de clock a entrada w se mantiver em* ***1****.*

Solução: Escolhendo projetar o circuito como uma Máquina de Mealy, podemos representar a sua operação através do seguinte

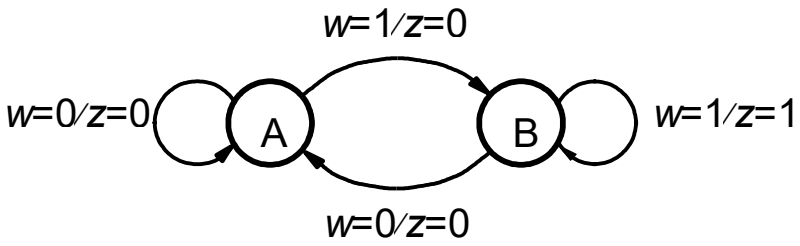


Diagrama de Estados:

Simbologia do Diagrama:

- A, B representam os estados (de interesse) a serem armazenados nos ffs.
- Uma seta indica o efeito de uma transição de clock.
- De acordo com a literatura, passaremos a escrever w=1/z=0, como 1/0. Isto é, entrada(s)/saída(s).

Este diagrama pode ser transformado numa **Tabela de Estados**:

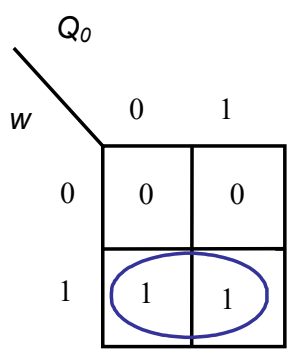
Estado Atual (**Q**)	Estado Futuro (**Q**$^+$)		Saída z	
	w=0	*w*=1	*w*=0	*w*=1
A	A	B	0	0
B	A	B	0	1

Fazendo, de forma arbitrária, A≡(Q_0=0), B≡(Q_0=1) e escolhendo flip-flops tipo D para a memória, reescrevemos a tabela de estados:

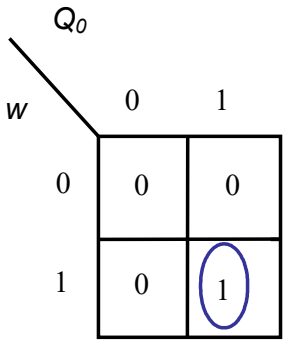
Estado Atual (**Q**)	Estado Futuro (**Q**$^+$)				Saída z	
	w=0		*w*=1		*w*=0	*w*=1
Q_0	Q_0^+	D_0	Q_0^+	D_0		
0	0	0	1	1	0	0
1	0	0	1	1	0	1

A partir deste diagrama devemos, agora, escrever as **Equações de Excitação**, que determinam o circuito combinacional responsável pela excitação dos flip-flops e a equação referente à saída z.

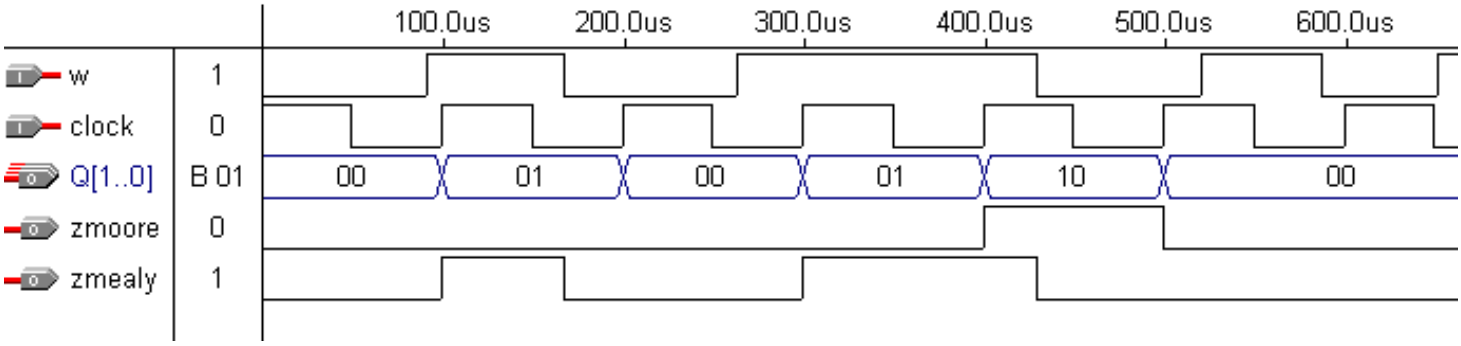
Estas equações expressam $D_0 = D_0(w,Q_0)$, $z = z(w,Q_0)$ e podem ser obtidas, por exemplo, através do método do Mapa de Karnaugh:



$$D_0 = w$$

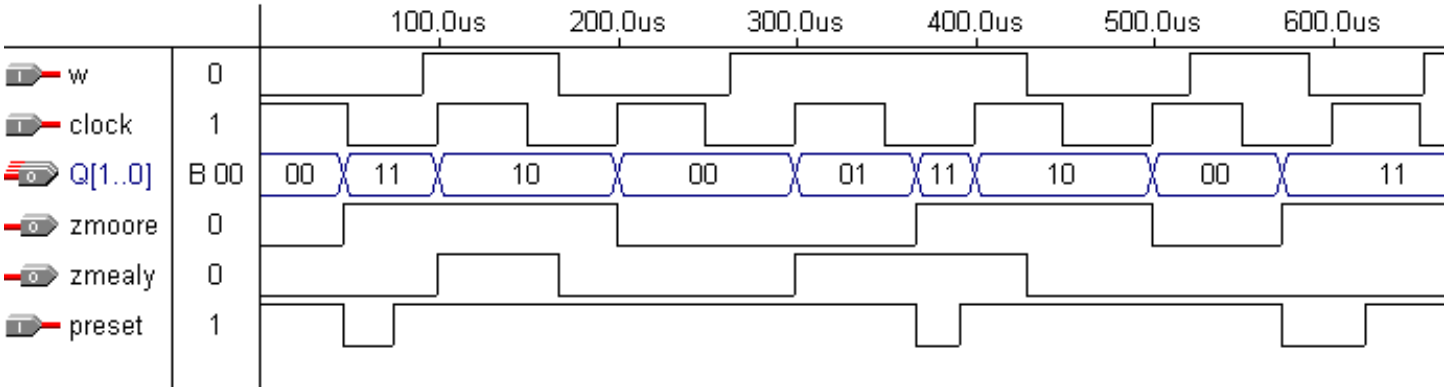


$$z = wQ_0$$



Exemplo de Temporização – Comparação com a Máquina de Moore.

Exemplo de Temporização – Impondo o DCS p/ máquina de Moore.



```
1  LIBRARY ieee ;
2  USE ieee.std_logic_1164.all ;
3
4  ENTITY mealy IS
5      PORT (  Clock, Resetn, w    : IN    STD_LOGIC ;
6              z                   : OUT   STD_LOGIC ) ;
7  END mealy ;
8
9  ARCHITECTURE Behavior OF mealy IS
10     TYPE State_type IS (A, B) ;
11     SIGNAL y : State_type ;
12 BEGIN
13     PROCESS ( Resetn, Clock )
14     BEGIN
15         IF Resetn = '0' THEN
16             y <= A ;
17         ELSIF (Clock'EVENT AND Clock = '1') THEN
18             CASE y IS
19                 WHEN A =>
20                     IF w = '0' THEN y <= A ;
21                     ELSE y <= B ;
22                     END IF ;
23                 WHEN B =>
24                     IF w = '0' THEN y <= A ;
25                     ELSE y <= B ;
26                     END IF ;
27             END CASE ;
28         END IF ;
29     END PROCESS ;
30
31     PROCESS ( y, w )
32     BEGIN
33         CASE y IS
34             WHEN A =>
35                 z <= '0' ;
36             WHEN B =>
37                 z <= w ;
38         END CASE ;
39     END PROCESS ;
40 END Behavior ;
```