

ماشین حالت متناهی ایمن

Safe FSM

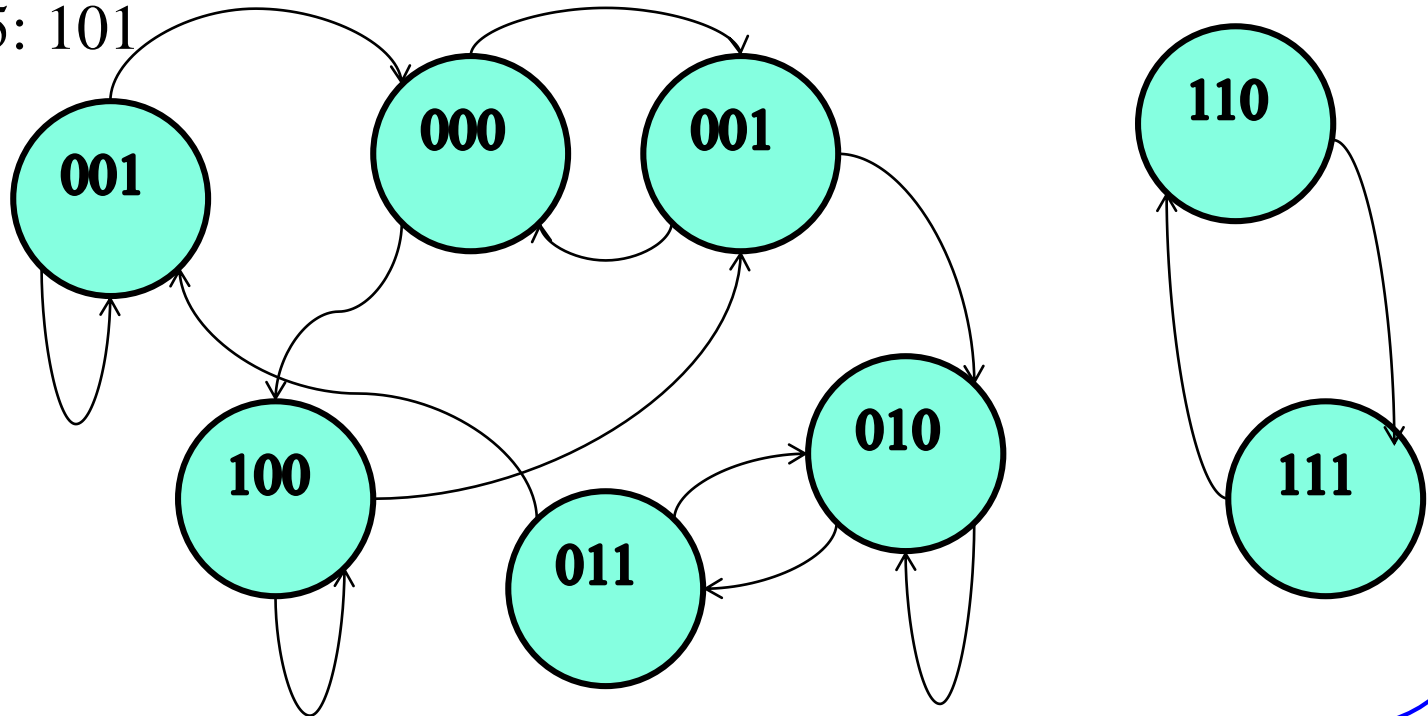
FSM ایمن

• اغتشاش در سیستم:

☐ ← رفتن به حالات اضافی

☐ بد اقبالی: در STATE5، بیت وسطی ۱ شود

STATE0: 000
STATE1: 001
STATE2: 010
STATE3: 011
STATE4: 100
STATE5: 101



FSM ایمن

• کد حالت ترتیبی:

□ برای N_s حالت:

$$N_e = N_s - 2^{\lceil \log_2 N_s \rceil} \text{ - تعداد حالات اضافی:}$$

حالات اضافی

• **when** : **راہ اول**
:others

□ **برخی ابزارها**

others را بی مورد

می بینند

← حذف می کنند!

```
architecture ARCH of ENTY is
  type STATE_TYPE is (START, S1, S2, S3, S4, S5);
  signal STATE: STATE_TYPE;
--
  process (CLK, RESET)
  begin
    if rising_edge (CLK) then
      if RESET = '1' then
        STATE <= START;
      else
        case STATE is
          when START =>
            if (INPUTS = ...) then
              STATE <= ...
            else
              STATE <= ...;
            end if;

            :
          when S5 =>
            if (INPUTS = ...)
              STATE <= ...;
            else
              STATE <= ...;
            end if;
          when others =>
            STATE <= START;
        end case;
      end if;
    end if;
  end process;
--
end architecture;
```

حالات اضافی

• راه دوم: حالات ساختگی:

□ دیگر others

بی مورد نیست.

□ تعداد حالات

ساختگی می تواند

زیاد باشد.

- برای ۱۸ حالت اصلی؟

- تغییر به one-hot?

```
architecture ARCH of ENTY is
    type STATE_TYPE is (START, S1, S2, S3, S4, S5,
        DUMMY1, DUMMY2);
    signal STATE: STATE_TYPE;
--
    process(CLK, RESET)
    begin
        if rising_edge(CLK) then
            if RESET = '1' then
                STATE <= START;
            else
                case STATE is
                    when START =>
                        if (INPUTS = ...) then
                            STATE <= ...
                        else
                            STATE <= ...;
                        end if;
                    :
                    when S5 =>
                        if (INPUTS = ...)
                            STATE <= ...;
                        else
                            STATE <= ...;
                        end if;
                    when others =>
                        STATE <= START;
                end case;
            end if;
        end if;
    end process;
--
end architecture;
```

حالات اضافی

• راه سوم: انتساب دستی:

□ دیگر others

بی مورد نیست.

□ کار طراح بیشتر

□ نیاز به تغییر در کد و کامپایل مجدد

```
architecture ARCH of ENTY is
  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";

  --
  process (CLK, RESET)
  begin
    if rising_edge(CLK) then
      if RESET = '1' then
        STATE <= START;
      else
        case STATE is
          :
          when S5 =>
            if (INPUTS = ...)
              STATE <= ...;
            else
              STATE <= ...;
            end if;
          when others =>
            STATE <= START;
          end if;
        end if;
      end process;
    --
  end architecture;
```

حالات اضافی

□ گاهی ابزار می‌فهمد حالات **others** حالت اضافی است ← در بهینه‌سازی، حذف می‌کند.

□ به ابزار بفهمانید که FSM ایمن باشد و حالت **recovery** را تعیین کنید.

```
attribute safe_implementation: string;
```

```
attribute safe_implementation of STATE is "yes";
```

```
type STATE_TYPE is (START, S1, S2, S3, S4, S5, RECOVERY);
```

```
;
```

```
attribute safe_recovery_state: string;
```

```
attribute safe_recovery_state of STATE: signal is "RECOVERY";
```

```
process (...)
```

```
begin
```

```
    case STATE is
```

```
        when START =>
```

```
            :
```

```
        when RECOVERY =>
```

```
            STATE <= START;
```

```
            :
```

```
    end case;
```

```
end process;
```

اشتراک منابع در FSM

- **اشتراک منابع (resource sharing):**

□ برای صرفه جویی در سخت افزار


```

process (STATE, A, B)
begin
case STATE is
  when ST0 => ...
  when ST1 =>
    if (A >= 0) then
      NEXT_STATE <= ST2;
      Z <= A + B;
    else
      NEXT_STATE <= ST1;
      Z <= A - B;
    end if;
  when ST2 =>
    NEXT_STATE <= ST3;
    if (B > A) then
      Z <= A + B;
    else
      Z <= A - B;
    end if;
  when ST3 =>
    ...

```

```

process (STATE, A, B)
  variable TEMP1: unsigned (7 DOWNTO 0);
  variable TEMP2: unsigned (7 DOWNTO 0);
begin
  TEMP1 := A + B;
  TEMP2 := A - B;
  case STATE is
    when ST0 => ...
    when ST1 =>
      if (A >= 0) then
        NEXT_STATE <= ST2;
        Z <= TEMP1;
      else
        NEXT_STATE <= ST1;
        Z <= TEMP2;
      end if;
    when ST2 =>
      NEXT_STATE <= ST3;
      if (B > A) then
        Z <= TEMP1;
      else
        Z <= TEMP2;
      end if;
    when ST3 =>

```

...

گزارش ابزار سنتز

• گزارش از سنتز FSM

```
=====
* HDL Synthesis *
```

```
=====
Synthesizing Unit <fsm_1>.
```

```
Found 1-bit register for signal <outp>.
```

```
Found 2-bit register for signal <state>.
```

```
Found finite state machine <FSM_0> for signal <state>.
```

```
-----
| States | 4 |
| Transitions | 5 |
| Inputs | 1 |
| Outputs | 2 |
| Clock | clk (rising_edge) |
| Reset | reset (positive) |
| Reset type | asynchronous |
| Reset State | s1 |
| Power Up State | s1 |
| Encoding | gray |
| Implementation | LUT |
-----
```

گزارش ابزار سنتز

• گزارش از سنتز FSM

...

```
=====
* Low Level Synthesis *
=====
```

Optimizing FSM <state> on signal <state[1:2]> with gray encoding.

```
-----
State | Encoding
```

```
-----
s1  | 00
```

```
s2  | 11
```

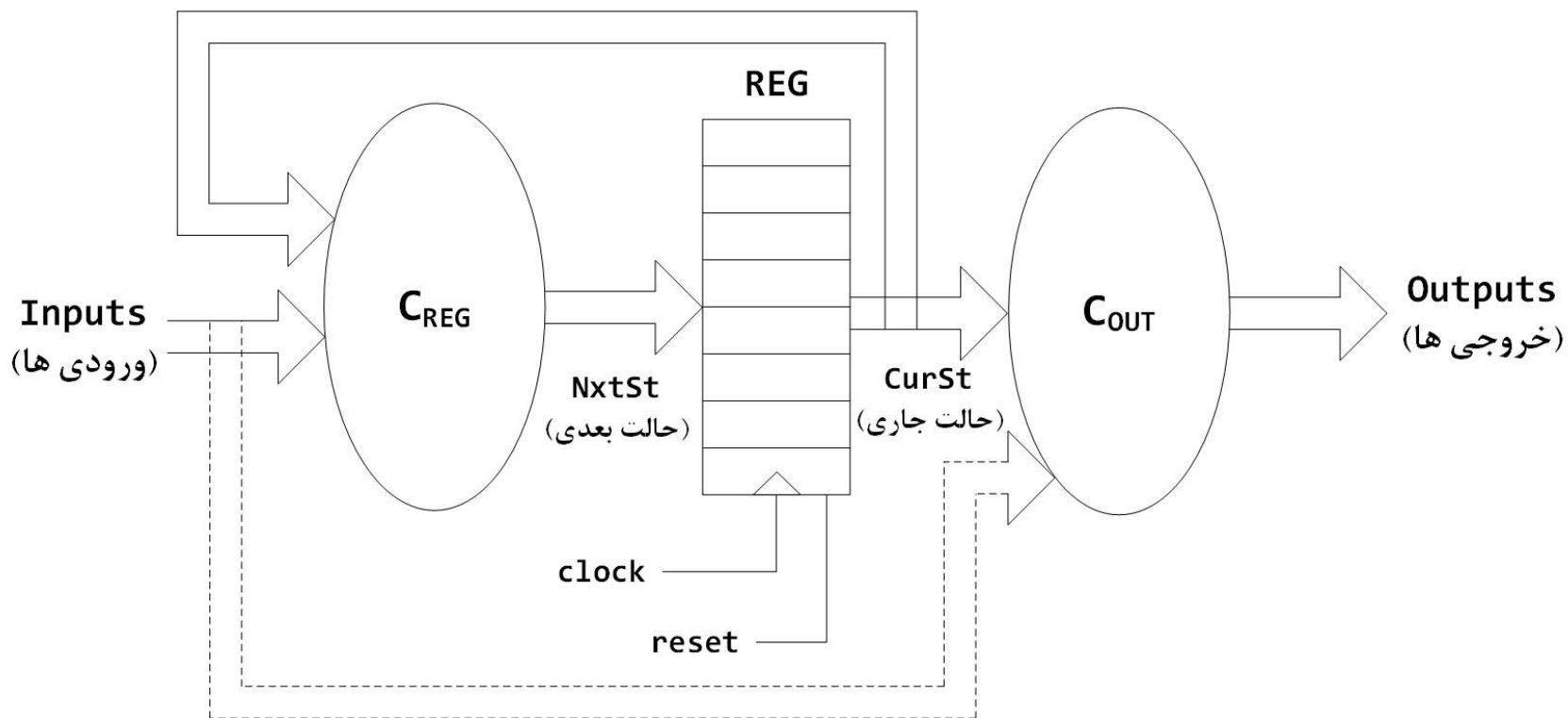
```
s3  | 01
```

```
s4  | 10
-----
```

تولید خروجی‌ها در FSM

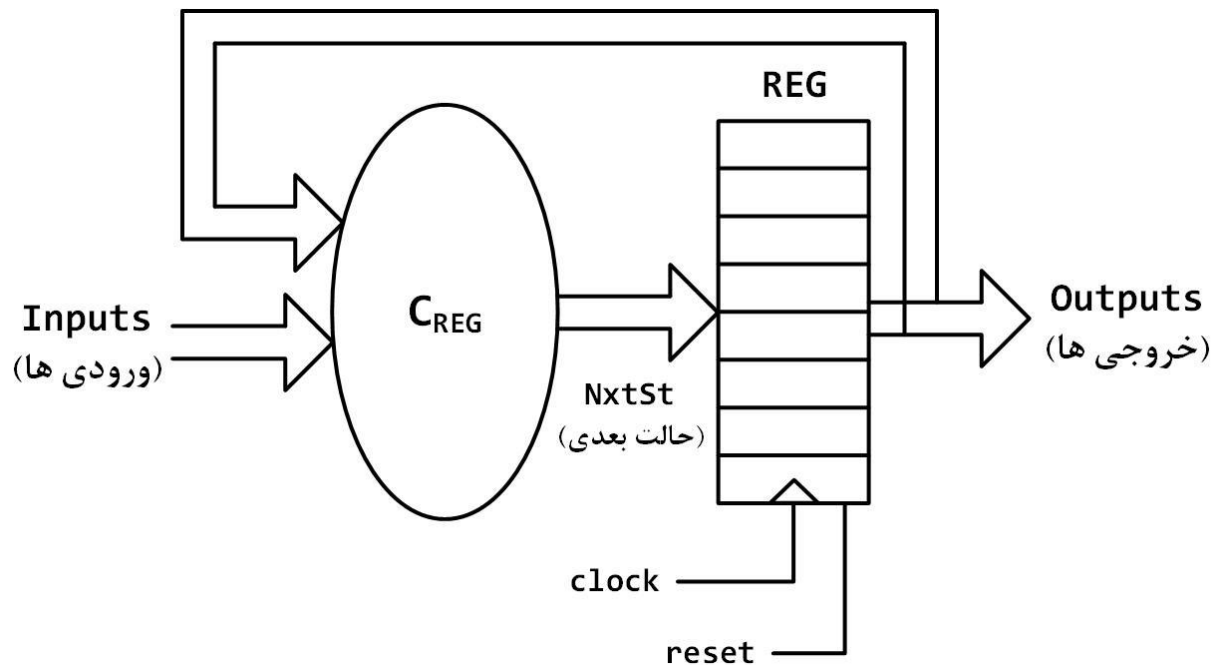
تولید خروجی‌ها

• مدل کلی FSM



تولید خروجی‌ها

• مدل محدود (Medvedev)



$$\text{outputs} = f_o(\text{cur_state}) = \text{cur_state}$$

تولید خروجی‌ها

• مدل مدودف (Medvedev):

□ مزیت:

- آماده شدن سریع خروجی‌ها

- تأخیر = ؟

□ برای ماشین مور، باید کدگذاری حالات به درستی انجام شود.

- ممکن است تعداد FF بیشتر بخواهد

تولید خروجی‌ها

• مدل مدودف (Medvedev): مثال



حالت جاری	کد حالت جاری (ترتیبی) $S_2 \dots S_0$	خروجی‌ها $Z_3 Z_2 Z_1$
ST0	000	000
ST1	001	100
ST2	010	110
ST3	011	000
ST4	100	100
ST5	101	000
ST6	110	111
ST7	111	000

حالت جاری	کد مدودف $S_2 S_1 S_0$	خروجی‌ها $Z_3 Z_2 Z_1$
ST0	000	000
ST1	100	100
ST2	110	110
ST3	000	000
ST4	100	100
ST5	000	000
ST6	111	111
ST7	000	000

تولید خروجی‌ها

• مدل مدودف (Medvedev): مثال

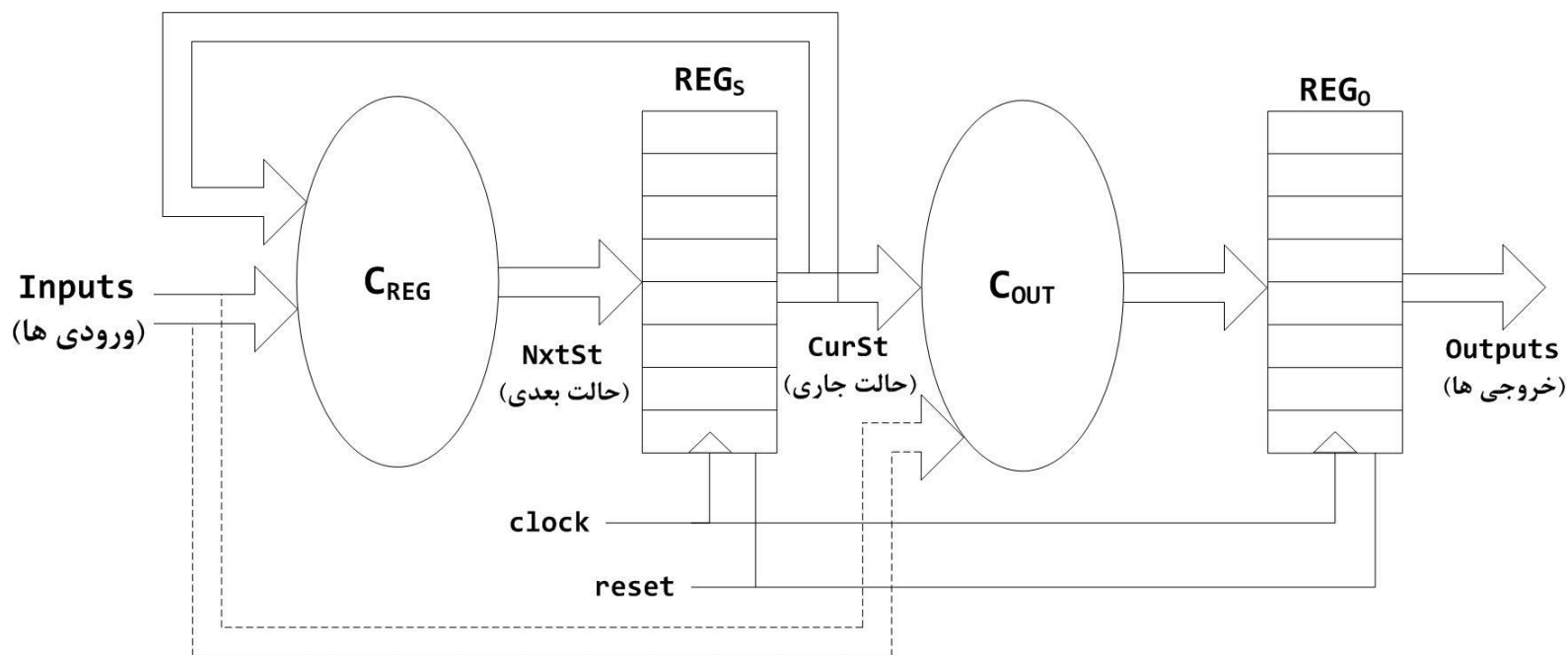


حالت جاری	کد مدودف S2S1S0	خروجی‌ها ·Z3Z2Z1
ST0	000	000
ST1	100	100
ST2	110	110
ST3	000	000
ST4	100	100
ST5	000	000
ST6	111	111
ST7	000	000

حالت جاری	کد مدودف S4S3S2S1S0	خروجی‌ها Z3Z2Z1
ST0	00000	000
ST1	100x0	100
ST2	110xx	110
ST3	00001	000
ST4	100 x1	100
ST5	00010	000
ST6	111xx	111
ST7	00011	000

تولید خروجی‌ها

- تولید خروجی‌ها به صورت ثبت شده (مدل اول)



تولید خروجی‌ها

• تولید خروجی‌ها به صورت ثبت شده:

□ مزیت:

- تغییر خروجی‌ها همگام با کلاک

□ اشکال:

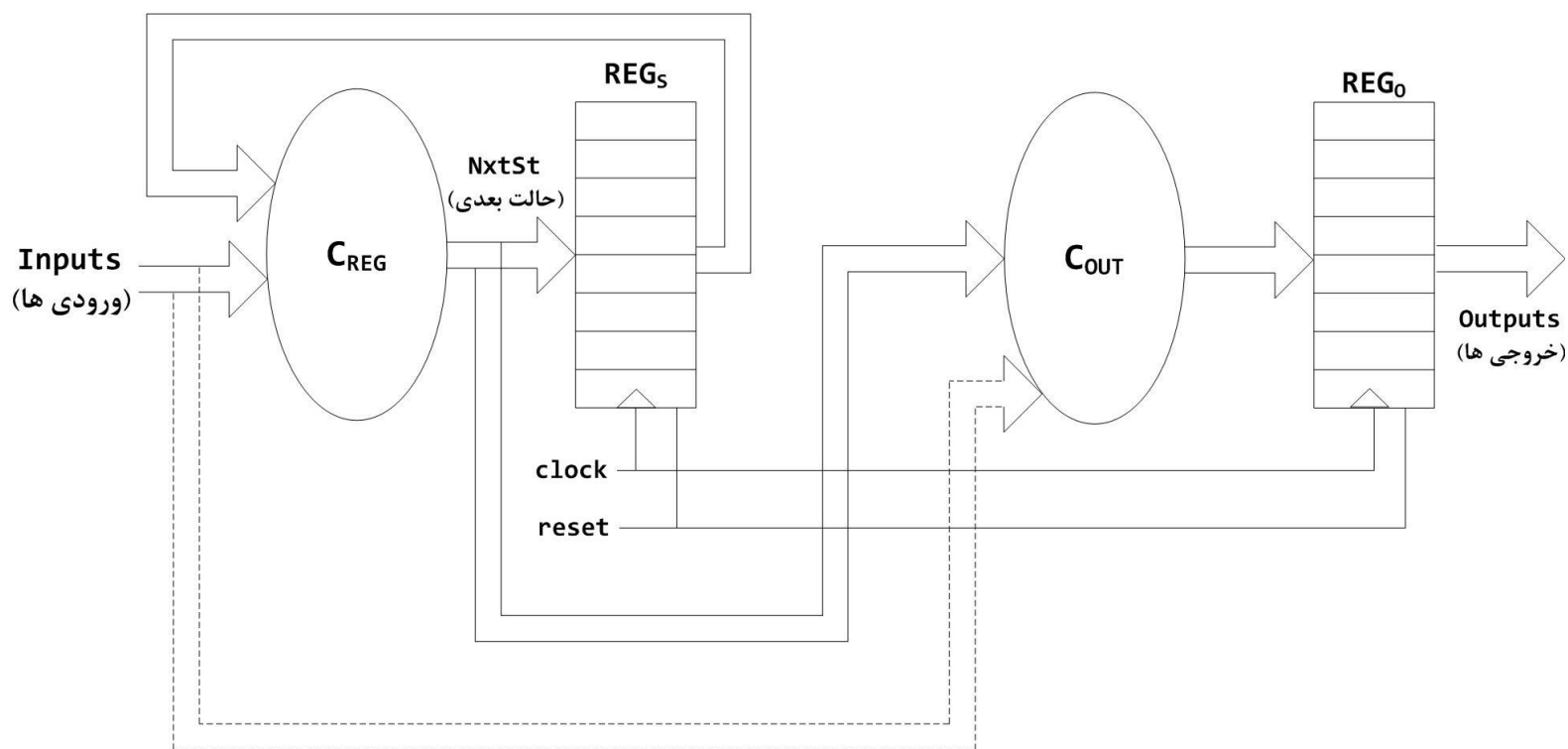
- عقب افتادن خروجی به اندازه یک سیکل ساعت

- در بعضی از کاربردها عیبی ندارد

□ راه حل؟

تولید خروجی‌ها

- تولید خروجی‌ها به صورت ثبت شده (مدل دوم)



تولید خروجی‌ها

- تولید خروجی‌ها به صورت ثبت شده (مدل دوم):

□ اشکال:

- تأخیر بیشتر مدار ترکیبی بین FFها
- مخصوصاً اگر باعث شود تعداد logic blockها بیشتر شود
- ممکن است مشکلی نباشد

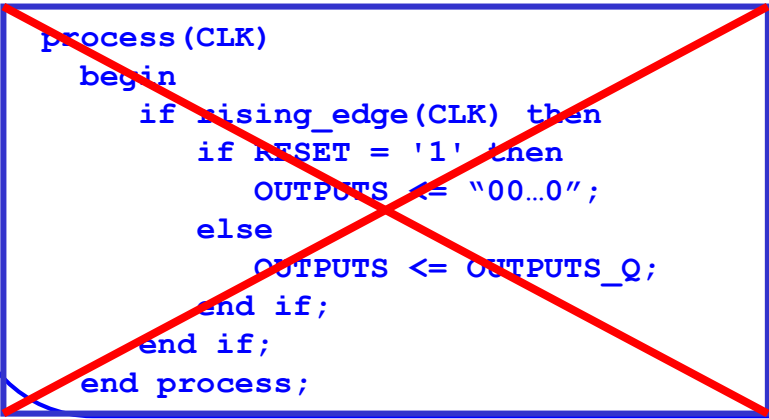
تولید خروجی‌ها

```
process (CLK)
begin
    if rising_edge(CLK) then
        if RESET = '1' then
            CUR_STATE <= RST_ST;
            OUTPUTS <= "00...0";
        else
            CUR_STATE <= NEXT_STATE;
            OUTPUTS <= OUTPUTS_Q;
        end if;
    end if;
end process;
```

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

• توصیف مدل اول:

```
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;
```



```
process (CLK)
begin
    if rising_edge(CLK) then
        if RESET = '1' then
            OUTPUTS <= "00...0";
        else
            OUTPUTS <= OUTPUTS_Q;
        end if;
    end if;
end process;
```

تولید خروجی‌ها

```
process (CLK)
begin
    if rising_edge(CLK) then
        if RESET = '1' then
            CUR_STATE <= RST_ST;
            OUTPUTS <= "00...0";
        else
            CUR_STATE <= NEXT_STATE;
            OUTPUTS <= OUTPUTS_Q;
        end if;
    end if;
end process;
```

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

• توصیف مدل دوم:

```
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;
```

بازنشانی و پیش‌نشانی

• Reset و Preset

□ شروع سیستم از یک حالت اولیه مشخص

- مثلاً همه FF ها صفر

□ نیاز به همزمانی دریافت توسط همه FF ها

- از خط reset سراسری در PLD ها با skew کم استفاده کنید

بازنشانی و پیش‌نشانی

• Reset همگام یا ناهمگام؟

□ اگر نمی‌دانید کدام برای سیستم شما بهتر است، با همگام شروع کنید (اگر لازم شد، ناهمگام)

□ مزایای همگام:

- همه سیستم با کلاک همگام است ← درک و اشکال‌زدایی آسان‌تر
- با glitch روی خط reset کل سیستم بازنشانی نمی‌شود.
- شبیه‌سازهای cycle-based آنها را می‌فهمند.

بازنشانی و پیش‌نشانی

• Reset همگام یا ناهمگام؟

□ مزایای ناهمگام:

- عدم نیاز به انتظار برای لبه کلاک
- گاهی کلاک بخش‌هایی را غیرفعال می‌کنند (صرفه‌جویی در توان)
- ← کنترل روی زمان reset کردن از بین نمی‌رود.
- برای reset همگام باید پهنای پالس به اندازه کافی بزرگ شود

بازنشانی و پیش‌نشانی

• synchronizer

- ❑ غیرفعال کردن Reset ناهمگام باید همگام شود
 - ❑ احتمال تخلف محدودیت‌های **set-up** و **hold** در زمان غیرفعال کردن **reset** ناهمگام
- ← - حالت meta-stable

