درس: آزمایشگاه معماری کامپیوتر(جلسه هفتم)

- مروری بر VHDL(ادامه)
- آزمایش ششم

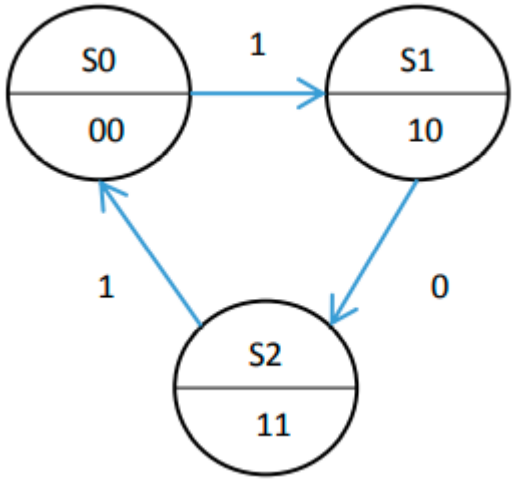
نیمسال اول ۱۴۰۱

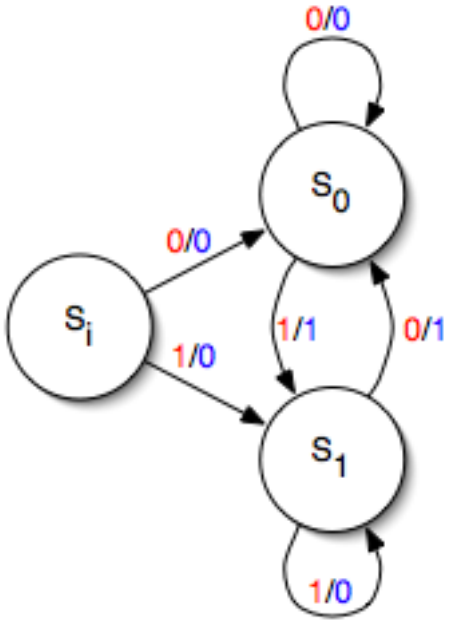
طراحی ماشین حالت (State Machine)

- دو نوع ماشین حالت Mealy و Moore وجود دارند که اختلاف آن‌ها در تاثیر یا عدم تاثیر مستقیم ورودی بر نتیجه خروجی است، اما در پیاده‌سازی HDL تفاوت چندانی ندارند. طراحی ماشین Moore معمولا آسان‌تر است و همچنین بر خلاف Mealy خروجی‌هایش سنکرون با کل مدار (در یک سیکل کامل) است اما در عوض تعداد حالت‌های بیشتری نسبت به Mealy دارد.
- در طراحی در سطح RTL (Register Transfer Level) معمولا طراحی را به دو بخش Controller و Datapath تقسیم می‌کنند. نحوه و نکات مربوط به طراحی RTL خارج از بحث ما است و نیاز به مبحثی جداگانه دارد اما در این بخش به صورت خلاصه، مثال‌هایی از آن نشان می‌دهیم. معادل Controller در این سبک طراحی، همان ماشین حالت است که پیاده‌سازی آن در زبان‌های HDL دشوار نیست به شرطی که اصول طراحی در آن رعایت شود.

- ماشین مور(moore machine) یک نوع از ماشین‌های حالات متناهیست که خروجی آن فقط توسط حالت کنونی آن به وجود می‌آید.

- ماشین میلی (mealy machine) یک نوع از ماشین‌های حالات متناهیست که خروجی آن به حالت کنونی و مقدار ورودی کنونی وابسته‌است. (این ماشین نقطه ی مقابل ماشین مور است که خروجی آن فقط به حالت کنونی آن وابسته می‌باشد)





طراحی ماشین حالت (State Machine) – ادامه

- به منظور طراحی یک ماشین حالت، ابتدا مسئله را به حالت‌های مختلف بخش‌بندی می‌کنیم. بهتر است ابتدا دیاگرام ماشین حالت را رسم کنیم. در ادامه سیگنال‌های کنترلی (معمولا مشترک بین DataPath و Controller) را به طور دقیق مشخص می‌کنیم.
- در نهایت، حالت‌های مختلف را به همراه تغییرات، عینا به صورت Behavioral و معمولا با استفاده از دستور **case-when** پیاده سازی می‌کنیم.

طراحی ماشین حالت (State Machine) - ادامه

- طراحی مدارهای دیجیتالی که دارای بخش‌های Combinational و Sequential هستند معمولا به دو صورت ترکیبی یا مجزا انجام می‌شود. یکی از مدل‌های مناسب برای پیاده‌سازی، مدل Huffman است. در این مدل مدار به دو بخش Sequential و Combinational تقسیم می‌شود. پیشنهاد می‌شود که در طراحی‌ها از همین مدل استفاده شود.
- در ادامه مقایسه‌ای بین انواع مختلف طراحی انجام می‌شود.

Processها ماهیت ترتیبی دارند و باید در بدنه‌ی Architecture نوشته شوند. این ساختارها به صورت ترتیبی اجرا می‌شوند لذا درون آن‌ها می‌توان از دستورالعمل‌های If ... then ... else استفاده کرد. هر Process برای اجرا به Sensitivity list نیاز دارد. منظور از Sensitivity list سیگنال‌هایی هستند که وقتی Event روی آن‌ها رخ می‌دهد Process را تحریک می‌کند.

برای طراحی ماشین‌های حالت از Processها استفاده می‌شود. همان‌طور که گفته شد اگر مقدار سیگنالی که در Sensitivity list وجود دارد تغییر کند Process اجرا می‌شود. اگر Process مدل کننده‌ی یک بلوک ترکیبی است، تمام ورودی‌های آن باید در Sensitivity list لحاظ شود. ماشین‌های حالت در لبه‌های کلاک تغییر حالت می‌دهند، لذا کلاک باید در ورودی Sensitivity list وارد شود. علاوه بر این حالت درونی سیستم هم باید در یک متغیر داخلی نگه‌داری شود. برای نگه‌داری حالت‌ها در خود VHDL می‌توان Type جدید تعریف کرد که در این‌جا مثلا دو مقدار A و B نشان‌دهنده دو حالت از سیستم هستند:

Type *state* is (*A,B*);

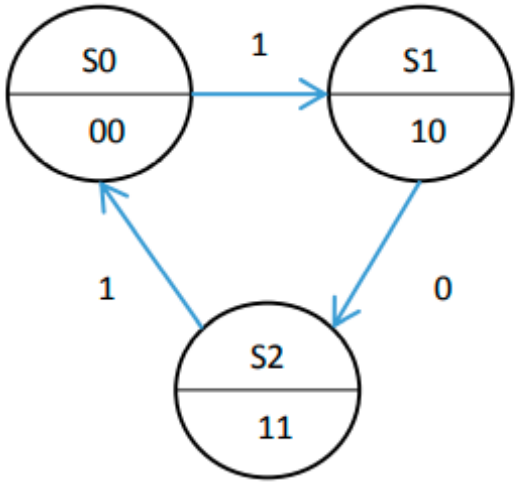
حال برای استفاده از Type تعریف شده می‌توان از آن Type یک سیگنال یا متغیر تعریف کرد. سپس از دستورالعمل Case روی سیگنال یا متغیر تعریف شده برای انشعاب به حالت‌های سیستم استفاده می‌شود. واضح است که این دستورالعمل Case بسته به اینکه ماشین حالت به صورت Mealy یا Moore باید طراحی شود، در داخل Process تعریف می‌شود. به عنوان مثال:

```
Signal st : state;
Process (...)
Case st is
When A => ....
When B => ....
```

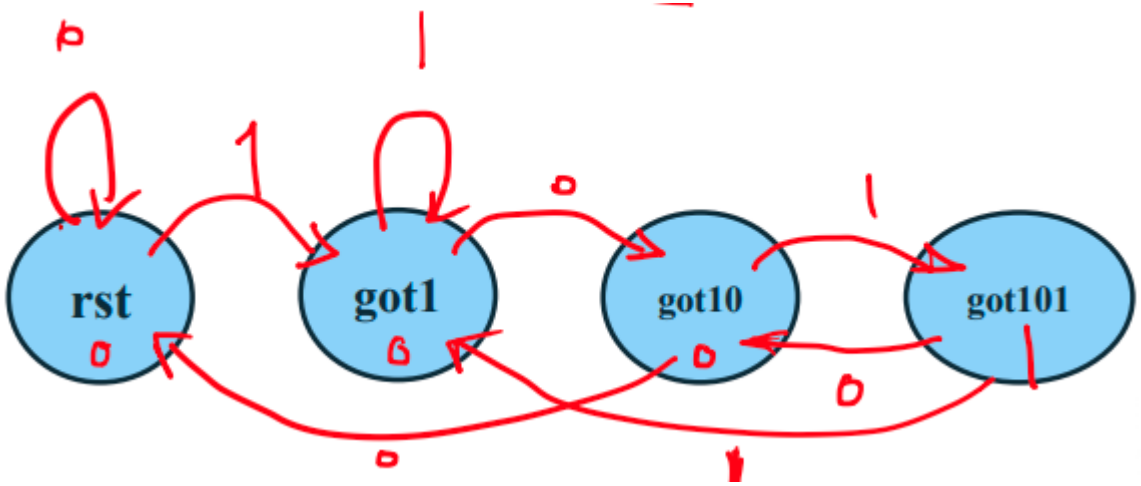
برای طراحی مدارات باحافظه مدلی به نام مدل هافمن وجود دارد که قسمت ترکیبی مدار را از قسمتی ترتیبی آن جدا می‌کند. قسمت ترتیبی آن معمولاً با Processی که نسبت به سیگنال کلاک و reset حساس است نوشته می‌شود و قسمت ترکیبی آن در Processی که نسبت به حالت‌های مدار حساس است نوشته می‌شود.

یک نمونه از کد ماشین مور

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity moore_machine is
        port(
                input : in std_logic ;
                output : out std_logic_vector(1 downto 0);
                clk : in std_logic
        );
end moore_machine;
architecture Behavioral of moore_machine is
        type state_t is (s0 , s1 , s2);
        signal state : state_t := s0;
        signal next_state : state_t := s0;
begin
        CMB : process(state , input)
        begin
                case state is
                        when s0=>
                                if(input = '1') then
                                        next_state <= s1;
                                else
                                        next_state <= state
                                end if;
                        when s1=>
                                ....
                        when s2=>
                                ....
                        when others=>
                                next_state <= s0;
                end case;
        end process;
        REG : process(clk)
                begin
                        if(clk'event and clk = '1') then
                                state <= next_state;
                        end if;
        end process;
        output <= "00" when state = s0   else
                  "10" when state = s1   else
                  "11" when state = s2;
end Behavioral;
```

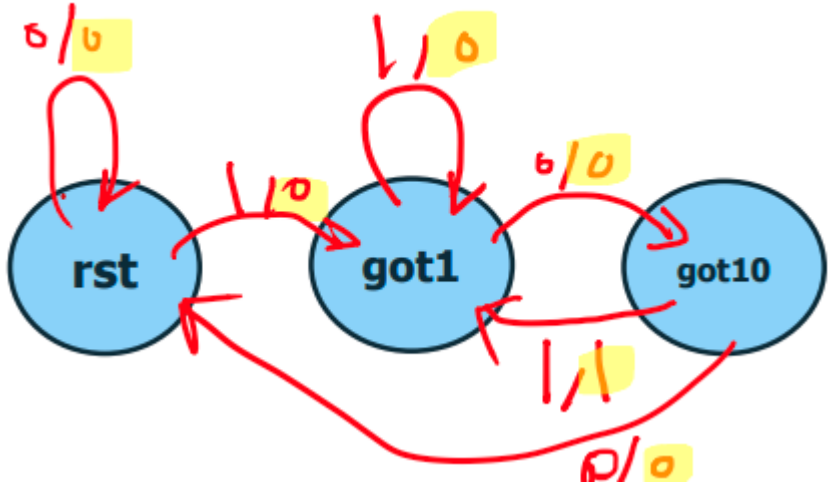


نمونه‌ای از ماشین حالت Moore



Moore Sequence Detector (101)

نمونه های کدهای تشخیص دنباله ۱۰۱ به صورت میلی و مور و هافمن به همراه بنچمارک ها در اختیار دانشجویان قرار داده می شود.



Mealy Sequence Detector (101)

```
library ieee;
use ieee.std_logic_1164.all;
entity moore_detector is
port (input, reset, clk : in std_logic;
                output : out std_logic);
end entity ;
architecture Behavioral of moore_detector is
        --type state is (rst, got1, got10, got101);
        --signal present_state : state := rst;
        constant rst : std_logic_vector(1 downto 0) := "00";
        constant got1  : std_logic_vector(1 downto 0) := "01";
        constant got10 : std_logic_vector(1 downto 0) := "10";
        constant got101: std_logic_vector(1 downto 0) := "11";
        signal present_state : std_logic_vector(1 downto 0) := "00"; --rst="00" as default
begin
process (clk) --reset should be synchronous with clk
begin
        if (clk = '1' and clk'event) then
                if reset = '1' then
                        present_state <= rst;
                else
                        case present_state is
                                when rst =>
                                        if input = '1' then
                                                present_state <= got1;
                                        else
                                                present_state <= rst;
```

```
                when got1 =>
                        if input = '0' then
                                present_state <= got10;
                        else
                                present_state <= got1;
                        end if;
                when got10 =>
                        if input = '1' then
                                present_state <= got101;
                        else
                                present_state <= rst;
                        end if;
                when got101 =>
                        if input = '1' then
                                present_state <= got1;
                        else
                                present_state <= got10;
                        end if;
                when others =>
                        present_state <= rst;
        end case;
    end if;
end if;
end process;

        output <= '1' when present_state = got101 else '0';

end Behavioral;
```

به صورت concurrentو خارج از بدنه پروسس
البته لابلای دستورات هم میشد اما اینجا ساده تر است.

تمرین کلاسی: طراحی دیاگرام حالت یک detector Sequence برای رشته ی "۱۱۰۱ "

گزارش کار

- طراحی یک مدار آشکار ساز با استفاده از ماشین های حالت میلی و مور برای دنباله ۰۱۱۰ به زبان Vhdl
- بررسی صحت عملکرد مدار