دوره مقدماتی آشنایی با FPGA و VHDL

ماشین حالت

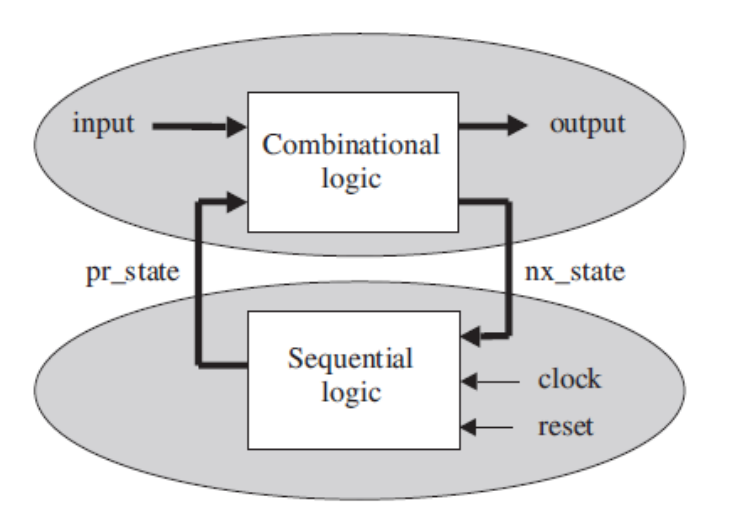
محمدرضا عزیزی

امیرعلی ابراهیمی

بهار ۱۳۹۷

مقدمه

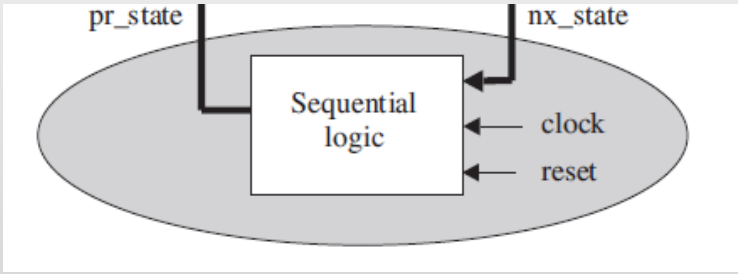
- از ماشین حالت ها برای پیاده سازی سیستم‌هایی که عملکرد آن‌ها به صورت یک دنباله است، استفاده می‌شود.
- انواع ماشین حالت:
  - میلی (Mealy): خروجی وابسته به *ورودی* و *حالت فعلی*
  - مور (Moore): خروجی وابسته به *حالت فعلی*
- تمامی مدارهای ترتیبی را می‌توان با ماشین حالت مدل‌سازی کرد، اما این کار می‌تواند مفید نباشد، زیرا کد طولانی تر شده و احتمال بروز خطا در کد نیز بیشتر می‌شود.



طراحی نوع #۱ (۱)

- طراحی بخش *پایینی* سیستم از بخش *بالایی* کاملا مجزاست.
- تمامی *حالت* ها با استفاده از یک نوع داده شمارشی بیان می‌شود.
- **طراحی بخش‌پایینی:**
  - دارای ریست آسنکرون
    - تعیین حالت اولیه سیستم
  - ذخیره سنکرون nx_state در pr_state

```
PROCESS (reset, clock)
BEGIN
    IF (reset='1') THEN
        pr_state <= state0;
    ELSIF (clock'EVENT AND clock='1') THEN
        pr_state <= nx_state;
    END IF;
END PROCESS;
```

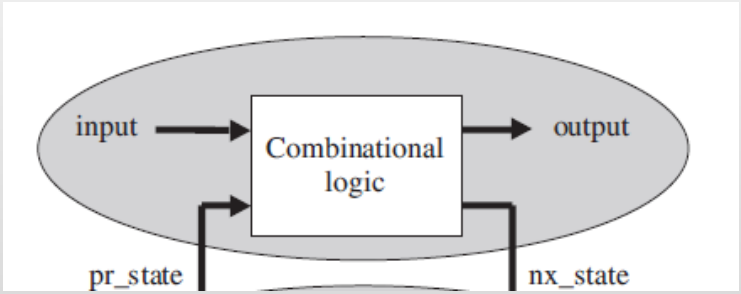


طراحی نوع #۱ (۲)

طراحی بخش بالایی:

- این بخش، یک مدار کاملا ترکیبی است، پس می‌توان با کد همروند نیز آن را پیاده‌سازی کرد، اما در این نوع طراحی ارائه شده، به دلیل استفاده از ساختار CASE، از کد ترتیبی استفاده شده است.

```
PROCESS (input, pr_state)
BEGIN
    CASE pr_state IS
        WHEN state0 =>
            IF (input = ...) THEN
                output <= <value>;      -- a. assigns the output value
                nx_state <= state1;     -- b. establishes the next state
            ELSE ...
            END IF;
        WHEN state1 =>
            IF (input = ...) THEN
                output <= <value>;
                nx_state <= state2;
            ELSE ...
            END IF;
...
    END CASE;
END PROCESS;
```



طراحی نوع #۱ (۳)

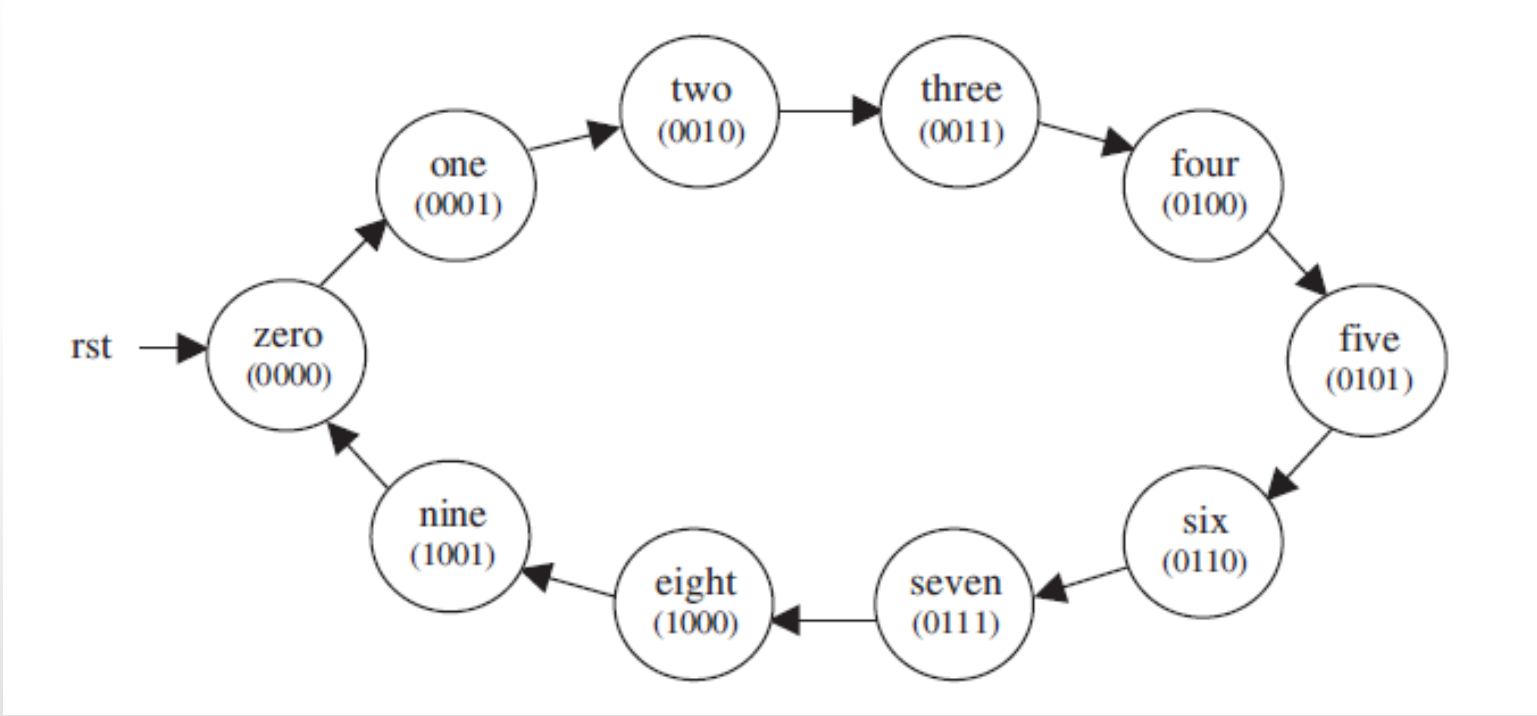
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------
ENTITY <entity_name> IS
   PORT ( input: IN <data_type>;
          reset, clock: IN STD_LOGIC;
          output: OUT <data_type>);
END <entity_name>;
-------------------------------------------------
ARCHITECTURE <arch_name> OF <entity_name> IS
   TYPE state IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: state;
BEGIN
   ---------- Lower section: ----------------------
   PROCESS (reset, clock)
   BEGIN
      IF (reset='1') THEN
         pr_state <= state0;
      ELSIF (clock'EVENT AND clock='1') THEN
         pr_state <= nx_state;
      END IF;
   END PROCESS;
```

طراحی نوع #۱ (۴)

```
---------- Upper section: ----------------------
PROCESS (input, pr_state)
BEGIN
   CASE pr_state IS
      WHEN state0 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state1;
         ELSE ...
         END IF;
      WHEN state1 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state2;
         ELSE ...
         END IF;
      WHEN state2 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state3;
         ELSE ...
         END IF;
      ...
   END CASE;
END PROCESS;
END <arch_name>;
```

طراحی نوع #۱ (۵)

- مثال: شمارنده **BCD**:



طراحی نوع #۱ (۶)

```
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
```

طراحی نوع #۱ (۷)

```
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;
```

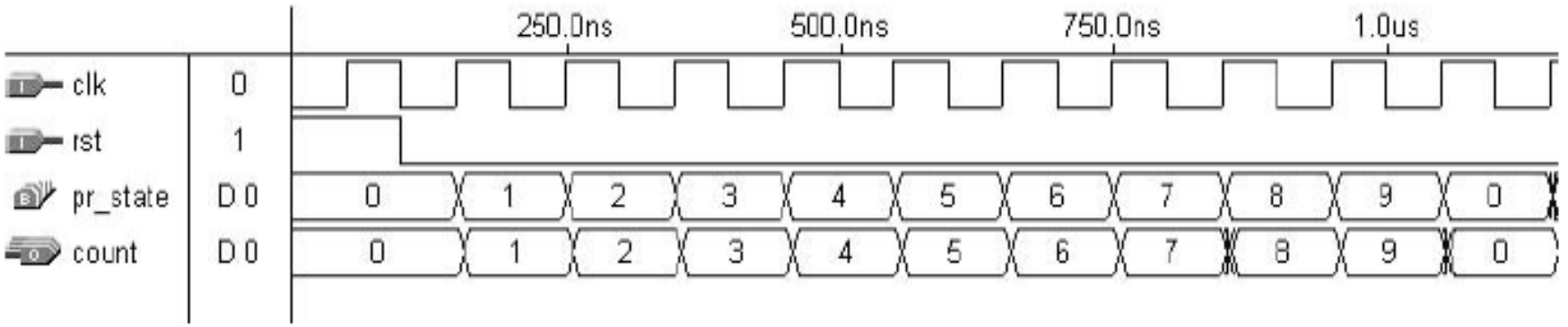
طراحی نوع #۱ (۸)

```
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;
```

طراحی نوع #۱ (۹)

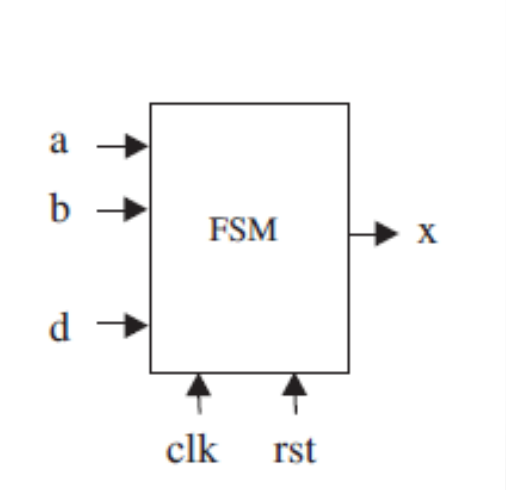
```
43                  WHEN five =>
44                       count <= "0101";
45                       nx_state <= six;
46                  WHEN six =>
47                       count <= "0110";
48                       nx_state <= seven;
49                  WHEN seven =>
50                       count <= "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 -------------------------------------------------
```

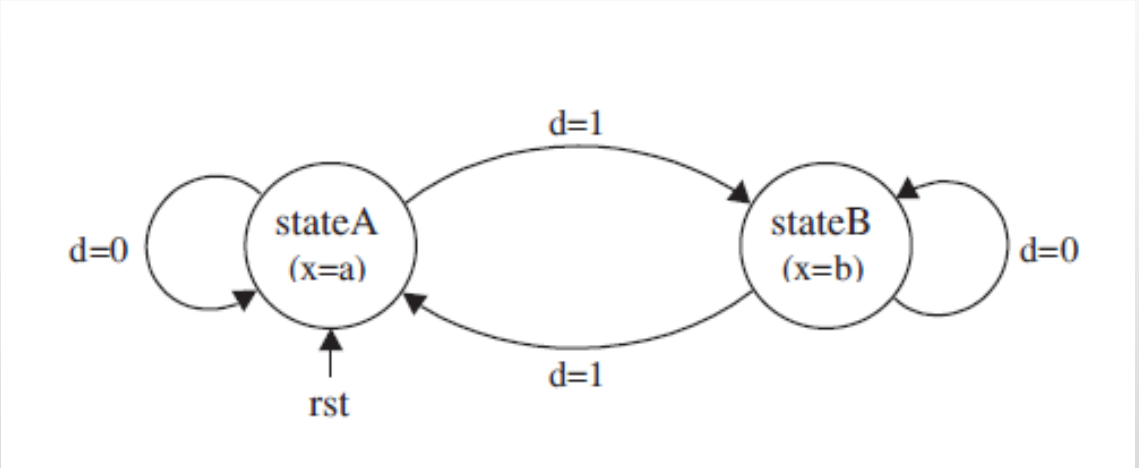
طراحی نوع #۱ (۱۰)



طراحی نوع #۱ (۱۱)

- **مثال:** ماشین حالت ساده #۱





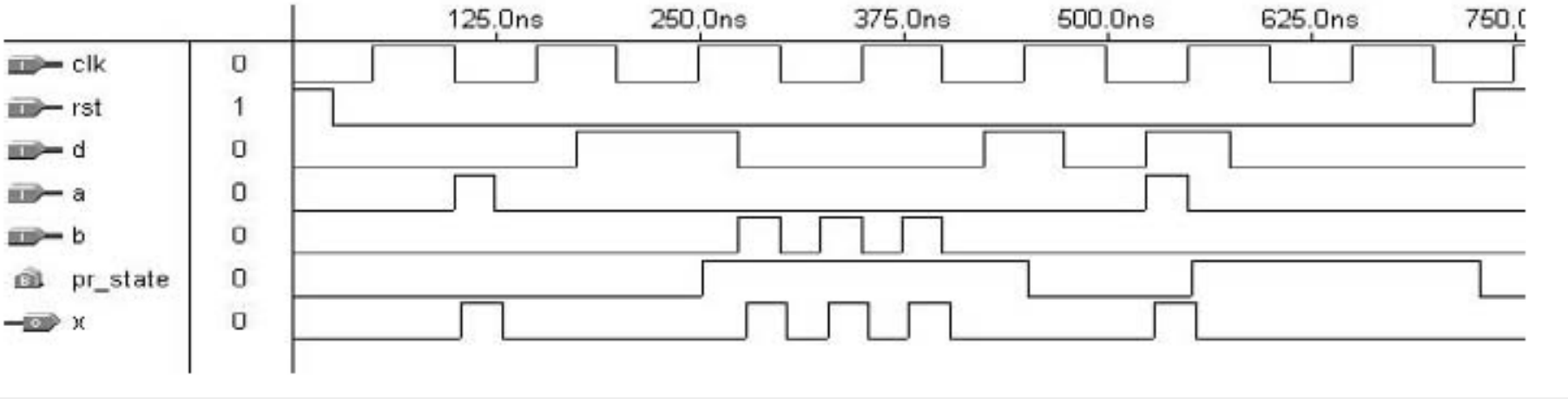
طراحی نوع #۱ (۱۲)

```
1 ------------------------------------------------
2 ENTITY simple_fsm IS
3      PORT ( a, b, d, clk, rst: IN BIT;
4             x: OUT BIT);
5 END simple_fsm;
6 ------------------------------------------------
7 ARCHITECTURE simple_fsm OF simple_fsm IS
8      TYPE state IS (stateA, stateB);
9      SIGNAL pr_state, nx_state: state;
10 BEGIN
11 ----- Lower section: ----------------------
12      PROCESS (rst, clk)
13      BEGIN
14          IF (rst='1') THEN
15              pr_state <= stateA;
16          ELSIF (clk'EVENT AND clk='1') THEN
17              pr_state <= nx_state;
18          END IF;
19      END PROCESS;
```

طراحی نوع #1 (۱۳)

```
20 ---------- Upper section: -----------------
21      PROCESS (a, b, d, pr_state)
22      BEGIN
23           CASE pr_state IS
24                WHEN stateA =>
25                     x <= a;
26                     IF (d='1') THEN nx_state <= stateB;
27                     ELSE nx_state <= stateA;
28                     END IF;
29                WHEN stateB =>
30                     x <= b;
31                     IF (d='1') THEN nx_state <= stateA;
32                     ELSE nx_state <= stateB;
33                     END IF;
34           END CASE;
35      END PROCESS;
36 END simple_fsm;
37 ------------------------------------------------
```

طراحی نوع #1 (۱۴)



طراحی نوع #۲ (خروجی ذخیره شده) (۱)

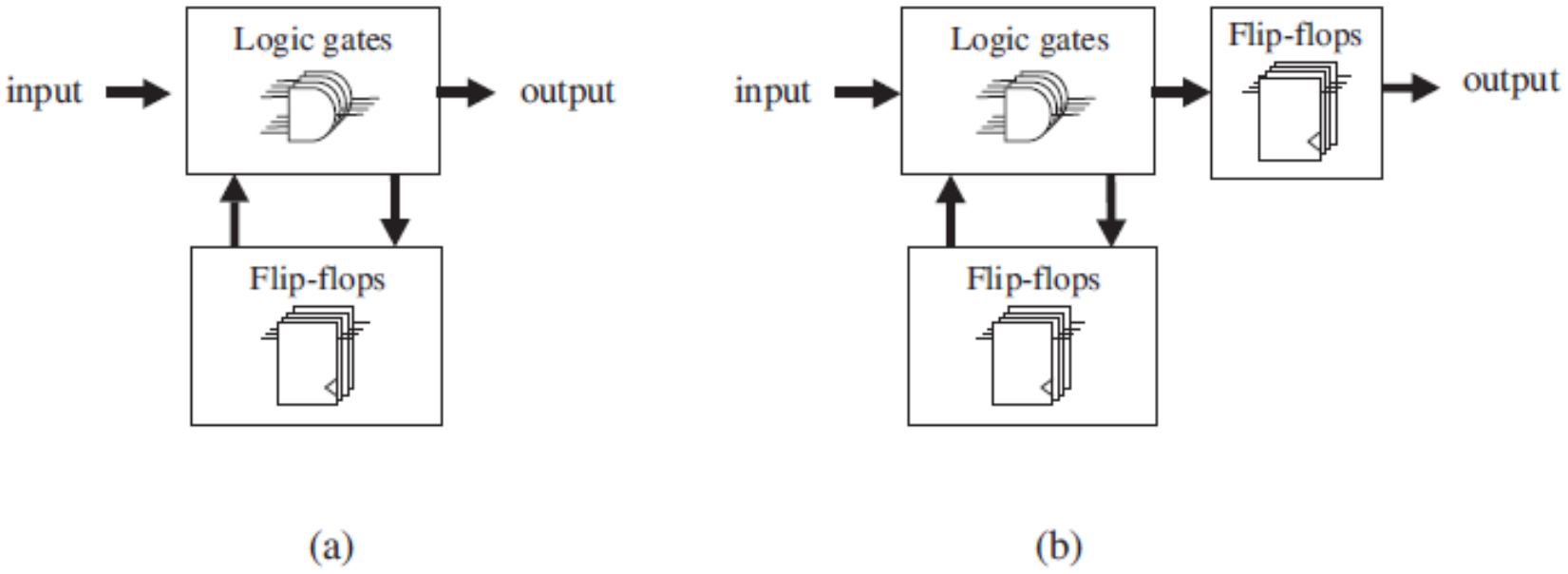


Figure 8.6
Circuit diagrams for (a) Design Style #1 and (b) Design Style #2.

طراحی نوع #۲ (خروجی ذخیره شده) (۲)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------------
ENTITY <ent_name> IS
   PORT (input: IN <data_type>;
         reset, clock: IN STD_LOGIC;
         output: OUT <data_type>);
END <ent_name>;
-------------------------------------------------------
ARCHITECTURE <arch_name> OF <ent_name> IS
   TYPE states IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: states;
   SIGNAL temp: <data_type>;
BEGIN
   ---------- Lower section: ------------------------
   PROCESS (reset, clock)
   BEGIN
      IF (reset='1') THEN
         pr_state <= state0;
      ELSIF (clock'EVENT AND clock='1') THEN
         output <= temp;
         pr_state <= nx_state;
      END IF;
   END PROCESS;
```

طراحی نوع #۲ (خروجی ذخیره شده) (۳)

```
---------- Upper section: ------------------------
PROCESS (pr_state)
BEGIN
   CASE pr_state IS
      WHEN state0 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state1;
         ...
         END IF;
      WHEN state1 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state2;
         ...
         END IF;
      WHEN state2 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state3;
         ...
         END IF;
      ...
   END CASE;
END PROCESS;
END <arch_name>;
```

طراحی نوع #۲ (خروجی ذخیره شده) (۴)

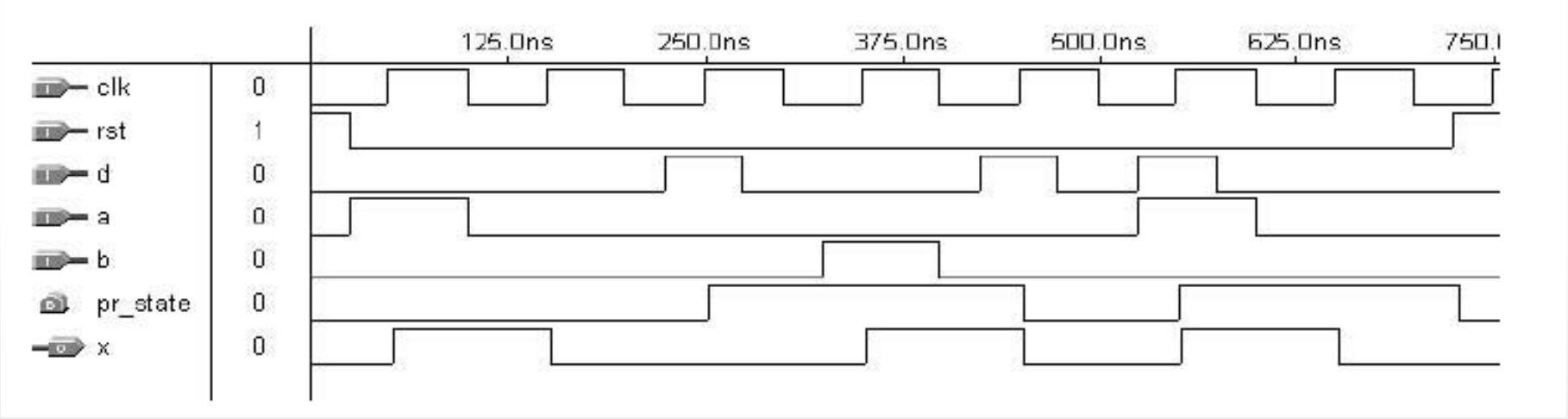
➢ **مثال:** ماشین حالت ساده #۲

```
1 ------------------------------------------
2 ENTITY simple_fsm IS
3      PORT ( a, b, d, clk, rst: IN BIT;
4             x: OUT BIT);
5 END simple_fsm;
6 ------------------------------------------
7 ARCHITECTURE simple_fsm OF simple_fsm IS
8      TYPE state IS (stateA, stateB);
9      SIGNAL pr_state, nx_state: state;
10     SIGNAL temp: BIT;
11 BEGIN
12     ----- Lower section: ---------------------
13     PROCESS (rst, clk)
14     BEGIN
15         IF (rst='1') THEN
16             pr_state <= stateA;
17         ELSIF (clk'EVENT AND clk='1') THEN
18             x <= temp;
19             pr_state <= nx_state;
20         END IF;
21 END PROCESS;
```

طراحی نوع #۲ (خروجی ذخیره شده) (۵)

```
22      ---------- Upper section: -----------------
23      PROCESS (a, b, d, pr_state)
24      BEGIN
25           CASE pr_state IS
26                WHEN stateA =>
27                     temp <= a;
28                     IF (d='1') THEN nx_state <= stateB;
29                     ELSE nx_state <= stateA;
30                     END IF;
31                WHEN stateB =>
32                     temp <= b;
33                     IF (d='1') THEN nx_state <= stateA;
34                     ELSE nx_state <= stateB;
35                     END IF;
36           END CASE;
37      END PROCESS;
38 END simple_fsm;
39 ------------------------------------------------
```

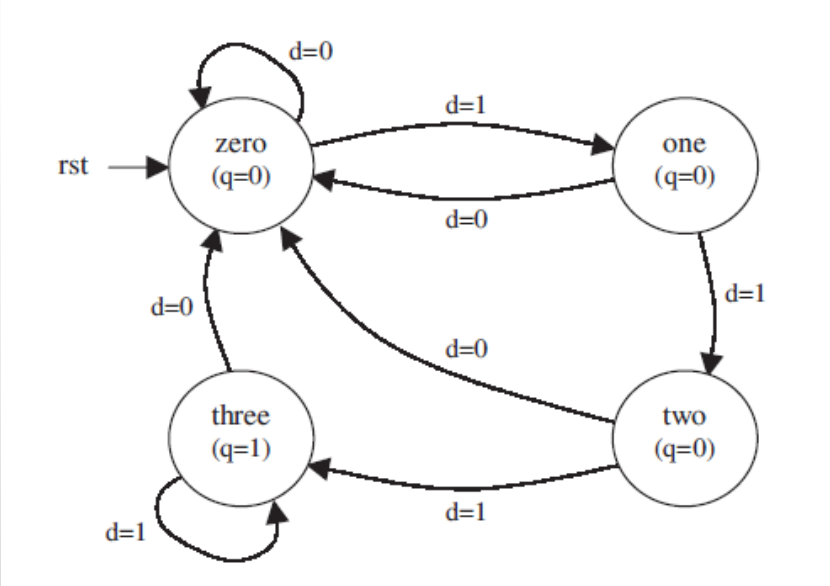
طراحی نوع #۲ (خروجی ذخیره شده) (۶)



تشخیص‌دهنده رشته (۱)

➢ مثال: تشخیص‌دهنده رشته (String Detector)

- سیستمی که یک جریان از ۰ و ۱ ها را به عنوان ورودی دریافت کند و هرگاه دنباله "111" را تشخیص داد، خروجی را '1' کند.
- تکرار نیز باید در نظر گرفته شود، یعنی در صورتی که جریان ورودی "...0111110" باشد، باید تا زمانی که سه ۱ کنار هم‌دیگر دیده می‌شوند، خروجی مقدار '1' داشته باشد.



تشخیص‌دهنده رشته (۲)

```
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;
```

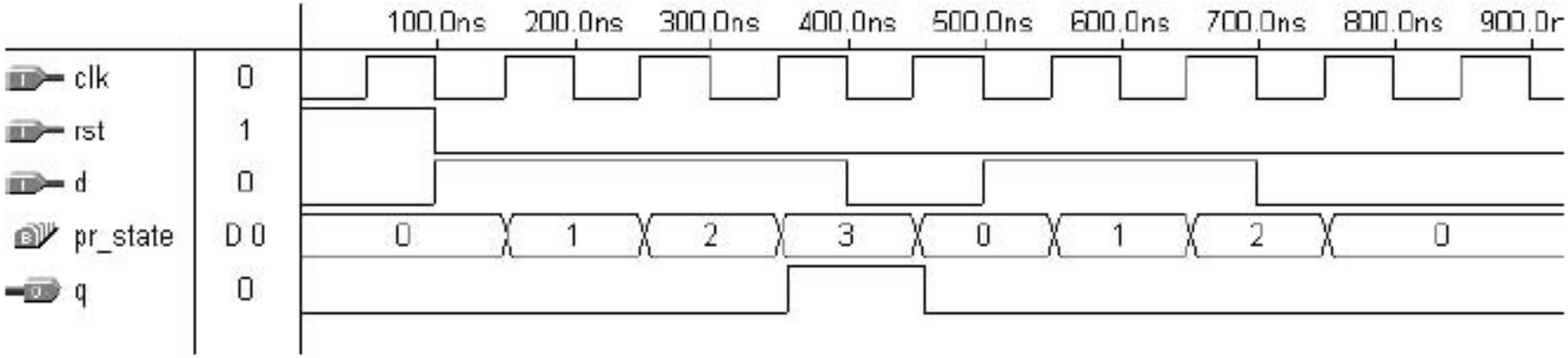
تشخیص‌دهنده رشته (۳)

```
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;
```

تشخیص‌دهنده رشته (۴)

```
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 ---------------------------------------------
```

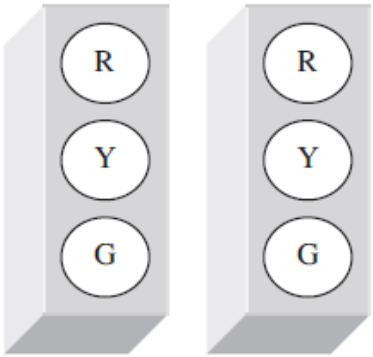
تشخیص‌دهنده رشته (۵)



کنترلرهای دیجیتال (۱)

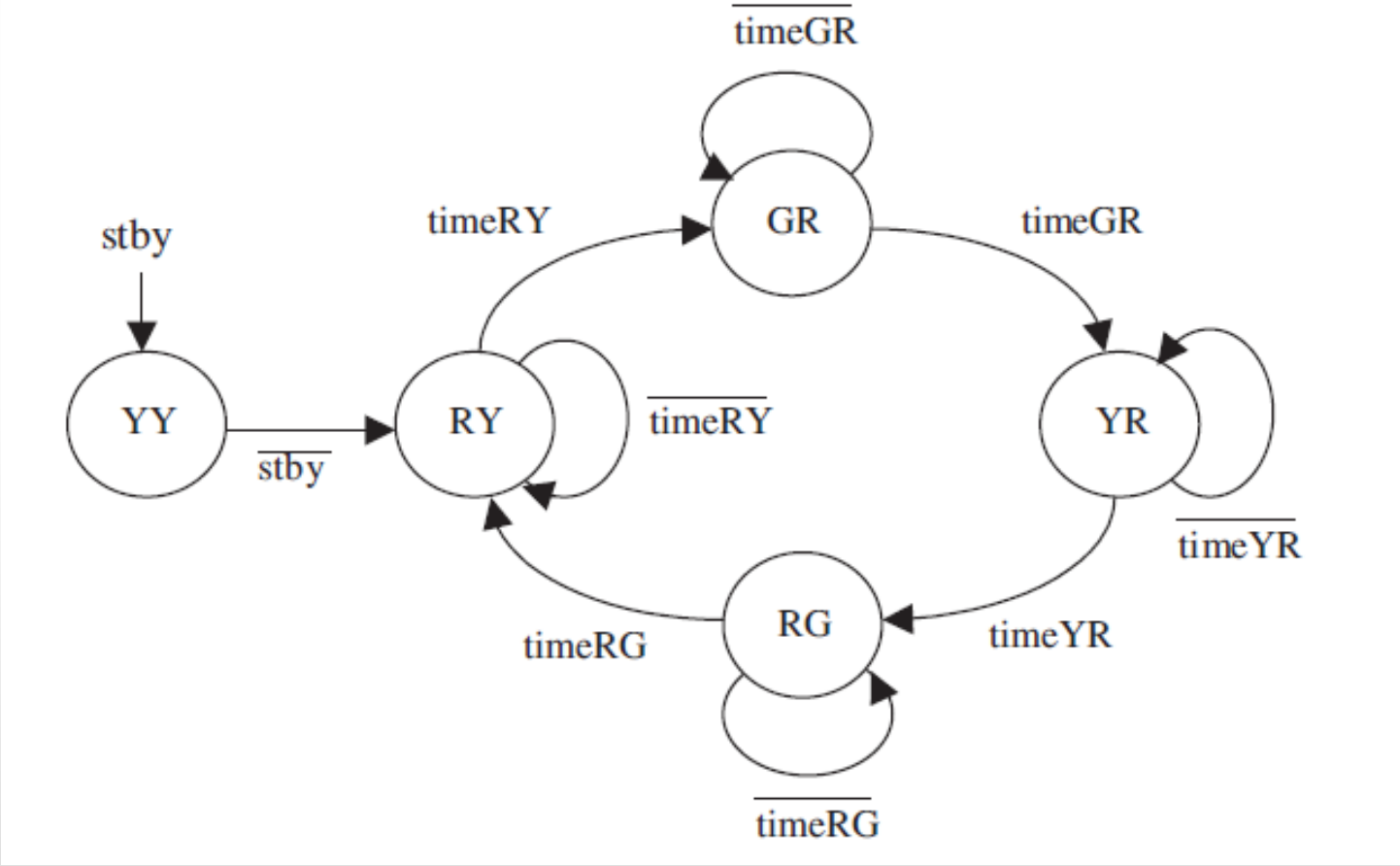
➢ مثال: کنترل کننده چراغ راهنمایی (Traffic Light Controller) (TLC)

- سه مد کاری: Regular، Test، Standby
- مد Regular: ۴ حالت، هر کدام دارای زمان مجزا و قابل تعیین با CONSTANT
- مد Test: امکان تغییر تمام زمان های مجزا از پیش برنامه‌ریزی شده در این مد توسط یک سوئیچ وجود دارد. زمان تست، توسط یک CONSTANT تعریف می‌شود.
- مد Standby: در صورتی که فعال شود، چراغ زرد در دو طرف روشن شده و تا زمانی که مد Standby فعال باشد، همین وضعیت ادامه پیدا می‌کند.
- فرض کنید پالس ساعتی با فرکانس ۶۰ هرتز موجود است.



State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite

کنترلرهای دیجیتال (۲)



کنترلرهای دیجیتال (۳)

```
1 ------------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------------
5 ENTITY tlc IS
6      PORT ( clk, stby, test: IN STD_LOGIC;
7             r1, r2, y1, y2, g1, g2: OUT STD_LOGIC);
8 END tlc;
9 ------------------------------------------------
10 ARCHITECTURE behavior OF tlc IS
11     CONSTANT timeMAX : INTEGER := 2700;
12     CONSTANT timeRG : INTEGER := 1800;
13     CONSTANT timeRY : INTEGER := 300;
14     CONSTANT timeGR : INTEGER := 2700;
15     CONSTANT timeYR : INTEGER := 300;
16     CONSTANT timeTEST : INTEGER := 60;
17     TYPE state IS (RG, RY, GR, YR, YY);
18     SIGNAL pr_state, nx_state: state;
19     SIGNAL time : INTEGER RANGE 0 TO timeMAX;
20 BEGIN
```

کنترلرهای دیجیتال (۴)

```
21      -------- Lower section of state machine: ----
22      PROCESS (clk, stby)
23      VARIABLE count : INTEGER RANGE 0 TO timeMAX;
24      BEGIN
25           IF (stby='1') THEN
26                pr_state <= YY;
27                count := 0;
28           ELSIF (clk'EVENT AND clk='1') THEN
29                count := count + 1;
30                IF (count = time) THEN
31                     pr_state <= nx_state;
32                     count := 0;
33                END IF;
34           END IF;
35 END PROCESS;
```

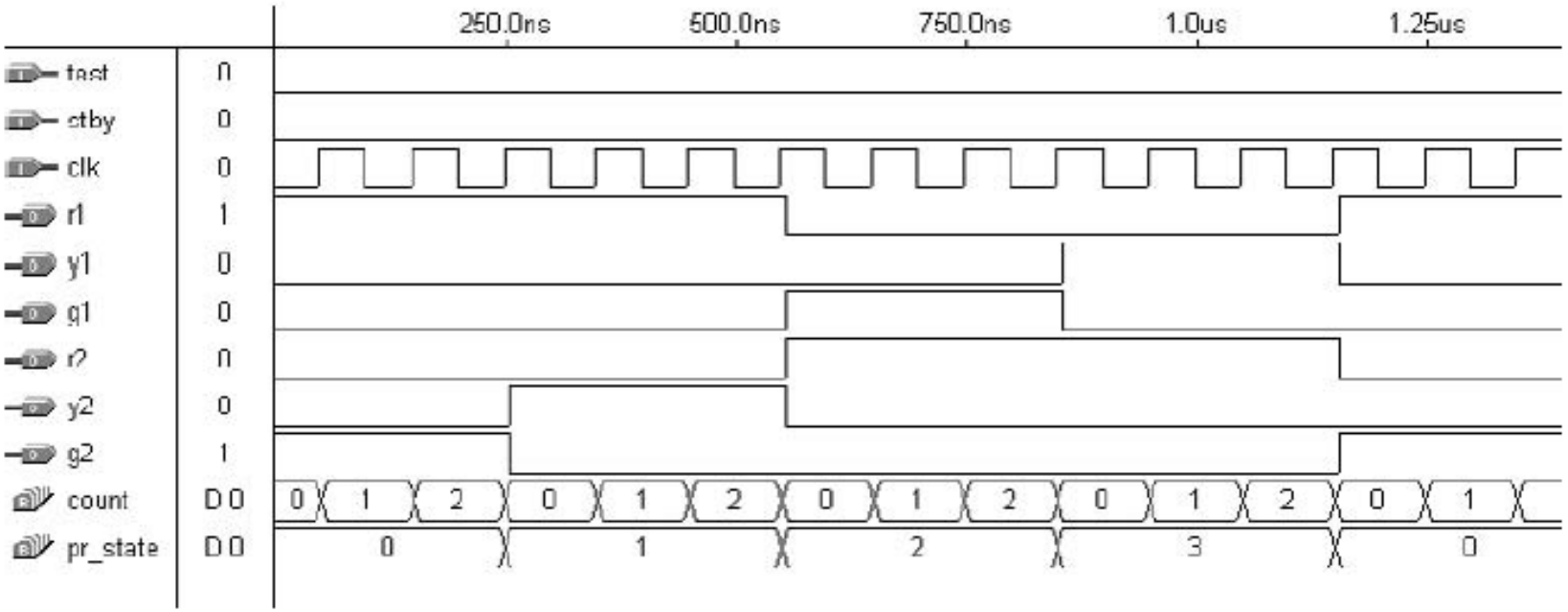
کنترلرهای دیجیتال (۵)

```
36      -------- Upper section of state machine: ----
37      PROCESS (pr_state, test)
38      BEGIN
39           CASE pr_state IS
40                WHEN RG =>
41                     r1<='1'; r2<='0'; y1<='0'; y2<='0'; g1<='0'; g2<='1';
42                     nx_state <= RY;
43                     IF (test='0') THEN time <= timeRG;
44                     ELSE time <= timeTEST;
45                     END IF;
46                WHEN RY =>
47                     r1<='1'; r2<='0'; y1<='0'; y2<='1'; g1<='0'; g2<='0';
48                     nx_state <= GR;
49                     IF (test='0') THEN time <= timeRY;
50                     ELSE time <= timeTEST;
51                     END IF;
```

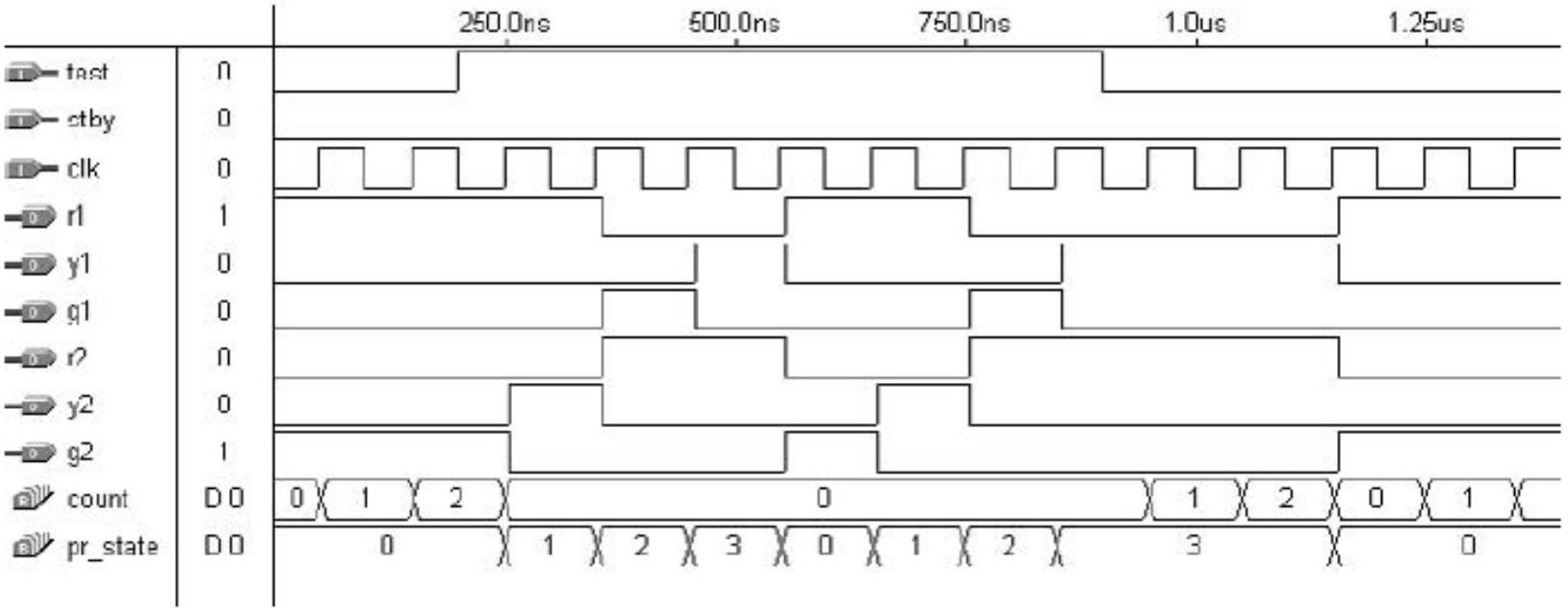
کنترلرهای دیجیتال (۶)

```
52                WHEN GR =>
53                     r1<='0'; r2<='1'; y1<='0'; y2<='0'; g1<='1'; g2<='0';
54                     nx_state <= YR;
55                     IF (test='0') THEN time <= timeGR;
56                     ELSE time <= timeTEST;
57                     END IF;
58                WHEN YR =>
59                     r1<='0'; r2<='1'; y1<='1'; y2<='0'; g1<='0'; g2<='0';
60                     nx_state <= RG;
61                     IF (test='0') THEN time <= timeYR;
62                     ELSE time <= timeTEST;
63                     END IF;
64                WHEN YY =>
65                     r1<='0'; r2<='0'; y1<='1'; y2<='1'; g1<='0'; g2<='0';
66                     nx_state <= RY;
67           END CASE;
68     END PROCESS;
69 END behavior;
70 ------------------------------------------------------
```

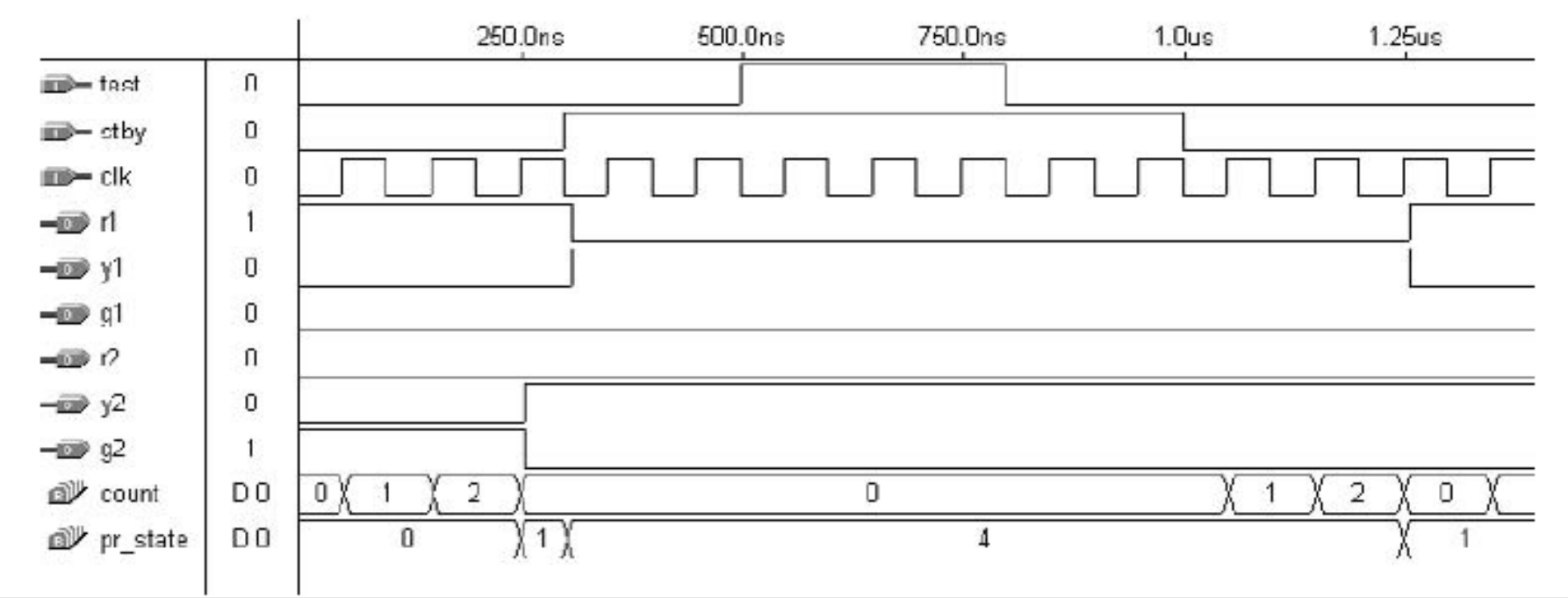
کنترلرهای دیجیتال (۷)



کنترلرهای دیجیتال (۸)



کنترلرهای دیجیتال (۹)



پایان بخش ماشین حالت