توصیف VHDL قابل سنتز

Synthesizable VHDL Description

- تاریخ امتحان
- تکالیف سری دوم
- مرحلهٔ اول پروژه: تا پنجشنبه
- ارسال ایمیل به szamani@aut.ac.ir
- مسابقه FPGA

کد قابل سنتز

- **چگونه کد قابل سنتز بنویسیم؟**
- **چگونه کد قابل سنتز خوب بنویسیم؟**
  - نکات کلی برای همهٔ ابزارها
  - نکات جزئی:
    - موردکاوی ابزار سنتز XST
    - مراجعه بر مستندات

کد قابل سنتز

- **یادآوری:**
  - ❑ طراح بی‌نیاز از سایر امکانات VHDL نیست.
    - توصیف testbench:
      - مثال: after برای تولید سیگنال‌های تست
      - مثال: خواندن و نوشتن فایل‌ها
      - مثال: مقایسهٔ خروجی‌های توصیف رفتاری طرح با خروجی‌های مدار سنتز شده
    - توصیف رفتاری پودمان‌های طرح‌های بزرگ در ابتدا

نیاز به کد رفتاری

- مثال:
  - سخت‌افزار محاسبهٔ $(x^2 + y^2)^{1/2}$
  - مقایسه با نتیجهٔ رفتاری

```
architecture TBENCH of ENT_BENCH is
   procedure WAVE_GEN (signal X:out integer, signal Y:out integer)
      variable I integer:= 0;
   begin
      for VX in 0 to 63 loop
         for VY in 0 to 63 loop
         begin
            X <= VX after I*10 ns;
            Y <= VY after I*10 ns;
            I:= I + 1;
          end loop;

    end procedure WAVE_GEN;
 begin
    WAVE_GEN (X, Y);
    L1:L_MODULE port map(X,Y,L);
    L_BEHAVE <= (X**2 + Y**2)**0.5;
    ERROR_SIG <= L_BEHAVE xor L;
 end architecture TBENCH;
```

انواع داده

- **انواع دادهٔ سنتزپذیر:**
  - bit و bit_vector
  - boolean
  - integer
    - به تعدادی بیت (سیم یا فلیپ فلاپ یا پورت) تبدیل می‌شود.
  - std_logic و std_ulogic
    - برخی از مقادیر آن قابل استفاده است
      - ‘1’, ‘0’, ‘Z’, ‘-’ (به طور محدود)
  - signed و unsigned
  - enumeration type
    - برای حالت‌های FSM

انواع داده

- انواع دادهٔ سنتزناپذیر:
  - real
  - character
  - string

دستورها

- **دستورهای سنتزپذیر:**
  - انتساب به سیگنال
  - انتساب به متغیر
  - if-then-else
  - case-when
  - when-else
  - with-select-when

دستورها

- دستورهای سنتزپذیر:
  - if-generate
    - با شرط static
  - for-generate
    - با تکرار static
  - for-loop
    - در بعضی ابزارها با تکرار static
  - while-loop
    - در بعضی ابزارها با شرط حلقهٔ static
  - wait
    - به شکل بسیار محدود (بعداً)

عملگرها

- **عملگرهای سنتزپذیر:**
  - منطقی:
    - and, or, xor, nand, nor, not, xnor
  - حسابی:
    - +, -, *
  - رابطه‌ای:
    - <, >, =, /=, <=, >=
  - شیفت
    - sra, ror, sra, …

عملگرها

- **عملگرهای سنتزناپذیر:**
  - **
  - /
  - mod
    - در شرایط خیلی خاص

توصیف سنتزپذیر RTL

توصیف مدارهای ترکیبی

- **با فرایند:**
  - ❑ عملیات منطقی یا حسابی: با عملگرهای مجاز
  - ❑ انتساب به سیگنال یا متغیرها
  - ❑ استفاده از دستورهای شرطی و مانند آن
  - ❑ همهٔ ورودی‌های مدار ترکیبی در لیست حساسیت
    - علت؟ نیاز به به‌هنگام شدن خروجی‌ها
    - اگر نگذاریم؟

توصیف مدارهای ترکیبی

- توصیف MUX

```
process (SEL, A, B, C, D)
begin
   if SEL = ”00” then
      Z <= A;
   elsif SEL = ”01” then
      Z <= B;
   elsif SEL = ”10” then
      Z <= C;
   else Z <= D;
   end if;
end process;
```

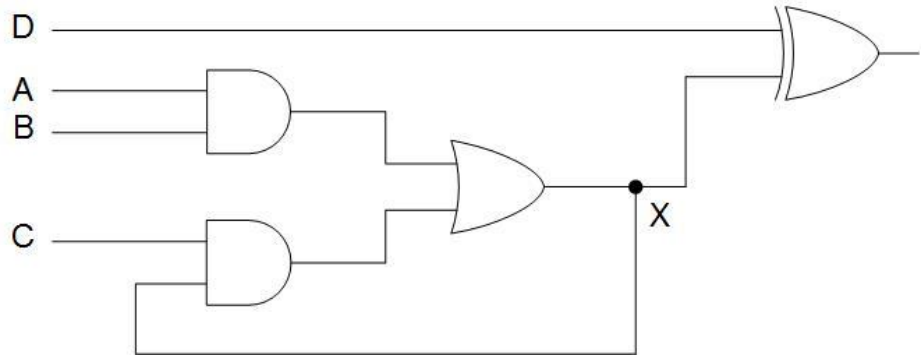
```
architecture …
begin
   Z <= A when SEL = "00" else
        B when SEL = "01" else
        C when SEL = "10" else
        D;
end architecture;
```

```
process (SEL, A, B, C, D)
begin
   case SEL is
      when "00" => Z <= A;
      when "01" => Z <= B;
      when "10" => Z <= C;
      when "11" => Z <= D;
   end case;
end process;
```

```
architecture …
begin
   with SEL select
      Z <= A when "00",
           B when "01",
           C when "10",
           D when others;
end architecture;
```

توصیف مدارهای ترکیبی

- **حلقهٔ ترکیبی:**
  - قابل سنتز
  - ولی اجتناب کنید: مشکلات زمانی:
    - ابهام نتیجه
    - نوسان
    - نمونه: SR-latch
  - ابزارهای سنتز معمولاً هشدار می‌دهند



- **توجه به هشدارها**
  - رفع کنید
  - حداقل بدانید چه می‌کنید

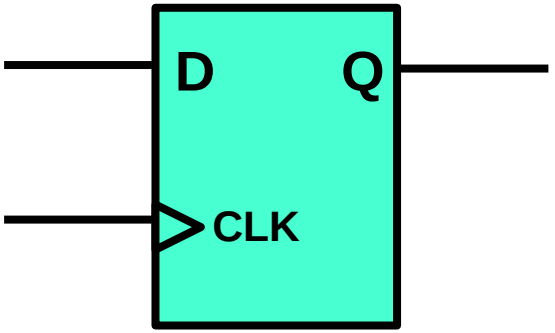
توصیف مدارهای ترتیبی

- **فلیپ فلاپ:**
  - در تراشه‌ها: عمدتاً DFF
    - بعضی تراشه‌ها: DFF با قابلیت برنامه‌ریزی به صورت TFF

```
process (CLK)
begin
   if (CLK'event and CLK = '1') then
      Q <= D;
   end if;
end process;
```

rising_edge(CLK)

not CLK'stable and CLK = '1'




```
process
begin
   wait until rising_edge(CLK);
   Q <= D;
end process;
```

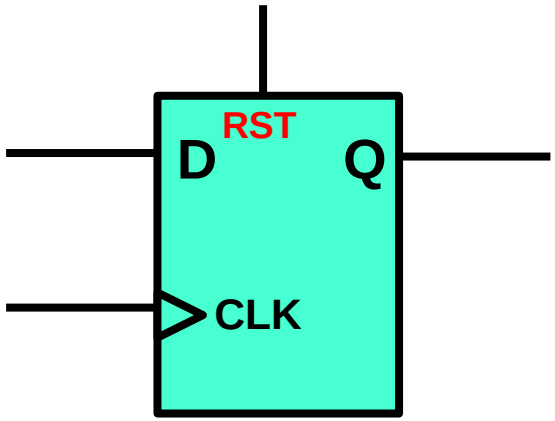
wait on CLK;
if (CLK = '1') then

توصیف مدارهای ترتیبی

- فلیپ فلاپ با بازنشانی:
  - همگام / ناهمگام

```
process (CLK, RST)
begin
   if (RST = '1') then
      Q <= '0';
   elsif (CLK'event and CLK = '1') then
      Q <= D;
   end if;
end process;
```

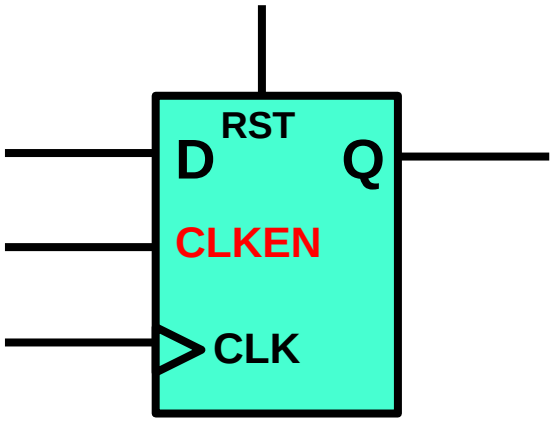
```
process (CLK)
begin
   if (CLK'event and CLK = '1') then
      if (RST = '1') then
         Q <= '0';
      else
         Q <= D;
      end if;
   end if;
end process;
```



توصیف مدارهای ترتیبی

- **فلیپ فلاپ با بازنشانی و فعال‌ساز کلاک:**
  - در زمان‌هایی نیازی به خروجی FFها نداریم
  - ← صرفه‌جویی در توان مصرفی

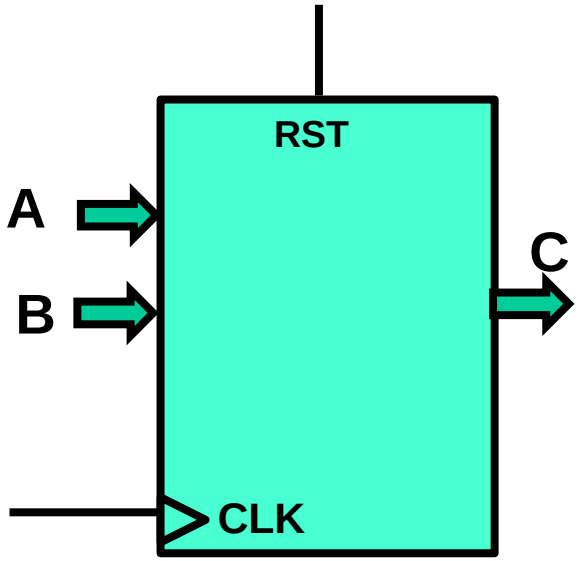
```
process
begin
   wait until rising_edge(CLK) and CLKEN = '1';
   Q <= D;
end process;
```



توصیف مدارهای ترتیبی

- عملیات منطقی/حسابی همگام با کلاک:

```
architecture …
begin
   signal A, B, C;
process (CLK)
begin
   if (CLK'event and CLK = '0') then
      C <= A * B;
   end if;
end process;
…
end architecture;
```



توصیف ماشین حالت متناهی

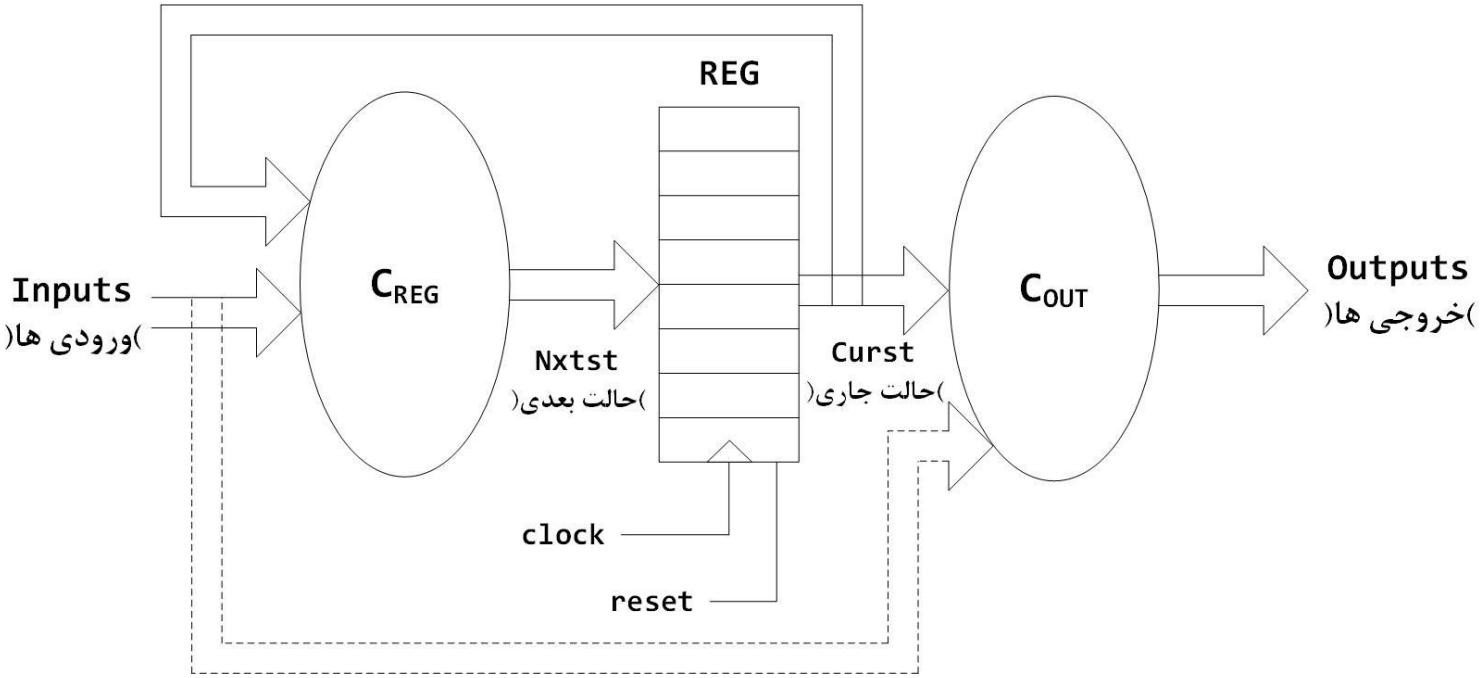
outputs = f_o (inputs, cur_state)

next_state = f_n (inputs, cur_state)

outputs = fo (cur_state)

next_state = fn (inputs, cur_state)

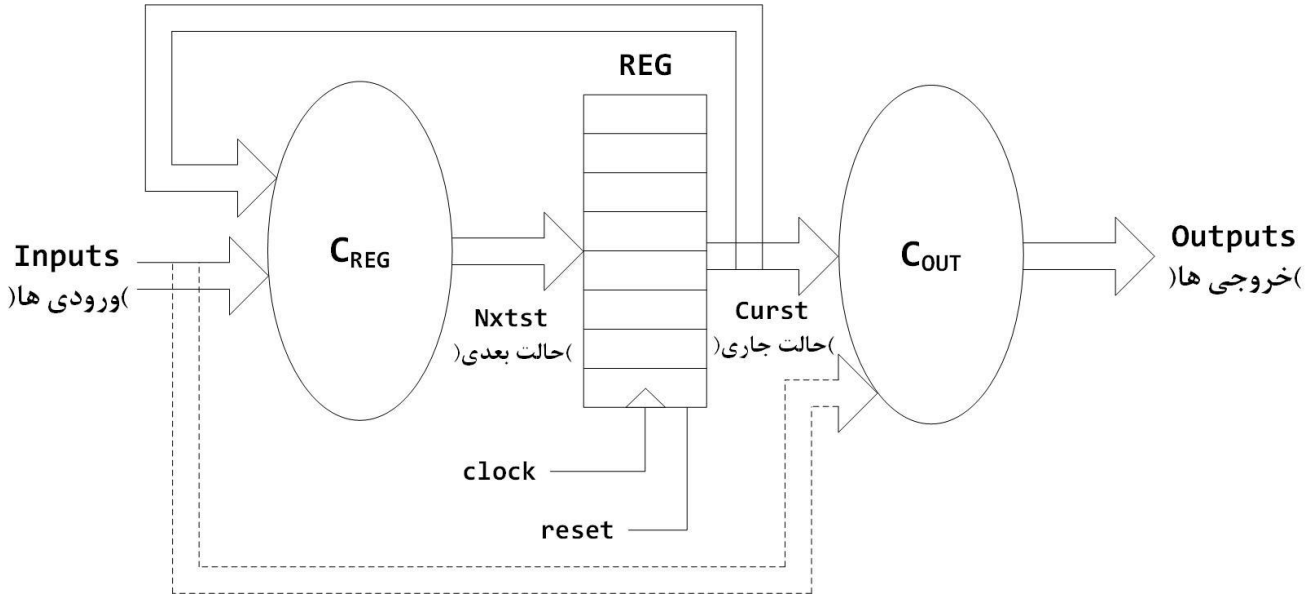
- **FSM:**
  - مدل کلی:
  - توصیف هر بخش با یک پروسس



توصیف ماشین حالت متناهی

- FSM:

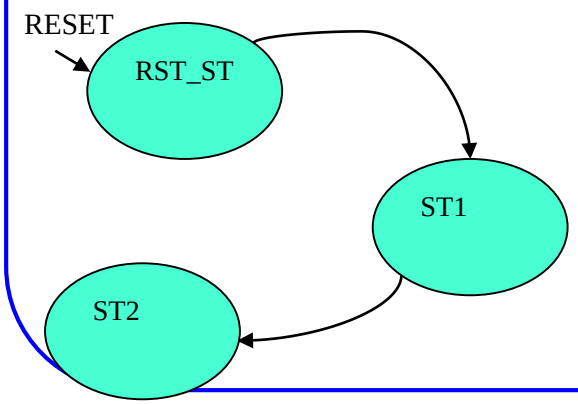
```
entity FSM is
   port(
      CLK : in ...
      RESET : in ...
      INPUTS : in ...
      OUTPUTS : out ...);
end FSM;
```

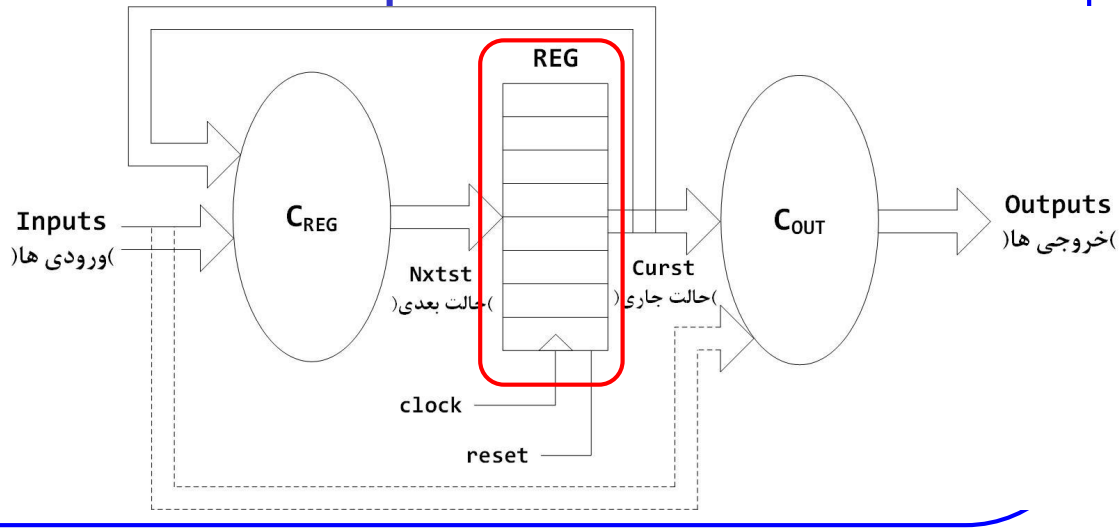


توصیف ماشین حالت متناهی

```
architecture arch ...

   type STATE_TYPE is (RST_ST, ST1, ...);
   signal CUR_STATE, NEXT_STATE : STATE_TYPE;

begin
   process(CLK)
   begin
      if rising_edge(CLK) then
         if RESET = '1' then
            CUR_STATE <= RST_ST;
         else
            CUR_STATE <= NEXT_STATE;
         end if;
      end if;
end process;
```

- **FSM: بخش ثبات‌ها**
  - ❑ تعداد FFها:
    - بستگی به تعداد حالت‌ها
    - بستگی به نوع کدگذاری

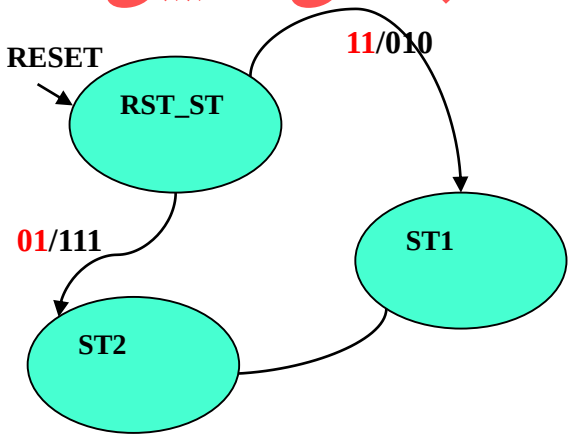


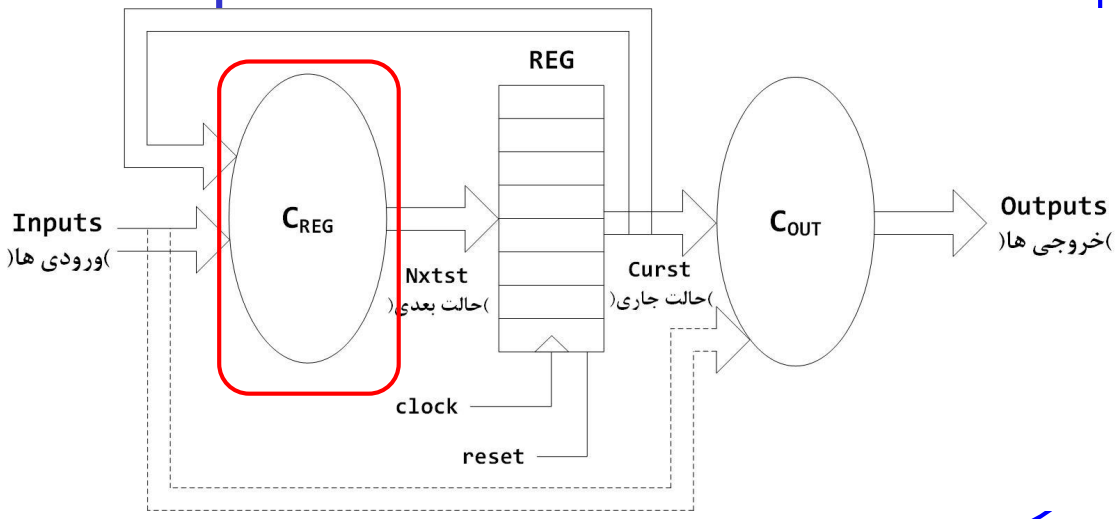


توصیف ماشین حالت متناهی

- FSM: بخش تعیین حالت بعدی

```
process(CUR_STATE, INPUTS)
begin
   case CUR_STATE is
      when RST_ST =>
         if (INPUTS = ...) then
            NEXT_STATE <= ...
         else
            NEXT_STATE <= ...;
         end if;
      when ST1 =>
         if (INPUTS = ...)
            NEXT_STATE <= ...
         else
            NEXT_STATE <= ...
         end if;
      when ...
...
end process;
```



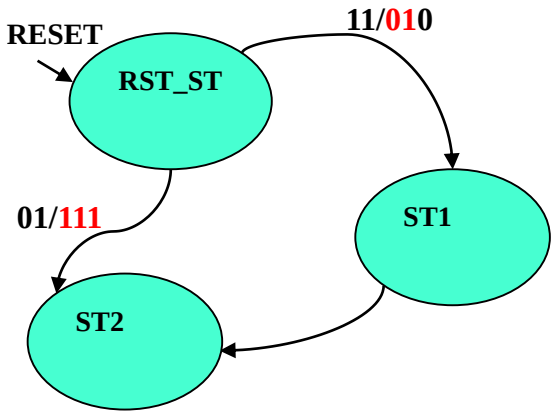


توصیف ماشین حالت متناهی

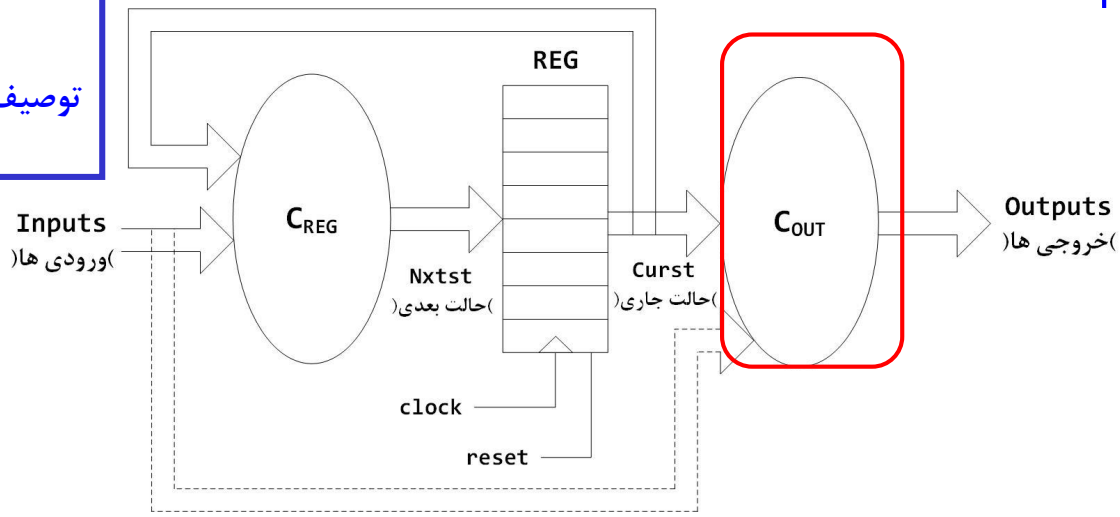
- **FSM: بخش تعیین خروجی**

```
process(CUR_STATE)
begin
    توصیف مدار ترکیبی و تولید خروجی‌ها در OUTPUTS
end process;
```

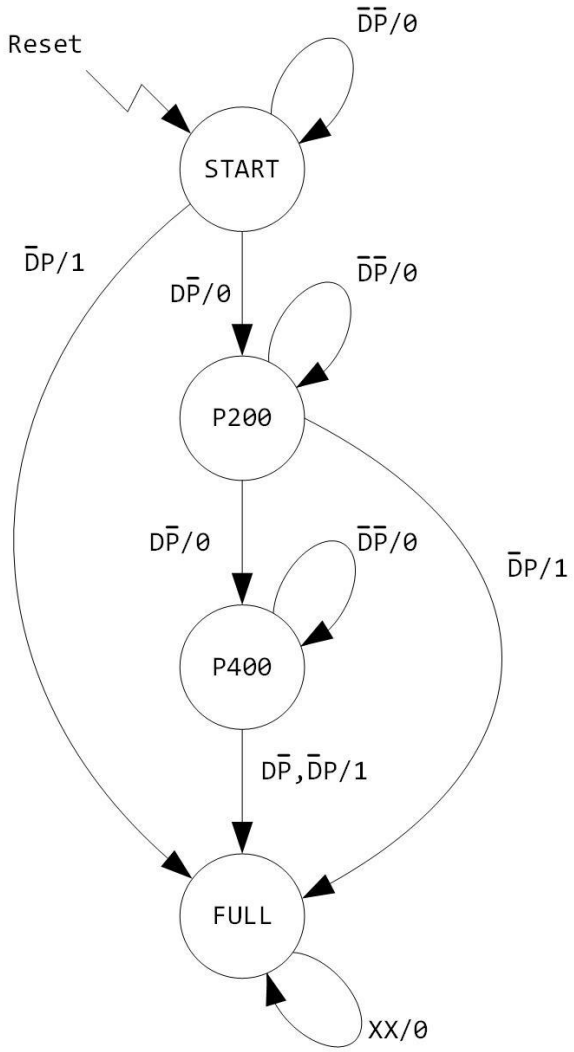
- میلی یا مور؟
  - میلی چطور؟




```
process(CUR_STATE, INPUTS)
begin
    توصیف مدار ترکیبی و تولید خروجی‌ها در OUTPUTS
end process;
```



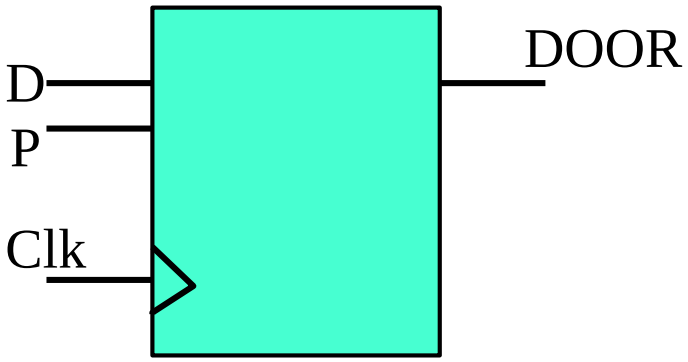
توصیف ماشین حالت متناهی



- **مثال: ماشین فروش شکلات:**
  - D: سکهٔ دویست تومانی
  - P: سکهٔ پانصد تومانی
  - هر دو با هم غیرممکن
  - قیمت شکلات: پانصد تومان
  - وقتی شکلات داد باید reset شود.

مثال: ماشین فروش شکلات

```
library ieee;
use ieee.std_logic_1164.all;
entity VENDING is
   port(
      CLK : in std_logic;
      RST : in std_logic;
      D, P : in std_logic;
      DOOR : out std_logic);
end VENDING;
--
```

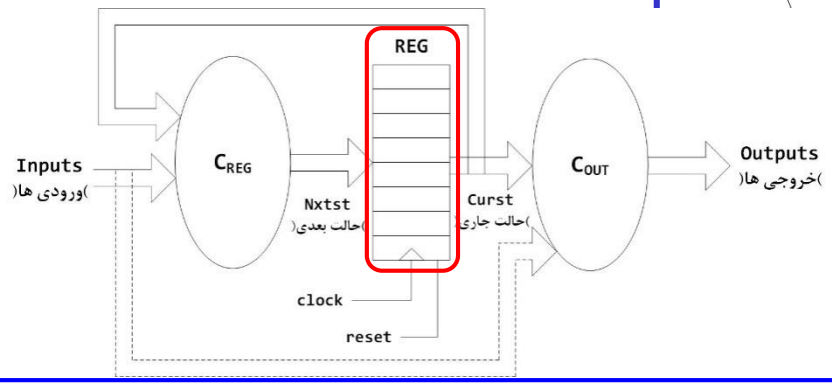


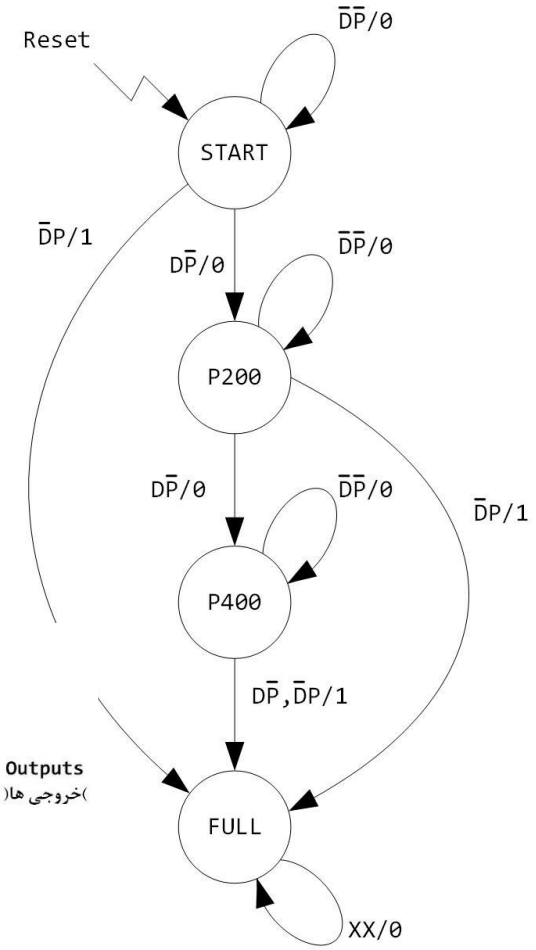
ماشین فروش شکلات

- توصیف ثبات حالت (FFها)

```
architecture RTL of VENDING is
   type ST_TYPE is (START, P200, P500, FULL);
   signal CUR, NEXT : ST_TYPE;
begin
   process(CLK)
   begin
      if rising_edge(CLK) then
         if RST = '1' then
            NEXT <= START;
         else
            NEXT <= CUR;
         end if;
      end if;
   end process;
```

Exchange!

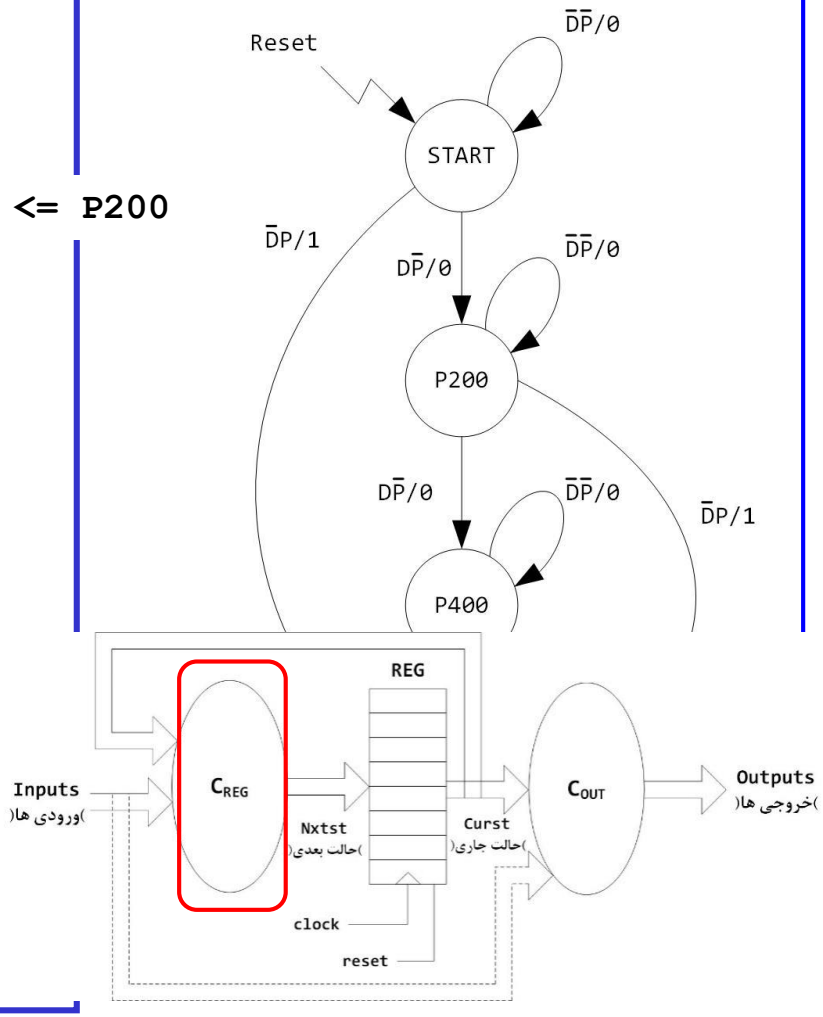




ماشین فروش شکلات

- توصیف حالت بعدی

```
process(CUR, D, P)
  begin
     case CUR is
        when START =>
           if (D, P) = "00" then
              NEXT <= START;
           elsif (D, P) = "01" then
              NEXT <= FULL;
           elsif (D, P) = "10" then
              NEXT <= P200;
           end if;                    else NEXT <= P200
        when P200 =>
           if (D, P) = "00" then
              NEXT <= P200;
           elsif (D, P) = "01" then
              NEXT <= FULL;
           else
              NEXT <= P400;
           end if;
        when P400 =>
           if (D, P) = "00" then
              NEXT <= P400;
           else
              NEXT <= FULL;
           end if;
        when FULL =>
           NEXT <= FULL;
  end process;
```



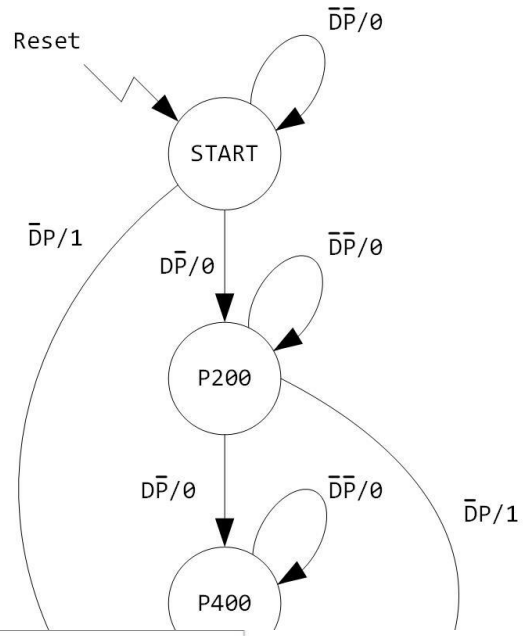
ماشین فروش شکلات

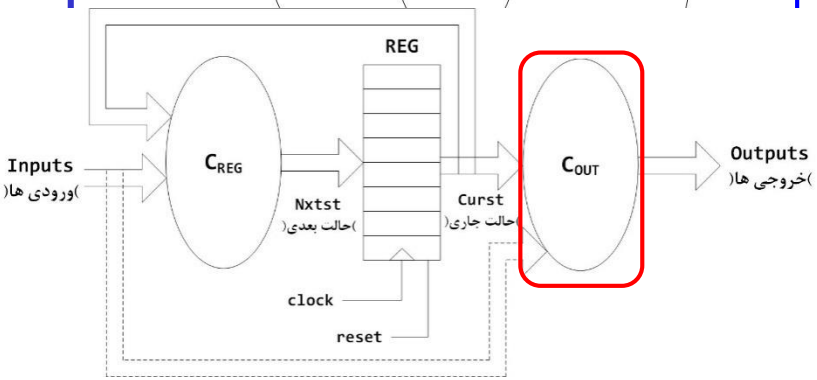
- توصیف خروجی

```
process(CUR, D, P)
     variable DP := (D, P);
  begin
     case CUR is
        when START =>
           if P = '0' then
              DOOR <= '0';
           else
              DOOR <= '1';
           end if;
        when P200 =>
           if P = '0' then
              DOOR <= '0';
           else
              DOOR <= '1';
           end if;
        when P400 =>
           if DP = "00" or DP = "11" then
              DOOR <= '0';
           else
              DOOR <= '1';
           end if;
        when FULL =>
           DOOR <= '0';
  end process;
end RTL;
```

DOOR <= P;

با if بهتر

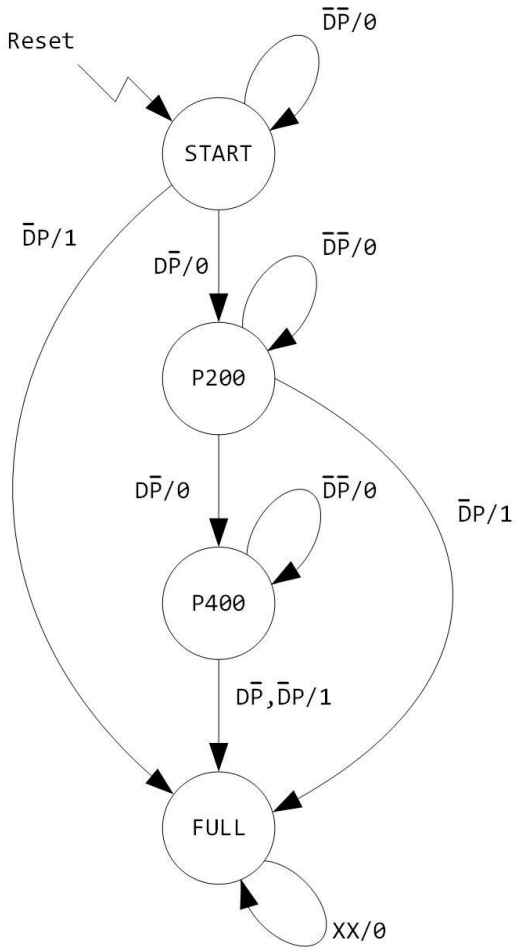


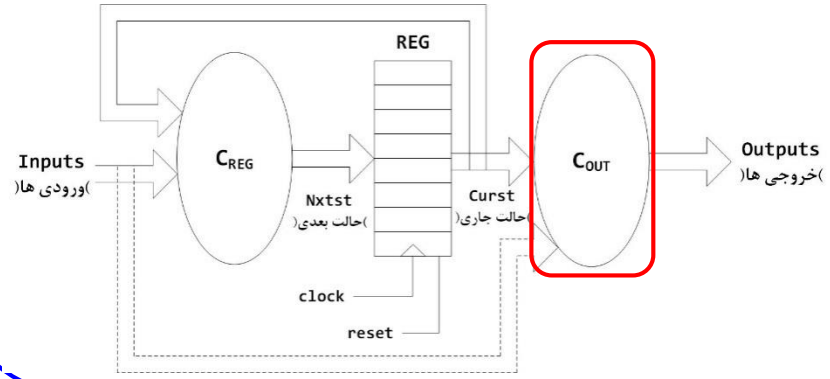


ماشین فروش شکلات

```
process(CUR)
  begin
     case CUR is
        when START || P200 || P400 =>
           DOOR <= '0';
        when FULL =>
           DOOR <= '1';
  end process;
```

- عملاً مور

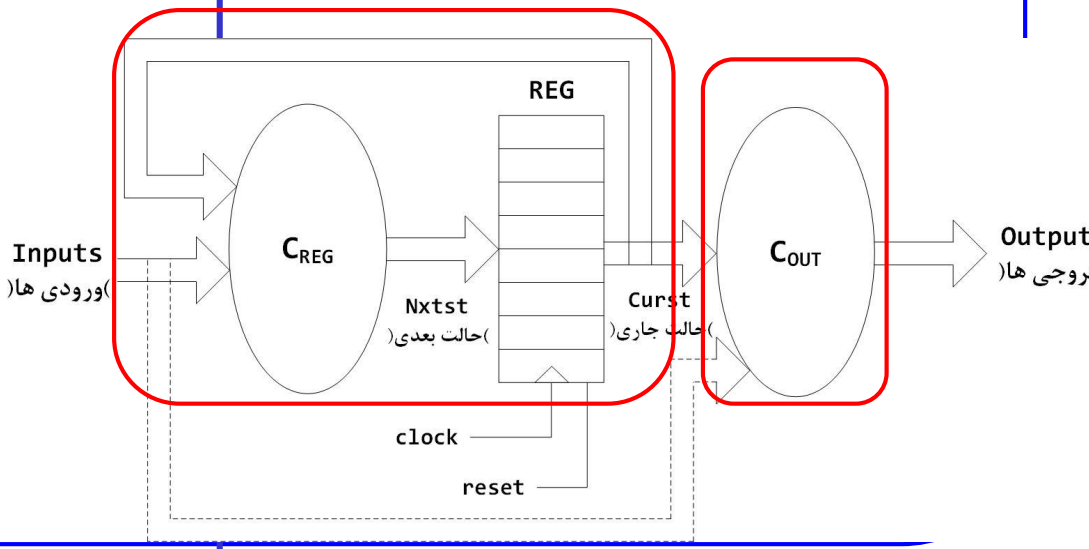




توصیف ماشین حالت متناهی

- FSM با دو فرایند:
  1. تغییر حالات
     - عدم نیاز به NEXT_STATE
  2. تولید خروجی

```
architecture arch ...

   type STATE_TYPE is (RST_ST, ST1, ...);
   signal STATE : STATE_TYPE;

begin
   process(CLK, RESET)
   begin
      if rising_edge(CLK) then
         if RESET = '1' then
            STATE <= RST_ST;
         else
            case STATE is
               when RST_ST =>
                  if (INPUTS = ...) then
                     STATE <= ...
                  else
                     STATE <= ...;
                  end if;
               when ST1 =>
                  ...
               when ...
...

         end if;
      end if;
   end process;
```

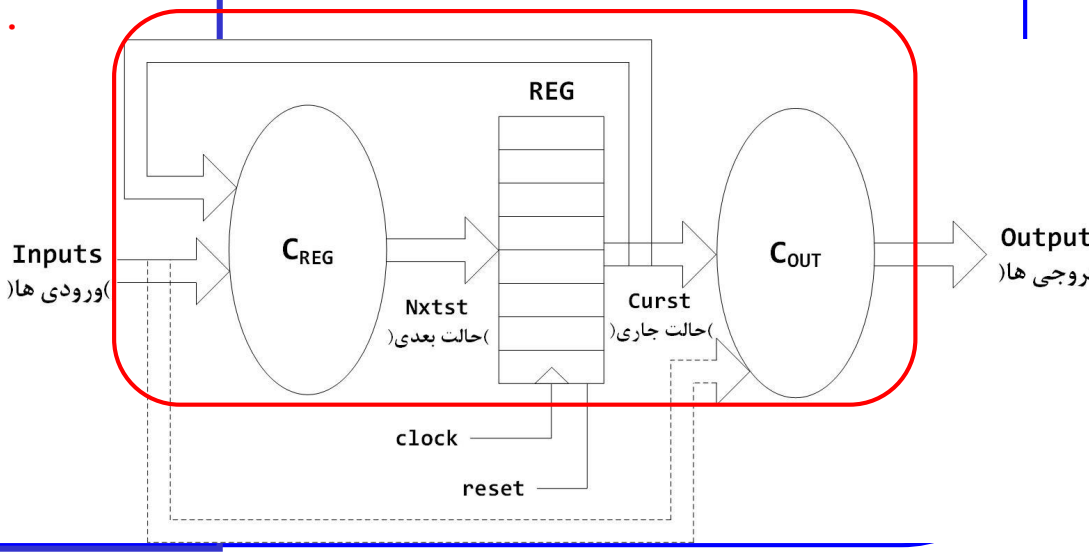


توصیف ماشین حالت متناهی

- FSM با یک فرایند:

1. تغییر حالات و تولید خروجی
   - حتماً به همهٔ خروجی‌ها مقدار داده شود.

```
architecture arch ...
   type STATE_TYPE is (RST_ST, ST1, ...);
   signal STATE : STATE_TYPE;
begin
   process(CLK, RESET)
   begin
      if rising_edge(CLK) then
         if RESET = '1' then
            STATE <= RST_ST;
            OUTPUTS <= ...
         else
            case STATE is
               when RST_ST =>
                  if (INPUTS = ...) then
                     STATE <= ...
                     OUTPUTS <= ...
                  else
                     STATE <= ...;
                     OUTPUTS <= ..
                  end if;
               when ST1 =>
...
               when ...
...
         end if;
      end if;
   end process;
```



تفاوت انواع توصیف FSM

- کدام بهتر است؟
  - ❑ حجم توصیف: یک یا دو فرایندی
    - حجم بیشتر: احتمال اشکال بیشتر (نه همیشه)
  - ❑ نزدیک بودن به سخت‌افزار: سه فرایندی
    - اشکال‌زدایی آسان‌تر: دسترسی به سیگنال‌های NEXT_STATE
      - در صورت وجود اشکال، مشاهدهٔ شکل موج آنها
  - ❑ نزدیک بودن به رفتار (دیاگرام حالت): دو فرایندی و یک فرایندی
    - دنبال کردن عملیات کد VHDL از روی دیاگرام حالت

انتساب کد حالات

- State Assignment یا State Encoding

  - دستی
    - کار بیشتر توسط طراح
    - نیاز به تغییر کد اگر انتساب عوض شود
    - توصیه نمی‌شود (مگر در موارد خاص)

```
subtype STATE_TYPE is std_logic_vector (2 downto 0);
signal STATE: STATE_TYPE;

constant START: STATE_TYPE := "000";
constant S1: STATE_TYPE := "001";
constant S2: STATE_TYPE := "010";
constant S3: STATE_TYPE := "011";
constant S4: STATE_TYPE := "100";
constant S5: STATE_TYPE := "101";
```

انتساب کد حالات

- **انتساب کد حالات**
  - تکیه به تشخیص ابزار (انتساب پیش‌فرض ابزار)
    - نامشخص (portability)
      - تغییر ابزار ← نتایج متفاوت
    - عدم کنترل کافی

انتساب کد حالات

- انتساب کد حالات
  - به صورت محدودیت
    - در کد VHDL

```
type STATE_T is (STATE0, STATE1, STATE2, STATE3);
signal STATE, NEXT_STATE : STATE_T;

attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "one-hot";
```

Xilinx

```
type STATE_T is (STATE0, STATE1, STATE2, STATE3);
attribute enum_encoding : string;
attribute enum_encoding of STATE_T : type is "one-hot";
```

Altera

انتساب کد حالات

- **انتساب کد حالات**
  - به صورت محدودیت
    - در فایل محدودیت‌ها (XCF)

```
BEGIN MODEL "entity_name"
NET "signal_name" fsm_encoding=one-hot;
END;
```

Xilinx

انتساب کد حالات

- انتساب کد حالات
  - به صورت محدودیت
    - به صورت script یا در خط فرمان

```
run … -fsm_encoding one-hot
```

Xilinx

انواع کدهای حالت

- **انواع کدها:**
  - باینری یا ترتیبی (sequential)
    - تعداد فلیپ فلاپ برای N_s حالت:

$$N_f = \lceil \log_2 N_s \rceil$$

STATE0: 00

STATE1: 01

STATE2: 10

STATE3: 11

```
type STATE_T is (STATE0, STATE1, STATE2, STATE3);
signal STATE, NEXT_STATE : STATE_T;

attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "sequential";
```

- اگر تعداد حالات $\neq 2^i$
  - حالات اضافی دردسرساز

انواع کدهای حالت

- **انواع کدها:**
  - ❑ تک فعال (one-hot)
    - تعداد فلیپ فلاپ:

$$N_f = N_s$$

STATE0: 0001
STATE1: 0010
STATE2: 0100
STATE3: 1000

```
type STATE_T is (STATE0, STATE1, STATE2, STATE3);
signal STATE, NEXT_STATE : STATE_T;

attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "one_hot";
```

انواع کدهای حالت

- **انواع کدها:**
  - کد گری (gray)
    - تعداد فلیپ فلاپ:

$$N_f = \lceil \log_2 N_s \rceil$$

STATE0: 000
STATE1: 001
STATE2: 011
STATE3: 010
STATE4: 110
STATE5: 111
STATE6: 101
STATE7: 100

```
type STATE_T is (STATE0, STATE1, STATE2, STATE3, STATE4, STATE5,
STATE6, STATE7);
signal STATE, NEXT_STATE : STATE_T;

attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "gray";
```

انواع کدهای حالت

- **انواع کدها:**
  - کد جانسون (johnson)
    - تعداد فلیپ فلاپ:

N_f = ?

STATE0: 0000
STATE1: 1000
STATE2: 1100
STATE3: 1110
STATE4: 1111
STATE5: 0111
STATE6: 0011
STATE7: 0001

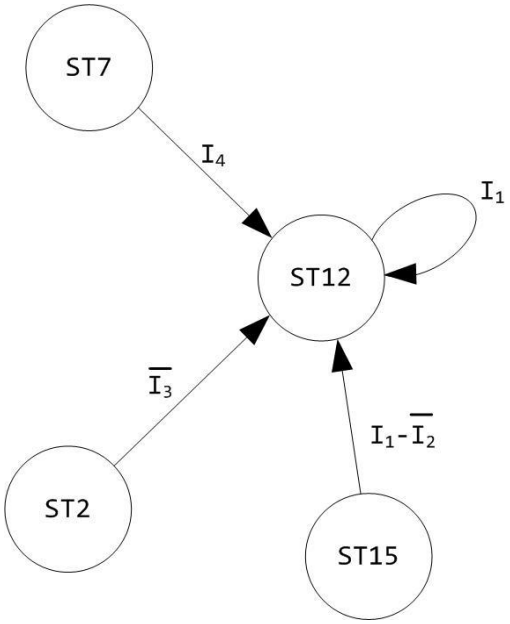
```
type STATE_T is (STATE0, STATE1, STATE2, STATE3, STATE4, STATE5,
STATE6, STATE7);
signal STATE, NEXT_STATE : STATE_T;

attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "johnson";
```

انتخاب نوع کدهای حالت

- **کدام بهتر است؟**
  - ❑ auto:
    - از یک ابزار به ابزار دیگر تغییر می‌کند
    - از یک نسخه به نسخهٔ دیگر تغییر می‌کند
    - ← بعد از ماه‌ها کار که طرح کار می‌کرده، با تغییر ابزار کار نمی‌کند
  - ❑ sequential و gray:
    - کمترین تعداد FF
    - مدارهای ترکیبی بزرگ‌تر
  - ❑ one-hot:
    - تعداد FF زیاد
    - مدارهای ترکیبی بیشتر ولی بسیار کوچک‌تر

انتخاب نوع کدهای حالت

❑ sequential: مثال



حالت جاری	کد حالت جاری				ورودی ها					حالت بعدی			
	S3	S2	S1	S0	I_0	I_1	I_2	I_3	I_4	S3	S2	S1	S0
ST0	0	0	0	0			⋮					⋮	
ST1	0	0	0	1			⋮					⋮	
ST2	0	0	1	0	X	X	X	0	X	1	1	0	0
⋮			⋮				⋮					⋮	
ST7	0	1	1	1	X	X	X	X	1	1	1	0	0
⋮			⋮				⋮					⋮	
ST12	1	1	0	0	X	1	X	X	X	1	1	0	0
⋮			⋮				⋮					⋮	
ST15	1	1	1	1	X	1	0	X	X	1	1	0	0

$D_3 = \ldots + (S3'.S2'.S1.S0').I_3' + \ldots + (S3'.S2.S1.S0).I_4 + \ldots + (S3.S2.S1'.S0').I_1 + \ldots + (S3.S2.S1.S0).I_1.I_2'$

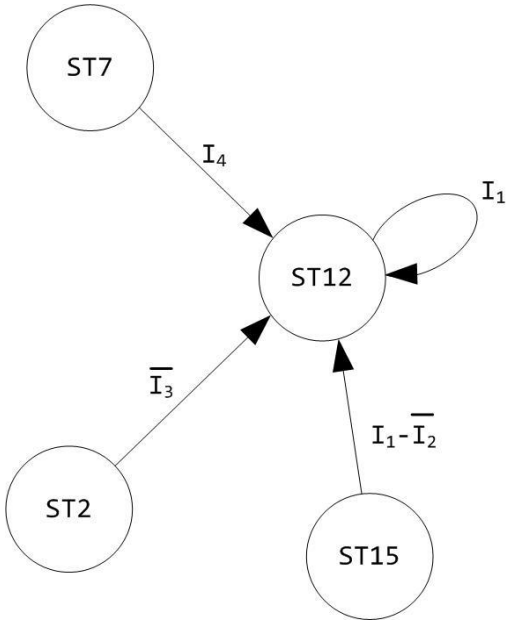
$D_2 = \ldots + (S3'.S2'.S1.S0').I_3' + \ldots + (S3'.S2.S1.S0).I_4' + \ldots + (S3.S2.S1'.S0').I_1 + \ldots + (S3.S2.S1.S0).I_1.I_2'$

$D_1 = \ldots$

$D_0 = \ldots$

انتخاب نوع کدهای حالت

❑ One-hot: مثال

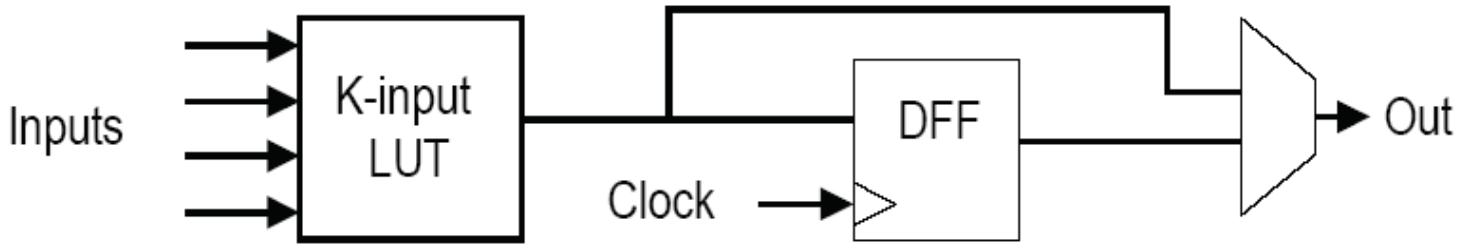


حالت جاری	کد حالت جاری S15 S14 …S0	ورودی‌ها I0 I1 I2 I3 I4	کد حالت بعدی
ST0	**0000000000000001**		
ST1	**0000000000000010**		
ST2	**0000000000000100**	**xxx0x**	**0001000000000000**
ST3	**0000000000001000**		
ST4	**0000000000010000**		
ST5	**0000000000100000**		
ST6	**0000000001000000**		
ST7	**0000000010000000**	**xxxx1**	**0001000000000000**
ST8	**0000000100000000**		
ST9	**0000001000000000**		
ST10	**0000010000000000**		
ST11	**0000100000000000**		
ST12	**0001000000000000**	**x1xxx**	**0001000000000000**
ST13	**0010000000000000**		
ST14	**0100000000000000**		
ST15	**1000000000000000**	**x10xx**	**0001000000000000**

$$D_{12} = S_2.I_3 + S_7.I_4 + S_{12}.I_1 + S_{15}.I_1.I_2'$$

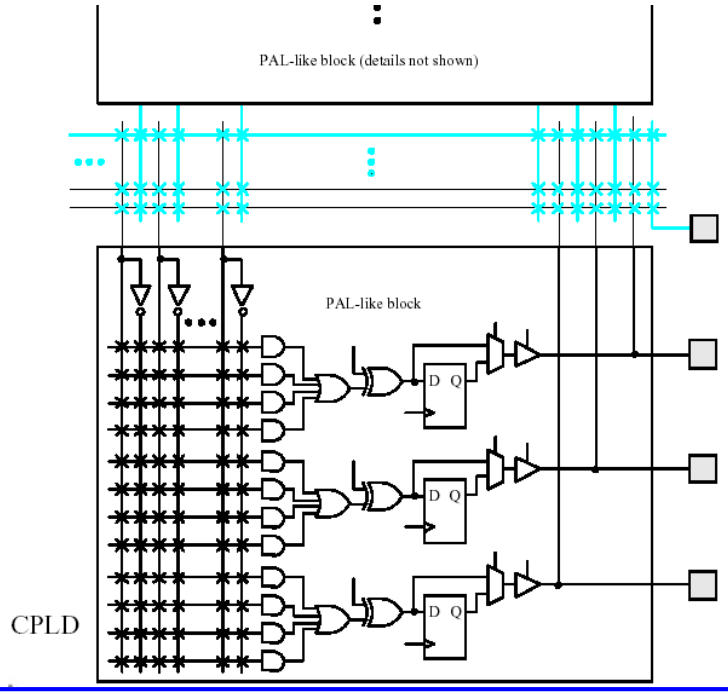
انتخاب نوع کدهای حالت

- کدام بهتر است؟
  - ❑ one-hot:
    - در FPGA، تعداد FFها زیاد، مدار ترکیبی به ازای هر FF کوچک
    - ← one-hot برای FPGA مناسب‌تر



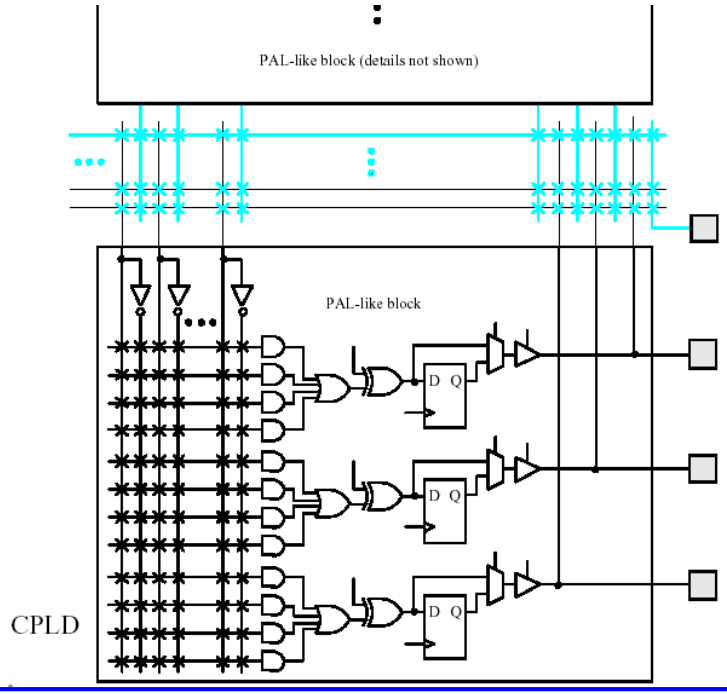
انتخاب نوع کدهای حالت

- **کدام بهتر است؟**
  - ❑ Sequential و gray:
    - در CPLD، تعداد FFها کم، مدار ترکیبی به ازای هر FF بزرگ
    - sequential و gray برای CPLD مناسب‌تر ←



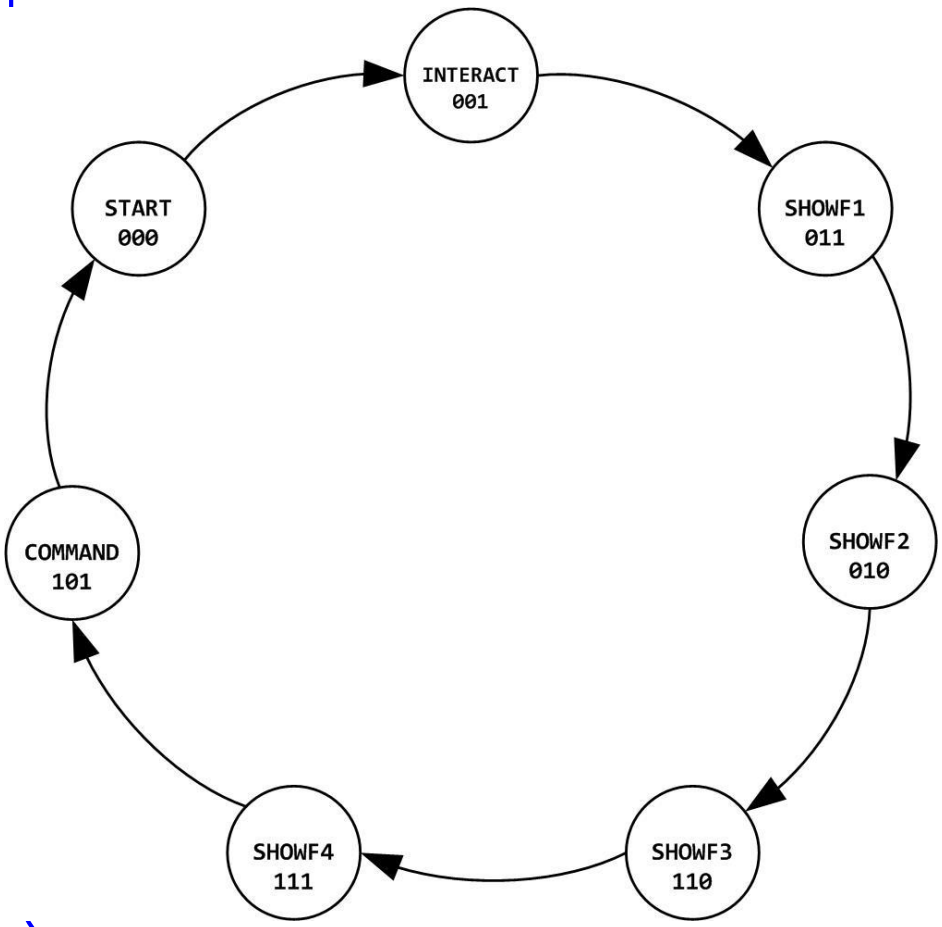
انتخاب نوع کدهای حالت

- کدام بهتر است؟
  - ❑ one-hot:
    - مدارهای ترکیبی کوچک ← تأخیر بین FFها: کم ← فرکانس کلاک بالاتر



انتخاب نوع کدهای حالت

- **کدام بهتر است؟**
  - gray (در برابر sequential):
    - برای FSMهای با ترتیب مشخص (مثل شمارنده)، انتخاب مناسب کد
    - ← تغییر کمتر بیت‌ها
    - ← توان مصرفی کمتر



انتخاب نوع کدهای حالت

- کدام بهتر است؟
  - johnson:
    - مانند gray، یک تغییر در هر بیت
    - تعداد FF بیشتر
    - مدار ترکیبی کوچک‌تر از gray

انتخاب نوع کدهای حالت

- **کدام بهتر است؟**
  - ❑ one-hot در برابر sequential از نظر توان مصرفی:
    - one-hot دو تغییر بیت در هر گونه تغییر حالت
      - ← توان کمتر از sequential
    - one-hot: تعداد FF بیشتر و مدارهای ترکیبی بیشتر از sequential
      - ← توان بیشتر از sequential
    - ← باید با ابزارهای تحلیل توان بررسی کرد

انتخاب نوع کدهای حالت

- کدام بهتر است؟
  - one-hot در برابر gray از نظر توان مصرفی:
    - one-hot دو تغییر بیت در هر گونه تغییر حالت
    - اما نیازی به ترتیب مشخص تغییر حالت‌ها ندارد

انتخاب نوع کدهای حالت

- کدام بهتر است؟
  - مقایسه از نظر خطاهای ناشی از تشعشعات:
    - Soft error: باعث تغییر مقدار FFها می‌شود
  - one-hot:
    - تعداد FF بیشتر ← احتمال خطای بیشتر
    - اما قابل تشخیص:
      - 0010000000 به 0010001000
      - در کد VHDL: با when others