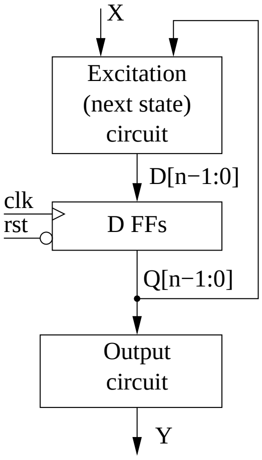
11 Synchronous State Machines

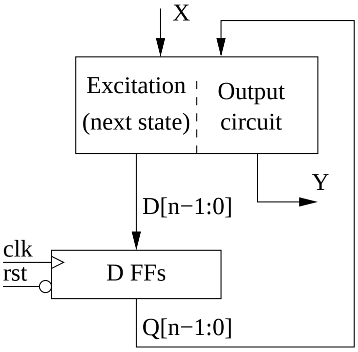
- Synchronous (finite) state machines are also known as clocked sequential circuits.
- Registers and counters are examples of specialised synchronous sequential circuits.
- Recall that in a general case we have two types of state machines, Mealy and Moore, that differ in the way the output signals are form.

$$\begin{aligned} \text{state equations:} \quad Q(t+1) &= f(Q(t), X(t)) \\ \text{D FFs excitation eqs:} \quad D(t) &= f(Q(t), X(t)) \end{aligned}$$

Output equations:

$$\begin{aligned} \text{Mealy:} \quad Y(t) &= g(Q(t), X(t)) \\ \text{Moore:} \quad Y(t) &= g(Q(t)) \end{aligned}$$

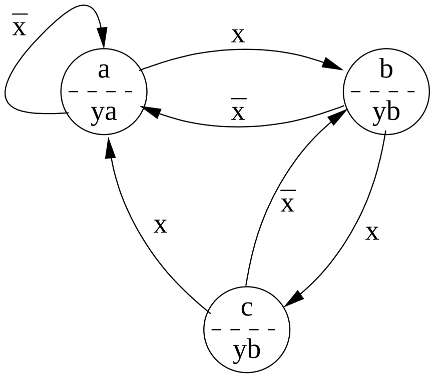




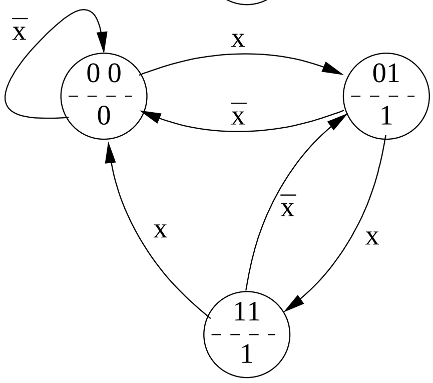
- In the Moore state machine the output signals are re-coded state signals.
- In the Mealy state machine the output signals depends both on state signals and input signals.
- Typically, state machines are primarily described by **state diagrams**.
- State diagrams are converted into **state and output tables** from which the structure of the excitation and output circuits can be derived.

11.1 Example of a Moore state machine

- Consider a Moore state machine described by the following symbolic state diagram:
- The state machine has a single input, x, three states labeled a, b, c and a single binary output $y \in \{ya, yb\}$
- At least two state signals, say q_1, q_0 are required to code three states.
- Unlike in the asynchronous case, the states can be flexibly coded, simplicity of the excitation circuit being the only limitation.



- We start with **state assignment** replacing symbolic names of the states with their binary equivalent. This results in the following state state diagram:
- The state machine now has two D flip-flops q_1, q_0.
- Note that in the synchronous case the machine is in any given state for at least one clock cycle.



- The next step is to convert the state diagram into the state and output equations.
- the excitation table for D flip-flops is identical with the state table.

x	Q_1	Q_0	D_1	D_0
0	0	0	0	0
1	0	0	0	1
0	0	1	0	0
1	0	1	1	1
0	1	0	–	–
1	1	0	–	–
0	1	1	0	1
1	1	1	0	0

Q_1	Q_0	y
0	0	0
0	1	1
1	0	–
1	1	1

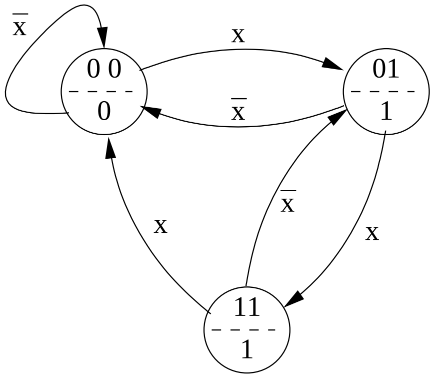
- Resulting Karnaugh maps

D_1 Q_1Q_0

x	00	01	11	10
0	0	0	0	–
1	0	1	0	–

D_0 Q_1Q_0

x	00	01	11	10
0	0	0	1	–
1	1	1	0	–



Excitation and output equations:

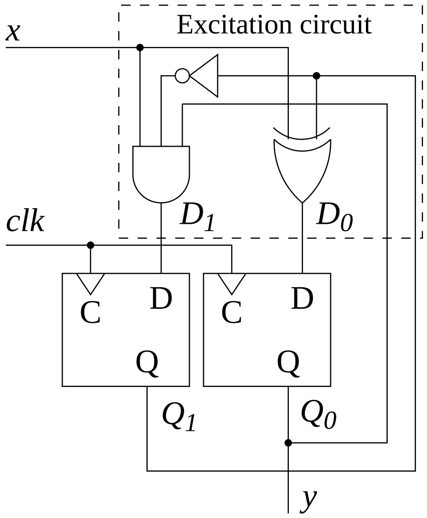
$$\begin{aligned} D_1 &= x \cdot \overline{Q}_1 \cdot Q_0 \\ D_0 &= x \cdot \overline{Q}_1 + \overline{x} \cdot Q_1 = x \oplus Q_1 \\ y &= Q_0 \end{aligned}$$

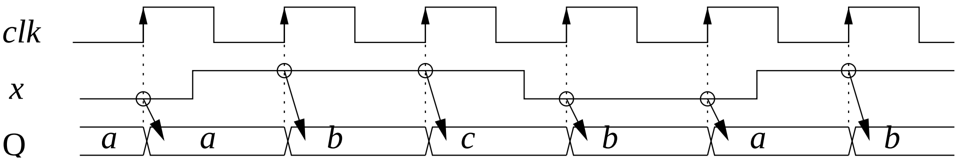
- Excitation and output equations:

$$\begin{aligned} D_1 &= x \cdot \overline{Q}_1 \cdot Q_0 \\ D_0 &= x \cdot \overline{Q}_1 + \overline{x} \cdot Q_1 = x \oplus Q_1 \\ y &= Q_0 \end{aligned}$$

- The excitation circuit generates input signals to the flip-flops
- These signals are clocked in during the rising edge of the clock.
- The output circuits is trivial in this example, but in general the output signals are generate by a combinational circuit from the state signals.
- Timing diagram:

Logic diagram:





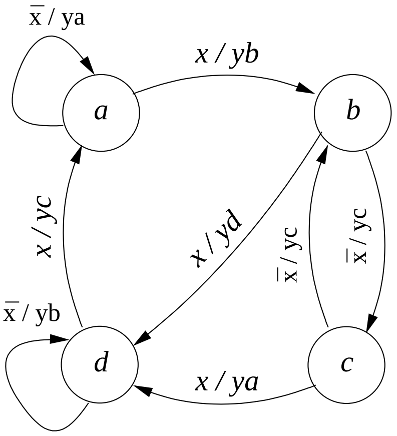
VHDL specification of the Moore state machine

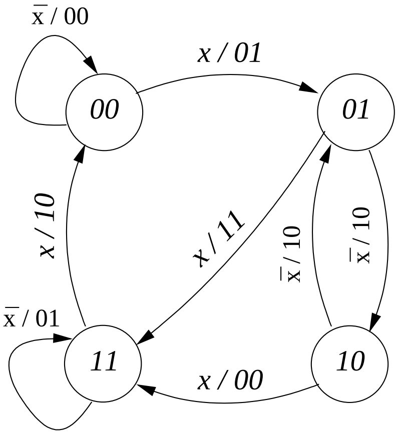
```
library IEEE ;
use IEEE.std_logic_1164.all ;

ENTITY MSTM IS
   PORT ( x, clk :  IN  STD_LOGIC ;
          Q : OUT  STD_LOGIC_VECTOR(1 DOWNTO 0) ;
          y   : OUT STD_LOGIC
   ) ;
END MSTM ;

ARCHITECTURE behv OF MSTM IS

--  TYPE states IS (sa, sb, sc) ;
--  SIGNAL  D  :  states := sa ;

    SIGNAL  D, stt  : STD_LOGIC_VECTOR(1 DOWNTO 0) ;
-- a "hard" encoding of states:
   CONSTANT sa : STD_LOGIC_VECTOR(1 DOWNTO 0) := "00" ;
   CONSTANT sb : STD_LOGIC_VECTOR(1 DOWNTO 0) := "01" ;
   CONSTANT sc : STD_LOGIC_VECTOR(1 DOWNTO 0) := "11" ;
-- encoding of output signals
   CONSTANT ya   : STD_LOGIC := ’0’ ;
   CONSTANT yb   : STD_LOGIC := ’1’ ;

BEGIN
       -- to synthesize edge-triggered flip-flops
 clkd: PROCESS ( clk )
 BEGIN
    IF  clk’EVENT AND clk = ’1’  THEN
       stt <= D ;
    END IF ;
 END PROCESS clkd ;

-- the stm process describes the transitions between states -- and the output signals
```

```
 stm: PROCESS ( Q )
 BEGIN
--  default assignments
     D <= Q  ;
     y <= ya  ;

   CASE stt IS  -- state transitions and output signals

   WHEN sa =>
     y <= ya  ;
     IF  x = ’0’  THEN
        D <= sa ;  ELSE  D <= sb ;
     END IF ;

   WHEN sb =>
     y <= yb  ;

     CASE  x  IS
       WHEN ’0’    =>  D <= sa ;
       WHEN OTHERS =>  D <= sc ;
     END CASE ;

  WHEN OTHERS =>    --- when sc
     y <= yb  ;

     CASE  x  IS
       WHEN ’0’    =>  D <= sb ;
       WHEN OTHERS =>  D <= sa ;
     END CASE ;

   END CASE ;
 END PROCESS stm ;
 Q <= stt ;
END behv ;
```

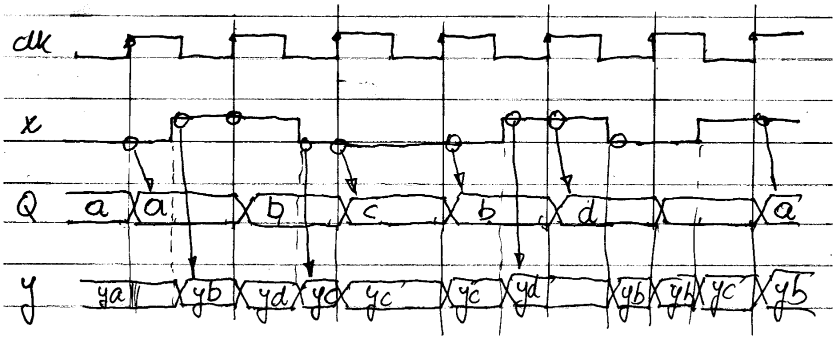
11.2 Example of a Mealy state machine

- Consider a Mealy state machine described by the following symbolic state diagram:
- The state machine has
  - a single input, x,
  - four states labeled a, b, c, d that will be coded using (at lest) two state signals q_1, q_0, and
  - binary output $y \in \{ya, yb, yc, yd\}$ that will be coded by two output signals y_1, y_0
- Note that the output signals depend now on both the state signals and the input signal(s).
- Labeling of the output signals might be confusing: it is the **state that arrow originates** from that generates the output signal together with the input signal.
- After **the state and output assignment** we have the following state state diagram:





- Timing diagram

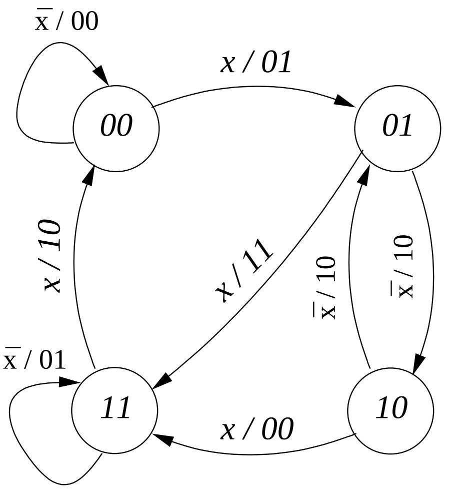


- Note that output signals change in response to a change in both input signals and state signals.
- The next step is to convert the state diagram into the state and output equations.
- the excitation table for D flip-flops is identical with the state table.

The excitation (state) and output tables:

x	Q_1	Q_0	D_1	D_0
0	0	0	0	0
1	0	0	0	1
0	0	1	1	0
1	0	1	1	1
0	1	0	0	1
1	1	0	1	1
0	1	1	1	1
1	1	1	0	0

x	Q_1	Q_0	y_1	y_0
0	0	0	0	0
1	0	0	0	1
0	0	1	1	0
1	0	1	1	1
0	1	0	1	0
1	1	0	0	0
0	1	1	0	1
1	1	1	1	0



Resulting Karnaugh maps:

D_1^+ Q_1Q_0

x	00	01	11	10
0	0	1	1	0
1	0	1	0	1

D_0^+ Q_1Q_0

x	00	01	11	10
0	0	0	1	1
1	1	1	1	0

y_1^+ Q_1Q_0

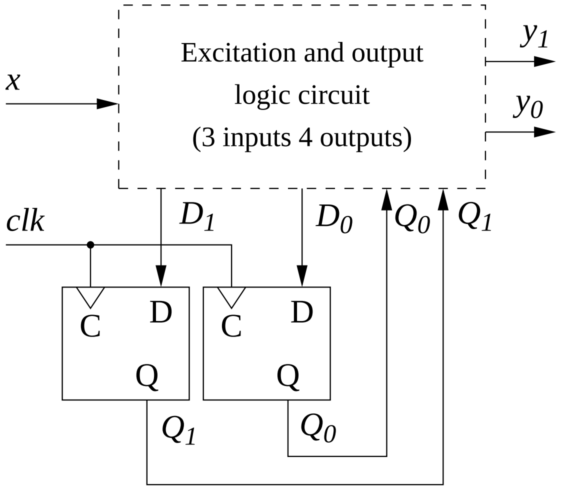
x	00	01	11	10
0	0	1	0	1
1	0	1	1	0

y_0^+ Q_1Q_0

SR	00	01	11	10
0	0	0	0	1
1	1	1	0	0

Write out the logic equations as an exercise.

Logic diagram:



Write a VHDL description as an exercise