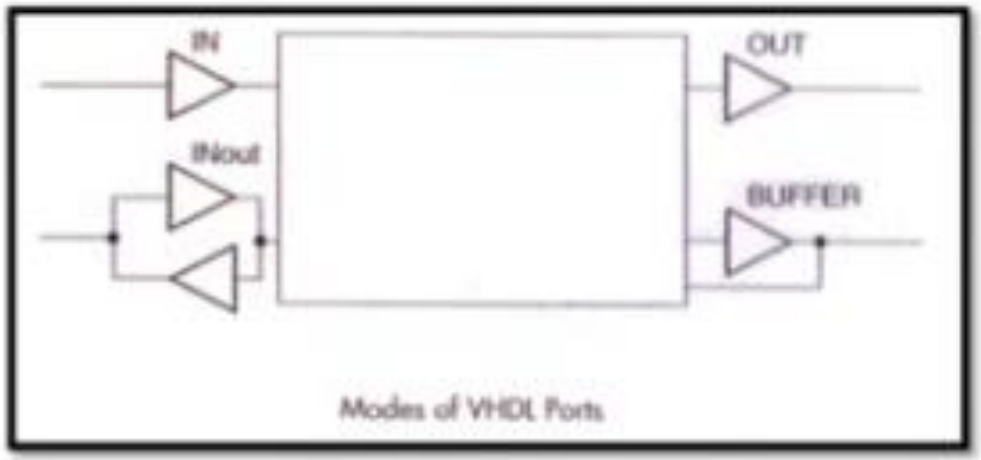
درس: آزمایشگاه معماری کامپیوتر(جلسه چهارم)

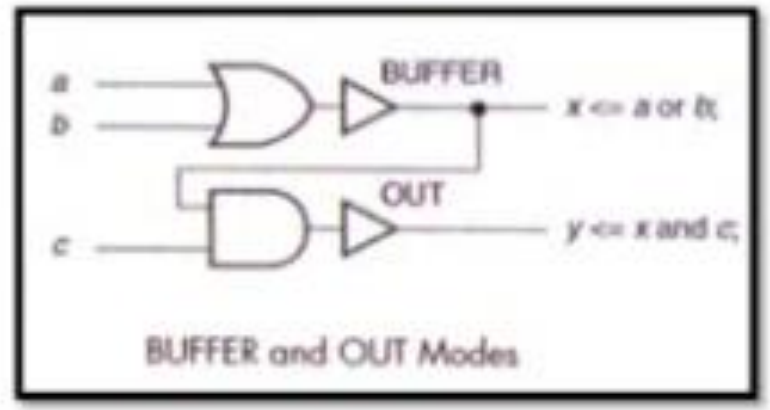
- مروری بر VHDL(ادامه)
- آزمایش سوم

نیمسال دوم ۱۴۰۰

Ports in VHDL

- انواع Port:





- تعریف port از نظر type و دیگر جزییات مشابه Signal است. تنها کافی است Mode به آن اضافه شود.
- با تعریف Port دیگر نمی‌توان با همان نام Signal در داخل برنامه تعریف نمود.

Port – مثال

```
entity test is
 port
      (
      Data : inout Std_Logic_Vector(7 downto 0);
      Addr : in Std_Logic_Vector(9 downto 0);
      NotCS : in Std_Logic;
      RdNotWr : in Bit
      );
end test;
```

Generic

- می توان مقادیر ثابتی را در ابتدای برنامه (درون Entity) تعریف نمود و از راه های مختلف مقدار آن را تعیین نمود.
- کاربرد آن به طور معمول تعیین تعداد بیت های Signal، Constant، Variable و غیره است.

```
entity my_design is
  generic (BusWidth : Integer := 16);
  port(DataBus : inout Std_Logic_Vector(BusWidth-1 downto 0));
  ...
end my_design
```

Generic

- تعریف generic مشابه تعریف Signal است.
- حتما باید مقداردهی اولیه شده باشد.

```
ENTITY My_Design IS
   GENERIC (
              data_width : integer range 1 to 8 := 4;
              ...
              delay : time := 10 ns
            );
   PORT ( ... );
END ENTITY My_Design; -- END ENTITY -- END My_Design
```

Testbench

- به منظور ارزیابی اولیه و درستی سنجی سیستم طراحی شده از Testbench استفاده می شود.
- مشابه آن است که واحد طراحی شده را در نظر گرفته و ورودی‌های مختلف را به آن اعمال کنیم و در نهایت خروجی را بررسی کنیم.
- در عمل چیزی بجز همان کد VHDL نیست که به منظور بررسی صحت عملکرد مدار دیجیتال طراحی شده می باشد.

- بهتر است که کد اصلی را داخل implementation و test benchها را دربخش simulation تعریف کنیم .
- ما فایل test bench را هیچ وقت سنتز نمی کنیم.
- در داخل test bench اسم سیگنال ها را مشابه و هم نام پورت های مداری که تست می کنید قرار دهید.(این موضوع با مطلبی که قبلا گفتیم در تناقض نمی باشد.)
- خود برنامه نام مدار مورد تست و پورت های آن را تحت عنوان component تعریف می کند و نیز خودش مقادیر اولیه ای برای ورودی ها در نظر می گیرد.
- بخش اصلی همان port map است که قبلا با آن آشنا شده اید و باز هم نرم افزار انجام می دهد.
- در مدارهای ترکیبی آن بخشی را که حاوی کلاک است حذف می کنیم.
- در tets bench می توانید به سادگی مقدار بدهید یا در یک process مقادیری را مثلا با دستورات سطح بالا شبیه while مقداردهی کنید.
- در اینجا از مقادیر تاخیر که قبلا اشاره نموده بودیم استفاده می کنیم تا بتوانیم نتایج را مشاهده کنیم در غیر این صورت تمام مقداردهی ها یکدفعه صورت می گیرند و نمی توانیم بررسی کنیم.
- در واقع کار اصلی ما مقداردهی سیگنال هاست.

در port mapها به جای استفاده از فلش ها و مقداردهی از طرق name میتوانیم به ترتیب نام سیگنال ها را فقط در port map بنویسیم که براساس ترتیب و مکان قرارگیری نگاشت شوند.

```
ARCHITECTURE behavior OF tb_Flip_Flop IS
 COMPONENT Flip_Flop    PORT(
                          d : IN  std_logic;
                         reset : IN  std_logic;
                         clk : IN  std_logic;
                         q : OUT  std_logic       );
  END COMPONENT;
            signal d : std_logic := '0';
            signal reset : std_logic := '0';
            signal clk : std_logic := '0';
             signal q : std_logic;   -- Clock period definitions
constant clk_period : time := 10 ns;
BEGIN
  uut: Flip_Flop PORT MAP (         d ,  reset,  clk, q       );
…
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
ENTITY tb_fulladder IS
END tb_fulladder;
ARCHITECTURE behavior OF tb_fulladder IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT full_adder
    PORT(
         a : IN  std_logic;
         b : IN  std_logic;
         cin : IN  std_logic;
         sum : OUT  std_logic;
         cout : OUT  std_logic
        );
    END COMPONENT;
               --Inputs
   signal a : std_logic := '0';
   signal b : std_logic := '0';
   signal cin : std_logic := '0';
               --Outputs
   signal sum : std_logic;
   signal cout : std_logic;
```

```
BEGIN
                -- Instantiate the Unit Under Test
(UUT)
   uut: full_adder PORT MAP (
          a => a,
          b => b,
          cin => cin,
          sum => sum,
          cout => cout
        );
   -- Stimulus process
   stim_proc: process
                variable count : std_logic_vector(2
downto 0) := "000";
   begin
                                while (count < "111")
loop
                                wait for 1 ns;
                                count := count + 1;
                                a <= count(0);
                                b <= count(1);
                                cin <= count(2);
                                end loop;
                                wait;
   end process;
END;
```

کلیه مدارهای دیجیتالی که در آزمایش های قبلی مورد بررسی قرار گرفته بودند از نوع مدارهای ترکیبی بودند.

در این مدارها خروجی ها همه به ورودی های دیجیتال وابسته اند. گرچه به نظر می رسد که هر سیستم دیجیتال دارای مدارهای ترکیبی است، بسیاری از سیستم هایی که در عمل با آن مواجه هستیم حاوی عناصر حافظه هم می باشند و بنابراین لازم است تا این سیستم ها بر حسب منطق ترتیبی مورد بررسی قرار گیرند.

همچنین لازم است در مواردی در فرایند طراحی گیت عمدا تاخیراتی اعمال گردد.

نمونه هایی را با هم بررسی می کنیم.

پالس های کلاک در سرتاسر سیستم توزیع می گردند به نحوی که عناصر حافظه تنها هنگام رسیدن هر پالس تحت تاثیر ورودی خود قرار می گیرند.

عناصر ذخیره سازی در مدارهای ترتیبی کلاک دار را فلیپ فلاپ می گویند.

فلیپ فلاپ یک وسیله ذخیره سازی دودویی بوده و قادر است یک بیت از اطلاعات را در خود ذخیره نماید.

در این آزمایش مباحث اصلی مربوط به مدارهای ترتیبی بالاخص مدارهای ترتیبی همزمان مورد بحث و بررسی قرار خواهند گرفت.

حالت فلیپ فلاپ ها تنها هنگام تغییر وضعیت یک پالس ساعت عوض می شود که ما حساسیت فلیپ فلاپها به لبه کلاک (پالس ساعت) را با مفهوم processتوصیف می کنیم.

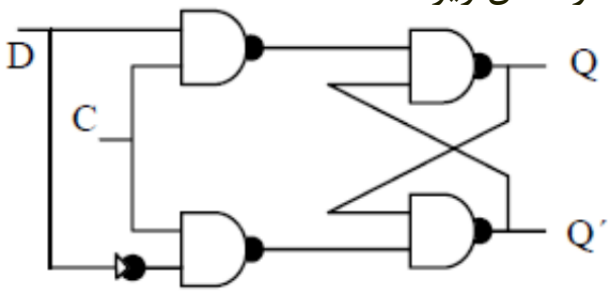
فلیپ فلاپها در ۳ گروه T-FF، JK-FFو D-FF قرار می گیرند.

ساختار داخلی یک لچ (حساس به سطح پالس نوع D)

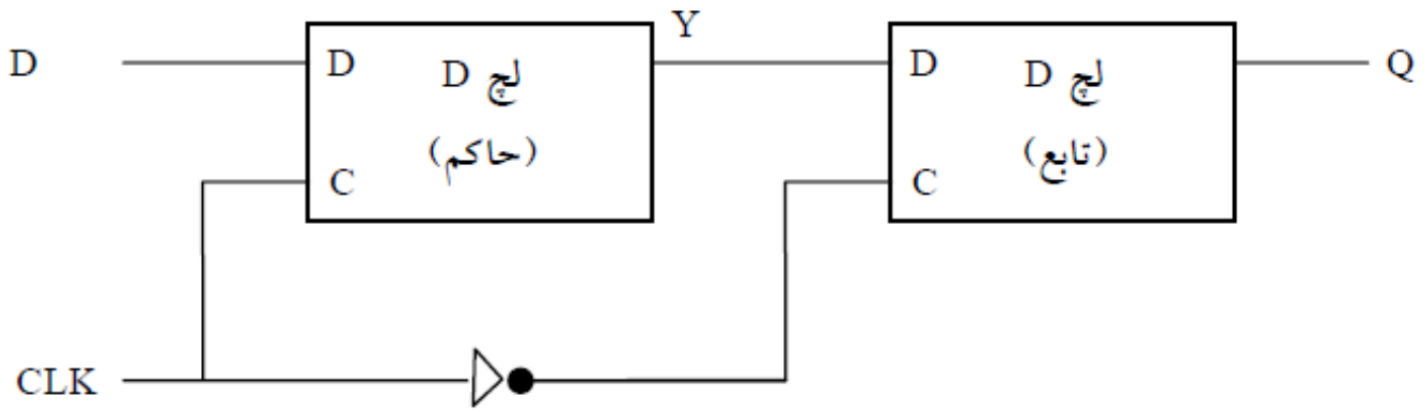
حساس به لبه پالس ساعت که با استفاده از ۲ تا لچ ساخته می شود) در شکل زیر

نشان داده شده است:

C	D	حالت بعدی Q
0	X	بلاتغییر
1	0	حالت بازنشانی Q=0
1	1	حالت نشاندن



لچ D



فلیپ‌فلاپ D

به غیر از فلیپ فلاپ نوع Dفلیپ فلاپ های دیگری نیز وجود دارند .اقتصادی ترین و بهترین فلیپ فلاپ قابل ساخت، نوع Dحساس به لبه می باشد که به تعداد کمتری گیت نیاز دارد .دیگر فلیپ فلاپ ها را می توان با فلیپ فلاپ D و مقداری مدار بیرونی به وجود آورد .دو فلیپ فلاپ رایج در طراحی سیستم های دیجیتال عبارتند از : فلیپ فلاپ JKو T. فلیپ فلاپ Tیک فلیپ فلاپ متمم ساز است که در طراحی شمارنده های دودویی بسیار مورد توجه است.

فلیپ فلاپ T

T	Q(t+1)	
0	Q(t)	بلا تغییر
1	Q´(t)	متمم

فلیپ فلاپ JK

J	K	Q(T+1)	
0	0	Q(t)	بلا تغییر
0	1	0	بازنشانی
1	0	1	نشاندن
1	1	Q´(t)	متمم

فلیپ فلاپ D

D	Q(t+1)	
0	0	بازنشانی
1	1	نشاندن

فرم کلی دستور case:

```
CASE expression IS

        WHEN constant_value =>
            statement;
            statement;
        WHEN constant_value =>
            statement;
            statement;
                .
                .
                .

        WHEN OTHERS =>
            statement;
            statement;

END CASE;
```

دستور case چون یک دستور sequential است یعنی عبارات داخلی آن به ترتیب و خط به خط اجرا می شوند لذا حتما می بایست داخل بدنه process نوشته شود.

Processها ماهیت ترتیبی دارند و باید در بدنه ی Architectureنوشته شوند.
این ساختارها به صورت ترتیبی اجرا می شوند لذا درون آنها می توان از دستورالعملهای If ... then ... else
استفاده کرد. هر Processبرای اجرا به Sensitivity listنیاز دارد. منظور از Sensitivity listسیگنال هایی هستند که وقتی Eventروی آنها رخ میدهد Processرا تحریک می کند
برای طراحی ماشین های حالت از Processها استفاده می شود. همان طور که گفته شد اگر مقدار سیگنالی که در Sensitivity listوجود دارد تغییر کند Processاجرا می شود. اگر Processمدل کننده ی یک بلوک ترکیبی است، تمام ورودی های آن باید در Sensitivity listلحاظ شود.
ماشین های حالت در لبه های کلاک تغییر حالت میدهند، لذا کلاک باید در ورودی Sensitivity listوارد شود. علاوه بر این حالت درونی سیستم هم باید در یک متغیر داخلی نگهداری شود. برای نگهداری حالتها در خود VHDLمی توان Typeجدید تعریف کرد که در این جا مثلا دو مقدار Aو Bنشان دهنده دو حالت از سیستم هستند:

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

حال برای استفاده از Typeتعریف شده می توان از آن Typeیک سیگنال یا متغیر تعریف کرد. سپس از دستورالعمل Caseروی سیگنال یا متغیر تعریف شده برای انشعاب به حالت های سیستم استفاده می شود. واضح است که این دستورالعمل Caseبسته به اینکه ماشین حالت به صورت Mealyیا Mooreباید طراحی شود، در داخل Processتعریف می شود. به عنوان مثال:

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

برای طراحی مدارات با حافظه مدلی به نام مدل هافمن وجود دارد که قسمت ترکیبی مدار را از قسمتی ترتیبی آن جدا می کند. قسمت ترتیبی آن معمولا با Processی که نسبت به سیگنال کلاک و resetحساس است نوشته می شود و قسمت ترکیبی آن در Processی که نسبت به حالت های مدار حساس است نوشته می شود.

کد vhdl فلیپ فلاپ D

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Flip_Flop is
    port(
        d,reset,clk : in std_logic;
        q : out std_logic
        );
end Flip_Flop;
architecture Behavioral of Flip_Flop is
begin
    q <= '0' when reset='1' else
    d  when clk'event and clk='1';
    --d  when clk'event and clk='1' else UNAFFECTED;
end Behavioral;
```

کد vhdl فلیپ فلاپ D

```
process (clk,reset)
begin
        if ( reset ='0' ) then
                Q <= '0';
        else if (clk'event and clk = '0')  then
                Q <= D;
                end if;
                end if;
end process ;
```

تست فلیپ فلاپ D

```
BEGIN

   uut: Flip_Flop PORT MAP (
          d => d,
          reset => reset,
          clk => clk,
          q => q
        );

   -- Clock process definitions
   clk_process :process
   begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
   end process;

        d <= '1', 'X' after 22 ns, '1' after 24 ns, 'U' after 35 ns, '1' after 40 ns;
        reset <= '1' after 50 ns, '0' after 60 ns;

END;
```

گزارش کار:
برای هر یک از فلیپ فلاپ های D,T,JK

- طراحی شماتیک طرح (محتوای توصیف) بر صورت فیزیکی
- پیاده‌سازی مدارها با استفاده از زبان‌های توصیف
- شبیه‌سازی مدارهای توصیف شده و تهیه Testbench برای آن و نشان دادن درستی عملکرد آن‌ها.