تصميم النظم المنطقية باستخدام الدارات المنطقية المبرمجة

المحاضرة العاشرة

د.م. خولة حموي
khawla.hamwi@gmail.com

العام الدراسي: 2023-2024

جَامعة المَنارة
MANARA UNIVERSITY

عناوين المحاضرة

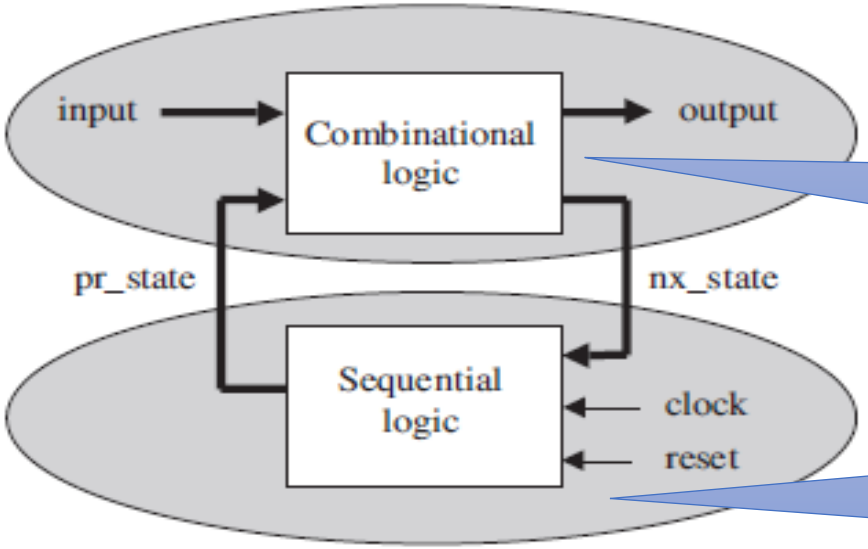
- ماكينة الحالة المنتهية
- ماكينة مور وماكينة ميلي
- عداد BCD
- ماكينة حالة منتهية بسيطة
- كاشف السلسلة

ماكينة الحالة المنتهية

ماكينة الحالة المنتهية Finite State Machine (FSM) هي دارة تتابعية متزامنة synchronous تتألف من جزء منطقي تتابعي Sequential(قلابات) وجزء منطقي تركيبي Combinational

عدد القلابات اللازمة (غالباً قلاب D هو المستخدم) للتصميم يساوي $\lceil \log_2 n \rceil$ حيث n تمثل عدد الحالات

خرج هذه الماكينة والحالات الداخلية للقلابات تتبع لتتالي الحالات المتوقعة بالإضافة إلى إشارة الدخل وفق إيقاع نبضة الساعة



يمتلك **دخلين**:
(الدخل والحالة السابقة (pr_state)) و**خرجين**:
(الخرج والحالة اللاحقة (nx_state))

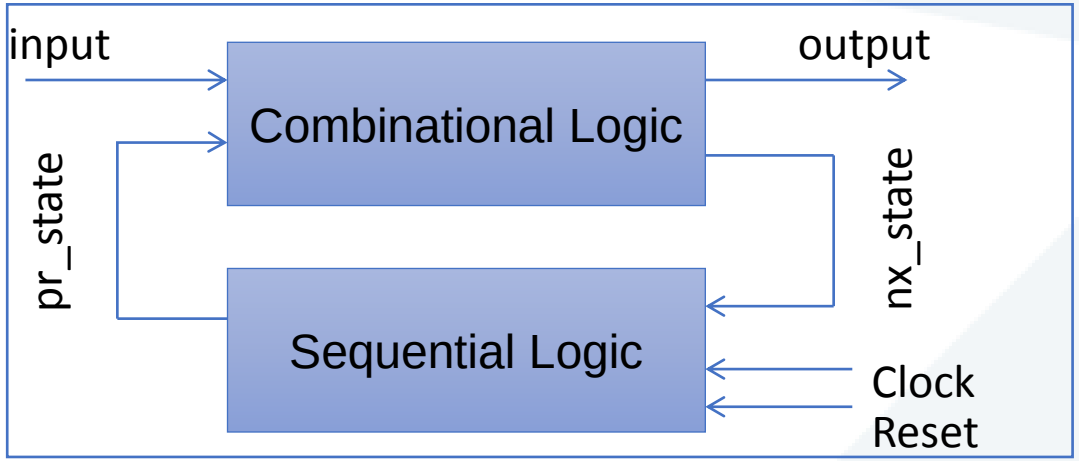
يمتلك **ثلاثة مداخل**:
(clock, reset, nx_state)
وخرج واحد:
(pr_state)

مخطط حالة بطور واحد

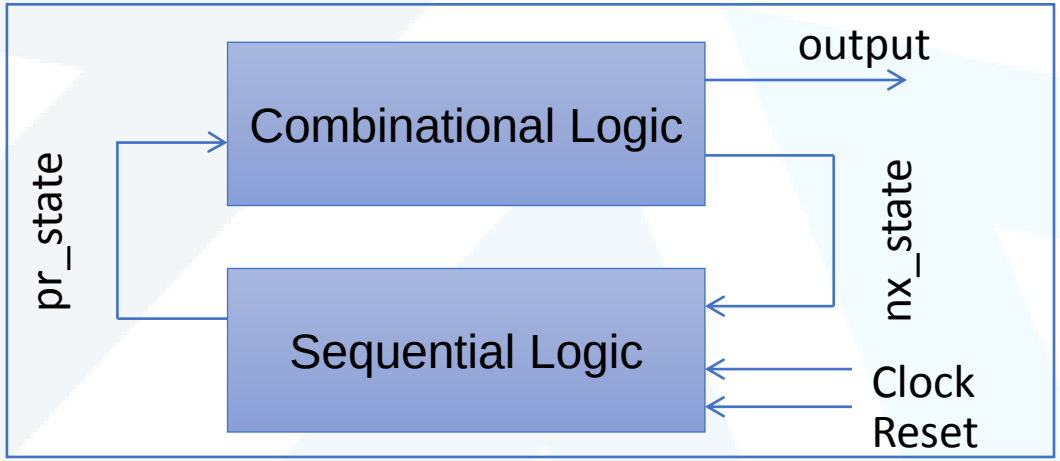
ماكينة مور و ماكينة ميلي

تصنف ماكينة الحالة إلى نوعين: ماكينة مور Mooreوماكينة ميلي Mealy

1. ماكينة مور: يتبع فيها الخرج للقسم التتابعي في الماكينة (الحالات الداخلية)
2. ماكينة ميلي: يتبع الخرج للقسم التتابعي (الحالات الداخلية) والقسم التركيبي (الدخل)



ماكينة ميلي

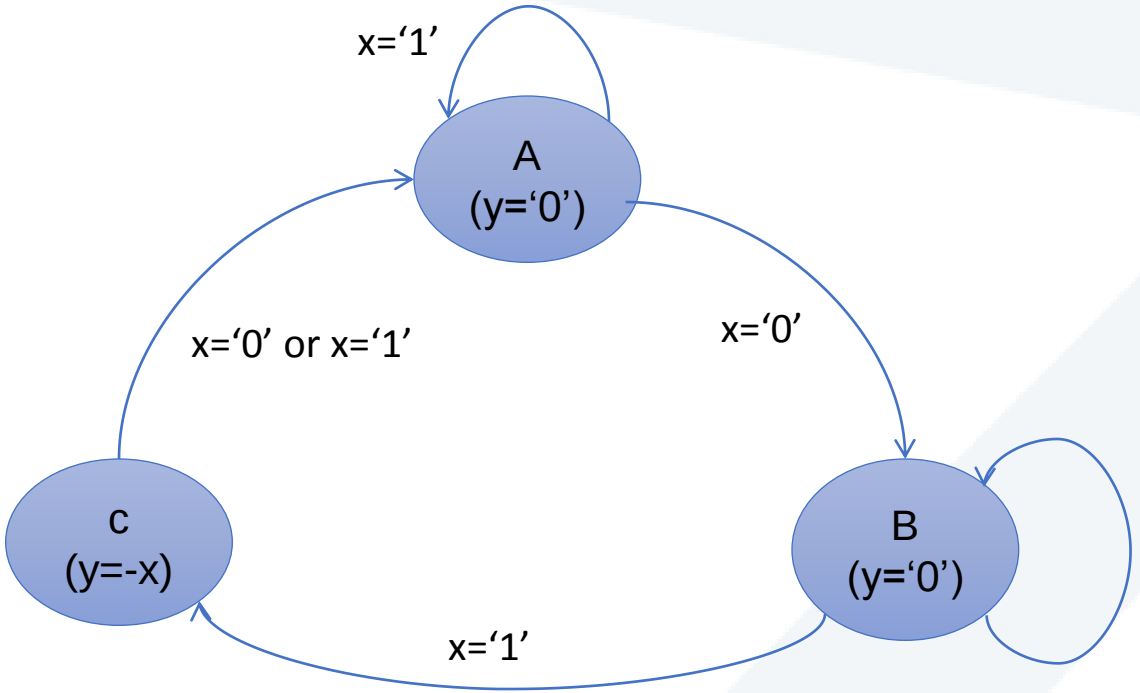


ماكينة مور

input: دخل ماكينة الحالة	output: خرج ماكينة الحالة
pr_state: تمثل الحالة الحالية present state	nx_state: تمثل الحالة التالية next state
clock: نبضة الساعة	reset: نبضة التصفير

ماكينة مور و ماكينة ميلي

- يمكن تمثيل عمل ماكينة الحالة باستخدام مخطط يسمى مخطط نقل الحالة state transition diagram

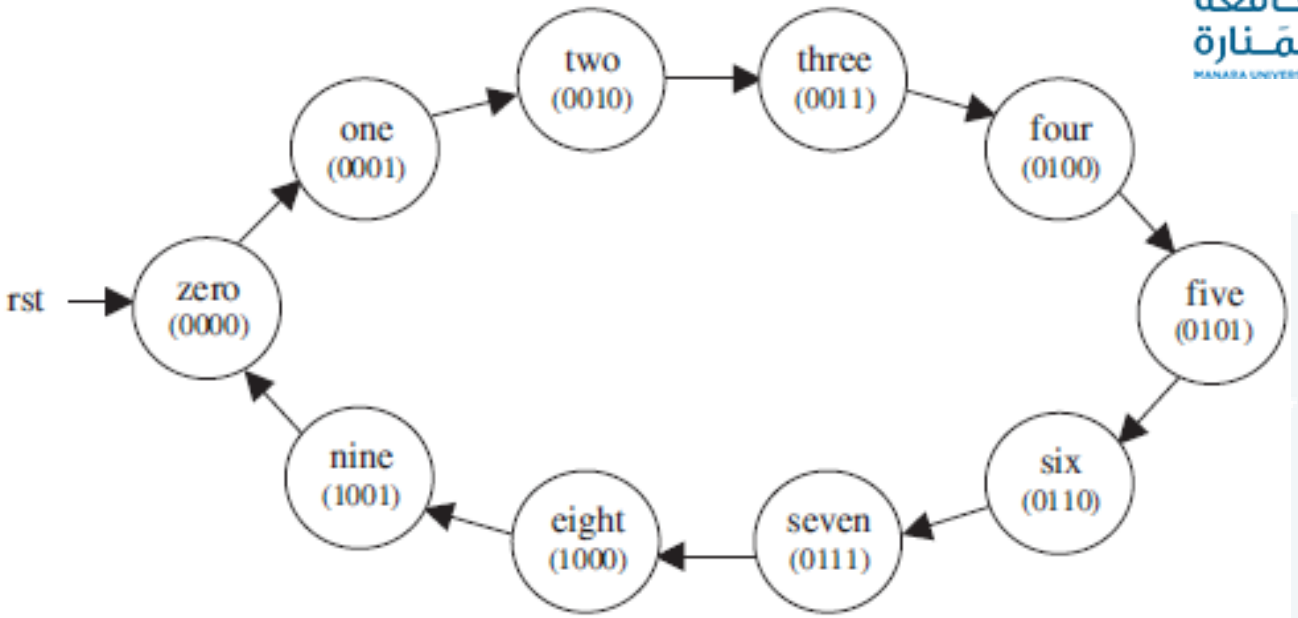


- يعبر **مخطط نقل الحالة** عن آلية الانتقال من حالة إلى أخرى وفق إشارة الدخل وما هو الخرج الناتج فعلى سبيل المثال إذا كانت ماكينة الحالة A عندها يكون الخرج y=0، فإذا كان الدخل x=1 فإننا نبقى في الحالة A ويبقى الخرج y=0, أما إذا كان الدخل x=0 فإننا ننتقل إلى الحالة B ويبقى الخرج y=0 وهكذا.

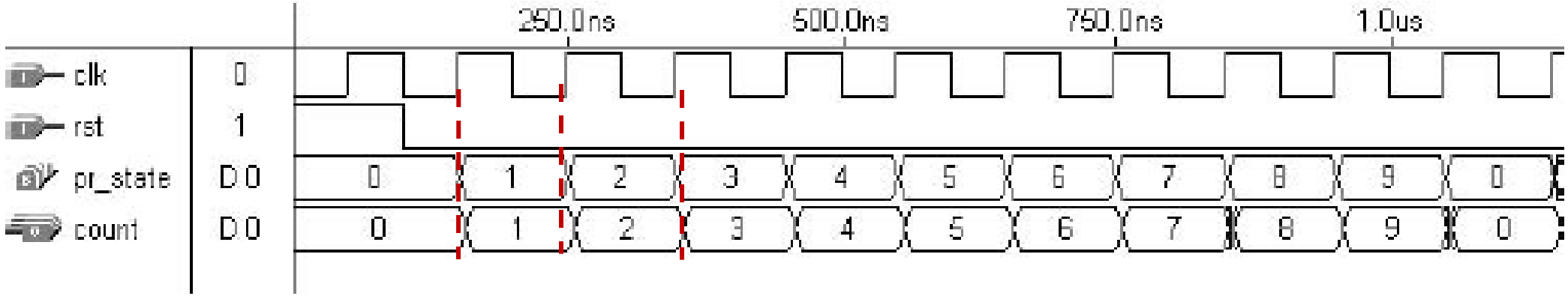
دارة عداد BCD

يمثل عداد BCDمثالاً عن ماكينة مور Moore Machineبحيث يتعلق الخرج فقط بالحالة الحالية المخزنة.

مخطط نقل الحالة للعداد مبين بالشكل



- نتائج المحاكاة مبينة بالشكل:

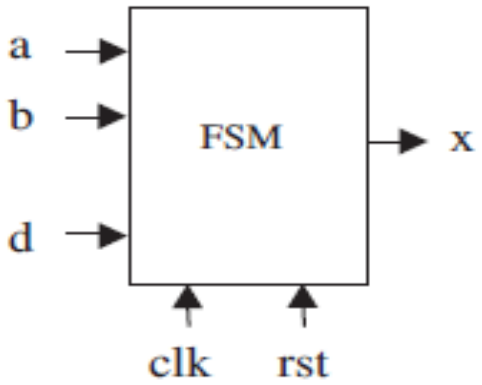


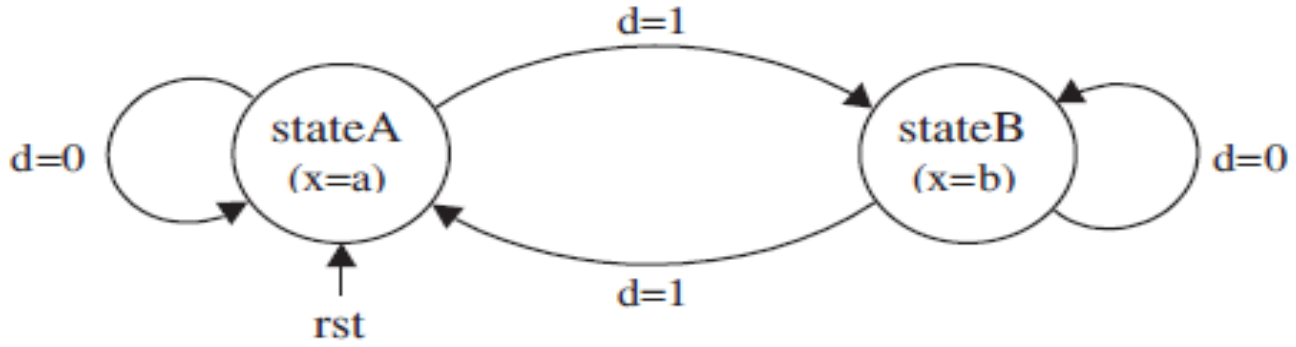
دارة عداد BCD

```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY counter IS
6     PORT ( clk, rst: IN STD_LOGIC;
7            count: OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
8  END counter;
9  ------------------------------------------------
10 ARCHITECTURE state_machine OF counter IS
11    TYPE state IS (zero, one, two, three, four,
12       five, six, seven, eight, nine);
13    SIGNAL pr_state, nx_state: state;
14 BEGIN
15    ------------- Lower section: ----------------
16    PROCESS (rst, clk)
17    BEGIN
18       IF (rst='1') THEN
19          pr_state <= zero;
20       ELSIF (clk'EVENT AND clk='1') THEN
21          pr_state <= nx_state;
22       END IF;
23    END PROCESS;
24    ------------- Upper section: ----------------
25    PROCESS (pr_state)
26    BEGIN
27       CASE pr_state IS
28          WHEN zero =>
29             count <= "0000";
30             nx_state <= one;
31          WHEN one =>
32             count <= "0001";
33             nx_state <= two;
34          WHEN two =>
35             count <= "0010";
36             nx_state <= three;
37          WHEN three =>
38             count <= "0011";
39             nx_state <= four;
40          WHEN four =>
41             count <= "0100";
42             nx_state <= five;
43          WHEN five =>
44             count <= "0101";
45             nx_state <= six;
46          WHEN six =>
47             count <= "0110";
48             nx_state <= seven;
49          WHEN seven =>
50             count <= "0111";
51             nx_state <= eight;
52          WHEN eight =>
53             count <= "1000";
54             nx_state <= nine;
55          WHEN nine =>
56             count <= "1001";
57             nx_state <= zero;
58       END CASE;
59    END PROCESS;
60 END state_machine;
61 ------------------------------------------------
```

ماكينة حالة منتهية بسيطة

•لتكن لدينا ماكينة الحالة المنتهية ومخطط نقل الحالة كما في الشكل والمطلوب كتابة برنامج بلغة VHDL لوصف مخطط نقل الحالة.





يتألف مخطط نقل الحالة من الحالتين stateA و stateB. إذا كانت الحالة الحالية هي stateA من أجل الدخل d=0 نبقى في نفس الحالة والخرج x=a، أما إذا كان الدخل d=1 تصبح الحالة الحالية stateB والخرج x=b. في الحالة stateB إذا كان الدخل d=0 نبقى في نفس الحالة ونفس الخرج، أما إذا كان الدخل d=1 فتصبح الحالة الحالية stateA والخرج x=a.

ماكينة حالة منتهية بسيطة

يكتب البرنامج على النحو التالي:

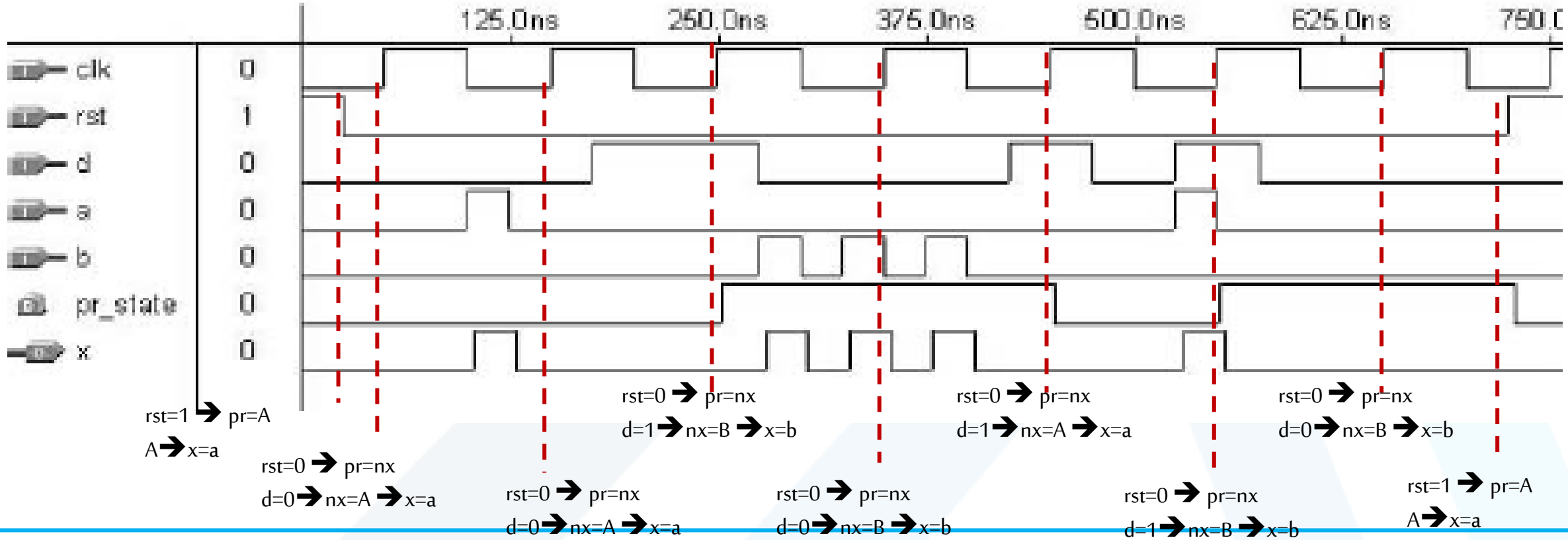
```
ENTITY simple_fsm IS
PORT ( a, b, d, clk, rst: IN BIT;                    // عرفنا مداخل ماكينة الحالة
                x: OUT BIT);                         // عرفنا خرج ماكينة الحالة
END simple_fsm;
-------------------------------------------
ARCHITECTURE simple_fsm OF simple_fsm IS             // سلوك ماكينة الحالة
TYPE state IS (stateA, stateB);                      //عرفنا نوع جديد يحوي حالتين A , B
SIGNAL pr_state, nx_state: state;                    // عرفنا اشارتين داخليتين من نوع الحالة الذي عرفناها
BEGIN
---------------------------- Lower section: ----------------
PROCESS (rst, clk)                                   // تحسس اشارات التصفير ونبضة الساعة
BEGIN
IF (rst='1') THEN                                    /// A اذا كان مدخل التصفير فعال سأكون بالحالة
pr_state <= stateA;
      ELSIF (clk'EVENT AND clk='1') THEN             /// اختبار ورود نبضة ساعة واذا تحقق الشرط
       pr_state <= nx_state;                         // ستصبح الحالة الحالية هي الحالة التالي
END IF;
END PROCESS;
---------------------------- Upper section: ----------------
PROCESS (a, b, d, pr_state)                          // تحسس اشارات الدخل والحالة الحالية
BEGIN
      CASE pr_state IS                               //لاختبار كافة حالة الماكينة /CASE/ تعليمة
            WHEN stateA =>                           // a هو /A/سيكون خرج /عندما تكون بالحالة
                  x <= a;
                  IF (d='1') THEN nx_state <= stateB;    // B اذا قيمة d=1 ننتقل إلى
                  ELSE nx_state <= stateA;               // A وإلا نبقى بالحالة
                  END IF;
```

ماكينة حالة منتهية بسيطة

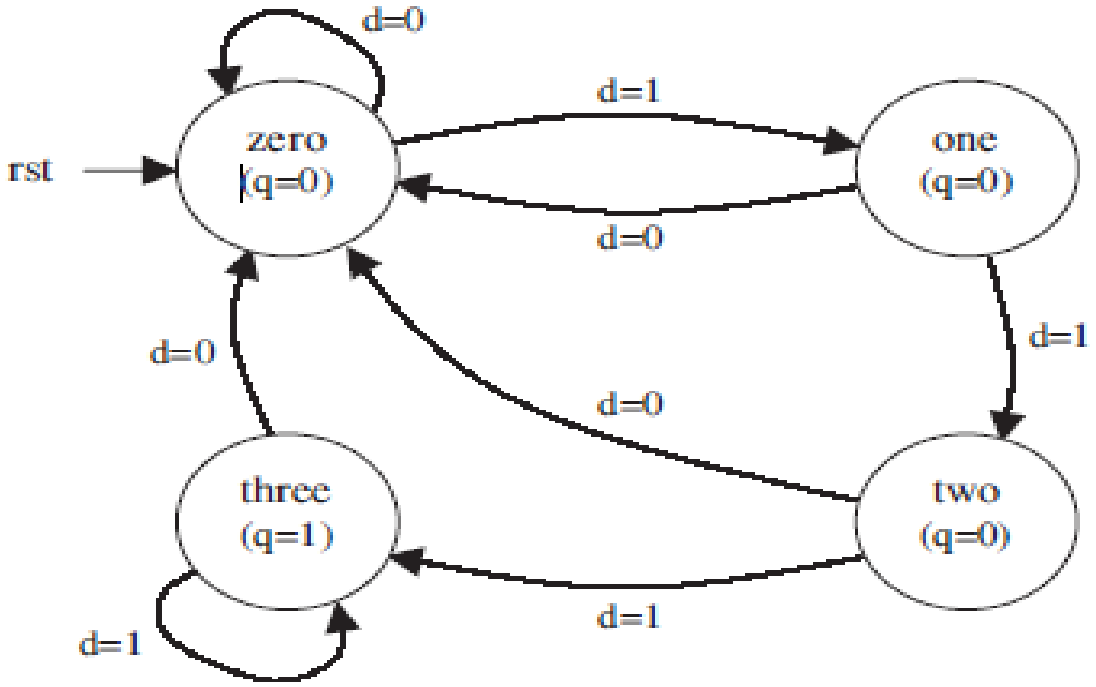
```
WHEN stateB =>                          // b  عندما نكون بالحالة /B/سيكون خرج هو
        x <= b;
        IF (d='1') THEN nx_state <= stateA;     // B اذا قيمة d=1 ننتقل إلى
        ELSE nx_state <= stateB;                //B وإلا نبقى بالحالة
    END IF;

END CASE;
END PROCESS;
END simple_fsm;
```

يكتب البرنامج على النحو التالي:

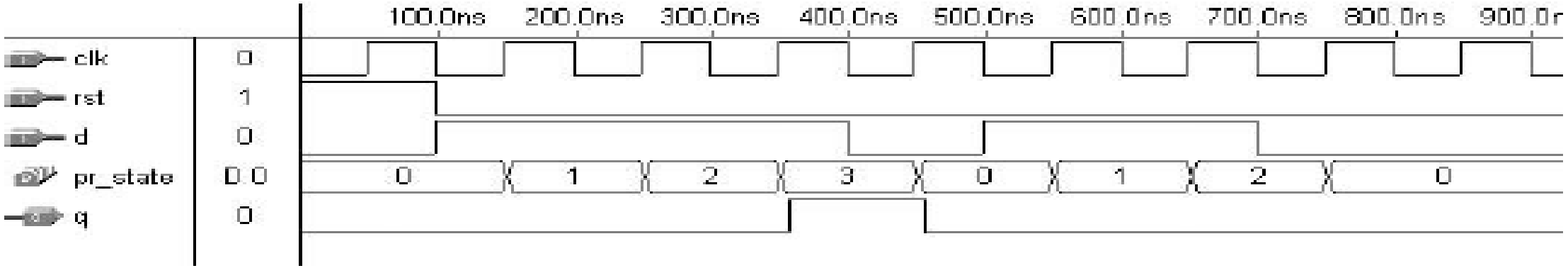
- نتائج المحاكاة مبينة بالشكل:



كاشف السلسلة



- يراد تصميم دارة تستقبل سلسلة من الخانات،بحيث تعطي على خرجها "1"إذا كان تتالي خانات الدخل "111". يجب أخذ التداخل بعين الاعتبار أي إذا ورد التتالي "0111110"، فيجب أن يبقى الخرج في المستوى "1" لثلاث نبضات متوالية.
- مخطط الحالة مبين في الشكل:
- نتائج المحاكاة مبينة بالشكل:



كاشف السلسلة

برنامج VHDL يُكتب كما يلي:

```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY string_detector IS
6     PORT ( d, clk, rst: IN BIT;
7            q: OUT BIT);
8  END string_detector;
9  ----------------------------------------
10 ARCHITECTURE my_arch OF string_detector IS
11    TYPE state IS (zero, one, two, three);
12    SIGNAL pr_state, nx_state: state;
13 BEGIN
14    ----- Lower section: -------------------
15    PROCESS (rst, clk)
16    BEGIN
17       IF (rst='1') THEN
18          pr_state <= zero;
19       ELSIF (clk'EVENT AND clk='1') THEN
20          pr_state <= nx_state;
21       END IF;
22    END PROCESS;
23    ---------- Upper section: --------------
24    PROCESS (d, pr_state)
25    BEGIN
26       CASE pr_state IS
27          WHEN zero =>
28             q <= '0';
29             IF (d='1') THEN nx_state <= one;
30             ELSE nx_state <= zero;
31             END IF;
32          WHEN one =>
33             q <= '0';
34             IF (d='1') THEN nx_state <= two;
35             ELSE nx_state <= zero;
36             END IF;
37          WHEN two =>
38             q <= '0';
39             IF (d='1') THEN nx state <= three;
40             ELSE nx_state <= zero;
41             END IF;
42          WHEN three =>
43             q <= '1';
44             IF (d='0') THEN nx_state <= zero;
45             ELSE nx_state <= three;
46             END IF;
47       END CASE;
48    END PROCESS;
49 END my_arch;
50 ----------------------------------------
```